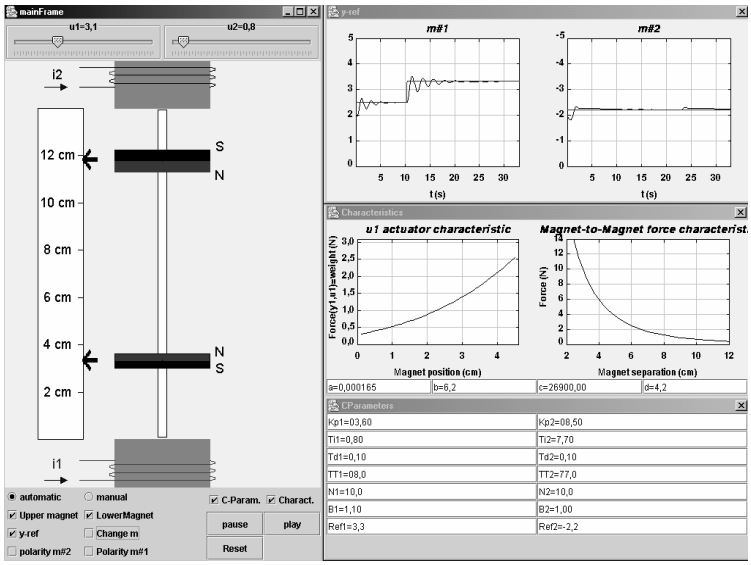
Universidad Nacional de Educación a Distancia

APLICACIÓN DE LA SIMULACIÓN POR ORDENADOR A LA ENSEÑANZA DE LAS CIENCIAS

Programa de Formación del Profesorado y Formación Continua de la UNED

Curso 2007-08



Alfonso Urquía

Carla Martín

Departamento de Informática y Automática

Escuela Técnica Superior de Ingeniería Informática, UNED

Juan del Rosal 16, 28040 Madrid, España

{aurquia,carla}@dia.uned.es

http://www.euclides.dia.uned.es/

Presentación

El Equipo Docente le da la bienvenida al curso "*Aplicación de la simulación por ordenador a la enseñanza de las Ciencias*", perteneciente al Programa de Formación del Profesorado y Formación Continua de la UNED.

Encontrará que la materia impartida en este curso tiene importantes aplicaciones prácticas. Una de ellas es la programación de *laboratorios virtuales*: herramientas docentes que, adecuadamente empleadas, resultan extremadamente eficaces. Se trata de programas informáticos ideados para reproducir algunos aspectos del comportamiento de un sistema real, permitiendo que el alumno interaccione con el modelo simulado de forma análoga a como lo haría con el sistema real. Una de las ventajas fundamentales de este enfoque es que el alumno desempeña un papel activo en su proceso de aprendizaje, con lo cual éste le resulta más ameno y motivador.

El entorno de simulación que se emplea en este curso es "*Easy Java Simulations*" (abreviado: Ejs), el cual ha sido ideado y desarrollado por el Prof. Dr. Francisco Esquembre[1]. Como quedará patente a lo largo del curso, la elección de este entorno responde a motivos bien fundamentados. Tres de las características por la cuales Ejs es adecuado para la docencia son las siguientes. En primer lugar, su gran facilidad de instalación y manejo: es un entorno concebido para ser usado por educadores y alumnos sin profundos conocimientos de programación. En segundo lugar, Ejs puede ser descargado, usado y distribuido gratuitamente. Finalmente, Ejs genera automáticamente el *laboratorio virtual* como una aplicación o como un applet de Java. En este último caso, el *laboratorio virtual* puede ser publicado en una página web para su uso a través de Internet.

Le animamos, por tanto, a que instale Ejs en su propio ordenador y a que vaya realizando por si mismo los ejemplos que vamos proponiendo a lo largo de este texto. Asimismo, nos gustaría que ideara y realizara sus propias simulaciones interactivas, acordes con sus intereses y con la actividad docente que usted realiza. Esperamos, en definitiva, que encuentre aplicables los conocimientos adquiridos en este curso, así como el material que en él le proporcionamos, a su labor docente cotidiana.

Para finalizar esta presentación, nos gustaría invitarle a que envíe sus dudas y sugerencias de mejora al profesor Alfonso Urquía (e-mail: aurquia@dia.uned.es) y a la profesora Carla Martín (e-mail: carla@dia.uned.es), de modo que podamos atenderlas.

El Equipo Docente.

[1] Prof. Dr. Francisco Esquembre, Dpto. de Matemáticas, Universidad de Murcia, Campus de Espinardo, 30071 Murcia (España). E-mail: fem@um.es

Sitio web del curso

En el sitio web

http://www.euclides.dia.uned.es/simulab-pfp/

puede encontrar información y noticias referentes al curso.

También, puede acceder a un curso online gratuito, escrito en inglés, sobre programación de laboratorios virtuales con Easy Java Simulations. Se trata de una versión muy resumida del curso que aquí le presentamos.

Este curso online recibió el segundo premio al "Curso mejor diseñado" (*Best Designed Course Award*) en la competición internacional celebrada con motivo de la 6^{th} Int. Conference Virtual University, que tuvo lugar en Bratislava (Eslovaquia), en diciembre de 2005.

Finalmente, en el sitio web puede visualizar los laboratorios virtuales explicados en el curso y algunos de los laboratorios virtuales más destacados programados por nuestros alumnos de años anteriores.

Registro de alumnos

El Negociado del Programa de Formación del Profesorado de la UNED facilita a los Equipos Docentes la dirección postal y el número de teléfono de cada alumno, pero no la dirección de e-mail.

Si dispone de correo electrónico y desea que, en caso de que surja la necesidad, nos pongamos en contacto con usted mediante e-mail, envíenos al Equipo Docente (aurquia@dia.uned.es, carla@dia.uned.es) un correo electrónico indicándonos simplemente que está matriculado en el curso. De esta manera, también sabremos que ha recibido este material.

Orientación metodológica

Objetivos docentes

En este curso se pretende que el alumno aprenda a hacer un uso eficaz de la simulación interactiva por ordenador como herramienta docente. Para ello, debe adquirir la capacidad de diseñar y programar *laboratorios virtuales* útiles para la enseñanza de las Ciencias, empleando para ello el entorno de simulación Ejs.

Contenidos

Los contenidos del curso van siendo expuestos gradualmente, a través de una sucesión de casos de estudio. Estos contenidos son:

1. Diseño, programación y distribución a través de Internet de laboratorios virtuales, usando el entorno de simulación Easy Java Simulations (Ejs).
2. Fundamentos de la aplicación de la simulación dinámica interactiva por ordenador a la enseñanza de las Ciencias. Metodología docente. Preparación del material docente.
3. Ejemplos de aplicación de laboratorios virtuales, programados con Ejs, a la enseñanza de las Ciencias.

Material didáctico

Al comienzo del curso se entregará al alumno el siguiente material:

– Una copia impresa del Texto Base del curso:

Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Alfonso Urquía y Carla Martín

En este texto no sólo se realiza una introducción tutorial al manejo de Ejs, sino que también se explican los fundamentos y la aplicación de una metodología para el modelado orientado a la simulación interactiva usando Ejs.

– Un CD-ROM, que contiene:
  - El Texto Base del curso en formato electrónico.
  - El entorno de simulación Ejs.
  - El entorno de desarrollo de Java.
  - El código de los laboratorios virtuales explicados en el Texto Base del curso.
  - Manuales y otra documentación de Ejs.

El alumno deberá emplear como material didáctico del curso, tanto el Texto Base, como los manuales y apéndices de Ejs incluidos en el CD-ROM.

Asimismo, es muy recomendable la lectura del libro:

Creación de Simulaciones Interactivas en Java
Francisco Esquembre
Editorial: Pearson Prentice-Hall

Metodología y actividades

El curso tiene un enfoque eminentemente aplicado. Se entregará al alumno el software Ejs, un texto en el que se explica paso a paso su manejo, ilustrado mediante una colección de ejemplos de aplicación a la enseñanza de las Ciencias. El alumno podrá realizar estos ejemplos en su propio ordenador y desarrollar su propio material docente.

Nivel del curso

Iniciación.

Alumnado

Este curso está destinado a profesores de materias de Ciencias (física, química, biología, matemáticas, economía, etc.) que estén interesados en el empleo de la simulación interactiva por ordenador como herramienta docente. Para poder realizar el curso, el alumno debe disponer de ordenador y de conexión a Internet.

Duración y dedicación

6 meses, 120 horas (12 créditos)

Criterios de evaluación y calificación

El alumno deberá realizar individualmente un trabajo en el cual aplique eficazmente la simulación interactiva por ordenador a la docencia. Para ello, deberá emplear el entorno de simulación Ejs.

Dicho trabajo consistirá en la realización de dos laboratorio virtuales, uno sencillo y otro más complejo, cuyos contenidos quedan a la elección del alumno.

El alumno deberá entregar el fichero .xml de cada laboratorio, así como todos los ficheros de imágenes que haya usado para la descripción de la *Introducción* y la *Vista* de los dos laboratorios. Con el fin de facilitar la revisión de los trabajos, *deberá guardar todas las figuras de los dos laboratorios en un mismo directorio*, y enviárnoslo junto con los dos ficheros .xml.

Es necesario que los dos laboratorios virtuales funcionen correctamente y que estén ampliamente documentados, empleando para ello el panel *Introducción*. Esta documentación debe incluir cinco ventanas, con los nombres y el contenido que se detalla a continuación:

1. **Objetivo.** Objetivo docente del laboratorio: qué concepto se pretende ilustrar mediante el laboratorio.
2. **Modelo matemático.** Explicación detallada del modelo matemático: descripción de las variables y las ecuaciones.
3. **Algoritmo de la simulación.** Clasificación de las variables en conocidas y desconocidas. Asignación de la causalidad computacional. Diagrama de flujo del algoritmo de la simulación.
4. **Actividades.** Actividades que se proponen al alumno, a realizar con el laboratorio virtual. Para ello, debe realizarse un descripción de todas las capacidades interactivas que posee el laboratorio virtual (sólo una descripción de la funcionalidad, no la explicación de cómo se ha programado).
5. **Autor.** Nombre del autor.

El alumno deberá enviar los dos laboratorios virtuales adjuntos en un correo electrónico, a las direcciones:

aurquia@dia.uned.es
carla@dia.uned.es

antes del **20 de junio**.

El "asunto" (subject) del mensaje deberá ser: *Trabajo Curso Formación Profesorado*.

En el cuerpo del mensaje deberá indicar su nombre y apellidos.

Atención al alumno

Las consultas deben dirigirse a los profesores Alfonso Urquía y Carla Martín, y pueden realizarse por cualquiera de los tres métodos siguientes:

- Llamando a los números de teléfono 91 398 84 59 / 82 53, cualquier lunes lectivo, de 16h a 20h.
- Enviando un correo electrónico a las direcciones:

  aurquia@dia.uned.es
  carla@dia.uned.es

- Acudiendo personalmente, cualquier lunes lectivo, entre las 16h y las 20h, al despacho 5.15 de la E.T.S. de Ingeniería Informática de la UNED. *En este caso, sería muy recomendable que enviara un correo electrónico a los profesores Alfonso Urquía y Carla Martín, manifestándoles su intención de venir, con el fin de concertar una cita y poder atenderle adecuadamente.*

Índice

Parte I

Fundamentos del modelado y la simulación

Tema 1

Conceptos básicos del modelado y la simulación

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Discutir los conceptos "sistema", "modelo", "simulación" y "marco experimental".
- Describir y comparar los diferentes tipos de modelos.
- Comparar y reconocer los distintos tipos de modelos matemáticos.

1.1. Sistemas y modelos

El concepto de *modelo* puede ser definido de varias maneras. En el sentido amplio del término, puede considerarse que un *modelo* es "*una representación de un sistema desarrollada para un propósito específico*".

Puesto que la finalidad de un modelo es ayudarnos a responder preguntas sobre un determinado sistema, el primer paso en la construcción de un modelo es definir cuál es el sistema y cuáles son las preguntas.

En este contexto, puede entenderse que un *sistema* es "*cualquier objeto o conjunto de objetos cuyas propiedades se desean estudiar*". Con una definición tan amplia, *cualquier fuente potencial de datos* puede considerarse un sistema.

Algunos ejemplos de sistema son:

- Un conjunto de planetas, que giran alrededor de una estrella.
- Una máquina fotocopiadora.
- El servicio de emergencias de un hospital, incluyendo el personal, las salas, el equipamiento y los medios para el transporte de los pacientes.
- Un circuito, compuesto por una fuente y una resistencia.
- El ecosistema de una determinada selva tropical.

Es característico de la naturaleza humana plantearse cuestiones acerca de las propiedades de los sistemas. Por ejemplo, para el sistema formado por el conjunto de planetas y la estrella, una pregunta típica es: ¿Cuál es el periodo de giro de cada uno de los planetas? Para la máquina fotocopiadora: ¿Cómo debo ajustar los mecanismos de la máquina a fin de obtener

copias de calidad óptima? Para el servicio de emergencias: ¿Cómo puedo organizar el servicio, de modo que el tiempo de espera de los pacientes sea mínimo?

Muchas cuestiones de este tipo pueden ser resueltas mediante experimentación. De hecho, éste ha sido el método empleado durante siglos para avanzar en el conocimiento: basta con conectar la fuente a la resistencia y observa qué ocurre.

Un *experimento* puede definirse como "*el proceso de extraer datos de un sistema sobre el cual se ha ejercido una acción externa*".

Cuando es posible trabajar directamente con el sistema real, el *método experimental* presenta indudables ventajas, ya que está basado en sólidos fundamentos científicos. Sin embargo, también presenta sus limitaciones, ya que en ocasiones es imposible o desaconsejable experimentar con el sistema real. Algunas de estas razones son las siguientes:

- Quizá la más evidente de ellas es que el sistema aun no exista físicamente. Esta situación se plantea frecuentemente en la fase de diseño de nuevos sistemas, cuando el ingeniero necesita predecir el comportamiento de los mismos antes de que sean construidos.
- Otra posible razón es el elevado coste económico del experimento.
- El experimento puede producir perjuicio o incomodidad. Por ejemplo, experimentar con un nuevo sistema de facturación en un aeropuerto puede producir retrasos y problemas imprevisibles que perjudiquen al viajero.
- En ocasiones el tiempo requerido para la realización del experimento lo hace irrealizable. Casos extremos pueden encontrarse en los estudios geológicos o cosmológicos, de evolución de las especies, sociológicos, etc.
- Algunos experimentos son peligrosos, y por tanto es desaconsejable realizarlos. Por ejemplo, sería inapropiado usar el sistema real para adiestrar a los operarios de una central nuclear acerca de cómo deben reaccionar ante situaciones de emergencia.
- En ocasiones el experimento requiere modificar variables que en el sistema real o bien no están accesibles o no pueden ser modificadas en el rango requerido. Con un modelo matemático adecuado, se pueden ensayar condiciones de operación extremas que son impracticables en el sistema real.

En cualquiera de los casos anteriores, el modelado y la simulación son las técnicas adecuadas para el análisis del sistema. *A excepción de la experimentación con el sistema real, la simulación es la única técnica disponible que permite analizar sistemas arbitrarios de forma precisa, bajo diferentes condiciones experimentales*.

1.2. Tipos de modelos

Los seres humanos, en nuestra vida cotidiana, empleamos continuamente modelos para comprender y predecir el comportamiento de sistemas (ver la Figura 1.1). Por ejemplo, considerar que alguien es "amable" constituye un modelo del comportamiento de esta persona. Este modelo nos ayuda a responder, por ejemplo, a la pregunta: "¿cómo reaccionará si le pedimos un favor?". También disponemos de modelos de los sistemas técnicos que están basados en la intuición y en la experiencia. Todos estos se llaman *modelos mentales* .

Por ejemplo, aprender a conducir un coche consiste parcialmente en desarrollar un modelo mental de las propiedades de la conducción del coche. Asimismo, un operario trabajando en determinado proceso industrial sabe cómo el proceso reacciona ante diferentes acciones: el operario, mediante el entrenamiento y la experiencia, ha desarrollado un modelo mental del proceso.

Otro tipo de modelos son los *modelos verbales*, en los cuales el comportamiento del sistema es descrito mediante palabras: si se aprieta el freno, entonces la velocidad del coche se reduce. Los *sistemas expertos* son ejemplos de modelos verbales formalizados. Es importante diferenciar entre los modelos mentales y los verbales. Por ejemplo, nosotros usamos un modelo mental de la dinámica de la bicicleta cuando la conducimos, sin embargo no es sencillo convertirlo a un modelo verbal.

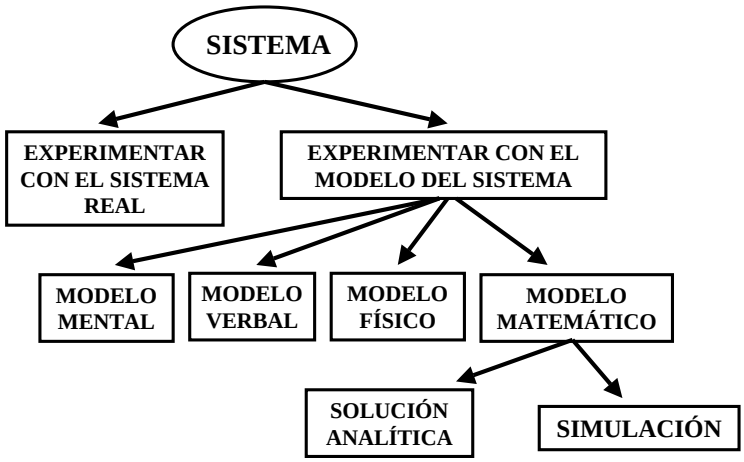


Figura 1.1: Formas de estudiar un sistema.

Además de los modelos mentales y verbales, existe otro tipo de modelos que tratan de imitar al sistema real. Son los *modelos físicos*, como las maquetas a escala que construyen los arquitectos, diseñadores de barcos o aeronaves para comprobar las propiedades estéticas, aerodinámicas, etc.

Finalmente, existe un cuarto tipo de modelos, los *modelos matemáticos*. En ellos, las relaciones entre las cantidades que pueden ser observadas del sistema (distancias, velocidades, flujos, etc.) están descritas mediante relaciones matemáticas. En este sentido, la mayoría de las leyes de la naturaleza son modelos matemáticos. En esencia, la Ciencia consiste en la construcción de modelos de una porción de la realidad y en el estudio de sus propiedades.

Por ejemplo, para el sistema "masa puntual", la Ley de Newton del movimiento describe la relación entre la fuerza y la aceleración. Asimismo, para el sistema "resistencia eléctrica", la Ley de Ohm describe la relación entre la caída de tensión y el flujo de corriente.

En algunos casos, las relaciones matemáticas que constituyen los modelos son sencillas y pueden resolverse analíticamente. Sin embargo, en la mayoría de los casos, los modelos no pueden resolverse analíticamente y deben estudiarse, con ayuda del ordenador, aplicando métodos numéricos. Este experimento numérico realizado sobre el modelo matemático, recibe el nombre de *simulación*.

1.3. Modelos matemáticos

La finalidad de un estudio de simulación (es decir, las preguntas que debe responder) condiciona las hipótesis empleadas en la construcción del modelo, y éstas a su vez determinan qué tipo de modelo resulta más adecuado al estudio. De hecho, un mismo sistema puede ser modelado de múltiples formas, empleando diferentes tipos de modelos, dependiendo de la finalidad perseguida en cada caso.

Existen diferentes clasificaciones de los modelos matemáticos, atendiendo a diferentes criterios. A continuación se describen algunas de las clasificaciones más comúnmente usadas.

Determinista vs Estocástico

Un modelo matemático es *determinista* cuando todas sus variables de entrada son deterministas, es decir, el valor de cada una de ellas es conocido en cada instante.

Por el contrario, un modelo es *estocástico* cuando alguna de sus variables de entrada es aleatoria. Las variables del modelo calculadas a partir de variables aleatorias son también aleatorias. Por ello, la evolución de este tipo de sistemas debe estudiarse en términos probabilísticos.

Por ejemplo, considérese el modelo de un parking, en el cual las entradas y salidas de coches se producen en instantes de tiempo aleatorios. La aleatoriedad de estas variables se propaga a través de la lógica del modelo, de modo que las variables dependientes de ellas también son aleatorias. Este sería el caso, por ejemplo, del tiempo que transcurre entre que un cliente deja aparcado su vehículo y lo recoge (tiempo de aparcamiento), el número de vehículos que hay aparcados en un determinado instante, etc.

Es importante tener en cuenta que realizar una única réplica de una simulación estocástica es equivalente a realizar un experimento físico aleatorio una única vez.

Por ejemplo, si se realiza una simulación del comportamiento del parking durante 24 horas, es equivalente a observar el funcionamiento del parking real durante 24 horas. Si se repite la observación al día siguiente, seguramente los resultados obtenidos serán diferentes, y lo mismo sucede con la simulación: si se realiza una segunda réplica independiente de la primera, seguramente los resultados serán diferentes.

La consecuencia que debe extraerse de ello es que el diseño y el análisis de los experimentos de simulación estocásticos debe hacerse teniendo en cuenta esta incertidumbre en los resultados, es decir, debe hacerse empleando técnicas estadísticas.

Las dificultades asociadas a la simulación de modelos estocásticos pueden invitarnos en ocasiones a realizar hipótesis adicionales, con el fin de eliminar la incertidumbre en el valor de las variables de entrada. Un ejemplo típico consiste en sustituir cada variable de entrada aleatoria por otra determinista, cuyo valor sea la media de la distribución de probabilidad de aquella. Este modelo determinista, obtenido de eliminar la incertidumbre en el valor de las variables de entrada, proporcionará resultados no aleatorios. Sin embargo, esta simplificación hará que probablemente el modelo ya no sea una representación del sistema válida para el objetivo del estudio.

Estático vs Dinámico

Un *modelo de simulación estático* es una representación de un sistema en un instante de tiempo particular, o bien un modelo que sirve para representar un sistema en el cual el tiempo no juega ningún papel. Ejemplo de simulaciones estáticas son las simulaciones de Monte Carlo.

Por otra parte, un *modelo de simulación dinámico* representa un sistema que evoluciona con el tiempo.

De tiempo continuo vs De tiempo discreto vs Híbrido

Un *modelo de tiempo continuo* está caracterizado por el hecho de que el valor de sus variables de estado puede cambiar infinitas veces (es decir, de manera continua) en un intervalo finito de tiempo. Un ejemplo es el nivel de agua en un depósito.

Por el contrario, en un *modelo de tiempo discreto* los cambios pueden ocurrir únicamente en instantes separados en el tiempo. Sus variables de estado pueden cambiar de valor sólo un número finito de veces por unidad de tiempo.

Pueden definirse modelos con algunas de sus variables de estado de tiempo continuo y las restantes de tiempo discreto. Este tipo de modelos, con parte de tiempo continuo y parte de tiempo discreto, de llama *modelos híbridos*.

Tal como se ha indicado al comienzo de la sección, la decisión de realizar un modelo continuo o discreto depende del objetivo específico del estudio y no del sistema en sí. Un ejemplo de ello lo constituyen los modelos del flujo de tráfico de vehículos. Cuando las características y el movimiento de los vehículos individuales son relevantes puede realizarse un modelo discreto. En caso contrario, puede resultar más sencillo realizar un modelo continuo.

En este punto es conveniente realizar una consideración acerca de los modelos de tiempo continuo y discreto. Al igual que las *variables continuas* (aquellas que pueden tomar cualquier valor intermedio en su rango de variación) son una idealización, también lo son los *modelos de tiempo continuo*. Cualquiera que sea el procedimiento de medida que se emplee para medir el valor de una variable, tendrá un límite de precisión. Este límite marca la imposibilidad de dar una medida continua y supone que, en la práctica, todas las medidas son discretas. Igualmente, los ordenadores trabajan con un número finito de cifras decimales. Sin embargo, para los razonamientos teóricos, conviene considerar ciertas variables como continuas.

Al simular mediante un computador digital un modelo de tiempo continuo, debe discretizarse el eje temporal a fin de evitar el problema de los infinitos cambios en el valor de los estados. Esta discretización constituye una aproximación (con su error asociado) que transforma el modelo de tiempo continuo en un modelo de tiempo discreto. Por ejemplo, si se discretiza el eje temporal del modelo de tiempo continuo

$$\frac{dx}{dt} = f\left(x, u, t\right) \tag{1.1}$$

con un intervalo de discretización Δt, se obtiene (empleando el método de Euler explícito) el siguiente modelo de tiempo discreto:

$$\frac{x_{K+1} - x_K}{\Delta t} = f\left(x_K, u_K, t_K\right) \;\longrightarrow\; x_{K+1} = x_K + \Delta t \cdot f\left(x_K, u_K, t_K\right) \tag{1.2}$$

1.4. El marco experimental

Al igual que se distingue entre el sistema real y el experimento, es conveniente distinguir entre la *descripción del modelo* y la *descripción del experimento*. Esto es así tanto desde el punto de vista conceptual como práctico. Sin embargo, en el caso del modelo esta separación implica cierto riesgo: su empleo en unas condiciones experimentales para las cuales no es válido.

Por supuesto, cuando se trabaja con sistemas reales este riesgo nunca existe, ya que un sistema real es válido para cualquier experimento. Por el contrario, cualquier modelo está fundamentado en un determinado conjunto de hipótesis. Cuando las condiciones experimentales son tales que no se satisfacen las hipótesis del modelo, éste deja de ser válido. Para evitar este problema, la descripción del modelo debe ir acompañada de la documentación de su *marco experimental*. Éste establece el conjunto de experimentos para el cual el modelo es válido.

Tema 2

Simulación de modelos de tiempo continuo

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Clasificar las variables de un modelo matemático de tiempo continuo en: parámetros, variables de estado y variables algebraicas.
- Aplicar el método de integración explícito de Euler.
- Realizar la asignación de la causalidad computacional.
- Plantear el algoritmo de la simulación de modelos de tiempo continuo sencillos.

En este tema se explican algunos aspectos fundamentales de la simulación de los modelos matemáticos de tiempo continuo. Estas explicaciones constituyen la base para la comprensión del algoritmo para la simulación interactiva que emplea Ejs, que se describe en el Tema 5.

2.1. Variables y ecuaciones

Los modelos matemáticos están compuestos por *ecuaciones*, que describen la relación entre las magnitudes relevantes del sistema. Estas magnitudes reciben el nombre de *variables*.

Ejemplo 2.1.1. *Sobre un objeto de masa constante (m) actúan dos fuerzas: la fuerza de la gravedad ($m \cdot g$) y una fuerza armónica de amplitud (F_0) y frecuencia (ω) constantes. La fuerza total aplicada sobre el objeto (F) es:*

$$F = m \cdot g + F_0 \cdot \sin(\omega \cdot t) \tag{2.1}$$

donde g la aceleración gravitatoria, que se considera constante. Se aplica el criterio siguiente: la aceleración tiene signo positivo cuando tiene sentido vertical ascendente. Por ejemplo, la aceleración gravitatoria terrestre sería $g = -9.8\ m \cdot s^{-2}$

La fuerza neta (F) aplicada sobre el objeto hace que éste adquiera una aceleración (a), que viene determinada por la relación siguiente:

$$m \cdot a = F \tag{2.2}$$

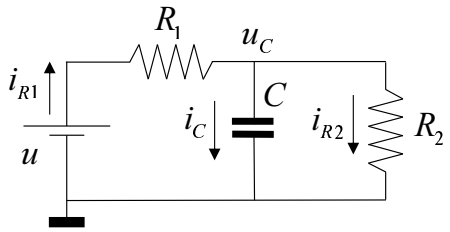


Figura 2.1: Circuito RC.

La posición y la velocidad del objeto pueden calcularse teniendo en cuenta que la derivada respecto al tiempo de la posición es la velocidad, y que la derivada respecto al tiempo de la velocidad es la aceleración. El modelo obtenido es el siguiente:

$$F = m \cdot g + F_0 \cdot \sin(\omega \cdot t) \quad (2.3)$$
$$m \cdot a = F \quad (2.4)$$
$$\frac{dx}{dt} = v \quad (2.5)$$
$$\frac{dv}{dt} = a \quad (2.6)$$

Este modelo está compuesto por cuatro ecuaciones, Ecs. (2.3) – (2.6), las cuales describen la relación existente entre las magnitudes relevantes del sistema, que son las variables del modelo:

- *La aceleración gravitatoria (g),*
- *La amplitud (F_0) y frecuencia (ω) de la fuerza armónica,*
- *La fuerza neta (F) aplicada sobre el objeto*
- *La masa (m), posición (x), velocidad (v) y aceleración (a) del objeto.*

□

2.2. Parámetros, variables de estado y variables algebraicas

El punto de partida para la simulación del modelo consiste en clasificar sus variables de acuerdo al criterio siguiente:

- **Parámetros**. Son aquellas variables cuyo valor permanece constante durante la simulación.
- **Variables de estado**. Son las variables que están derivadas respecto al tiempo.
- **Variables algebraicas**. Son las restantes variables del modelo. Es decir, aquellas que no aparecen derivadas en el modelo y que no son constantes.

Ejemplo 2.2.1. *De acuerdo con la clasificación anterior, el modelo descrito en el Ejemplo 2.1.1 tiene:*

- *Cuatro parámetros: g, m, F_0, ω.*
- *Dos variables de estado: x, v.*
- *Dos variables algebraicas: a, F.*

Obsérvese que la variable tiempo (t) no se incluye en la clasificación. □

Ejemplo 2.2.2. *Considérese el circuito eléctrico mostrado en la Figura 2.1, el cual está compuesto por un generador de tensión, dos resistencias y un condensador. El modelo de este circuito consta de las ecuaciones siguientes:*

$$u = u_0 \cdot \sin(\omega \cdot t) \tag{2.7}$$
$$i_{R1} = i_{R2} + i_C \tag{2.8}$$
$$u - u_C = R_1 \cdot i_{R1} \tag{2.9}$$
$$C \cdot \frac{du_C}{dt} = i_C \tag{2.10}$$
$$u_C = i_{R2} \cdot R_2 \tag{2.11}$$

La Ec. (2.7) es la relación constitutiva del generador de tensión. La amplitud (u_0) y la frecuencia (ω) son independientes del tiempo (t).

Las Ecs. (2.9) y (2.11) son las relaciones constitutivas de las resistencias. La Ec. (2.10) es la relación constitutiva del condensador. Los valores de la capacidad (C) y de las resistencias (R_1, R_2) son independientes del tiempo.

Finalmente, la Ec. (2.8) impone que la suma de las corrientes entrantes a un nodo debe ser igual a la suma de las corrientes salientes del mismo.

Las variables de este modelo se clasifican de la manera siguiente:

- *Parámetros:* u_0, ω, C, R_1, R_2.
- *Variable de estado:* u_C.
- *Variables algebraicas:* u, i_{R1}, i_{R2}, i_C.

□

2.3. Un algoritmo para la simulación de modelos de tiempo continuo

En la Figura 2.2 se muestra un algoritmo para la simulación de modelos de tiempo continuo. Puede comprobarse que la clasificación de las variables del modelo en parámetros, variables de estado y variables algebraicas constituye la base para la simulación del modelo:

- Al comenzar la simulación, se asignan valores a los *parámetros*. Estos valores permanecen constantes durante toda la simulación.
- Las *variables de estado* son calculadas mediante la integración numérica de sus derivadas. Por ejemplo, la función de paso del método explícito de Euler para la ecuación diferencial ordinaria
$$\frac{dx}{dt} = f(x,t) \tag{2.12}$$
es la siguiente:
$$x_{i+1} = x_i + f(x_i, t_i) \cdot \Delta t \tag{2.13}$$
donde x_i y x_{i+1} representan el valor de la variable de estado x en los instantes t_i y $t_i + \Delta t$ respectivamente, y $f(x_i, t_i)$ representa el valor de la derivada de x (es decir, $\frac{dx}{dt}$) en el instante t_i.
- El valor de las *variables algebraicas* se calcula, en cada instante de tiempo, a partir de valor de las variables de estado en ese instante y del valor de los parámetros.

La condición de terminación de la simulación depende del estudio en concreto que vaya a realizarse sobre el modelo. Puede ser, por ejemplo, que se alcance determinado valor de la variable tiempo, o que una determinada variable satisfaga cierta condición.

El valor del *tamaño del paso de integración* (Δt) debe escogerse alcanzando un compromiso entre precisión y carga computacional. Cuanto menor sea el valor de Δt, menor es el error

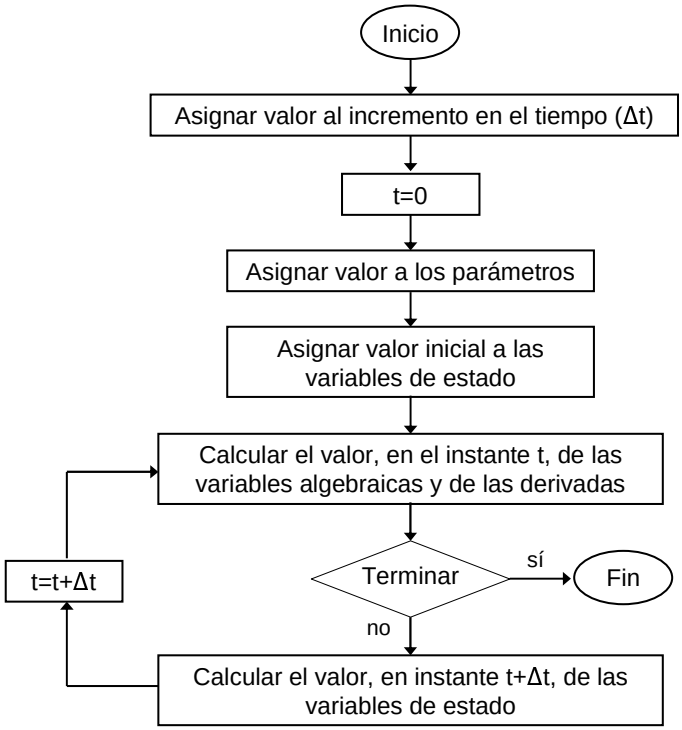


Figura 2.2: Algoritmo de la simulación de los modelos matemáticos de tiempo continuo.

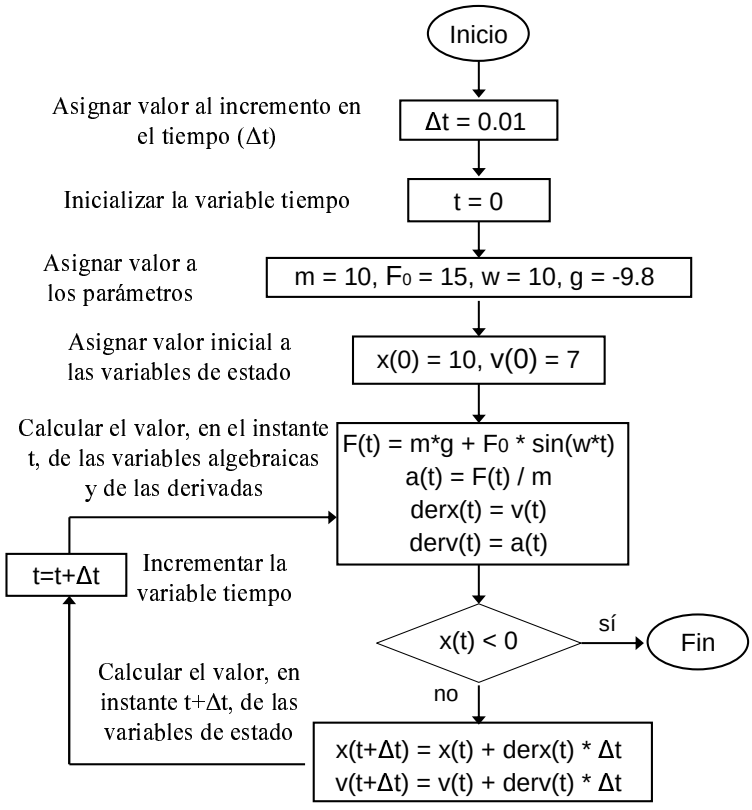


Figura 2.3: Algoritmo de la simulación del modelo descrito en el Ejemplo 2.1.1.

que se comete en el cálculo de las variables del modelo, pero mayor es el tiempo de ejecución de la simulación.

Un procedimiento para estimar el error cometido al escoger un determinado valor de Δt es comparar los resultados obtenidos usando ese valor y los obtenidos usando un valor menor, por ejemplo, $\frac{\Delta t}{2}$. Si la diferencia entre ambos resultados es aceptable para los propósitos del estudio de simulación que se está realizando, entonces el valor Δt es adecuado. En caso contrario, se comparan los resultados obtenidos usando $\frac{\Delta t}{2}$ y $\frac{\Delta t}{4}$. Si el error es aceptable, de emplea $\frac{\Delta t}{2}$. Si el error es demasiado grande, se investigan los valores $\frac{\Delta t}{4}$ y $\frac{\Delta t}{8}$, y así sucesivamente.

Ejemplo 2.3.1. *En la Figura 2.3 se muestra el algoritmo de la simulación del modelo descrito en el Ejemplo 2.1.1. Obsérvese que:*

- *Las derivadas de las variables de estado se han sustituido por variables auxiliares, cuyo nombre es igual al de la variable de estado, pero anteponiéndole el prefijo "der":*

$$\frac{dx}{dt} \rightarrow derx \tag{2.14}$$

$$\frac{dv}{dt} \rightarrow derv \tag{2.15}$$

- *Para realizar el cálculo de las variables algebraicas y las derivadas, se ha despejado de cada ecuación la variable a calcular y se han ordenado las ecuaciones, de modo que sea posible resolverlas en secuencia. Se ha obtenido la siguiente secuencia de asignaciones:*

$$F = m \cdot g + F_0 \cdot \sin(\omega \cdot t) \tag{2.16}$$

$$a = \frac{F}{m} \tag{2.17}$$

$$derx = v \tag{2.18}$$

$$derv = a \tag{2.19}$$

La secuencia de cálculos, según se muestra en la Figura 2.3, es la siguiente:

$t = 0$ $m = 10, F_0 = 15, \omega = 10, g = -9.8$

$x(0) = 10, v(0) = 7$

$F(0) = m \cdot g + F_0 \cdot \sin(\omega \cdot t) \rightarrow F(0) = 10 \cdot (-9.8) + 15 \cdot \sin(10 \cdot 0) = -98$

$a(0) = F(0)/m \rightarrow a(0) = -98/10 = -9.8$

$derx(0) = v(0) \rightarrow derx(0) = 7$

$derv(0) = a(0) \rightarrow derv(0) = -9.8$

$x(0.01) = x(0) + derx(0) \cdot \Delta t \rightarrow x(0.01) = 10 + 7 \cdot 0.01 = 10.07$

$v(0.01) = v(0) + derv(0) \cdot \Delta t \rightarrow v(0.01) = 7 + -9.8 \cdot 0.01 = 6.902$

$t = 0.01$ $F(0.01) = m \cdot g + F_0 \cdot \sin(\omega \cdot t) \rightarrow F(0.01) = 10 \cdot (-9.8) + 15 \cdot \sin(10 \cdot 0.01) = -96.5025$

$a(0.01) = F(0.01)/m \rightarrow a(0.01) = -96.5025/10 = -9.65025$

$derx(0.01) = v(0.01) \rightarrow derx(0.01) = 6.902$

$derv(0.01) = a(0.01) \rightarrow derv(0.01) = -9.65025$

$x(0.02) = x(0.01) + derx(0.01) \cdot \Delta t \rightarrow x(0.02) = 10.07 + 6.902 \cdot 0.01 = 10.139$

$v(0.02) = v(0.01) + derv(0.01) \cdot \Delta t \rightarrow v(0.02) = 6.902 + -9.65025 \cdot 0.01 = 6.8055$

$t = 0.02$ *Y así sucesivamente ...*

En la Figura 2.4 se muestra la evolución de la posición, la velocidad y la aceleración del objeto, obtenidas ejecutando el algoritmo mostrado en la Figura 2.3. En el eje horizontal de ambas gráficas se representa el tiempo.

La condición de finalización de la simulación es que el objeto toque el suelo. Es decir, que se satisfaga la condición: $x < 0$.

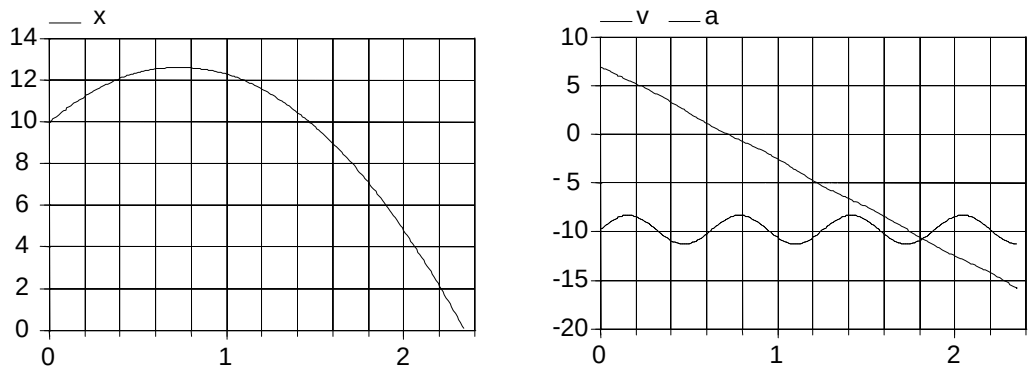


Figura 2.4: Resultado de la ejecución del algoritmo mostrado en la Figura 2.3.

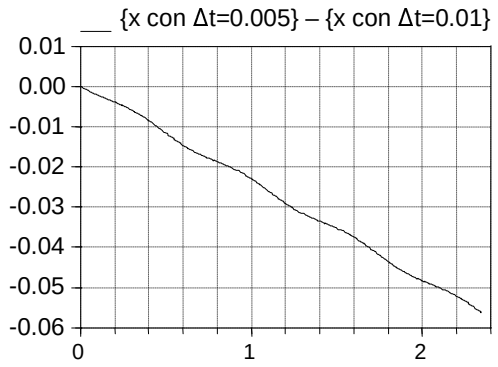


Figura 2.5: Diferencia en la evolución temporal de la posición para dos valores de Δt.

Cabe plantearse si el valor 0.01 *para el tamaño del paso de integración (*Δt*) es adecuado. En la Figura 2.5 se muestra la diferencia entre la posición del objeto, calculada usando* $\Delta t = 0.005$*, y la posición calculada usando* $\Delta t = 0.01$*. En el eje horizontal está representado el tiempo. Esta gráfica permite obtener una idea de cómo va acumulándose el error y cuál es la magnitud de éste. Dependiendo cuál sea el objetivo del estudio de simulación, esta magnitud del error (y consecuentemente, el valor* $\Delta t = 0.01$*) será o no aceptable.* □

2.4. Causalidad computacional

Como se ha mostrado en el Ejemplo 2.3.1, para realizar el cálculo de las variables algebraicas y las derivadas, es preciso despejar de cada ecuación la variable a calcular y ordenar las ecuaciones, de modo que sea posible resolverlas en secuencia.

Esto tiene un carácter general. Para plantear el algoritmo de la simulación de un modelo, es preciso realizar las tareas siguientes:

1. Decidir qué variable debe calcularse de cada ecuación y cómo deben ordenarse las ecuaciones del modelo, de modo que puedan ser resueltas en secuencia. A esta decisión se la denomina *asignación de la causalidad computacional*.
2. Una vez se ha decidido qué variable debe evaluarse de cada ecuación, debe manipularse simbólicamente la ecuación a fin de despejar dicha variable.

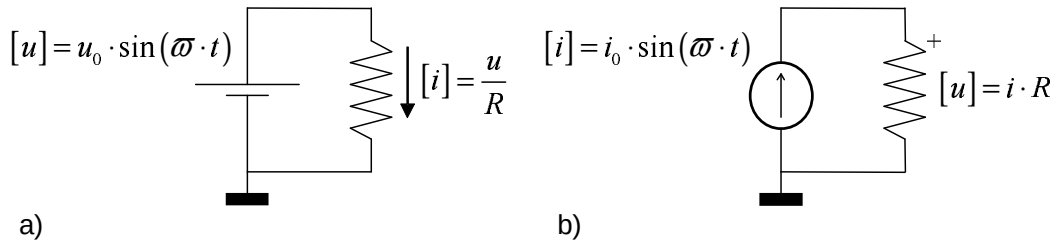


Figura 2.6: La causalidad computacional es una propiedad del modelo completo.

Ejemplo 2.4.1. *Supóngase que se desea modelizar una resistencia eléctrica mediante la Ley de Ohm. La Ley de Ohm establece que la caída de potencial, u, entre los bornes de una resistencia es igual al producto de un parámetro característico de la resistencia, R, por la intensidad de la corriente eléctrica, i, que circula a través de la resistencia:*

$$u = i \cdot R \qquad \textit{(Ley de Ohm)} \tag{2.20}$$

Esta ecuación constituye una relación entre las tres variables u, R e i, que es válida para cualquiera de las tres posibles causalidades computacionales admisibles de la ecuación:

$$[u] = i \cdot R \qquad [i] = \frac{u}{R} \qquad [R] = \frac{u}{i} \tag{2.21}$$

donde se ha señalado la variable a evaluar de cada ecuación incluyéndola entre corchetes y se ha despejado escribiéndola en el lado izquierdo de la igualdad. □

La consideración fundamental que debe hacerse llegado este punto es que:

> la causalidad computacional de una determinada ecuación del modelo no sólo depende de ella misma, sino que también depende del resto de las ecuaciones del modelo. Es decir, la **causalidad computacional es una propiedad global del modelo completo**.

Ejemplo 2.4.2. *La casualidad computacional de la ecuación de la resistencia (vea Ejemplo 2.4.1) depende del resto de las ecuaciones del modelo.*

Si la resistencia se conecta a un generador de tensión senoidal, las ecuaciones del modelo son:

$$u = [i] \cdot R \qquad \textit{(Relación constitutiva de la resistencia)} \tag{2.22}$$
$$[u] = u_0 \cdot \sin(\varpi \cdot t) \qquad \textit{(Relación constitutiva del generador de tensión)} \tag{2.23}$$

donde R, u_0 y ϖ son parámetros del modelo, es decir, su valor se supone conocido y no es preciso evaluarlos de las ecuaciones del modelo.

En las Ecuaciones (2.22) y (2.23) se ha señalado entre corchetes la variable que debe evaluarse de cada ecuación.

Para simular el modelo es preciso ordenar las ecuaciones y despejar la variable a evaluar en cada una de ellas. Realizando la ordenación y la manipulación simbólica, se obtiene (vea la Figura 2.6a):

$$[u] = u_0 \cdot \sin(\varpi \cdot t) \tag{2.24}$$
$$[i] = \frac{u}{R} \tag{2.25}$$

Como puede verse, en primer lugar debe calcularse el valor de la caída de tensión, a partir de la relación constitutiva del generador, y a continuación debe calcularse la corriente, a partir de la relación constitutiva de la resistencia (para lo cual debe usarse el valor de la tensión que ha sido previamente calculado).

En cambio, si esta misma resistencia se conecta a un generador sinusoidal de corriente, las ecuaciones del modelo, una vez ordenadas y manipuladas simbólicamente, serán (vea la Figura 2.6b):

$$[i] = i_0 \cdot \sin(\varpi \cdot t) \tag{2.26}$$
$$[u] = i \cdot R \tag{2.27}$$

En este caso, en primer lugar debe calcularse el valor de la corriente, a partir de la relación constitutiva del generador, y a continuación debe calcularse la caída de tensión, a partir de la relación constitutiva de la resistencia (para lo cual debe usarse el valor de la corriente que se ha sido previamente calculado).

Se observa que en el caso de la Ecuación (2.25) la relación constitutiva de la resistencia se usa para calcular la corriente, mientras que en el caso de la Ecuación (2.27) se usa para calcular la caída de tensión. La variable a evaluar depende no sólo de la relación constitutiva de la resistencia, sino también del resto del modelo. □

A continuación, se describe un procedimiento sistemático para asignar la causalidad computacional de un modelo[1]. Sin embargo, para aplicarlo deben previamente clasificarse las variables del modelo en *conocidas* y *desconocidas*, según sean conocidas o desconocidas en el instante de evaluación. En primer lugar, por tanto, se explicará cómo realizar esta clasificación.

2.5. Variables conocidas y desconocidas

A efectos de la asignación de la causalidad computacional, las variables del modelo se clasifican en desconocidas (o incógnitas) y conocidas. Dicha clasificación se realiza en función de que el valor de la variable deba o no ser evaluado de las ecuaciones del modelo.

Variables conocidas:

- La variable *tiempo*.
- Los *parámetros del modelo*. La persona que formula el modelo decide qué variables son parámetros. La característica distintiva de este tipo de variables es que no son calculadas de las ecuaciones del modelo, sino que se les asigna valor al inicio de la simulación (en la llamada fase de "inicialización del modelo") y éste permanece constante durante toda la simulación. Puesto que los parámetros no deben ser calculados de las ecuaciones del modelo, a efectos de la asignación de la causalidad computacional, se considera que los parámetros son variables conocidas.
- Las *entradas globales* al modelo, es decir, aquellas variables cuyo valor se especifica, independientemente del de las demás variables, para cada instante de la simulación.
- Las variables que aparecen derivadas en el modelo, ya que son consideradas *variables de estado*, es decir, se asume que se calculan mediante la integración numérica de su derivada. El motivo es evitar tener que derivar numéricamente en tiempo de simulación.

Por tanto, toda variable que aparece derivada en el modelo se clasifica como una variable de estado. Con el fin de formular el modelo de manera adecuada para su simulación, se sustituye la derivada de cada variable de estado, allí donde aparezca, por una variable auxiliar (por ejemplo, de nombre igual al de la variable de estado, pero anteponiendo el prefijo "der").

[1] Esta explicación está extraída del texto (Elmqvist 1978)

Estas variables auxiliares se clasifican como desconocidas y se añade al modelo la condición de que cada variable de estado se calcula por integración numérica de su correspondiente variable auxiliar.

Variables desconocidas:

- Las *variables auxiliares* introducidas, de la forma descrita anteriormente, sustituyendo a las derivadas de las variables de estado.
- Las restantes variables del modelo, es decir, aquellas que, no apareciendo derivadas, dependen para su cálculo en el instante de evaluación del valor de otras variables. Estas variables se denominan *variables algebraicas*.

Ejemplo 2.5.1. *Considérese el circuito mostrado en la Figura 2.1. El modelo está compuesto por las ecuaciones siguientes:*

$$u = u_0 \cdot \sin(\omega \cdot t) \quad (2.28)$$
$$i_{R1} = i_{R2} + i_C \quad (2.29)$$
$$u - u_C = R_1 \cdot i_{R1} \quad (2.30)$$
$$C \cdot \frac{du_C}{dt} = i_C \quad (2.31)$$
$$u_C = i_{R2} \cdot R_2 \quad (2.32)$$

La variable u_C aparece derivada, con lo cual es una variable de estado del modelo.

Con el fin de realizar la asignación de la causalidad computacional, se sustituye en el modelo $\frac{du_C}{dt}$ por la variable auxiliar $deru_C$. Realizando esta sustitución ($\frac{du_C}{dt}$ por $deru_C$), se obtiene el modelo siguiente:

$$u = u_0 \cdot \sin(\omega \cdot t) \quad (2.33)$$
$$i_{R1} = i_{R2} + i_C \quad (2.34)$$
$$u - u_C = R_1 \cdot i_{R1} \quad (2.35)$$
$$C \cdot deru_C = i_C \quad (2.36)$$
$$u_C = i_{R2} \cdot R_2 \quad (2.37)$$

A efectos de la asignación de la causalidad computacional, se considera que:

- *La variable de estado, u_C, es conocida.*
- *La derivada de la variable de estado, $deru_C$, es desconocida.*

Al realizar la simulación, u_C se calculará integrando $deru_C$.

Descontando del número total de ecuaciones, el número de variables de estado, se obtiene el número de ecuaciones disponibles para el cálculo de variables algebraicas. Este número determina el número de variables algebraicas del modelo. El resto de las variables deberán ser parámetros. La persona que realiza el modelo deberá decidir, de entre las variables que no son estados, cuáles son las variables algebraicas y cuáles los parámetros.

En este ejemplo sólo hay una variable de estado: u_C. Así pues, debe emplearse una de las ecuaciones del modelo para evaluar su derivada: $deru_C$. Puesto que $deru_C$ sólo interviene en la Ec. (2.36), debe emplearse esta ecuación para calcular $deru_C$.

Las restantes cuatro ecuaciones permiten calcular cuatro variables algebraicas. Una posible elección de las variables algebraicas es la siguiente: u, i_{R1}, i_{R2}, i_C. En consecuencia, las variables desconocidas son las siguientes: u, i_{R1}, i_{R2}, i_C y $deru_C$. El resto de las variables del modelo deberán ser parámetros: u_0, ω, R_1, R_2 y C.

Este modelo tiene sólo 5 ecuaciones y 5 incógnitas, con lo cual la causalidad computacional puede asignarse de manera sencilla (en el siguiente epígrafe se explica un método sistemático para hacerlo). Asignar la causalidad computacional es decidir en qué orden deben evaluarse las ecuaciones y qué incógnita debe evaluarse de cada ecuación, a fin de calcular las incógnitas u, i_{R1}, i_{R2}, i_C y $deru_C$ del conjunto de Ecuaciones (2.33) – (2.37).

A continuación, se señala en cada ecuación del modelo qué incógnita debe calcularse de ella. Para ello, se incluye entre corchetes la incógnita a evaluar de cada ecuación:

$$\begin{aligned}
[u] &= u_0 \cdot \sin(\omega \cdot t) && (2.38)\\
i_{R1} &= i_{R2} + [i_C] && (2.39)\\
u - u_C &= R_1 \cdot [i_{R1}] && (2.40)\\
C \cdot [deru_C] &= i_C && (2.41)\\
u_C &= [i_{R2}] \cdot R_2 && (2.42)
\end{aligned}$$

Ordenando las ecuaciones del modelo, se obtiene:

$$\begin{aligned}
[u] &= u_0 \cdot \sin(\omega \cdot t) && (2.43)\\
u_C &= [i_{R2}] \cdot R_2 && (2.44)\\
u - u_C &= R_1 \cdot [i_{R1}] && (2.45)\\
i_{R1} &= i_{R2} + [i_C] && (2.46)\\
C \cdot [deru_C] &= i_C && (2.47)
\end{aligned}$$

Finalmente, despejando en cada ecuación la incógnita que debe ser evaluada de ella, se obtiene el modelo ordenado y resuelto:

$$\begin{aligned}
[u] &= u_0 \cdot \sin(\omega \cdot t) && (2.48)\\
[i_{R2}] &= \frac{u_C}{R_2} && (2.49)\\
[i_{R1}] &= \frac{u - u_C}{R_1} \cdot && (2.50)\\
[i_C] &= i_{R1} - i_{R2} && (2.51)\\
[deru_C] &= \frac{i_C}{C} && (2.52)
\end{aligned}$$

Además, debe realizarse el cálculo de la variable de estado mediante la integración de su derivada:

$$\frac{d[u_C]}{dt} = deru_C \qquad (2.53)$$

Obsérvese que, en general, la clasificación de las variables en algebraicas y parámetros puede realizarse de varias maneras, en función de cuál sea el experimento que desee realizarse sobre el modelo.

Para comprobar si una determinada clasificación es válida, es preciso realizar la asignación de la causalidad computacional, y comprobar que efectivamente las variables algebraicas seleccionadas pueden evaluarse de las ecuaciones del modelo.

Una selección de las variables algebraicas alternativa a la anterior consistiría en escoger R_1 y R_2 como variables algebraicas, en lugar de i_{R1} e i_{R2}. En este caso, i_{R1} e i_{R2} serían parámetros. □

2.6. Asignación de la causalidad computacional

En esta sección se describe un procedimiento sistemático para asignar la causalidad computacional de un modelo. Consta de dos pasos. En primero consiste en comprobar que el modelo no es estructuralmente singular. El segundo paso es la asignación en sí de la causalidad.

Singularidad estructural del modelo

Antes de realizar la partición se comprueba la no *singularidad estructural* del modelo. Es decir, se comprueba:

- que el número de ecuaciones y de incógnitas (obtenido siguiendo el criterio anterior de clasificación de las variables en conocidas y desconocidas) es el mismo; y
- que cada incógnita puede emparejarse con una ecuación en que aparezca y con la cual no se haya emparejado ya otra incógnita.

Si alguna de estas dos condiciones no se verifica, se dice que el modelo es *singular* y es necesario reformularlo para poder simularlo.

Si el modelo no es singular, se procede a asignar la casualidad computacional.

Ejemplo 2.6.1. *Considere un modelo con tres variables: x, y, z. El modelo está compuesto por tres ecuaciones, en cada una de las cuales intervienen las variables siguientes:*

$$f_1(x,y,z) = 0 \tag{2.54}$$
$$f_2\left(x, \frac{dx}{dt}, y\right) = 0 \tag{2.55}$$
$$f_3(x,z) = 0 \tag{2.56}$$

Sustituyendo la derivada de la variable x por la variable auxiliar ($derx$), se obtiene un modelo compuesto por las tres ecuaciones siguientes:

$$f_1(x,y,z) = 0 \tag{2.57}$$
$$f_2(x,derx,y) = 0 \tag{2.58}$$
$$f_3(x,z) = 0 \tag{2.59}$$

Dicho modelo tiene tres incógnitas: $derx$, y, z. La variable x, por aparecer derivada, se supone que es una variable de estado: se calcula integrando $derx$, y no de las ecuaciones del modelo.

Se comprueba que este modelo no es singular, ya que cada ecuación puede asociarse con una incógnita que interviene en ella y que no se ha asociado con ninguna otra ecuación (obsérvese que, en general, esta asociación puede no ser única):

$$f_1(x,y,z) = 0 \quad \rightarrow \quad y \tag{2.60}$$
$$f_2(x,derx,y) = 0 \quad \rightarrow \quad derx \tag{2.61}$$
$$f_3(x,z) = 0 \quad \rightarrow \quad z \tag{2.62}$$

□

Ejemplo 2.6.2. *Considere un modelo con tres variables: x, y, z. El modelo está compuesto por tres ecuaciones, en cada una de las cuales intervienen las variables siguientes:*

$$f_1(z) = 0 \tag{2.63}$$
$$f_2\left(\frac{dx}{dt}, y\right) = 0 \tag{2.64}$$
$$f_3(x) = 0 \tag{2.65}$$

Dicho modelo tiene tres incógnitas: $derx$, y, z. Observe que la tercera ecuación ($f_3(x) = 0$) no contiene ninguna incógnita, mientras que la segunda ecuación contiene dos incógnitas que no aparecen en ninguna otra ecuación del modelo ($f_2(derx, y) = 0$). En consecuencia, el modelo es singular y por tanto es necesario reformularlo. □

Asignación de la causalidad computacional

Una vez se ha comprobado que el modelo no es *singular*, se realiza la asignación de causalidad computacional siguiendo las tres reglas siguientes:

1. Las variables que aparecen derivadas se consideran variables de estado y se suponen conocidas, ya que se calculan por integración a partir de sus derivadas. Las derivadas de las variables de estado son desconocidas y deben calcularse de las ecuaciones en que aparezcan.
2. Las ecuaciones que poseen una única incógnita deben emplearse para calcularla.
3. Aquellas variables que aparecen en una única ecuación deben ser calculadas de ella.

Aplicando las tres reglas anteriores a sistemas no singulares pueden darse dos situaciones:

1. Se obtiene una solución que permite calcular todas las incógnitas usando para ello todas las ecuaciones. Esto significa que las variables pueden ser resueltas, una tras otra, en secuencia. En este caso, el algoritmo proporciona una ordenación de las ecuaciones tal que en cada ecuación hay una y sólo una incógnita que no haya sido previamente calculada. En ocasiones será posible despejar la incógnita de la ecuación. En otros casos, la incógnita aparecerá de forma implícita y deberán emplearse métodos numéricos para evaluarla.
2. Se llega a un punto en que todas las ecuaciones tienen al menos dos incógnitas y todas las incógnitas aparecen al menos en dos ecuaciones. Corresponde al caso en que hay sistema de ecuaciones. Si en este sistema las incógnitas intervienen linealmente, será posible despejarlas resolviendo el sistema simbólicamente. Si al menos una de las incógnitas interviene de forma no lineal, deberán emplearse métodos numéricos para evaluar las incógnitas. El algoritmo de partición asegura que la dimensión de los sistemas de ecuaciones obtenidos es mínima.

Ejemplo 2.6.3. *A continuación va a realizarse la partición del modelo descrito en el Ejemplo 2.6.1. Indicando únicamente las incógnitas que intervienen en cada ecuación se obtiene:*

$$f_1(y,z) = 0 \tag{2.66}$$
$$f_2(derx,y) = 0 \tag{2.67}$$
$$f_3(z) = 0 \tag{2.68}$$

La variable z es la única incógnita que aparece en la ecuación f_3, por tanto debe emplearse esta ecuación para calcular z. Las ecuaciones que quedan por emplear, y las incógnitas que quedan por evaluar son:

$$f_1(y) = 0 \tag{2.69}$$
$$f_2(derx,y) = 0 \tag{2.70}$$

Debe emplearse f_1 para calcular y. Seguidamente, debe emplearse f_2 para calcular $derx$. Las ecuaciones ordenadas, con la causalidad computacional señalada, son las siguientes:

$$f_3(x,[z]) = 0 \tag{2.71}$$
$$f_1(x,[y],z) = 0 \tag{2.72}$$
$$f_2(x,[derx],y) = 0 \tag{2.73}$$

□

Parte II

Easy Java Simulations

Tema 3

Fundamentos de Ejs

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Quién y con qué finalidad ha desarrollado Ejs.
- Cuál es la metodología de Ejs para la creación de laboratorios virtuales.

3.1. ¿Qué es Easy Java Simulations?

Easy Java Simulations (abreviado: Ejs) es un entorno de simulación que ha sido diseñado y desarrollado por el profesor Francisco Esquembre[1]. Según indica el Prof. Esquembre, Ejs ha sido especialmente ideado para el desarrollo de aplicaciones docentes, permitiendo a profesores y alumnos crear de forma sencilla sus propios laboratorios virtuales, sin que para ello requieran de conocimientos avanzados de programación.

El entorno de simulación Ejs, así como su documentación y algunos casos de estudio, puede ser descargado gratuitamente del sitio web http://fem.um.es/Ejs. Para su comodidad, todo este material ha sido incluido en el CD que se le ha entregado como material de este curso. En el Tema 4 se explicará cómo realizar la instalación de Ejs.

Condiciones de uso de Ejs

Las condiciones de uso de Ejs y el copyright de los ejemplos de simulaciones disponibles en el sitio web http://fem.um.es/Ejs, son las indicadas en dicho sitio web (en diciembre de 2004), y que a continuación se reproducen textualmente:

> Easy Java Simulations es propiedad exclusiva de su autor, Francisco Esquembre, que lo distribuye bajo licencia Open Source.
>
> Los ejemplos de simulaciones contenidos en este servidor son propiedad de sus autores.
>
> **Copyright del Software Easy Java Simulations**

[1] Prof. Dr. Francisco Esquembre, Dpto. de Matemáticas, Universidad de Murcia, Campus de Espinardo, 30071 Murcia (España). E-mail: fem@um.es

3.2. Paradigma modelo-vista-control

La metodología para la creación de laboratorios virtuales de Ejs está basada en una simplificación del *paradigma "modelo-vista-control"*. Este paradigma[2] establece que el laboratorio virtual se compone de las tres partes siguientes:

1. El **modelo**: describe los fenómenos bajo estudio. Está compuesto por un conjunto de variables y por las relaciones entre estas variables.
2. El **control**: define las acciones que el usuario puede realizar sobre la simulación.
3. La **vista**: representación gráfica de los aspectos más relevantes del fenómeno simulado.

Estas tres partes están interrelacionadas entre sí:

- El modelo afecta a la vista, ya que debe mostrarse al usuario cuál es la evolución del valor de las variables del modelo.
- El control afecta al modelo, ya que las acciones ejercidas por el usuario pueden modificar el valor de las variables del modelo.
- La vista afecta al modelo y al control, ya que la interfaz gráfica puede contener elementos que permitan al usuario modificar el valor de las variables o realizar ciertas acciones.

Simplificación del paradigma modelo-vista-control realizada por Ejs

Ejs se basa en una simplificación del paradigma modelo-vista-control, suprimiendo la parte del control como tal, e integrando sus funciones tanto en la vista como en el modelo. Esta simplificación se basa en el hecho de que el usuario puede usar la interfaz gráfica del laboratorio virtual (es decir, la vista) para interaccionar con la simulación, empleando para ello el ratón, el teclado, etc.

Así pues, en un laboratorio virtual programado usando Ejs, el usuario interacciona con el modelo a través de la vista. Por tanto, al programar el modelo es preciso especificar de qué forma las acciones realizadas por el usuario durante la simulación sobre los componentes gráficos o los controles de la vista afectan al valor de las variables del modelo.

Las propiedades de los elementos gráficos de la vista (posición, tamaño, etc.) pueden asociarse con las variables del modelo, dando lugar a un flujo de información bidireccional entre la vista y el modelo. Cualquier cambio en el valor de una variable del modelo es automáticamente representado en la vista. Recíprocamente, cualquier interacción del usuario con la vista de laboratorio virtual, modifica el valor de la correspondiente variable del modelo.

[2]Esta discusión está extraída de la Sección 3.1 del documento EjsManual3.1.pdf, que puede encontrar en el CD del curso.

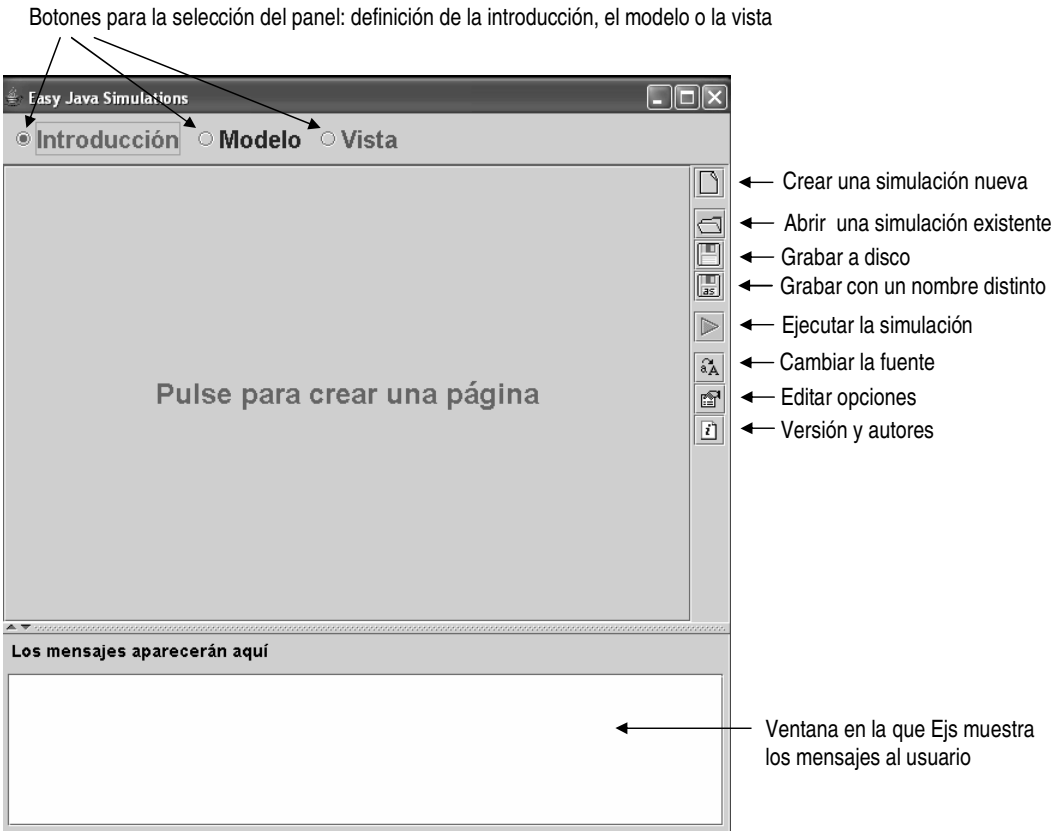


Figura 3.1: Interfaz de usuario de Ejs.

Además del modelo y la vista, Ejs permite incluir en el laboratorio virtual páginas HTML que realicen las funciones de documentación, informando acerca de la finalidad del laboratorio, sus instrucciones de uso, recomendaciones pedagógicas, etc. Este conjunto de páginas recibe el nombre de *Introducción*.

Resumiendo lo anterior, la definición de un laboratorio virtual mediante Ejs se estructura en las siguientes tres partes:

- **Introducción**: páginas html que incluyen los contenidos educativos relacionados con el laboratorio virtual.
- **Modelo**: modelo dinámico cuya simulación interactiva es la base del laboratorio virtual.
- **Vista**: interfaz entre el usuario y el modelo. La vista del laboratorio virtual tiene dos funciones. Por una parte, proporciona una representación visual del comportamiento dinámico del modelo. Por otra parte, proporciona los mecanismos para que el usuario pueda interaccionar con el modelo durante la simulación.

En la Figura 3.1 se muestra la interfaz de usuario de Ejs. Se trata de la pantalla que aparece al arrancar Ejs. Como puede verse, en la parte superior hay tres botones: *Introducción*, *Modelo*, *Vista*. Mediante estos botones puede seleccionarse el panel para la definición de las páginas de la introducción, el panel para la definición del modelo o el panel para la definición de la vista.

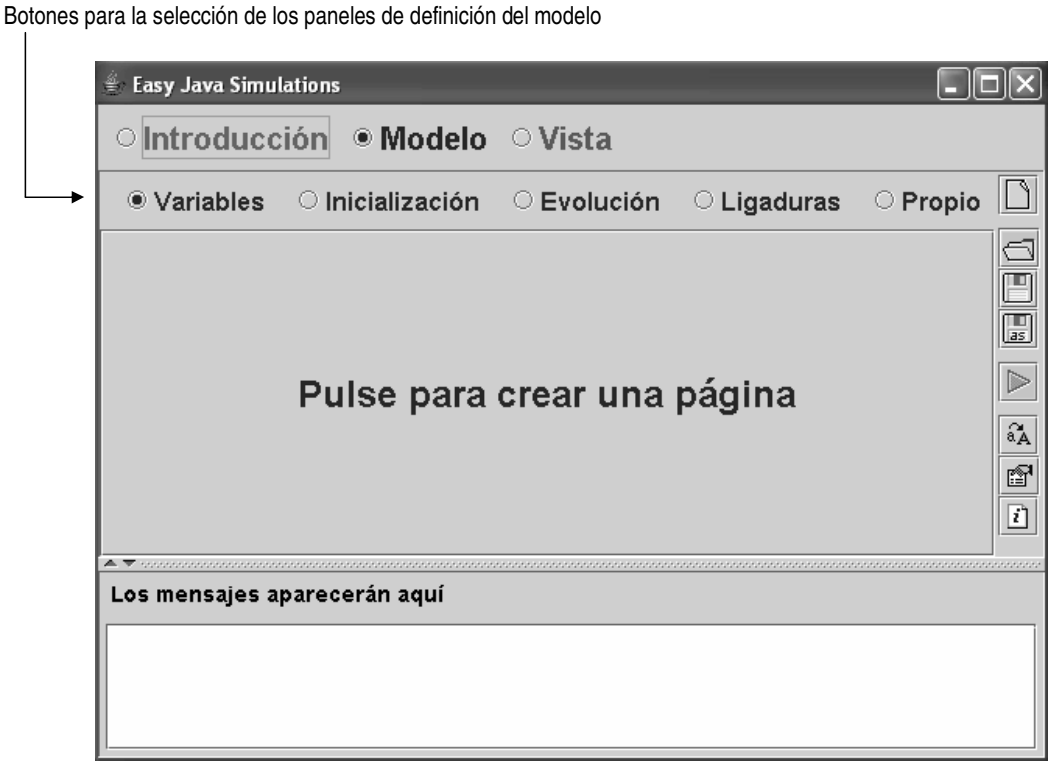


Figura 3.2: Panel para la definición del modelo.

3.3. Definición del modelo

Ejs proporciona un procedimiento sencillo para la definición del *modelo*: un conjunto de paneles que el usuario debe completar para especificar las variables y los algoritmos que componen el modelo.

En la Figura 3.2 se muestra el panel para la definición del modelo, dentro del cual, a su vez, puede accederse a cinco paneles: Variables, Inicialización, Evolución, Ligaduras y Propio. La finalidad de cada uno de estos paneles se explicará en el Tema 5.

Además de este método para definir el modelo, Ejs soporta la opción de describir y simular el modelo usando Matlab/Simulink[3]:

- En cualquier punto de la definición del modelo en Ejs pueden incluirse código Matlab, así como llamadas a cualquier función de Matlab, tanto las funciones propias de Matlab como cualquier función definida en un fichero M.
- El modelo puede ser descrito parcial o completamente usando bloques de Simulink.

El modelado y la simulación combinando el uso de Ejs y Matlab/Simulink excede los objetivos planteados en este curso.

3.4. Definición de la vista

Ejs proporciona un conjunto de elementos predefinidos que pueden usarse de manera sencilla para componer la *vista*.

[3]Puede encontrar información acerca de Matlab y Simulink en el sitio web http://www.mathworks.com/

Figura 3.3: Panel para la definición de la vista.

En la Figura 3.3 se muestra el panel para la definición de la vista. Ésta se realiza empleando las dos ventanas siguientes:

- La ventana situada en la parte derecha del panel, que tiene el letrero *Elementos para la vista*. Los elementos se encuentran clasificados en tres grupos: *Contenedores*, *Básicos* y *Dibujo*. Estos elementos son clases de componentes gráficos que Ejs tiene predefinidas y que el usuario va a emplear para componer la vista.
- La ventana que está situada en la parte izquierda del panel, y tiene el letrero *Árbol de elementos*. En esta ventana el usuario va componiendo la vista, usando para ello los elementos de la ventana *Elementos para la vista*.

Los elementos gráficos proporcionados por Ejs están programados usando la librería *Swing* de Java y las herramientas creadas en el proyecto *Open Source Physics*. Puede encontrar información acerca de este proyecto en el sitio web http://www.opensourcephysics.org/

Como ilustración de qué tipo de laboratorios virtuales puede programarse empleando Ejs, y en particular, qué capacidades de visualización e interactividad tienen, se muestra a continuación un laboratorio virtual que ha sido programado para la enseñanza del control automático.

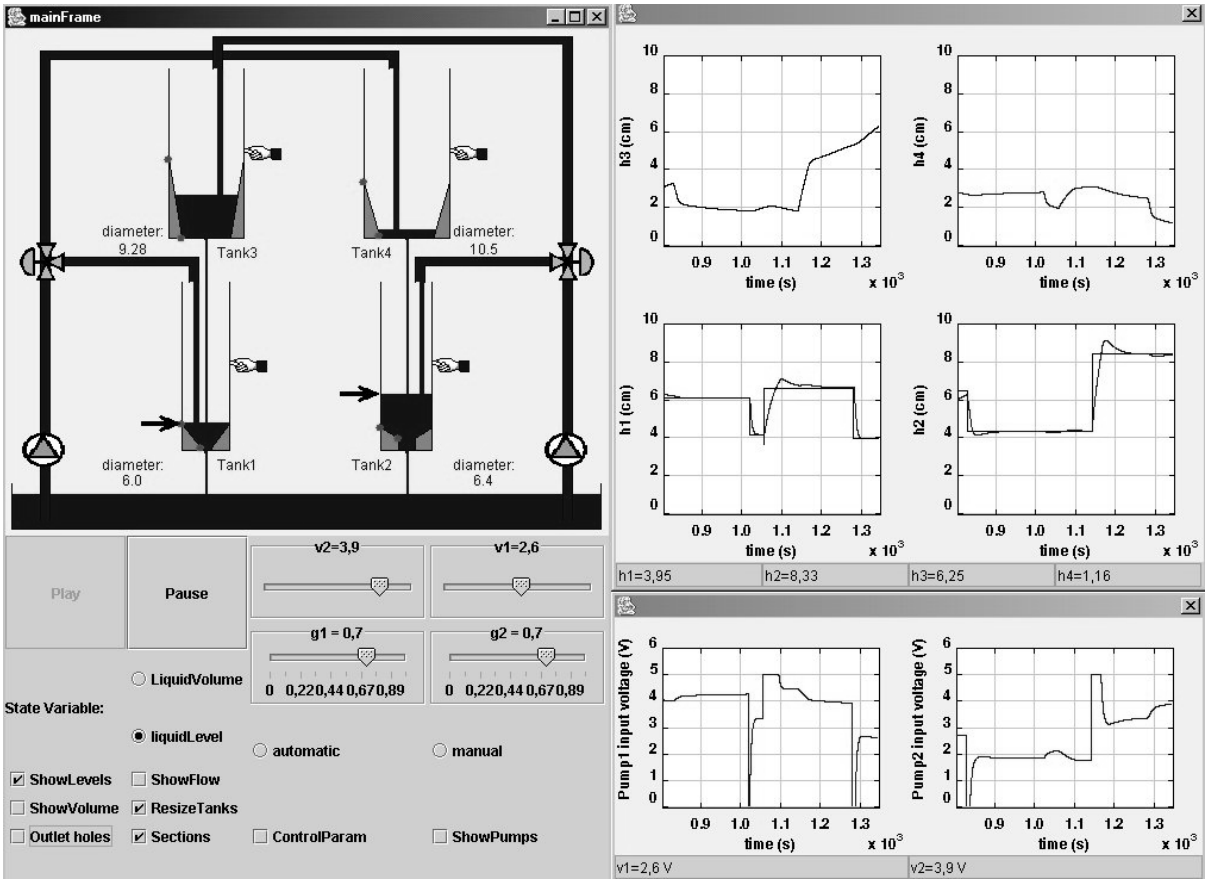


Figura 3.4: Vista de un laboratorio virtual programado con Ejs.

Ejemplo 3.4.1. *En la Figura 3.4 se muestra la vista de un laboratorio virtual programado con Ejs. El modelo consta de una planta*[4]*, compuesta por cuatro depósitos de líquido, y de su sistema de control. Este laboratorio virtual se emplea para ilustrar diferentes aspectos de la teoría del control automático. El objetivo es controlar el nivel de los dos depósitos inferiores mediante dos bombas. Para ello, pueden emplearse dos estrategias de control - manual y automático (PID descentralizado)- entre las cuales el alumno puede conmutar en tiempo de simulación.*

Dos bombas hacen circular el líquido desde la cubeta inferior a los tanques. En el diagrama de la planta, mostrado en la parte superior izquierda de la Figura 3.4, cada bomba se representa mediante un triángulo inscrito en un círculo.

Una válvula de doble via distribuye el causal suministrado por cada una de las bombas. La primera de estas válvulas determina qué proporción del líquido suministrado por la bomba va al Tanque 1 y qué proporción (la restante) va al Tanque 4. Igualmente, una segunda válvula determina cómo se reparte el flujo entre los Tanques 2 y 3.

En la base de los Tanques 3 y 4 existen sendos agujeros, a través de los cuales se vierte líquido a los Tanques 1 y 2 respectivamente. Igualmente, a través de los agujeros situados en

[4]Puede encontrarse información adicional acerca de esta planta en (Johansson 2000). La programación y el manejo del laboratorio virtual se explican en (Dormido & Esquembre 2003, Martin, Urquia, Sanchez, Dormido, Esquembre, Guzman & Berenguel 2004).

la base de los Tanques 1 y 2 se vierte líquido a la cubeta inferior. Estos flujos son modelados usando la relación de Bernoulli[5].

La vista del laboratorio virtual, mostrada en la Figura 3.4, consta de:

- *La ventana principal, situada en la parte izquierda, que contiene la representación esquemática del proceso (en la parte superior) y los botones de control (en la parte inferior). Ambos permiten al alumno experimentar con el modelo. En la representación esquemática del sistema, pueden modificarse (haciendo clic y arrastrando el ratón) los niveles del líquido en los tanques, la forma y la sección de los tanques y los valores de consigna para el nivel de líquido. Los deslizadores y botones situados en la parte inferior permiten modificar interactivamente el valor de los parámetros del controlador PID y el valor de la sección de los agujeros realizados en la base de los tanques. Asimismo, los botones permiten seleccionar las variables de estado del modelo (el volumen o el nivel del líquido), abrir y cerrar ventanas gráficas, etc.*
- *En la parte izquierda, se muestran seis ventanas en las cuales se representa gráficamente la evolución en el tiempo de las variables relevantes del sistema: altura del líquido en cada tanque y voltaje aplicado a cada una de las bombas por el sistema de control.*

□

3.5. Ejecución y distribución del laboratorio virtual

Una vez que el usuario ha definido el modelo, la vista y la introducción del laboratorio virtual, Ejs genera automáticamente el código Java del programa, lo compila, empaqueta los ficheros resultantes en un fichero comprimido, y genera páginas html que contienen la introducción y la simulación como un applet.

Entonces, existen tres posibles formas de ejecutar el laboratorio virtual:

- Como un *applet*, abriendo con un navegador web (Internet Explorer, Netscape, etc.) el documento html generado por Ejs para el laboratorio. Esta opción permite publicar el laboratorio virtual en Internet.

  Una vez han sido generadas por Ejs, puede usted (si lo desea) editar las páginas html del laboratorio virtual y añadirles otros contenidos. Deberá usar para ello un editor de páginas web.

  Es importante tener en cuenta que, por motivos de seguridad, los applets de Java no pueden escribir datos en el disco (pero si pueden leer datos del disco o de Internet). Por ello, si ha programado la simulación de modo que escriba datos en el disco, no podrá ejecutarla como un applet, sino que deberá usar cualquiera de las dos formas de ejecución siguientes.
- Ejecución desde el entorno Ejs.
- Ejecución como una aplicación Java independiente.

[5]El flujo de líquido (F) se relaciona con la sección recta del orificio (a), el nivel del líquido en el tanque (h) y la aceleración gravitatoria (g) mediante la relación de Bernoulli: $F = a \cdot \sqrt{2 \cdot g \cdot h}$.

Tema 4

Instalación y arranque de Ejs

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Cómo instalar Ejs.
- Cómo ejecutar un laboratorio virtual ya existente.
- Cómo acceder a los ejemplos que incluye Ejs.

4.1. Requisitos para la instalación de Ejs

Ejs ha sido probado en las plataformas siguientes: Windows 95, 98, ME, 2000 y XP; Apple MacIntosh bajo Mac OS X; y PC-compatibles bajo Linux.

Ejs es un programa escrito en lenguaje Java, que compila programas escritos en Java. Por ello, Ejs requiere para su funcionamiento que previamente se haya instalado el *Kit de Desarrollo de Software de Java 2 (Java 2 SDK)*.

Dependiendo de la versión de Ejs, es aconsejable instalar una versión del Java 2 SDK u otra. En el caso de Ejs 3.4, se recomienda usar la versión 1.5.0_09 del kit de desarrollo de Java 2.

Una vez haya instalado Java 2 SDK, deberá instalar Ejs. La instalación bajo Windows se explica en la Sección 4.2. La instalación bajo Linux y Mac OS X, que no explica en este texto, está explicada detalladamente en el sitio web de Ejs (http://fem.um.es/Ejs/Ejs_es/index.html) y en el manual de Ejs que se incluye en el CD-ROM del curso.

La documentación de Ejs consiste en ficheros con formato pdf. Si no dispone de un visualizador de pdf, puede descargar uno gratuitamente del sitio web http://www.adobe.com/.

4.2. Instalación de Ejs para Windows

Instalación del Java SDK

Ejecute el fichero de instalación de Java SDK, jdk-1_5_0_09-windows-i586-p.exe, y siga las instrucciones del mismo. Este fichero se encuentra en el CD-ROM del curso.

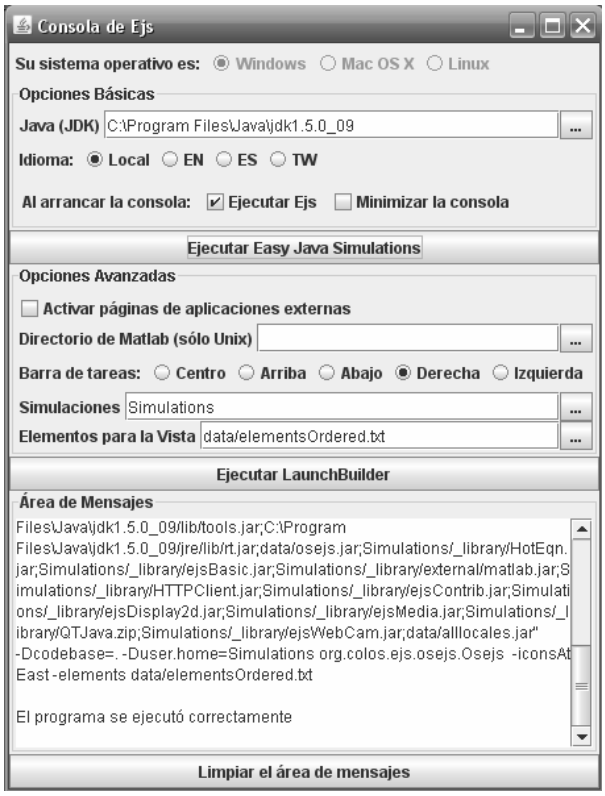


Figura 4.1: Consola de Ejs versión 3.4.

En realidad, la única elección importante que debe hacer es el directorio donde debe instalarse el software. La recomendación que hacemos es que acepte el directorio que la instalación proporciona por defecto.

Instalación de Easy Java Simulations

Mueva el directorio Ejs desde el CD-ROM al disco duro de su ordenador. Es recomendable que ni dicho de los directorios del path contenga espacios en blanco en su nombre.

4.3. El directorio de instalación de Ejs

El directorio Ejs contiene:

- El fichero **EjsConsole.jar**. Debe ejecutarlo para arrancar la **consola de Ejs** (véase la Figura 4.1). Desde dicha consola podrá especificar en qué directorio se encuentra instalado Java2SDK, cuál es el directorio de trabajo de Ejs, podrá arrancar Ejs, seleccionar el idioma de la interfaz gráfica de usuario de Ejs,etc. Asimismo, en la parte inferior de la consola hay una pantalla en la cual Ejs muestra los mensajes de error generados al compilar y ejecutar los laboratorios virtuales.
- El directorio **data**. Contiene el código de la aplicación Ejs, con lo cual usted no debe modificar el contenido de este directorio.
- El directorio **Simulations**. Es el directorio de trabajo por defecto: donde arranca la ejecución de Ejs, donde Ejs espera encontrar la definición de los laboratorios virtuales

y donde Ejs almacena los resultados de las simulaciones. En la Sección 4.4 se explica cómo escoger otro directorio de trabajo.

A su vez, el directorio Simulations contiene los tres subdirectorios siguientes:

- El directorio **_examples**. Contiene ejemplos de uso de Ejs, que usted puede inspeccionar y modificar si lo desea.
- El directorio **_library**. Contiene ficheros necesarios para el funcionamiento de Ejs, por tanto, usted tampoco debe modificar el contenido de este directorio.
- El directorio **laboratoriosTexto**, que contiene los laboratorios virtuales descritos en el presente texto.

4.4. Selección del directorio de trabajo

Ejs toma por defecto el directorio **Simulations** como directorio de trabajo. Si desea cambiar el directorio de trabajo, puede hacerlo desde la consola de Ejs. Para ello, debe además copiar el directorio completo **_library** en su directorio de trabajo.

4.5. Arranque de Ejs y ejecución de un laboratorio virtual

Para arrancar el entorno de simulación Ejs debe arrancar la consola de Ejs, ejecutando el fichero EjsConsole.jar. Pulse entonces el botón con la etiqueta "Ejecutar Easy Java Simulations" que hay en la consola (vea la Figura 4.1).

Con ello, se abrirá la ventana mostrada en las Figura 4.2, que es la interfaz de usuario del entorno de simulación Ejs, en la cual se definen las tres partes que componen el laboratorio virtual: la introducción, el modelo y la vista.

Para comprobar que la instalación se ha realizado correctamente, abra alguno de los laboratorios contenidos en el directorio *Simulations_examples*. Para ello:

1. Pulse el segundo botón de la regleta de botones situada en la parte derecha de la interfaz de Ejs (ver la Figura 4.2). Dicho botón representa una carpeta abierta, y si se sitúa sobre él aparece el mensaje: "Abrir una simulación existente". Al pulsar dicho botón se muestra una ventana, en la que debe escoger qué laboratorio virtual desea abrir.
2. Entre en el directorio *_examples*.
3. Los ficheros en los que Ejs guarda la definición del laboratorio virtual tienen extensión xml. Seleccione el fichero *EarthSunAndMoon.xml* y ábralo (vea la Figura 4.3). Al abrir el fichero *EarthSunAndMoon.xml* se abren las dos ventanas mostradas en la Figura 4.4. En la ventana situada a la derecha puede ver la introducción del laboratorio. Si pulsa los botones *Modelo* o *Vista* accederá a la definición del modelo y de la vista respectivamente.
4. Para iniciar la ejecución de la simulación, debe pulsar el botón que tiene un triángulo de color verde dibujado. Si se sitúa con el ratón sobre él, aparece el mensaje "Ejecutar la simulación" (vea la Figura 4.2). Púlselo para arrancar la simulación

Al arrancar la simulación, Ejs genera varios ficheros en el directorio de trabajo:

- *earthSunAndMoon.jar*, en el cual Ejs ha empaquetado todos los ficheros Java necesarios para la ejecución del laboratorio virtual.
- *EarthSunAndMoon.html*, documento html que recoge la introducción del laboratorio y la simulación. Esta última como un applet. Además, Ejs genera otros documentos html a los que se enlaza desde EarthSunAndMoon.html. Estos son: EarthSunAndMoon_Activities.html, EarthSunAndMoon_Author.html, etc.
- *EarthSunAndMoon.bat*, fichero por lotes que arranca la ejecución del laboratorio virtual como una aplicación Java independiente. Por defecto, Ejs no genera este fichero. En la Sección 4.6 se explicará cómo configurar Ejs para que lo haga.
- Posiblemente Ejs almacene algún fichero en el directorio *_library*. Recuerde que, si cambia el directorio de trabajo, debe copiar en el nuevo directorio de trabajo el directorio *_library* con todo su contenido.

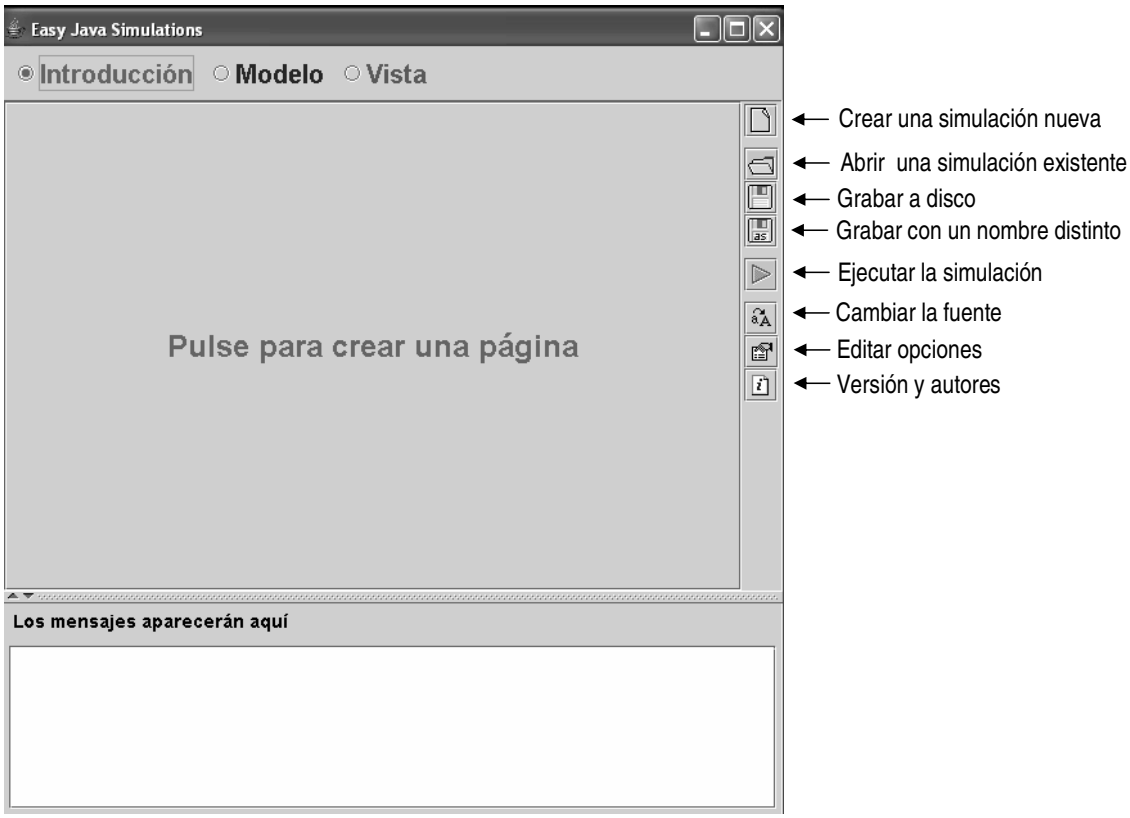


Figura 4.2: Interfaz de usuario de Ejs.

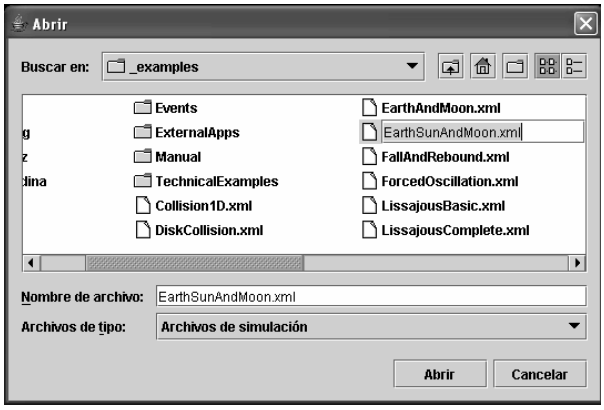


Figura 4.3: Selección del fichero *EarthSunAndMoon.xml*, que se encuentra dentro del directorio *_examples*.

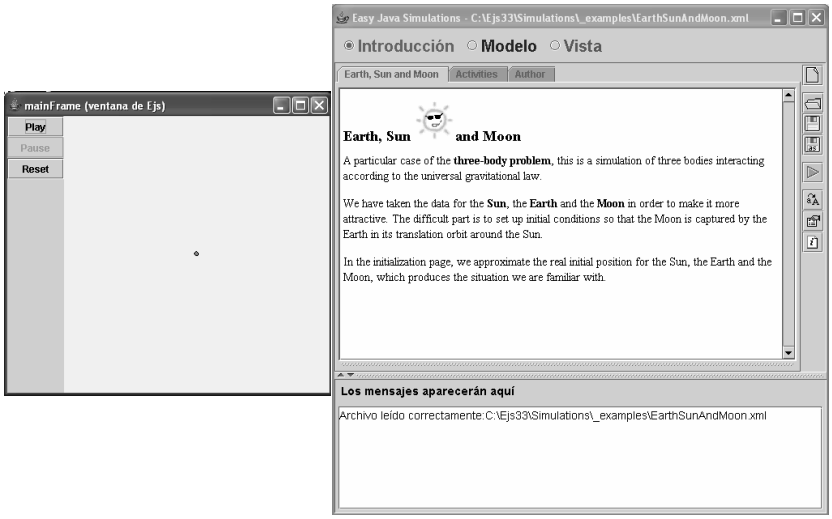


Figura 4.4: Ventanas del laboratorio virtual EarthSunAndMoon.xml.

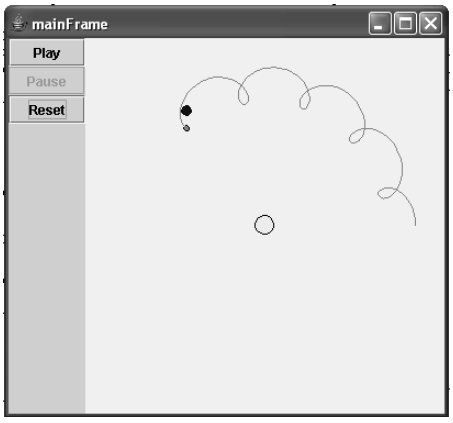


Figura 4.5: Vista del laboratorio EarthSunAndMoon.xml.

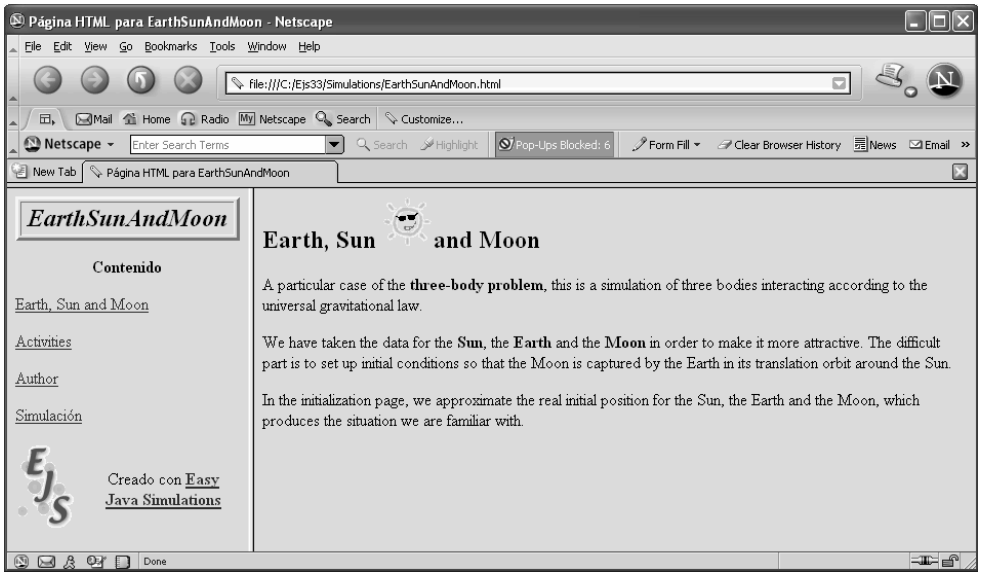


Figura 4.6: Página web del laboratorio virtual: EarthSunAndMoon.html.

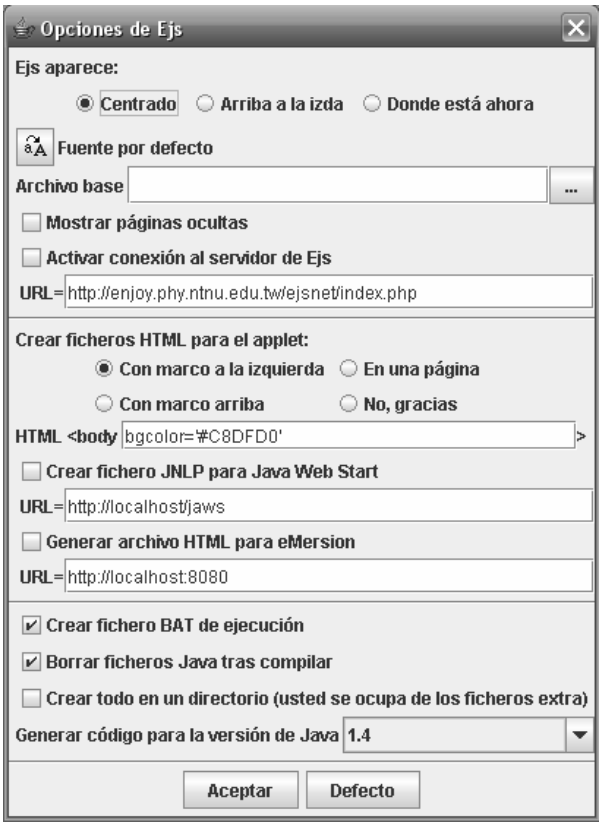


Figura 4.7: Ventana de configuración de Ejs versión 3.4.

La ejecución del laboratorio virtual hace que se abra una nueva ventana, en la cual se muestra la evolución de la tierra y de la luna en sus respectivas órbitas (vea la Figura 4.5).

Si detiene la simulación pulsando el botón “Pause”, observará que situando el ratón sobre cualquiera de los tres objetos celestes (el sol, la tierra o la luna), pulsando y arrastrando, podrá cambiar su posición. Pulsando el botón “Play”, reinicia la simulación partiendo de las nuevas posiciones.

Si lo desea, puede publicar el laboratorio virtual en Internet. Como se ha indicado anteriormente, Ejs ha creado en el directorio de trabajo los ficheros necesarios para ello. Abra el fichero *EarthSunAndMoon.html* con un navegador web, y observará la página web mostrada en la Figura 4.6.

4.6. Opciones de configuración de Ejs

Para asignar valor a las opciones de configuración de Ejs, debe pulsar el botón “Editar opciones”, que tiene dibujada una mano que sujeta un papel. Se trata del penúltimo botón de la regleta lateral izquierda de botones de Ejs (vea la Figura 4.2).

Pulsando dicho botón se abre la ventana mostrada en la Figura 4.7. En ella el usuario puede indicar sus preferencias acerca de:

- La posición en la que aparece la ventana principal de Ejs cuando se arranca la aplicación.
- El estilo, tamaño, etc. de la fuente de caracteres.

- La creación o no de determinados ficheros, entre ellos el fichero .bat al que se hacía referencia en la Sección 4.5.
- Eliminar o no el fichero fuente de Java que contiene el código para la ejecución del laboratorio virtual. Por defecto, Ejs elimina este fichero una vez lo ha compilado y ha generado el fichero .jar. En este fichero con extensión .jar están empaquetados y comprimidos los ficheros obtenidos de la compilación del fichero fuente .java.
- Mostrar o no las páginas “ocultas” del laboratorio.

Tema 5

Conceptos básicos para la descripción del modelo

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Cuáles son los componentes de un modelo en Ejs.
- Cuál es el algoritmo de simulación de Ejs.
- Cómo se declaran variables y se inicializan en Ejs.
- Cuál es la utilidad de los paneles Evolución y Ligaduras.
- Cómo definir métodos propios.
- Cuáles son los algoritmos para la integración de ODE que soporta Ejs.

En este tema se explica cuál es el algoritmo para la simulación que emplea Ejs. Su comprensión es esencial para poder describir correctamente los modelos usando Ejs.

Las explicaciones se fundamentan en los conceptos expuestos en el Tema 2, apareciendo dos elementos adicionales:

- Ejs está diseñado para permitir la interactividad del usuario sobre el modelo durante la simulación.
- La descripción de cómo debe Ejs realizar el cálculo de los parámetros y de las variables algebraicas debe realizarse mediante fragmentos de código escritos empleando el lenguaje de programación Java.

A continuación, se explica detalladamente todo ello.

5.1. Componentes del modelo escrito en Ejs

Para definir un modelo interactivo en Ejs, es preciso proporcionar la información siguiente:

- Declarar las variables que intervienen en el modelo.
- Describir los algoritmos necesarios para calcular el valor de las variables:
  - En el instante inicial de la simulación.
  - En función del tiempo.
  - Cuando el usuario realiza sobre la vista del laboratorio virtual alguna acción interactiva.

Ejs proporciona un procedimiento sencillo para introducir esta información. En concreto, la definición del modelo en Ejs se compone de las partes siguientes:

- La inicialización de las variables realizada al declararlas. A tal fin, la expresión cuyo valor debe asignarse a la variable debe escribirse en la columna *Valor* del panel *Variables*, en la fila correspondiente a la declaración de la variable.
- Los algoritmos para la inicialización de las variables, que se definen en el panel *Inicialización*.
- Los algoritmos escritos en el panel *Evolución*.
- Los algoritmos escritos en el panel *Ligaduras*.
- Los métodos definidos en el panel *Propio*, que pueden ser invocados desde cualquier punto de la descripción del modelo o de la vista. Se emplean típicamente para describir las acciones interactivas del usuario.

En este tema se explica cómo debe introducirse esta información en los cinco paneles (*Variables*, *Inicialización*, *Evolución*, *Ligaduras* y *Propio*) que componen el panel *Modelo* de Ejs (vea la Figura 3.2).

5.2. Descripción algorítmica del modelo

Como se ha indicado anteriormente, el modelo se describe en Ejs mediante fragmentos de código en lenguaje Java, los cuales deben ser escritos en los diferentes paneles que proporciona Ejs para la descripción del modelo: *Inicialización*, *Evolución*, *Ligaduras* y *Propio*.

Esto implica que el modelo matemático debe ser manipulado por el usuario, previamente a escribirlo en los paneles de Ejs, con el fin de formularlo como una secuencia ordenada de asignaciones. Para ello, es conveniente emplear las técnicas para la asignación de la causalidad computacional descritas en el Tema 2.

Las ecuaciones del modelo deben reescribirse como asignaciones, de la forma siguiente:

$$\text{variable} = \text{expresión};\tag{5.1}$$

Ejs realiza la ejecución de una asignación como la mostrada en (5.1) de la forma siguiente:

1. Se evalúa la expresión del lado derecho de la igualdad, empleando para ello el valor que en ese punto de la ejecución tengan las variables que intervienen en dicha expresión.
2. Se asigna el resultado obtenido a la variable situada al lazo izquierdo de la igualdad.

Además de asignaciones, para la descripción del modelo pueden usarse las cláusulas que proporciona Java para el control del flujo del programa. Por ejemplo, en la Figura 5.5 se muestra un ejemplo de uso de la cláusula *for*.

Al uso conjunto de estas cláusulas y de asignaciones lo denominaremos *algoritmo*.

5.3. El algoritmo de simulación de Ejs

Para poder entender cómo estructurar la definición del modelo en los diferentes paneles, es preciso previamente comprender el *algoritmo de simulación* de Ejs.

En este contexto, se entiende por *algoritmo de simulación* de Ejs el orden en el que Ejs ejecuta los diferentes paneles y las diferentes ventanas dentro de cada panel.

El orden es el siguiente:

1. Ejs ejecuta los diferentes paneles que componen la descripción del modelo en el orden mostrado en la Figura 5.1.

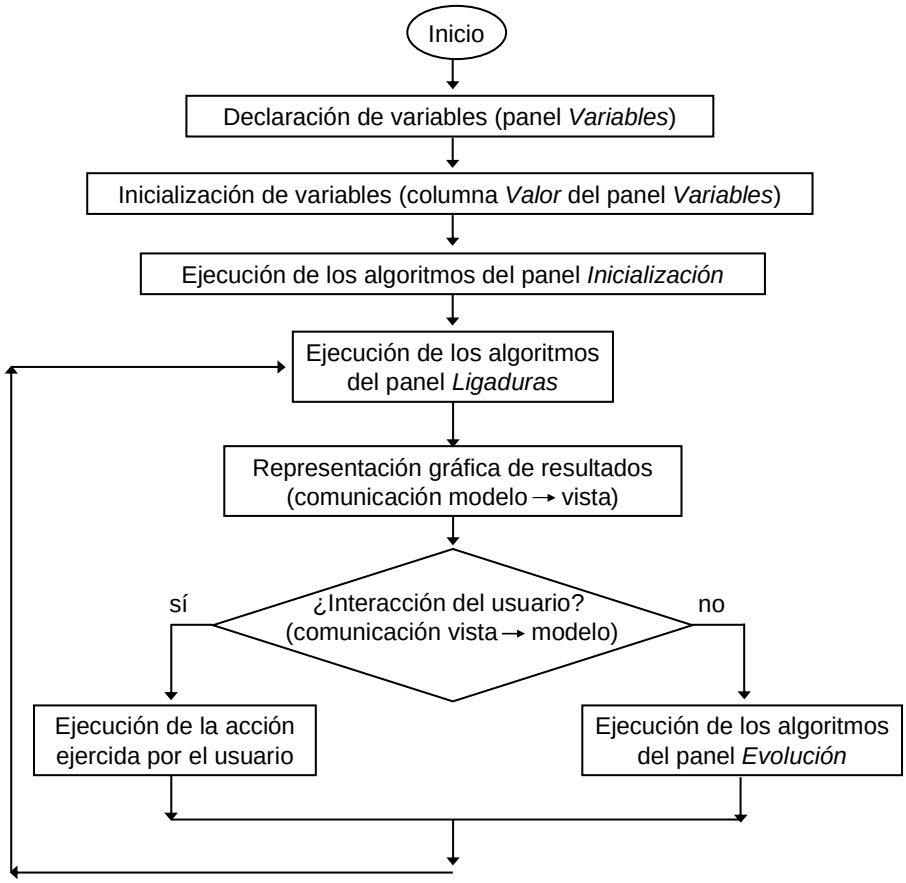


Figura 5.1: Algoritmo de simulación de Ejs.

2. Si un determinado panel consta de varias páginas, éstas se ejecutan siguiendo el orden relativo en que están dispuestas[1], empezando por la que está situada más a la izquierda y terminando con la que está más a la derecha.

   Por ejemplo, en la Figura 7.9 se observa que el panel *Variables* consta de dos ventanas:
   - *Variables principales*, que está situada más a la izquierda, y que por tanto se ejecuta en primer lugar.
   - *Variables auxiliares*, situada más a la derecha, y que por ello se ejecuta en segundo lugar.
3. Ejs ejecuta los algoritmos de una página siguiendo el orden en que están escritos, comenzando por la parte superior de la página y finalizando por su parte inferior, de forma completamente análoga a como si se tratara de la ejecución de un fragmento de código de un programa escrito en Java.

En la Figura 5.1 se muestra el *algoritmo de simulación* de Ejs, es decir, la secuencia de tareas que realiza Ejs para ejecutar cualquier laboratorio virtual. Éstas son las siguientes[2]:

1. Ejs crea las variables y les asigna sus correspondientes valores de inicialización. Para ello, primero realiza las asignaciones descritas en el panel *Variables* y a continuación ejecuta los algoritmos descritos en el panel *Inicialización*.

[1] Ejs permite cambiar la posición relativa de una página respecto a las demás páginas del panel. Es decir, permite desplazarla hacia la izquierda o hacia la derecha. Esta opción aparece en el menú para la gestión de páginas (vea la Figura 7.3). Para desplegar este menú es preciso situar el ratón sobre la lengüeta de la ventana que se desea desplazar y pulsar el botón derecho del ratón.

[2] Esta explicación está extraída de la Sección 3.3 del texto (Esquembre 2004*b*)

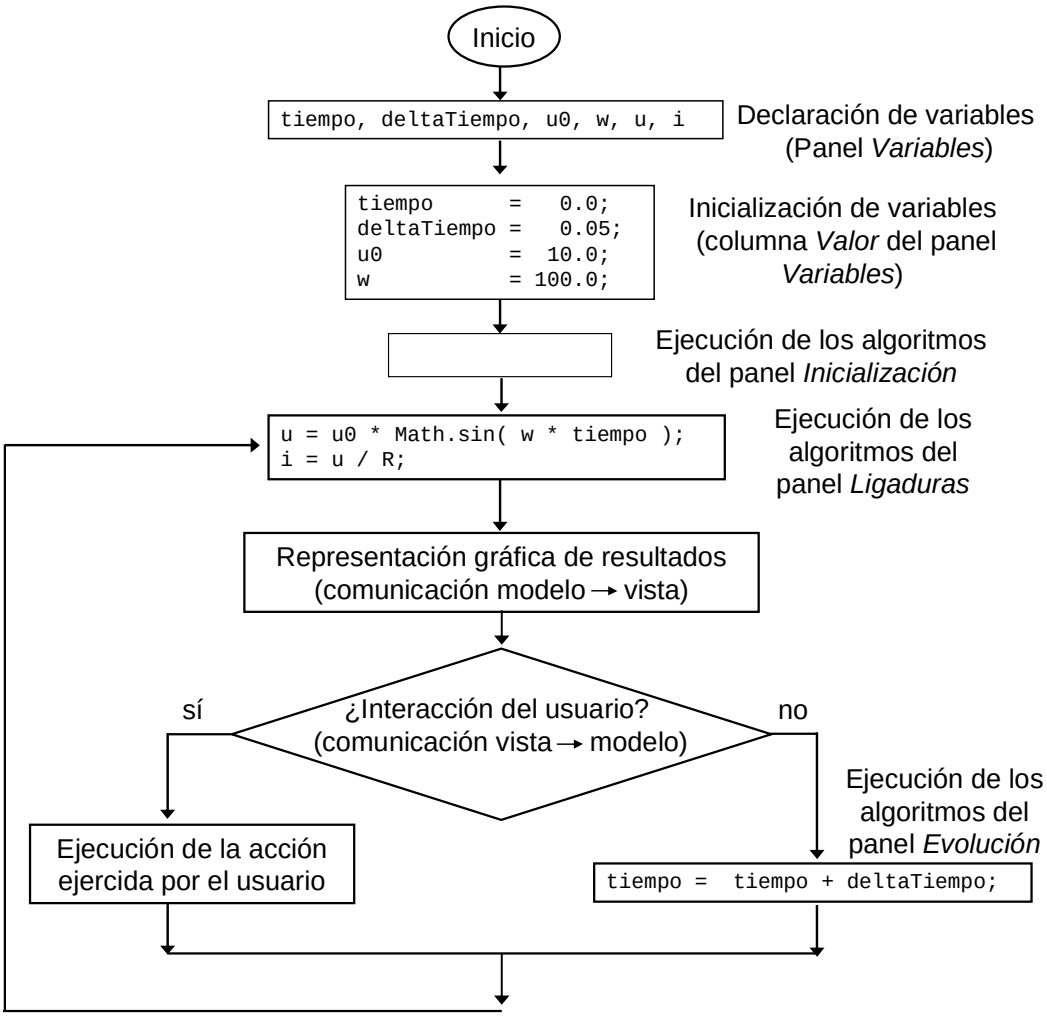


Figura 5.2: Algoritmo de simulación del modelo mostrado en la Figura 2.6a.

2. Ejs ejecuta los algoritmos contenidos en el panel *Ligaduras*.
3. Ejs crea la vista de la simulación y la muestra en pantalla, estableciendo las asociaciones entre las propiedades de los elementos gráficos y las variables del modelo, así como la forma en la que debe reaccionarse ante las acciones del usuario sobre el modelo.

   En este punto, el modelo se encuentra en su estado inicial, y la vista refleja los valores de las variables en este estado.
4. Ejs examina si el usuario ha interaccionado con la vista. Pueden darse dos situaciones:
   a) Si el usuario interacciona con la vista, Ejs ejecuta los algoritmos asociados a dicha interacción y, a continuación, ejecuta los algoritmos del panel *Ligaduras*.
   b) Si el usuario no interacciona con la vista, se ejecutan los algoritmos del panel *Evolución* y seguidamente los del panel *Ligaduras*.
5. Ejs representa en la vista del laboratorio el nuevo estado del modelo que acaba de evaluarse en el paso anterior.
6. Ejs salta al Paso 4.

Ejemplo 5.3.1. *En el caso del modelo mostrado en la Figura 2.6a, que está compuesto por un generador de tensión y una resistencia, el algoritmo de la simulación podría ser el mostrado en la Figura 5.2. El modelo ordenado y resuelto, y con la causalidad computacional señalada es el representado por las Ecs. (2.24 – 2.25).*

Una vez que se ha realizado la asignación de la causalidad computacional del modelo, la descripción del mismo empleando los paneles de Ejs es un proceso sistemático:

- *La asignación de valor a los parámetros se realiza en el panel Variables, al declararlos, o bien en el panel Inicialización.*
- *En el modelo de este ejemplo no hay variables de estado. Si las hubiera, se fijaría su valor inicial de forma análoga a como se asigna valor a los parámetros: en el panel Variables, al declararlas, o bien en el panel Inicialización.*
- *La secuencia ordenada de asignaciones que realiza el cálculo de las variables algebraicas se escribe en el panel Ligaduras.*
- *En el panel Evolución se describe cómo debe realizarse el cálculo de las variables de estado.*
- *Cuando el panel Evolución contiene alguna página EDO, Ejs realiza automáticamente el incremento de la variable tiempo, de acuerdo con los parámetros del método de integración fijados por el usuario. Cuando no hay ningún panel EDO, como en este caso, el usuario del modelo debe ocuparse de escribir el código de avance de la variable tiempo:*
  - *Se define un parámetro (en este ejemplo, $deltaTiempo$), que representa el tamaño del paso de avance en el tiempo. Cambiando el valor asignado a este parámetro, el usuario puede controlar en qué instantes de tiempo evalúa Ejs el modelo.*
  - *Se escribe la ecuación de avance en el tiempo ($tiempo = tiempo + deltaTiempo$).*

□

Ejemplo 5.3.2. *La simulación del modelo del circuito eléctrico representado en la Figura 2.1 puede realizarse empleando Ejs. La asignación de la causalidad computacional de este modelo se explicó en el Ejemplo 2.5.1. El modelo ordenado, resuelto, y con la causalidad computacional señalada es el siguiente:*

$$
\begin{aligned}
[u] &= u_0 \cdot \sin(\omega \cdot t) && (5.2)\\
[i_{R2}] &= \frac{u_C}{R_2} && (5.3)\\
[i_{R1}] &= \frac{u - u_C}{R_1}\,. && (5.4)\\
[i_C] &= i_{R1} - i_{R2} && (5.5)\\
[deru_C] &= \frac{i_C}{C} && (5.6)\\
\frac{d\,[u_C]}{dt} &= deru_C && (5.7)
\end{aligned}
$$

Si se desea emplear uno de los métodos de integración de Ejs para el cálculo de las variables de estado, es preciso:

- *Expresar la derivada de cada una de las variables de estado en función únicamente de variables de estado, parámetros y la variable tiempo. Puesto que se ha asignado la causalidad computacional del modelo, las manipulaciones necesarias para ello se pueden realizar de manera sencilla. En este ejemplo:*

$$
\frac{d\,[u_C]}{dt} = deru_C = \frac{i_C}{C} = \frac{i_{R1} - i_{R2}}{C} = \frac{\frac{u-u_C}{R_1} - \frac{u_C}{R_2}}{C} = \frac{\frac{u_0 \cdot \sin(\omega \cdot t) - u_C}{R_1} - \frac{u_C}{R_2}}{C} \qquad (5.8)
$$

  Es decir, para calcular u_C debe emplearse la expresión siguiente:

$$
\frac{d\,[u_C]}{dt} = \frac{\frac{u_0 \cdot \sin(\omega \cdot t) - u_C}{R_1} - \frac{u_C}{R_2}}{C} \qquad (5.9)
$$

- *Deben escribirse las expresiones calculadas anteriormente para la derivada de las variables de estado en una página ODE, dentro del panel Evolución. En este ejemplo, habría que escribir la Ec. (5.9) en una página EDO del panel Evolución.*

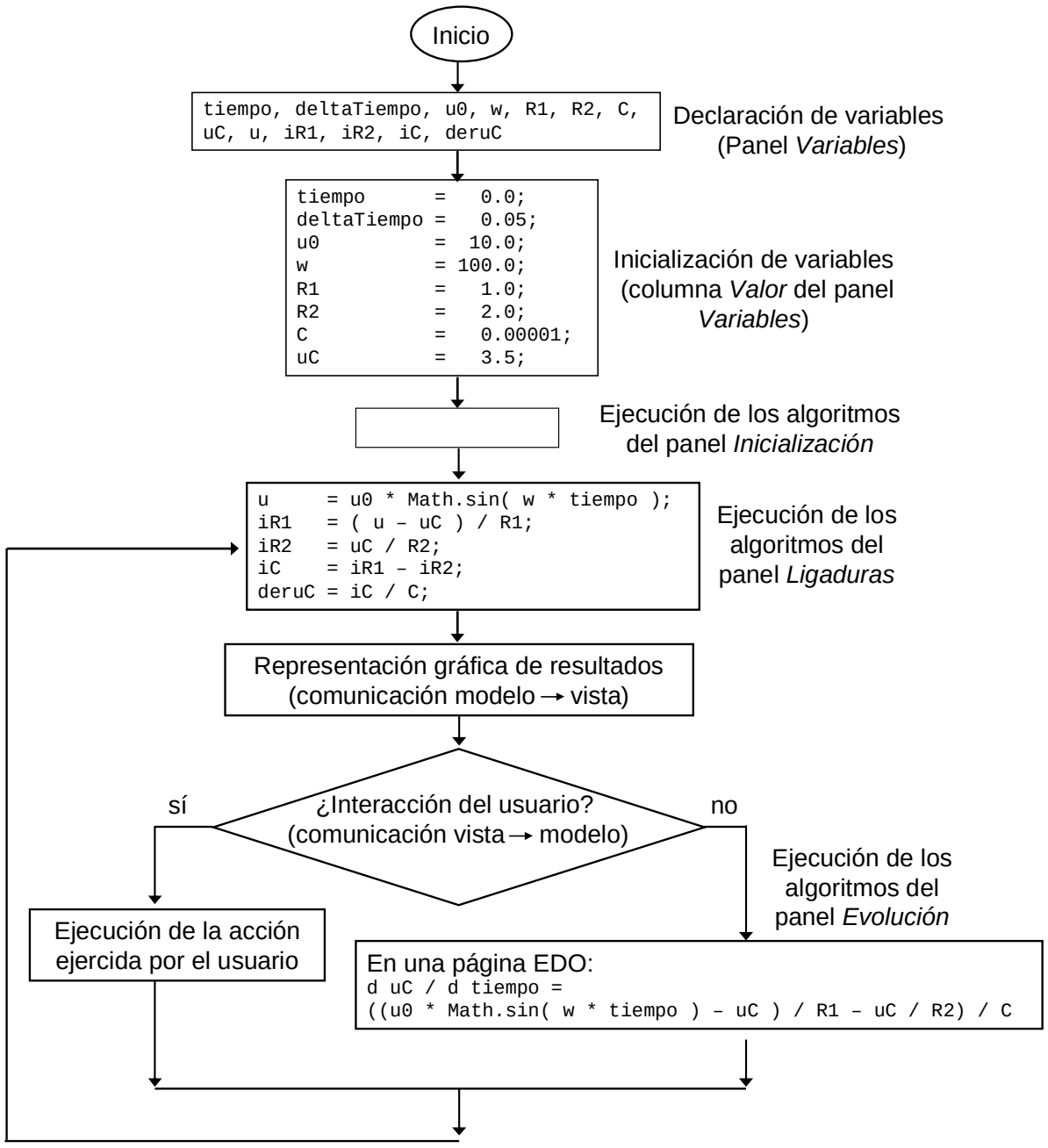


Figura 5.3: Algoritmo de simulación del modelo del circuito RC de la Figura 2.1.

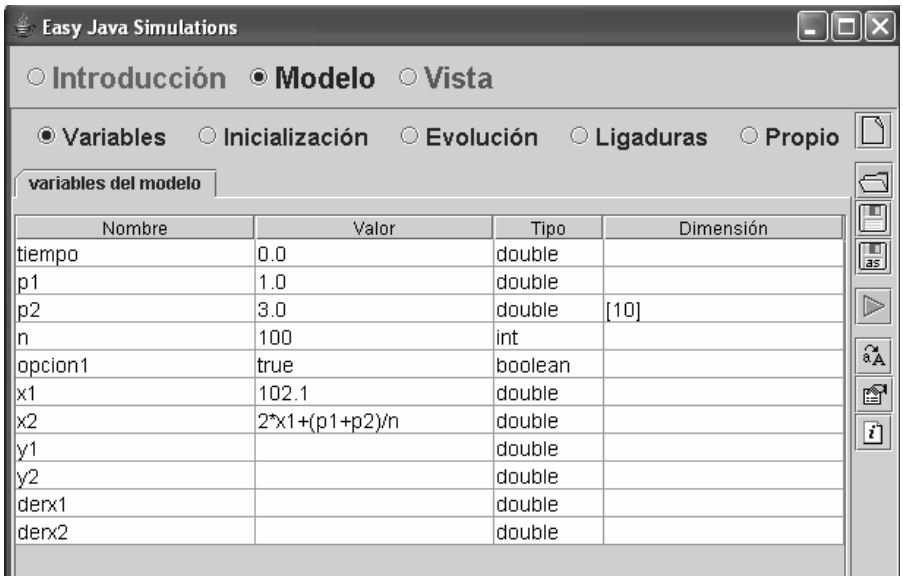
El algoritmo de la simulación se muestra en la Figura 5.3. Observe que:

- *Las asignaciones que permiten calcular las variables algebraicas y las derivadas de las variables de estado, se incluyen en el panel Ligaduras.*
- *Es preciso asignar la causalidad computacional a las ecuaciones del modelo para saber cómo deben ordenarse dentro del panel Ligaduras y qué variable debe despejarse de cada ecuación.*
- *El cálculo de las variables de estado, mediante integración de sus derivadas, debe incluirse en el panel Evolución. Si se desea emplear uno de los métodos de integración de Ejs, debe emplearse una página EDO y expresar la derivada de cada variable de estado en función únicamente de variables de estado, parámetros y la variable tiempo.*
- *Cuando se emplea una página EDO, Ejs gestiona automáticamente el incremento de la variable tiempo a lo largo de la simulación.*

Los cálculos que realiza Ejs para ejecutar la simulación son los indicados a continuación, suponiendo que no se produce ninguna interacción por parte del usuario, y que se emplea el método de integración de Euler explícito con un tamaño del paso igual al valor del parámetro $deltaTiempo$.

1. *Asignación de valor a la variable* $tiempo$*:*
```
tiempo      =   0.0;
```
2. *Asignación de valor a los parámetros del modelo:*
```
deltaTiempo =   0.05;
u0          =  10.0;
w           = 100.0;
R1          =   1.0;
R2          =   2.0;
C           =   0.00001;
```
3. *Asignación de valor inicial a la variable de estado:*
```
uC          =   3.5;
```
4. *Cálculo del valor inicial de las variables algebraicas:*
```
u     = u0 * Math.sin( w * tiempo );
iR1   = ( u - uC ) / R1;
iR2   = uC / R2;
iC    = iR1 - iR2;
deruC = iC / C;
```
5. *Comunicación de los valores de las variables y los parámetros a la vista del laboratorio virtual para su representación gráfica.*
6. *Cálculo del valor de la variable de estado en el instante* $tiempo + deltaTiempo$ *(aplicando el método de Euler explícito):*
```
uC = uC + deltaTiempo * ( (u0*Math.sin(w*tiempo)-uC)/R1 - uC/R2 ) / C;
```
7. *Incremento de la variable* $tiempo$*:*
```
tiempo = tiempo + deltaTiempo;
```
8. *Cálculo de las variables algebraicas en el instante* $tiempo$*:*
```
u     = u0 * Math.sin( w * tiempo );
iR1   = ( u - uC ) / R1;
iR2   = uC / R2;
iC    = iR1 - iR2;
deruC = iC / C;
```
9. *Volver al Paso 5.*

Si se produce interacción por parte del usuario, típicamente será para modificar el valor de alguno de los parámetros o de las variables de estado del modelo. Una vez realizado el cambio, se ejecuta el panel Ligaduras: se recalcula el valor de las variables algebraicas correspondientes a ese nuevo valor de los parámetros y de las variables de estado. Una vez hecho esto, se establece la comunicación con la vista, para representar los valores calculados, y se continua con la integración del modelo (es decir, se ejecuta el panel Evolución). □



Nombre	Valor	Tipo	Dimensión
tiempo	0.0	double	
p1	1.0	double	
p2	3.0	double	[10]
n	100	int	
opcion1	true	boolean	
x1	102.1	double	
x2	2*x1+(p1+p2)/n	double	
y1		double	
y2		double	
derx1		double	
derx2		double	

Figura 5.4: Ejemplo de declaración e inicialización de variables.

5.4. Declaración e inicialización de las variables

De las explicaciones anteriores, se concluye que es preciso declarar y asignar valor inicial a los siguientes tipos de variables del modelo:

- La variable tiempo, que típicamente se inicializará al valor cero.
- Las constantes y parámetros.
- Las variables de estado, tanto continuas como discretas.

Es preciso declarar, pero no inicializar, los siguientes tipos de variables:

- Las variables algebraicas.
- Las derivadas de las variables de estado (variables auxiliares que se han introducido al analizar la causalidad computacional del modelo).

No es preciso inicializar estas variables debido a que su valor en el instante inicial de la simulación lo calcula Ejs de ejecutar el código del panel *Ligaduras* (vea la Figura 5.1). En cualquier caso, si se inicializa alguna de estas variables, este valor será sobreescrito al ejecutar el panel *Ligaduras*.

La inicialización de las variables se realiza en dos paneles: *Variables* e *Inicialización* (vea nuevamente la Figura 5.1):

- En primer lugar, Ejs ejecuta las asignaciones que el usuario haya escrito en el panel *Variables*.
- A continuación, Ejs ejecuta los algoritmos que el usuario haya definido en el panel *Inicialización*

A continuación se describe cada uno de ellos.

El panel *Variables*

En la Figura 5.4 se muestra un ejemplo de declaración e inicialización de variables. A continuación, se explica el significado de cada uno de los campos.

El nombre de la variable debe escribirse en la correspondiente casilla de la columna *Nombre*.

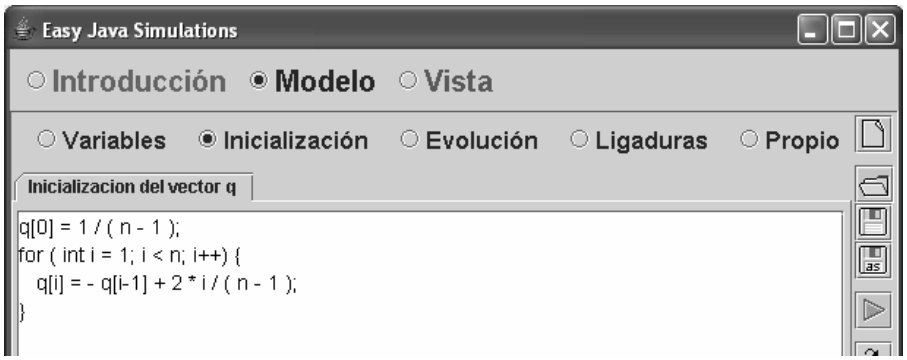


Figura 5.5: Ejemplo de inicialización de una variable vectorial.

Al asignar nombres, debe tenerse en cuenta que no puede asignarse el mismo nombre a dos variables diferentes. Para evitar que, por error, se asigne el mismo nombre a dos variables diferentes del laboratorio virtual (por ejemplo, a una variable del modelo y a una variable definida en la vista), es recomendable asignar a las variables nombres descriptivos de su función.

En la columna *Tipo* debe indicarse el tipo de cada variable. En Ejs las variables pueden ser de los tipos siguientes:

- *boolean*: cuando sólo puede tomar dos valores: true o false.
- *int*: variable de tipo entero.
- *double*: variables de tipo real.
- *String*: variables del tipo cadena de caracteres.

En la columna *Dimensión* debe indicarse la dimensión de la variable. Si es una variable escalar (es decir, no es un vector ni una matriz), la casilla debe dejarse vacía. Si alguna de las variables es un vector o una matriz, debe indicarse su dimensión.

Ejemplo 5.4.1. *Si se escribe* $[10]$ *en la casilla Dimensión de la variable* q*, se está declarando un vector de* 10 *componentes:* $q\,[0]$*,* $q\,[1]$*,* $\cdots$*,* $q\,[9]$*. Si se escribe* $[10]\,[20]$*, se está declarando una matriz de* 10 *filas (de la* 0 *a la* 9*) y* 20 *columnas (de la* 0 *a la* 19*).* □

En la columna *Valor*, puede introducirse el valor inicial de la variable (es opcional), que puede ser un valor constante o una expresión. Este "valor inicial" es el valor que se asigna a la variable al comienzo de la simulación. Cuando se asigna un valor inicial a una variable vectorial o matricial, Ejs inicializa todos los elementos de la variable igualándolos a ese determinado valor.

El panel *Inicialización*

En algunos modelos es preciso realizar determinados cálculos para asignar valor inicial a alguna de las variables. El código Java que describe cómo han de realizarse estos cálculos puede escribirse en el panel *Inicialización*.

Ejemplo 5.4.2. *Cuando se quiere asignar un valor inicial diferente a cada uno de los elementos de una variable vectorial o matricial, debe hacerse en el panel Inicialización.*

En la Figura 5.5 se muestra el código para realizar la inicialización de los componentes de un vector mediante la fórmula:

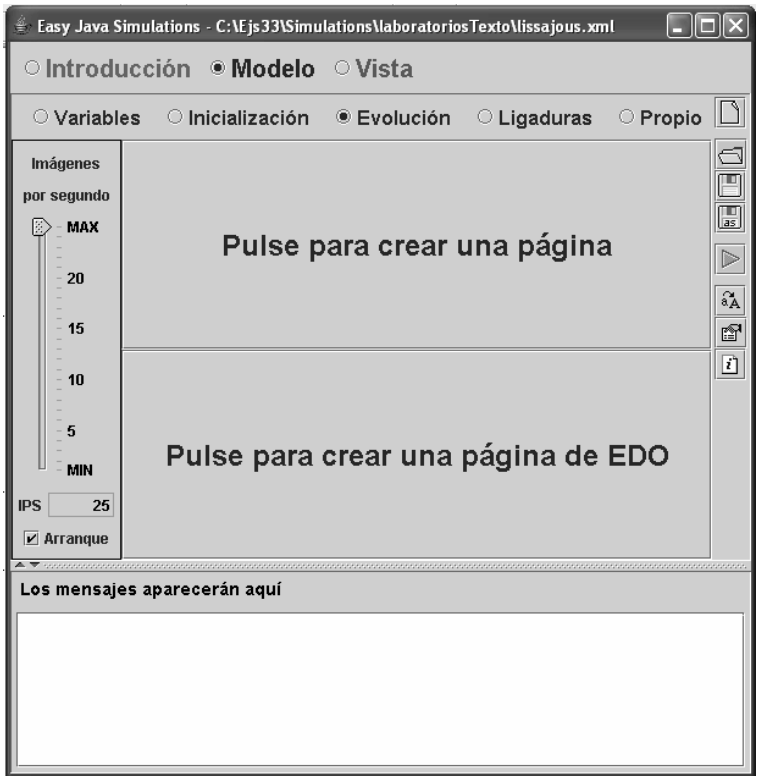


Figura 5.6: Panel para la descripción de la *Evolución*.

$$q[0] = \frac{1}{n-1} \tag{5.10}$$

$$q[i] = -q[i-1] + \frac{2 \cdot i}{n-1} \qquad \textit{para } i : 1, \ldots, n-1 \tag{5.11}$$

donde se supone que la variable q *es un vector de* n *componentes:* $q[0], \ldots, q[n-1]$. □

5.5. Descripción de la evolución

En la Figura 5.6 de muestra el panel *Evolución*. Observe que el panel se encuentra dividido en las tres partes siguientes:

- En la parte superior derecha, hay un subpanel con el letrero *Pulse para crear una página*. Pulsando se crea una página en la cual el usuario puede escribir los algoritmos que desee, por ejemplo, puede programar sus propios métodos de integración para calcular las variables de estado.
- En la parte inferior derecha, hay un subpanel con el letrero *Pulse para crear una página de EDO* (EDO es el acrónimo de *Ecuación Diferencial Ordinaria*). Esta página es un asistente para la definición de ecuaciones diferenciales ordinarias. Ejs genera automáticamente el código para integrar numéricamente las EDO definidas en esta página. Ejs soporta cuatro métodos de integración, entre los cuales el usuario puede escoger. Estos son:
  - Euler

  - Punto medio (Euler-Richardson)
  - Runge-Kutta (4º orden)
  - Runge-Kutta-Fehlberg (4º-5º orden)

  En la Sección 5.8 se describen brevemente los fundamentos de estos algoritmos. En cualquier caso, el código que genera Ejs para una página EDO realiza las dos tareas siguientes:

  1. Calcula el valor de las variables de estado, aplicando para ello el algoritmo correspondiente al método de integración que ha seleccionado el usuario. Si el valor actual de la variable tiempo es t, y el tamaño del paso de integración es Δt, este valor calculado de las variables de estado es el valor de las mismas en el instante de tiempo $t + \Delta t$.
  2. Incrementa el valor de la variable tiempo en Δt.

  Puesto que el código generado por Ejs para una página EDO incrementa el valor de la variable tiempo, **en un laboratorio virtual no puede haber más de una página de EDO**.

  Igualmente, si el laboratorio no tiene ninguna página EDO, el usuario debe gestionar por sí mismo el incremento de la variable tiempo. Este es el caso del laboratorio virtual descrito en la Sección 7.

- En la parte izquierda, hay un subpanel que contiene el letrero *Imágenes por segundo* (abreviado: IPS), debajo del cual hay un indicador que el usuario puede desplazar verticalmente arrastrándolo con el ratón. Con ello puede regular la velocidad de ejecución de la simulación. En concreto, determina el número de pasos en el tiempo que avanzará la simulación en un segundo de tiempo real[3]. Cuanto mayor sea el número de imágenes por segundo, mayor será la velocidad con que se visualizará la evolución del sistema.

  En la parte inferior se encuentra el botón *Arranque*. Cuando está activado, la simulación comienza tan pronto como es ejecutado el laboratorio virtual.

  Si está desactivado la simulación no comenzará cuando se ejecute el laboratorio virtual. La utilidad de ello es permitir que el usuario realice ciertas tareas antes de que comience la simulación (por ejemplo, cambiar el valor de ciertas variables). En este caso, es preciso definir un botón de arranque (en la vista del laboratorio virtual) que el usuario deberá pulsar para que se inicie la simulación. Este botón de arranque deberá tener asociada como acción una llamada al método *_play()*. Se trata de un método predefinido de Ejs que hace que comience la ejecución de la simulación.

5.6. Descripción de las ligaduras

En general, en el panel *Ligaduras* se escriben los algoritmos para el cálculo de las variables algebraicas del modelo.

Como puede verse en la Figura 5.1, las ecuaciones de ligadura son aquellas que describen el comportamiento del modelo, no sólo durante su evolución en el tiempo, sino también en el caso de que el usuario realice cualquier cambio interactivo en el valor de alguna variable. Esta es una de las diferencias conceptuales entre las ecuaciones de evolución y las ecuaciones de ligadura.

5.7. Métodos propios del usuario

El usuario puede definir, en el panel *Propio*, todos aquellos *métodos* en lenguaje Java que precise para la definición del modelo o de la vista. Típicamente, los métodos se emplean para definir acciones sobre el modelo que son activadas desde la vista (por ejemplo, cuando el usuario pulsa un botón).

[3]La simulación avanza a pasos Δt en el tiempo simulado, con lo cual, en un segundo de tiempo real el tiempo simulado avanzará IPS $\cdot\, \Delta t$.

Estos métodos pueden ser invocados desde cualquier parte de la simulación, y está es su única finalidad. Es decir, a excepción de los puntos donde son invocados, durante la simulación no se realiza ninguna otra llamada a estos métodos.

5.8. Los algoritmos de integración de Ejs

Como se ha indicado anteriormente, Ejs soporta cuatro algoritmos para la integración de las ecuaciones diferenciales ordinarias:

- Euler explícito
- Punto medio (Euler-Richardson)
- Runge-Kutta (4º orden)
- Runge-Kutta-Fehlberg (4º-5º orden)

A continuación se describe cada uno de ellos.

Método de Euler explícito

La función de paso del método de Euler para la ecuación diferencial ordinaria

$$\frac{dx}{dt} = f(x,t) \tag{5.12}$$

es la siguiente:

$$x_{i+1} = x_i + f(x_i, t_i) \cdot \Delta t \tag{5.13}$$

que corresponde con un desarrollo de Taylor truncado en el primer orden.

Ejemplo 5.8.1. *El modelo cinemático de una masa puntual (m), sobre la que actúa una fuerza (F), consta de las ecuaciones siguientes:*

$$v = \frac{dy}{dt} \tag{5.14}$$

$$a = \frac{dv}{dt} \tag{5.15}$$

$$a = \frac{F(y,v,t)}{m} \tag{5.16}$$

El algoritmo para integrar este modelo, empleando el método de Euler explícito, es el siguiente:

1. *Deben conocerse los valores de las variables de estado (y, v) en el instante inicial de la simulación (t_0). Sean estos valores: y_0, v_0.*

   También debe fijarse el tamaño del paso de integración (Δt).
2. *Cálculo de la aceleración de la partícula (a_i), que es una variable algebraica:*

$$a_i = \frac{F(y_i, v_i, t_i)}{m} \tag{5.17}$$

3. *Se calcula el valor de las variables de estado en el nuevo instante de tiempo:*

$$v_{i+1} = v_i + a_i \cdot \Delta t \tag{5.18}$$

$$y_{i+1} = y_i + v_i \cdot \Delta t \tag{5.19}$$

La velocidad al final del intervalo (v_{i+1}) se calcula de la aceleración al principio del intervalo (a_i). Igualmente, la posición al final del intervalo (y_{i+1}) se calcula de la aceleración al principio del intervalo (v_i).

4. *Avance de un paso en el tiempo e incremento del índice i:*

$$t_{i+1} = t_i + \Delta t \tag{5.20}$$
$$i = i + 1 \tag{5.21}$$

5. *Volver al Paso 2.*

□

Método de Euler-Richardson

El algoritmo de Euler-Richardson usa la derivada en el principio del intervalo para estimar el valor de la variable de estado en el punto medio del intervalo ($t_{med} = t + \frac{\Delta t}{2}$). A continuación, emplea la derivada en este punto medio para calcular la variable de estado al final del intervalo.

Este algoritmo también se conoce como algoritmo de Runge-Kutta de 2º orden o del punto medio.

La función de paso del método de Euler-Richardson para la ecuación diferencial ordinaria

$$\frac{dx}{dt} = f(x,t) \tag{5.22}$$

es la siguiente:

$$x_{med} = x_i + f(x_i, t_i) \cdot \frac{\Delta t}{2} \tag{5.23}$$
$$x_{i+1} = x_i + f\left(x_{med}, t_i + \frac{\Delta t}{2}\right) \cdot \Delta t \tag{5.24}$$

Ejemplo 5.8.2. *El modelo de la masa puntual descrito en el Ejemplo 5.8.1 puede integrarse, empleando el algoritmo de Euler-Richardson, de la forma siguiente:*

$$v_{med} = v_i + a_i \cdot \frac{\Delta t}{2} \tag{5.25}$$
$$y_{med} = y_i + v_i \cdot \frac{\Delta t}{2} \tag{5.26}$$
$$a_{med} = \frac{F\left(y_{med}, v_{med}, t_i + \frac{\Delta t}{2}\right)}{m} \tag{5.27}$$

y

$$v_{i+1} = v_i + a_{med} \cdot \Delta t \tag{5.28}$$
$$y_{i+1} = y_i + v_{med} \cdot \Delta t \tag{5.29}$$

□

Método de Runge-Kutta (4º orden)

Este método se emplea con mucha frecuencia. Requiere cuatro evaluaciones de la derivada por cada paso en el tiempo. Se denomina de 4º orden porque considera el desarrollo de Taylor hasta 4º orden. La función de paso para la ecuación diferencial ordinaria

$$\frac{dx}{dt} = f(x,t) \tag{5.30}$$

es la siguiente:

$$k_1 = \Delta t \cdot f(x_i, t_i) \tag{5.31}$$
$$k_2 = \Delta t \cdot f\left(x_i + \frac{k_1}{2}, t_i + \frac{\Delta t}{2}\right) \tag{5.32}$$
$$k_3 = \Delta t \cdot f\left(x_i + \frac{k_2}{2}, t_i + \frac{\Delta t}{2}\right) \tag{5.33}$$
$$k_4 = \Delta t \cdot f(x_i + k_3, t_i + \Delta t) \tag{5.34}$$
$$x_{i+1} = x_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \tag{5.35}$$

Método de Runge-Kutta-Fehlberg (4º-5º orden)

Se trata de un método de 4º orden embebido dentro de un método de 5º orden. El error cometido al aplicar el método de cuarto orden se obtiene restando las funciones paso de ambos métodos, y corresponderá con el término de 5º orden del desarrollo de Taylor ($\frac{d^5 x_i}{dt^5} \cdot \frac{\Delta t^5}{5!}$).

La función de paso para la ecuación diferencial ordinaria

$$\frac{dx}{dt} = f(x, t) \tag{5.36}$$

es la siguiente:

$$k_1 = \Delta t \cdot f(x_i, t_i) \tag{5.37}$$
$$k_2 = \Delta t \cdot f\left(x_i + \frac{k_1}{4}, t_i + \frac{\Delta t}{4}\right) \tag{5.38}$$
$$k_3 = \Delta t \cdot f\left(x_i + \frac{3}{32} \cdot k_1 + \frac{9}{32} \cdot k_2, t_i + \frac{3}{8} \cdot \Delta t\right) \tag{5.39}$$
$$k_4 = \Delta t \cdot f\left(x_i + \frac{1932}{2197} \cdot k_1 - \frac{7200}{2197} \cdot k_2 + \frac{7296}{2197} \cdot k_3, t_i + \frac{12}{13} \cdot \Delta t\right) \tag{5.40}$$
$$k_5 = \Delta t \cdot f\left(x_i + \frac{439}{216} \cdot k_1 - 8 \cdot k_2 + \frac{3680}{513} \cdot k_3 - \frac{845}{4104} \cdot k_4, t_i + \Delta t\right) \tag{5.41}$$
$$k_6 = \Delta t \cdot f\left(x_i - \frac{8}{27} \cdot k_1 + 2 \cdot k_2 - \frac{3544}{2565} \cdot k_3 + \frac{1859}{4104} \cdot k_4 - \frac{11}{40} \cdot k_5, t_i + \frac{\Delta t}{2}\right) \tag{5.42}$$

La fórmula de paso de 4º orden es:

$$x_{i+1,\ 4^{\text{o}}\ \text{orden}} = x_i + \frac{25}{216} \cdot k_1 + \frac{1408}{2565} \cdot k_3 + \frac{2197}{4104} \cdot k_4 - \frac{k_5}{5} \tag{5.43}$$

y la de 5º orden es:

$$x_{i+1,\ 5^{\text{o}}\ \text{orden}} = x_i + \frac{16}{135} \cdot k_1 + \frac{6656}{12825} \cdot k_3 + \frac{28561}{56430} \cdot k_4 - \frac{9}{50} \cdot k_5 + \frac{2}{55} \cdot k_6 \tag{5.44}$$

Descripción de las ecuaciones diferenciales en la página EDO

Anteriormente se ha indicado que, en una página EDO, la descripción de las derivadas de las variables de estado debe hacerse únicamente en función de:

- variables de estado,
- constantes y parámetros, y
- la variable tiempo.

A la vista de los algoritmos de los métodos de integración de Euler-Richardson, R-K y R-K-F, la razón de ello es comprensible: estos métodos requieren la evaluación de la derivada en instantes intermedios del paso de integración.

Por la forma de trabajar del algoritmo de Ejs, para posibilitar la evaluación de la derivada en los instantes intermedios del paso de integración, es necesario que ésta se exprese únicamente en función de variables de estado, variables que permanezcan constantes durante el paso de integración (es decir, las constantes y los parámetros del modelo), y la variable tiempo.

Lo anterior es equivalente a decir que la expresión de la derivada que se introduce en una página EDO no puede contener variables algebraicas.

Tema 6

Conceptos básicos para la descripción de la vista

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Realizar las acciones básicas para configurar la vista de un laboratorio virtual.
- Cuál es la utilidad del panel *Árbol de elementos*.
- Cuál es la utilidad del panel *Elementos para la vista*.
- Reconocer las características fundamentales de algunas clases de elementos gráficos.

La vista del laboratorio virtual es la interfaz entre el usuario y el modelo. Mediante la manipulación de los controles de la vista, el usuario puede:

- Controlar la ejecución de la simulación, por ejemplo, deteniéndola o reiniciándola.
- Cambiar el valor de los parámetros, de las variables de entrada y de las variables de estado del modelo.

La definición de la vista de un laboratorio virtual se realiza en el panel *Vista* (vea la Figura 6.1). Este panel se subdivide en dos paneles: el panel *Árbol de elementos* y el panel *Elementos para la vista*.

El panel *Elementos para la vista* se divide en tres áreas, donde se encuentran los iconos de todas las clases que se pueden usar para componer la vista. Estas tres áreas se denominan *Contenedores*, *Básicos* y *Dibujo*. A su vez, el área de elementos llamada *Dibujo* contiene tres paneles con iconos que representan clases de elementos gráficos: *Básicos*, *Grafos y cuerpos*, y *Campos*. Al situar el ratón sobre cualquiera de los iconos se obtiene un mensaje con información acerca de su finalidad.

El panel *Árbol de elementos* muestra la composición de la vista del laboratorio virtual. La vista está compuesta por elementos gráficos organizados formando una estructura en árbol.

Para desplegar el menú de un elemento gráfico, hay que seleccionar dicho elemento en el panel *Árbol de la vista* y hacer clic con el botón derecho del ratón.

En la siguiente sección se explica cómo crear la vista del laboratorio, así como el procedimiento para seleccionar y editar el menú de un determinado elemento gráfico. Las secciones posteriores describen algunas clases de elementos gráficos.

Figura 6.1: Panel *Vista* de un laboratorio. Se muestra el menú del elemento *Traza*.

6.1. Árbol de elementos

La vista del laboratorio virtual se compone de elementos gráficos, los cuales pueden situarse unos dentro de otros, formando una estructura de árbol.

Decimos que un elemento b es *padre* de un elemento c, si b contiene a c. Si b es padre de c, entonces c es hijo de b. En el árbol de elementos se observará la existencia de un enlace directo entre c y b.

Un padre puede tener varios hijos, pero un hijo sólo puede tener un padre. Así, por ejemplo, en la Figura 6.1 el elemento gráfico PanelDibujo tiene dos hijos (Particula1 y Particula2) y un padre (Ventana).

Es importante el orden relativo de los hijos, ya que un hijo se dibuja encima de los hijos situados por debajo del mismo en el árbol de elementos. Por ejemplo, en la Figura 6.1 Particula1 se encuentra por encima de Particula2 en el árbol de elementos de la vista. Si estos dos elementos tienen el mismo tamaño y posición dentro del PanelDibujo, sólo se verá Particula1, ya que Particula1 oculta a Particula2.

Como se observa en la Figura 6.1, en el árbol de elementos hay siempre un elemento raíz, denominado *Vista de la simulación*. Este es el elemento de nivel superior, en el cual se ubican todos los elementos gráficos que componen la vista. Este elemento debe tener siempre como hijos elementos gráficos de la clase *Ventana* o *VentanaDiálogo*.

Para situar un elemento gráfico en el árbol de la vista, hay que seguir los pasos siguientes:

1. *Seleccionar la clase del elemento gráfico*. Para ello, debe hacer clic con el ratón sobre el icono de la clase de elemento gráfico en cuestión. Éste estará situado dentro del panel de *Elementos para la vista*. Al hacer esto, el icono queda inscrito en un recuadro oscuro, que señaliza que esa clase de elemento ha sido seleccionada. Observe que al posicionar el ratón sobre el elemento se obtiene información sobre su finalidad.
2. *Definir tantos objetos como desee de la clase anteriormente seleccionada*. Para ello, debe hacer clic con el ratón sobre el elemento gráfico padre del nuevo elemento que está a punto de crear. Observe que al situar el ratón sobre cualquier punto de esta ventana, el curso adquiere la forma de una varita mágica.

Para obtener el menú de un elemento gráfico hay que situar el ratón sobre dicho elemento y hacer clic con el botón derecho. En la Figura 6.1 se muestra el menú del elemento *Traza*. Algunas opciones de los menús de los elementos gráficos son:

- **Propiedades**: permite acceder al menú de propiedades del elemento.
- **Renombrar**: permite cambiar el nombre del elemento. No puede haber dos elementos del árbol de elementos de la vista con el mismo nombre.
- **Cambiar padre**: permite cambiar al elemento de padre.
- **Mover arriba**: permite subir el elemento una posición dentro de los elementos que comparten el mismo elemento padre.
- **Mover abajo**: permite bajar el elemento una posición dentro de los elementos que comparten el mismo elemento padre.
- **Eliminar**: permite eliminar al elemento del árbol de elementos.

En el siguiente apartado, se describen las características fundamentales de algunas clases.

6.2. Clases de elementos de la vista

Las clases de elementos de la vista se dividen en los tres tipos siguientes:

- **Contenedores**: se caracterizan porque pueden tener hijos.
- **Básicos**: no pueden tener hijos. El padre de estas clases son elementos de tipo *Contenedor* salvo las clases *PanelDibujo*, *PanelDibujo3D* y *PanelConEjes*.
- **Dibujo**: no pueden tener hijos. El padre de estas clases son elementos de la clases *PanelDibujo*, *PanelDibujo3D* o *PanelConEjes*.

A continuación, se describe brevemente cada uno de ellos.

6.3. Algunas clases de elementos de tipo *Contenedor*

Las clases de este tipo pueden contener otros elementos gráficos. A continuación se describen las características fundamentales de las clases *Ventana*, *VentanaDialogo*, *Panel*, *PanelDibujo*, *PanelDibujo3D* y *PanelConEjes*.

Clases Ventana y VentanaDialogo

El padre de los elementos de estas clases es la *Vista de la Simulación*. Estas dos clases tienen muchas características comunes y comparten las mismas propiedades. Se diferencian fundamentalmente en que el cierre de un elemento de la clase *Ventana* conlleva la finalización de la simulación. Sin embargo, al cerrar un elemento de la clase VentanaDialogo la simulación continúa.

La ventana de propiedades se muestra en la Figura 6.2. A continuación, se describen sus propiedades:

- **Distribución**. La distribución de la ventana según la política de distribución especificada. Se puede elegir una de las siguientes políticas de distribución:

Figura 6.2: Propiedades del elemento *Ventana*.

  - *Márgenes*: los hijos se sitúan en cinco áreas: centro, izquierda, derecha, arriba o abajo. Cambiar el tamaño de la ventana sólo cambia el tamaño del área del centro sin cambiar el área de los márgenes (izquierda, derecha, arriba o abajo).
  - *Caja horizontal*: los hijos se sitúan horizontalmente de izquierda a derecha.
  - *Caja vertical*: los hijos se sitúan verticalmente de arriba a abajo.
  - *Rejilla*: se indica el número de filas y columnas. Los hijos se van colocando por filas comenzando por arriba. Dentro de una fila se van colocando de izquierda a derecha. Todos los hijos ocupan un área del mismo tamaño.
  - *Flujo izquierda/centro/derecha*: la distribución flujo hace que los hijos se alineen horizontalmente. Cuando no hay espacio en una línea se crea una fila adicional para el hijo. Izquierda/centro/derecha indica que dentro de una fila los hijos se alinean comenzando por la izquierda/centro/derecha.
- **Visible**. La ventana es visible cuando el valor de esta propiedad es true. Si se escribe el nombre de una variable en la casilla de la propiedad Visible, la ventana será visible cuando dicha variable valga true.
- **Situación**. Posición de la ventana dentro de la pantalla del ordenador.
- **Tamaño**. Tamaño de la ventana.
- **Redimensionable**. La ventana cambia su tamaño al arrastrar un extremo de la misma con el ratón cuando el valor de esta propiedad es true.

Clase Panel

Esta clase siempre tiene como padre a otro elemento de alguna clase de tipo contenedor. Se usa para situar otros elementos, hijos de este elemento, según una determinada política de distribución. La propiedad distribución se explicó al describir las propiedades de las clases *Ventana* y *VentanaDialogo*.

Clase PanelDibujo

Los hijos de los elementos de esta clase son del tipo *Dibujo*. Los elementos de la clase *PanelDibujo* representan una región del plano en 2 dimensiones y proporcionan su propio sistema de coordenadas. Para situar a un elemento dentro del plano del panelDibujo hay que especificar las coordenadas x e y que ocupa en este plano (ver la Figura 6.3).

Los elementos de la clase *PanelDibujo* son interactivos, es decir, se puede asociar una acción que se ejecute cuando se produzca una manipulación del ratón sobre el elemento. Responden así a los diferentes gestos del ratón según el esquema siguiente:

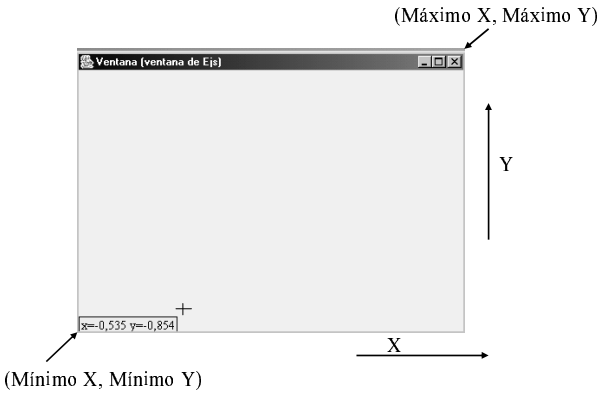


Figura 6.3: Sistema de coordenadas del elemento *PanelDibujo*.

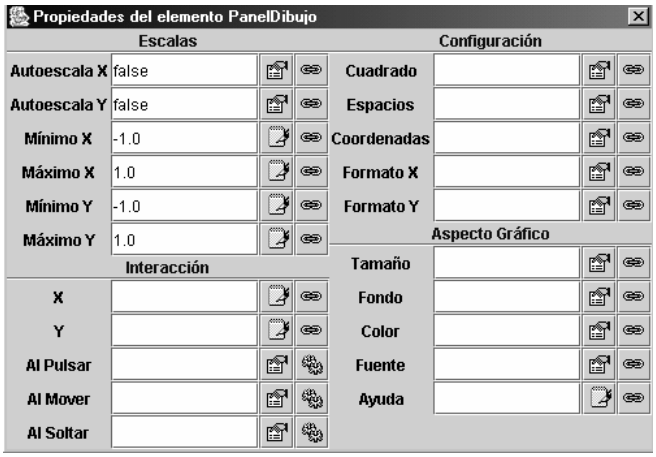


Figura 6.4: Propiedades del elemento *PanelDibujo*.

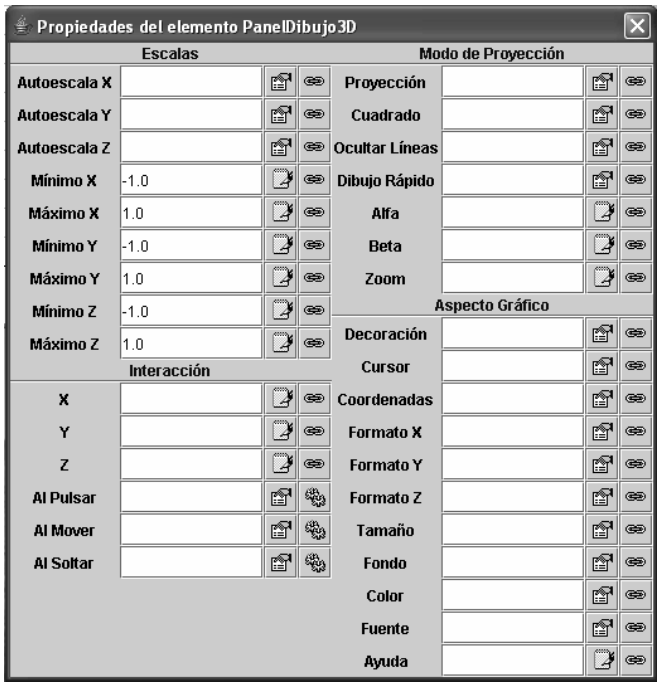


Figura 6.5: Propiedades del elemento *PanelDibujo3D*.

- Cuando el usuario hace clic en el panel, se invoca la acción asociada a la propiedad "Al Pulsar". Inmediatamente después, se fijan las propiedades "X" e "Y" del panel a las coordenadas de la posición del ratón, lo que invoca a su vez a la acción asociada a la propiedad "Al Mover", si la hubiera.
- Cuando el usuario mueve el ratón (con el botón del ratón apretado), se actualizan las propiedades "X" e "Y" y se invoca la acción asociada a la propiedad "Al Mover".
- Cuando el usuario finalmente suelta el botón del ratón (si lo hace dentro del elemento) se invoca la acción asociada a la propiedad "Al Soltar".

El menú de propiedades de estos elementos se muestra en la Figura 6.4. Se puede especificar las dimensiones del plano que representa el elemento especificando sus coordenadas "X Mínimo", "X Máximo", "Y Mínimo" e "Y Máximo". También se puede dejar que se ajusten las escalas automáticamente. Para ello, hay que introducir en las propiedades "Autoescala X" y "Autoescala Y" el valor false.

Clase PanelDibujo3D

Es un contenedor especial en tres dimensiones para elementos gráficos de dibujo.

Los elementos de esta clase representan una región tridimensional del espacio y proporcionan su propio sistema de coordenadas. Para situar un elemento dentro del espacio que representan los elementos de la clase *panelDibujo3D*, hay que especificar las coordenadas X, Y y Z que ocupa en este espacio.

El menú de propiedades de estos elementos se muestra en la Figura 6.5. Las dimensiones de la región del espacio se especifican asignando valor a las propiedades "Mínimo X", "Máximo X", "Mínimo Y", "Máximo Y", "Mínimo Z" y "Máximo Z". También se puede dejar que se ajusten las escalas automáticamente de modo que se muestren todos los elementos gráficos ubicados en el contenedor asignando el valor true a las propiedades "Autoescala X", "Autoescala Y" y "Autoescala Z".

Los elementos de esta clase responden a la interacción del ratón del siguiente modo:

- Cuando el usuario hace clic con el ratón y arrastra el panel, cambia el punto de perspectiva de la vista.
- Si se presiona la tecla Control mientras se opera con el ratón, entonces la escena se mueve en su conjunto.
- Si se presiona la tecla de mayúsculas, se aplica un zoom (positivo o negativo, según el movimiento del ratón) a toda la escena.
- Si la tecla que se mantiene pulsada es "Alt", aparece un cursor tridimensional que permite seleccionar un punto de la escena. El movimiento de este cursor responde al movimiento bidimensional del ratón afectando solamente a dos coordenadas espaciales. Para forzar el movimiento en una dimensión particular, deberá mantenerse pulsada la tecla correspondiente, x, y o z. Al cambiar de punto se aplican las acciones "Al pulsar", "Al mover" y "Al soltar", exactamente igual que en el caso de los paneles de dibujo bidimensionales.

Clase PanelConEjes

La clase *PanelConEjes* es una variante de la clase *PanelDibujo*. Incluye, por defecto, un sistema de ejes coordenados. Estos elementos son interactivos y responden a la acción del usuario del mismo modo que los elementos de la clase PanelDibujo.

A continuación, se describen algunas propiedades de los elementos de esta clase (ver Figura 6.6):

- **Título**: título mostrado en la parte superior del panel.
- **Tipo de Ejes**: tipo de ejes mostrados. Se puede escoger entre tres tipos de ejes cartesianos y dos tipos de ejes polares.

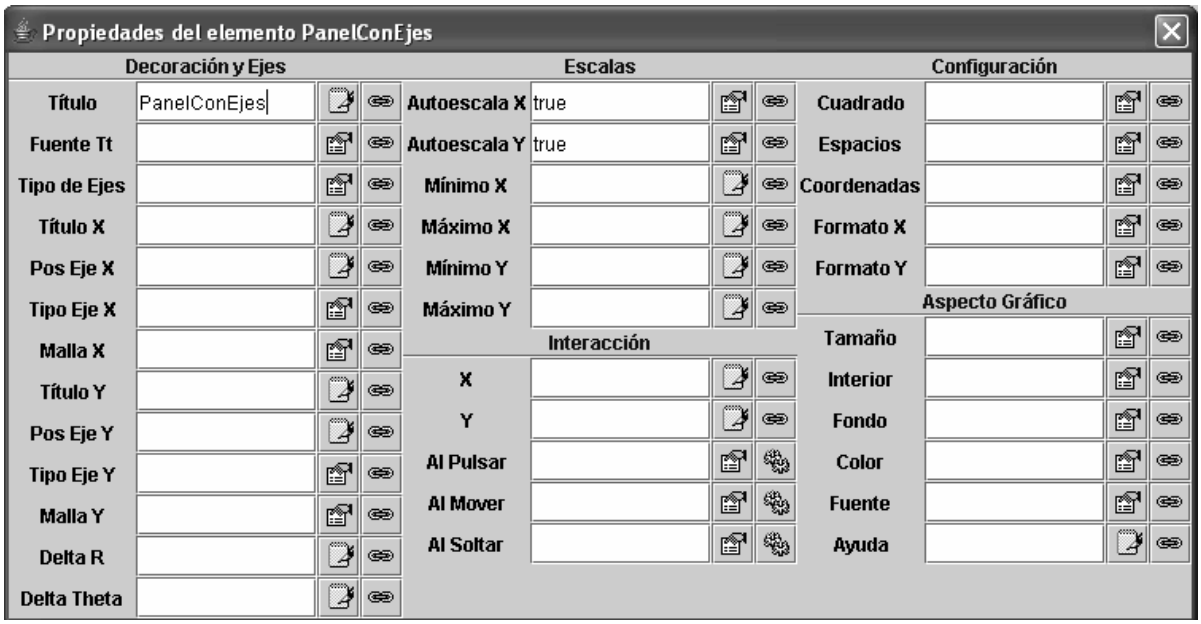


Figura 6.6: Propiedades del elemento *PanelConEjes*.

Figura 6.7: Propiedades del elemento *Boton*.

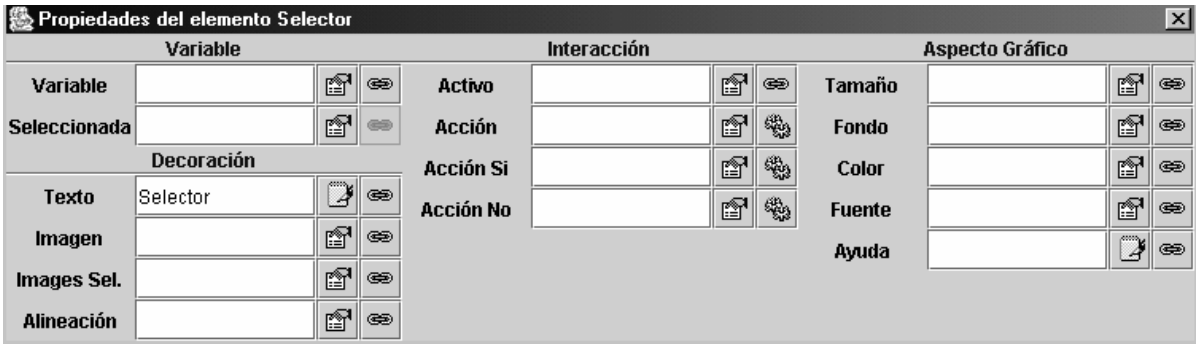


Figura 6.8: Propiedades del elemento *Selector*.

- **Título X**: título mostrado en el eje X.
- **Pos Eje X**: indica la coordenada X del cruce del eje coordenado Y con el X. Sólo tiene efecto cuando la propiedad "Tipo de Ejes" tiene asignado el valor Cartesian3.
- **Tipo Eje X**: lineal o logarítmico en base 10. La constante LINEAR se usa para designar un tipo de eje lineal, mientras que la constante LOG10 se usa para ejes logarítmicos.
- **Malla X**: a esta propiedad se le asigna una variable lógica o las constantes true o false. Cuando esta propiedad toma el valor false no se dibuja una malla en el eje X.
- **Título Y**: título mostrado en el eje Y.
- **Pos Eje Y**: indica la coordenada Y del cruce del eje coordenado X con el Y. Sólo tiene efecto cuando la propiedad "Tipo de Ejes" tiene asignado el valor Cartesian3.
- **Tipo Eje Y**: lineal o logarítmico en base 10. La constante LINEAR se usa para designar un tipo de eje lineal y la constante LOG10 un tipo de eje logarítmico.
- **MallaY**: a esta propiedad se le asigna una variable lógica o las constantes true o false. Cuando esta propiedad toma el valor false no se dibuja una malla en el eje Y.
- **Delta R**: cuando se selecciona un tipo de ejes polares, el valor de esta propiedad indica el paso entre líneas de radio polar constante.
- **Delta Theta**: cuando se selecciona un tipo de ejes polares, el valor de esta propiedad indica el paso entre líneas de ángulo polar constante.
- **Autoescala X**: a esta propiedad se le asigna una variable lógica o las constantes true o false. Cuando esta propiedad toma el valor true, se calcula automáticamente la escala en el eje X.
- **Autoescala Y**: a esta propiedad se le asigna una variable lógica o las constantes true o false. Cuando esta propiedad toma el valor true, se calcula automáticamente la escala en el eje Y.
- **Mínimo X**: menor valor de la coordenada X que resulta visible en el panel.
- **Máximo X**: mayor valor de la coordenada X que resulta visible en el panel.
- **Mínimo Y**: menor valor de la coordenada Y que resulta visible en el panel.
- **Máximo Y**: mayor valor de la coordenada Y que resulta visible en el panel.

6.4. Algunas clases de elementos de tipo *Básicos*

Este tipo de clases no pueden contener otros elementos gráficos. A continuación se describen las características fundamentales de las clases *Boton*, *Selector*, *Deslizador* y *CampoNumerico*.

Boton

Un Boton es un elemento gráfico al que se le pueden asociar acciones. En la Figura 6.7 se muestran las propiedades de los elementos de esta clase. Alguna de sus propiedades más relevantes son:

- **Texto**: texto mostrado por el elemento, pudiendo ser cualquier constante o variable de tipo String.
- **Alineación**: forma de alinear el texto en el elemento.
- **Activo**: el elemento responde a la acción del usuario cuando el valor de esta propiedad es true. Si en esta propiedad se escribe una variable lógica (tipo boolean) el elemento responderá a la acción del usuario cuando la variable tome el valor true.
- **Acción**: la acción que se invoca cuando se pulsa el botón.

Selector

Un Selector es un elemento gráfico que se usa para mostrar y modificar un valor lógico. Alguna de las propiedades más relevantes de estos elementos son (ver Figura 6.8):

- **Variable**: variable lógica (tipo boolean) que se "enlaza" con el selector. Este elemento gráfico escribe en la variable un valor true o false, en función de que esté o no seleccionado.
- **Acción**: acción que se ejecuta siempre que se modifica el valor del elemento Selector.
- **Acción Si**: acción que se ejecuta cuando la variable toma el valor true.
- **Acción No**: acción que se ejecuta cuando la variable toma el valor false. En el caso de que estén asignadas tanto la propiedad Acción Si como Acción No se ejecuta primero la acción asociada a Acción Si y después la asociada a Acción No.
- **Activo**: el elemento está deshabilitado cuando el valor de esta propiedad es false. Si en esta propiedad se escribe una variable lógica (tipo boolean) el elemento se deshabilita cuando la variable tome el valor false. Cuando el elemento está deshabilitado se oscurece.

Deslizador

Un Deslizador es un elemento gráfico que muestra y permite modificar un valor numérico. Sus propiedades se muestran en la Figura 6.9. Alguna de sus propiedades fundamentales son:

- **Variable**: propiedad que permite "enlazar" una variable con el deslizador. De este modo, el valor de esta variable se modifica desplazando el deslizador entre sus dos valores extremos.
- **Formato**: si su valor es no nulo, se visualiza también el valor de la variable en forma de texto.
- **Al pulsar**: acción que se invoca cuando se pulsa el deslizador.
- **Al mover**: acción que se invoca cuando se desplaza el deslizador.
- **Al soltar**: acción que se invoca cuando se suelta el deslizador.

CampoNumerico

Un CampoNumerico es un elemento que muestra y permite modificar un valor numérico. Al modificar el valor mostrado el fondo del elemento cambia de color. Sólo cuando se pulsa la tecla Intro se acepta el nuevo valor y vuelve el fondo del elemento al color original. Si el valor introducido es erróneo (es decir, no puede ser entendido por la propiedad "Formato"), el valor será rechazado y se mostrará el fondo del elemento gráfico en rojo.

Sus propiedades se muestran en la Figura 6.10. Alguna de sus propiedades fundamentales son:

- **Variable**: permite enlazar una variable de tipo double o int con el CampoNumérico. De esta forma, al introducir un nuevo valor numérico y pulsar a Intro, se modifica el valor de la variable.
- **Formato**: formato con que se visualiza el valor de la variable.
- **Acción**: acción que se invoca cuando el valor mostrado por el elemento se modifica.

6.5. Algunas clases de elementos de tipo *Dibujo*

Este tipo de clases no pueden contener otros elementos gráficos. Se ubican en contenedores de las clases PanelDibujo, PanelDibujo3D o PanelConEjes. A continuación se describen las características fundamentales de las clases Particula, Flecha, Imagen, Texto, Traza, Polígono.

Figura 6.9: Propiedades del elemento *Deslizador*.

Figura 6.10: Propiedades del elemento *CampoNumerico*.

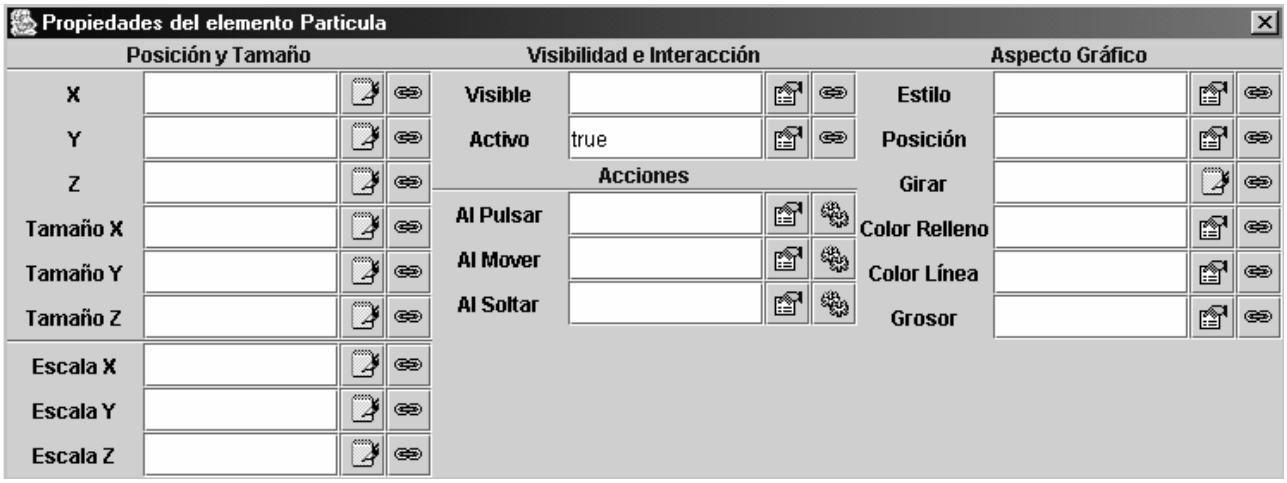


Figura 6.11: Propiedades del elemento *Particula*.

Particula

Este elemento gráfico representa una forma geométrica sencilla. La posición del elemento dentro del contenedor se especifica introduciendo las coordenadas X, Y y Z de su posición. Sólo es necesario especificar la coordenada Z si el elemento se ubica en un contenedor de la clase PanelDibujo3D. Sus propiedades se muestran en la Figura 6.11. Alguna de sus propiedades fundamentales son:

- **Estilo**: permite seleccionar la forma del elemento.
- **X**: coordenada x del elemento.
- **Y**: coordenada y del elemento.
- **Z**: coordenada z del elemento.
- **Activo**: indica si el elemento responde a la acción del usuario.
- **Al pulsar**: acción que se produce cuando el ratón se sitúa sobre el elemento y se presiona su botón.
- **Al mover**: acción que se produce cuando se mueve el objeto con el ratón.
- **Al soltar**: acción que se produce cuando se deja de presionar el botón del ratón.

Flecha

Este elemento gráfico representa un vector interactivo. La posición del elemento dentro del contenedor se especifica dando las coordenadas cartesianas del origen y el tamaño del vector según cada uno de los ejes coordenados. En la Figura 6.12 se muestra su ventana de propiedades. A continuación se describen algunas de sus propiedades:

- **Estilo**: forma gráfica del elemento, pudiendo seleccionar entre la forma de flecha, línea o cajita.
- **X**: coordenada x del origen del vector.
- **Y**: coordenada y del origen del vector.
- **Z**: coordenada z del origen del vector.
- **Tamaño X**: tamaño del vector en el eje x.
- **Tamaño Y**: tamaño del vector en el eje y.
- **Tamaño Z**: tamaño del vector en el eje z.
- **Activo**: indica si el elemento responde a la acción del usuario sobre su extremo.
- **Movible**: indica si el elemento responde a la acción del usuario sobre su origen.
- **Al pulsar**: acción que se invoca cuando el usuario hace clic con el ratón sobre el elemento.
- **Al mover**: acción que se invoca cuando el usuario arrastra el elemento con el ratón.
- **Al soltar**: acción que se invoca cuando se suelta el elemento.

Imagen

Elemento gráfico que muestra una imagen gif o gif animada. El elemento se ubica en el contenedor especificando su posición en coordenadas cartesianas y su tamaño según cada uno de los ejes coordenados. En la Figura 6.13 se muestra su ventana de propiedades. A continuación se describen algunas de sus propiedades:

- **X**: coordenada x del elemento.
- **Y**: coordenada y del elemento.
- **Z**: coordenada z del elemento.
- **Tamaño X**: longitud del elemento en el eje x.
- **Tamaño Y**: longitud del elemento en el eje y.
- **Tamaño Z**: longitud del elemento en el eje z.
- **Activo**: indica si el elemento responde a la acción del usuario.

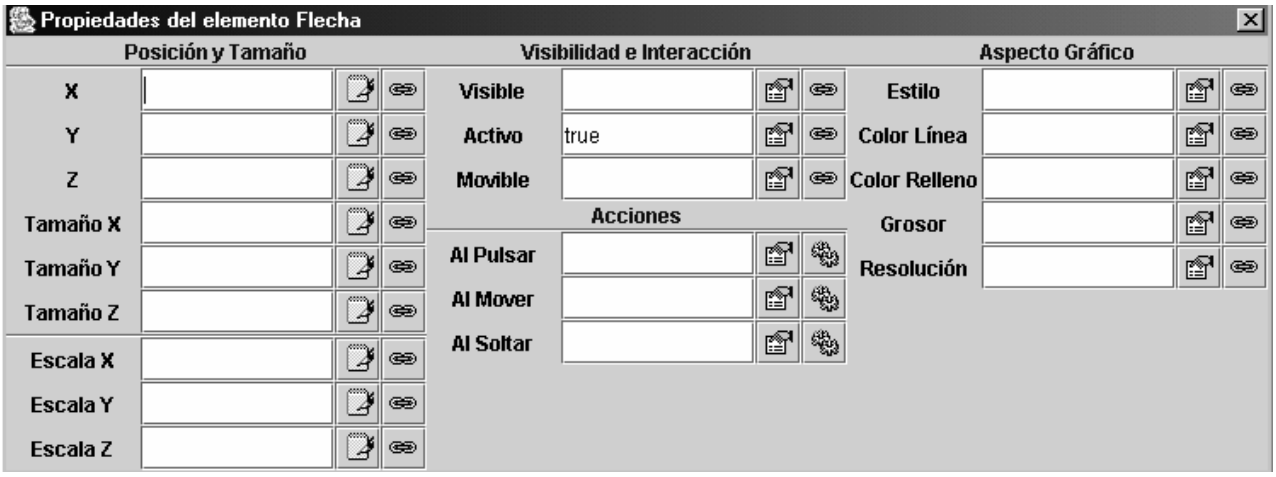


Figura 6.12: Propiedades del elemento *Flecha*.

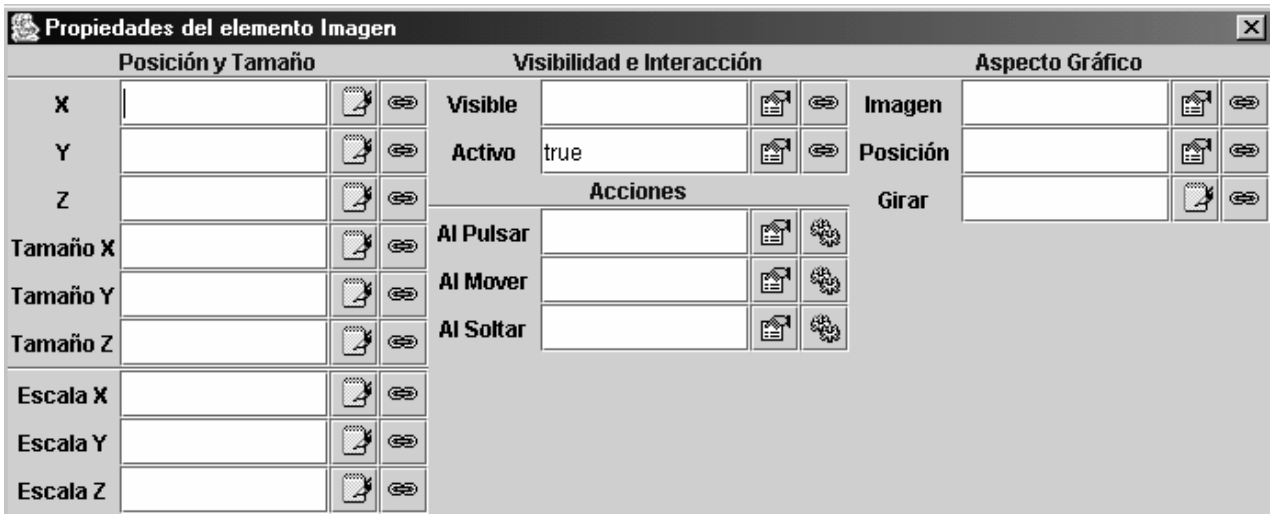


Figura 6.13: Propiedades del elemento *Imagen*.

Figura 6.14: Propiedades del elemento *Texto*.

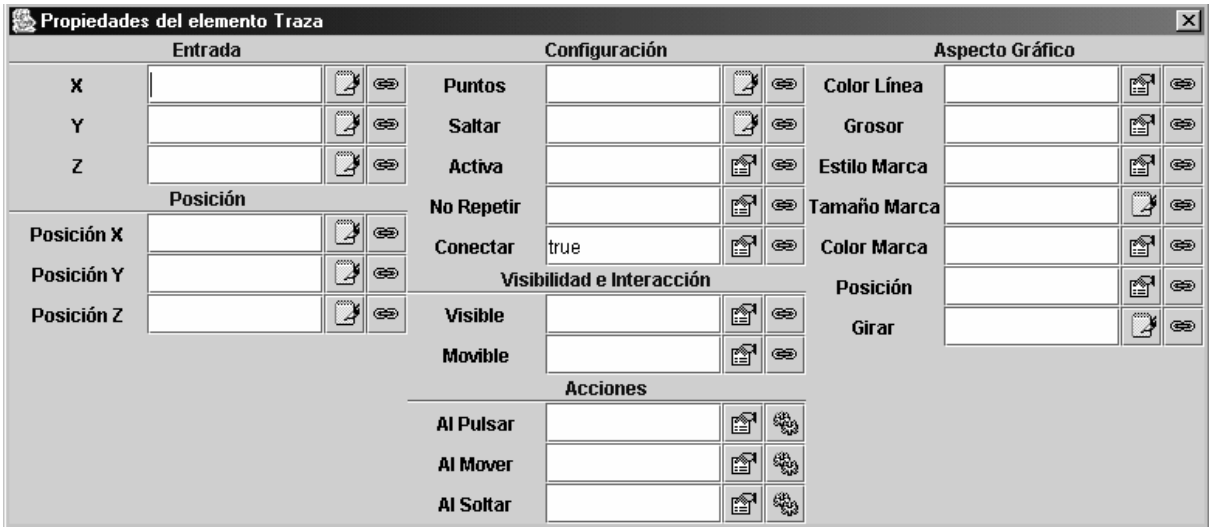


Figura 6.15: Propiedades del elemento *Traza*.

- **Al pulsar**: acción que se invoca cuando el usuario hace clic con el ratón sobre el elemento.
- **Al mover**: acción que se invoca cuando el usuario arrastra el elemento con el ratón.
- **Al soltar**: acción que se invoca cuando se suelta el elemento.
- **Imagen**: Cualquier constante o variable de tipo String. El String indica la ruta hasta el fichero de imagen correspondiente. Puede ser una ruta relativa al directorio de trabajo o una URL (dirección) de Internet. La imagen ha de corresponder a una imagen GIF o GIF animada.

Texto

Elemento gráfico que muestra un texto en unas coordenadas determinadas del contenedor donde está ubicado. En la Figura 6.14 se muestra la ventana de propiedades del elemento. A continuación se describen algunas de sus propiedades:

- **X**: coordenada x del elemento.
- **Y**: coordenada y del elemento.
- **Z**: coordenada z del elemento.
- **Activo**: indica si el elemento responde a la acción del usuario.
- **Al pulsar**: acción que se invoca cuando el usuario hace clic con el ratón sobre el elemento.
- **Al mover**: acción que se invoca cuando el usuario arrastra el elemento con el ratón.
- **Al soltar**: acción que se invoca cuando se suelta el elemento.
- **Texto**: texto a mostrar por el elemento, que puede ser cualquier constante o variable de tipo String.
- **Fuente**: tipo de fuente a usar para el texto del elemento.

Traza

Elemento gráfico que dibuja una serie de puntos en unas determinadas coordenadas del contenedor donde está ubicado. Los puntos se añaden secuencialmente, y pueden visualizarse usando marcadores en cada puntos, segmentos de unión entre puntos o ambos.

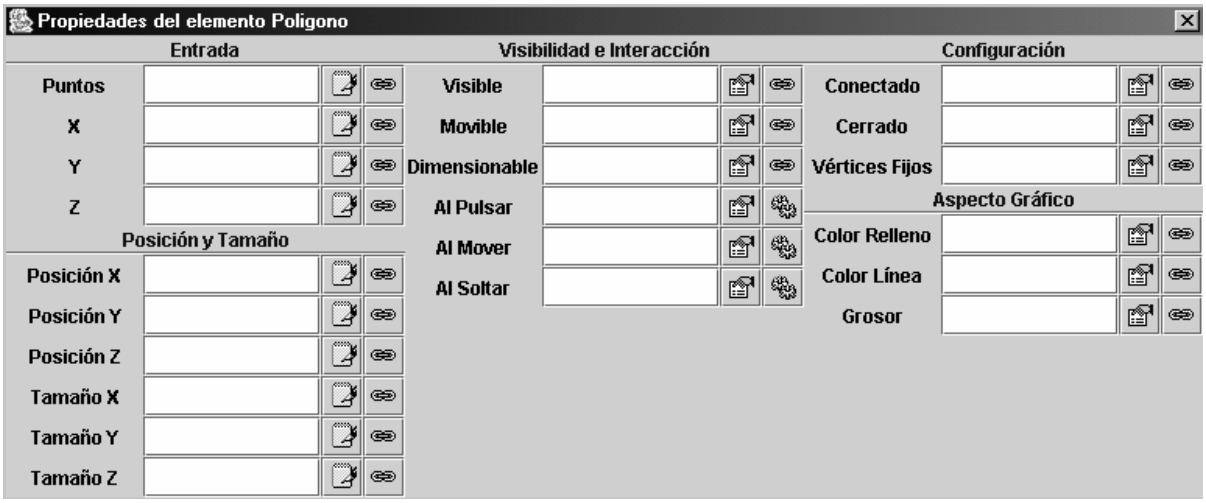


Figura 6.16: Propiedades del elemento *Poligono*.

En la Figura 6.15 se muestra la ventana de propiedades del elemento. A continuación, se describen algunas de sus propiedades:

- **X**: coordenada x del nuevo punto que se añade a la traza.
- **Y**: coordenada y del nuevo punto que se añade a la traza.
- **Z**: coordenada z del nuevo punto que se añade a la traza.
- **Puntos**: número de puntos a dibujar. Si se asigna el valor 0 a la propiedad, se dibujan todos los puntos.
- **Saltar**: número de puntos que no se dibujan hasta que se vuelve a dibujar un punto.
- **Activo**: cuando se le asigna el valor false esta propiedad, no se dibujan puntos. A esta propiedad se le puede asignar una variable lógica. De este modo, cuando la variable tome el valor false no se dibujarán nuevos puntos de la traza.
- **No repetir**: cuando se le asigna el valor true a esta propiedad, no se dibuja un punto si éste tiene las mismas coordenadas que el punto anterior. A esta propiedad se le puede asignar una variable lógica. De este modo, cuando tome la variable el valor true, no se dibujará un punto cuando tenga las mismas coordenadas que el punto anterior.
- **Conectar**: se le asigna una variable lógica o las constantes true o false. Cuando esta propiedad toma el valor false, no se conecta el nuevo punto con el anterior.
- **Al pulsar**: acción que se invoca cuando el usuario hace clic con el ratón sobre el elemento.
- **Al mover**: acción que se invoca cuando el usuario arrastra el elemento con el ratón.
- **Al soltar**: acción que se invoca cuando se suelta el elemento.

Polígono

Elemento gráfico que representa un prisma poligonal o un polígono. Cuando se trabaja en dos dimensiones, la posición del polígono en el contenedor donde está ubicado se especifica dando las coordenadas cartesianas de sus vértices.

En la Figura 6.16 se muestra la ventana de propiedades del elemento. A continuación se describen algunas de sus propiedades:

- **Puntos**: número de vértices del polígono. Puede especificarse dando una constante o variable de tipo int.

- **X**: coordenadas x de los vértices del elemento.
- **Y**: coordenadas y de los vértices del elemento.
- **Z**: coordenadas z de los vértices del elemento.
- **Movible**: si el elemento puede moverse al arrastrar sus vértices.
- **Dimensionable**: si pueden modificarse interactivamente las coordenadas de sus vértices.
- **Al pulsar**: acción que se invoca cuando el usuario hace clic con el ratón sobre el elemento.
- **Al mover**: acción que se invoca cuando el usuario arrastra el elemento con el ratón.
- **Al soltar**: acción que se invoca cuando se suelta el elemento.
- **Conectado**: vector de dimensión igual al número de vértices del polígono de valores lógicos. Un valor lógico true en la posición i del vector indica que el vértice i se conecta con una línea al vértice i-1.
- **Cerrado**: se le asigna a esta propiedad una variable o constante lógica. Cuando está a true, el conjunto de vértices representa un polígono cerrado.
- **Vertices fijos**: vector de dimensión igual al número de vértices del polígono del tipo boolean. Un valor lógico true en la posición i del vector indica que no se pueden modificar interactivamente las coordenadas del vértice i.

Parte III

Casos de estudio

Tema 7

Programación de un osciloscopio virtual con Ejs

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Crear una página de Introducción con sólo texto.
- Diseñar el algoritmo de simulación de un modelo estático y programar dicho modelo usando Ejs.
- El uso de las clases de elementos gráficos: Ventana, PanelDibujo, Panel, Traza y Botón.
- Programar métodos propios e invocarlos desde la vista.

En este tema se describe, de forma tutorial, cómo programar un laboratorio virtual con Ejs. En concreto, cómo programar un *osciloscopio virtual* en el cual puedan visualizarse las figuras de Lissajous[1]. El objetivo de dicho laboratorio es permitir que el alumno pueda modificar interactivamente el valor de las frecuencias de las señales, así como el desfase, observando las correspondientes figuras de Lissajous.

Este laboratorio está disponible en el CD del curso. Se trata del fichero *lissajous.xml*, que está grabado en el directorio *laboratoriosTexto*.

7.1. Las figuras de Lissajous

Las figuras de Lissajous fueron descubiertas por el físico francés Jules Antoine Lissajous (vea la Figura 7.1). Lissajous empleaba sonidos de diferentes frecuencias para hacer vibrar un espejo. La luz reflejada en el espejo trazaba una curva cuya forma dependía de la frecuencia del sonido.

Las figuras de Lissajous se obtienen de la superposición de dos movimientos armónicos perpendiculares:

[1]Las explicaciones dadas en este Tema acerca de la programación del osciloscopio virtual están basadas en el Capítulo 3 de (Esquembre 2002*b*).

Figura 7.1: Jules Antoine Lissajous (1822–1880).

$$x = A \cdot \cos(\omega_1 \cdot t) \quad \text{(movimiento horizontal)} \quad (7.1)$$

$$y = A \cdot \cos(\omega_2 \cdot t + \delta) \quad \text{(movimiento vertical)} \quad (7.2)$$

La trayectoria resultante, $(x(t), y(t))$, depende de la relación de las frecuencias, $\frac{\omega_2}{\omega_1}$, y de la diferencia de fase, δ.

Una de las aplicaciones de las figuras de Lissajous fue determinar la frecuencia de sonidos o señales de radio. Se aplicaba en el eje horizontal de un osciloscopio una señal de frecuencia conocida, y la señal cuya frecuencia se deseaba medir se aplicaba en el eje vertical. La forma de la figura resultante es función del cociente de las dos frecuencias.

A continuación se describen los pasos necesarios para programar el *osciloscopio virtual* con Ejs. Para una mejor comprensión de las explicaciones, se recomienda que los vaya realizando por usted mismo. Para ello, comience por arrancar el entorno de simulación Ejs (según se explica en la Sección 4.5).

7.2. Descripción de la introducción

Al arrancar Ejs, aparece por defecto seleccionado el panel *Introducción*. Dentro de este panel deben crearse las páginas que componen la introducción del laboratorio virtual. Para crear una página, haga clic con el ratón sobre la frase "Pulse para crear una página" (vea la Figura 4.2).

Al hacerlo, se abre una ventana en la cual Ejs pregunta qué nombre se desea dar a la nueva página. Déle un nombre a la página de introducción que está a punto de crear, por ejemplo: *Figuras de Lissajous*. Escriba en la zona de texto de la página lo que desee. Por ejemplo (ver la Figura 7.2):

> *Las figuras de Lissajous se obtienen de la superposición de dos movimientos armónicos perpendiculares.*

Lo que usted escriba en esta página, será traducido por Ejs a código HTML. Para ver el código HTML generado, seleccione *Ver / Ver Código Fuente*.

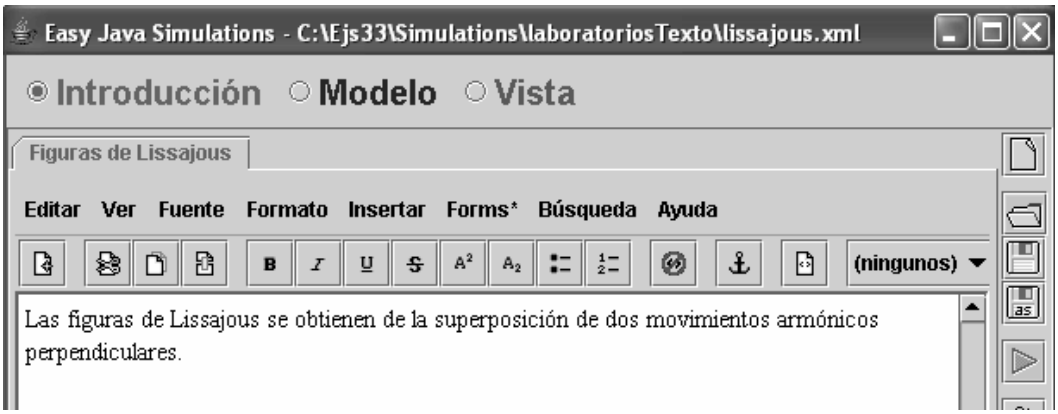


Figura 7.2: Contenido de la página *Figuras de Lissajous*, del panel *Introducción*.

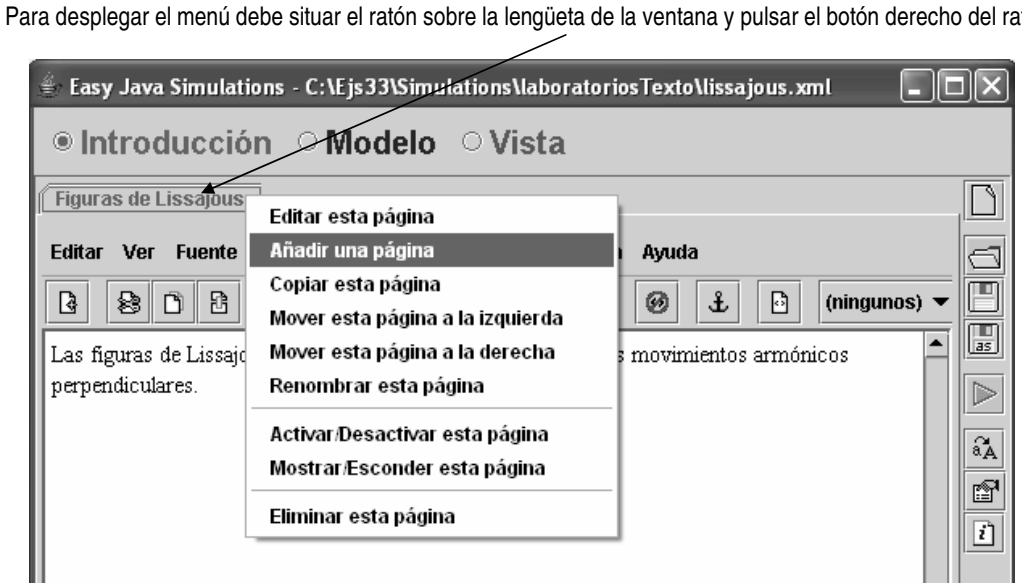


Figura 7.3: Menú para la gestión de las páginas.

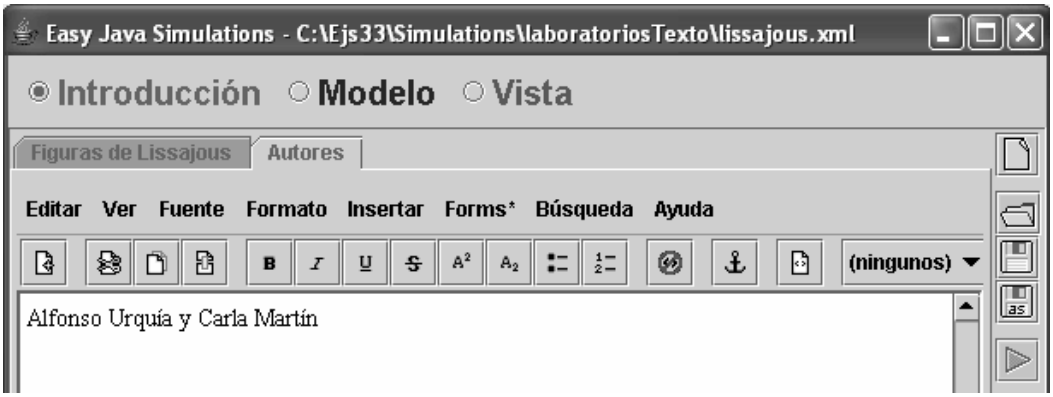


Figura 7.4: El panel *Introducción* contiene dos páginas: *Figuras de Lissajous* y *Autores*.

Inspeccione las diferentes opciones de la barra de herramientas con el fin de familiarizarse con ellas.

Para añadir una segunda página de introducción, debe pulsar el botón derecho del ratón sobre la lengüeta de la página de introducción que ha creado anteriormente (*Figuras de Lissajous*). Se despliega un menú, en el cual puede escoger realizar operaciones sobre las páginas (vea la Figura 7.3).

Entre estas operaciones está la creación de páginas nuevas. Para ello debe hacer clic sobre la opción *Añadir una página*. De a esta segunda página de introducción el nombre *Autor*, y escriba su nombre en el espacio de texto de la página. En la Figura 7.4 se muestra un ejemplo.

7.3. El algoritmo de la simulación

Como paso previo a programar el modelo de un laboratorio virtual empleando Ejs, deben seguirse los dos pasos siguientes:

1. Clasificar las variables del modelo en conocidas y desconocidas. En el caso del modelo del osciloscopio, que esta compuesto por las Ecuaciones (7.1) y (7.2):
   - Las variables conocidas son los parámetros (A, ω_1, ω_2, δ) y la variable tiempo (t).
   - Las variables desconocidas son las dos variables algebraicas: x, y.
2. Aplicar el algoritmo de asignación de la causalidad computacional, con el fin de decidir cómo deben ordenarse las ecuaciones y qué variable debe evaluarse de cada ecuación. La aplicación de dicho algoritmo es trivial en el caso del modelo del osciloscopio:

$$[x] = A \cdot \cos(\omega_1 \cdot t) \quad (7.3)$$

$$[y] = A \cdot \cos(\omega_2 \cdot t + \delta) \quad (7.4)$$

En la Figura 5.1 se mostró el algoritmo de la simulación de Ejs, que es el que emplea Ejs para ejecutar cualquier laboratorio virtual.

En la Figura 7.5 se muestra una posible forma de programar el osciloscopio virtual. Como se explicará más adelante, los parámetros $maximo$, $minimo$ y n van a emplearse en la definición de la vista:

- Las variables $maximo$ y $minimo$ son de tipo *double*, y van a emplearse en la definición de los extremos de escala del osciloscopio.
- La variable n, de tipo entero (*int*), representa el número de puntos de que va a constar la traza de la figura de Lissajous. Es decir, representa la "memoria" de la pantalla del osciloscopio.

El parámetro $deltaTiempo$ se emplea para incrementar la variable $tiempo$. Es decir, el valor de este parámetro determina el tamaño del paso de avance en el tiempo en la simulación.

Esta situación se produce frecuentemente: las variables del laboratorio virtual (que está descrito mediante algoritmos codificados en los diferentes paneles) no sólo son las variables del modelo matemático (descrito mediante ecuaciones) sino además otras variables que se emplean:

- *En los métodos numéricos.* En el caso del osciloscopio, el parámetro $deltaTiempo$ no interviene en el modelo matemático: se emplea para especificar el tamaño del paso de avance en el tiempo de la simulación. Esto significa que la simulación consistirá en resolver el modelo (es decir, calcular el valor de sus variables) en los instantes de tiempo 0, $deltaTiempo$, $2 \cdot deltaTiempo$, $3 \cdot deltaTiempo$, ...
- *En la definición de las acciones y de la vista.* En este caso, los parámetros $maximo$, $minimo$ y n se definen para personalizar las propiedades de la vista.

A continuación se explica cómo declarar e inicializar las variables del modelo en Ejs.

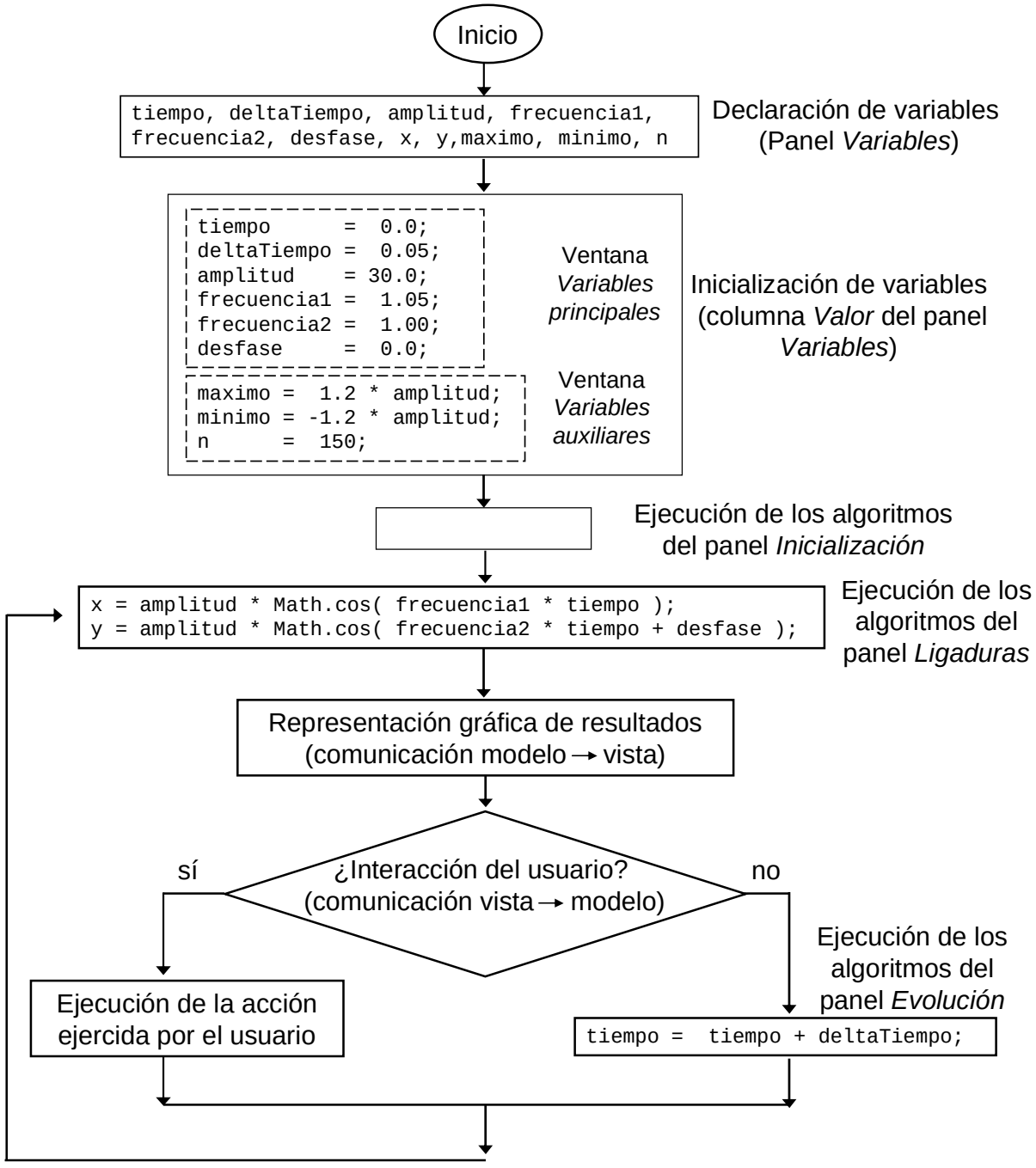


Figura 7.5: Algoritmo de la simulación del osciloscopio virtual.

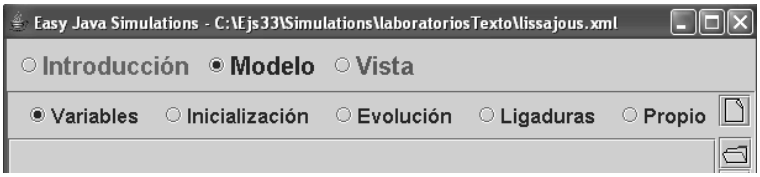


Figura 7.6: Selección del panel *Variables*.

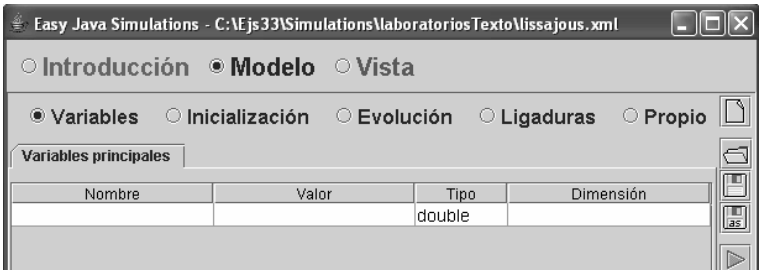


Figura 7.7: Página *Variables principales*, dentro del panel *Variables*.

7.4. Declaración e inicialización de las variables

La declaración de las variables se realiza en el panel *Variables*. Para acceder a este panel, seleccione *Modelo*, en la regleta superior de la interfaz de Ejs, y *Variables* en la regleta situada debajo (vea la Figura 7.6).

Dentro del panel *Variables* pueden crearse una o varias páginas, en las cuales se realiza la declaración de las variables y, opcionalmente, su inicialización. En el caso del osciloscopio virtual van a definirse dos páginas:

– En una página, a la que se dará el nombre *Variables principales*, se declaran las variables que intervienen en el modelo matemático y las variables necesarias para su resolución numérica:

Modelo matemático	Modelo en Ejs
t	tiempo
x	x
y	y
A	amplitud
ω_1	frecuencia1
ω_2	frecuencia2
δ	desfase
	deltaTiempo

– En la segunda página, se declaran las variables empleadas para la definición de la vista: $maximo$, $minimo$, n. A esta segunda página se le asignará el nombre: *Variables auxiliares*.

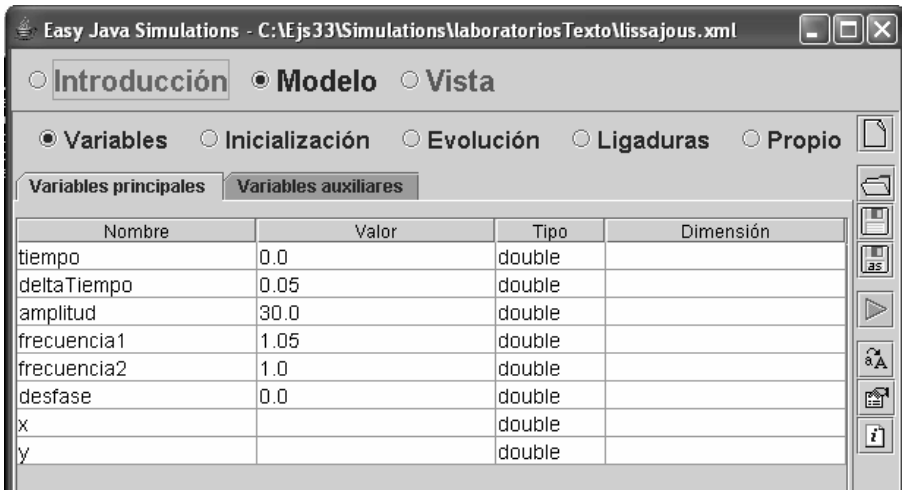


Figura 7.8: Variables principales del modelo.

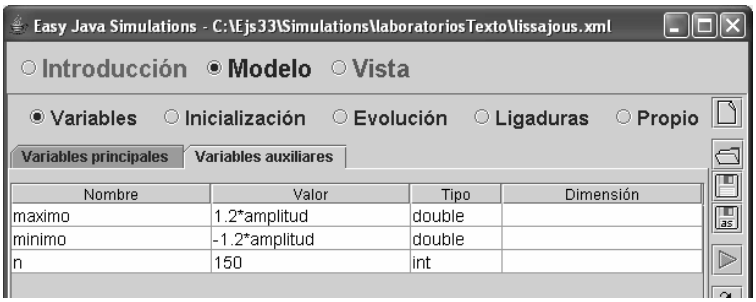


Figura 7.9: Variables auxiliares del modelo.

Creación de la página *Variables principales*. Para crear una nueva página de definición de variables, pulse con el ratón sobre la frase "Pulse para crear una página", que aparece en el panel *Variables* .

A consecuencia de ello, se abre una ventana, en la que debe asignar un nombre a la página de variables que está a punto de crear. Déle el nombre *Variables principales*. Una vez hecho esto, la interfaz de Ejs presenta el aspecto mostrado en la Figura 7.7.

Creación de la página *Variables auxiliares*. Para añadir una nueva página de definición de variables, debe pincharse con el botón derecho del ratón sobre la lengüeta de la página de variables ya existente y a continuación, en el menú desplegable que aparece (como el mostrado en la Figura 7.3), haga clic con el ratón sobre *Añadir una página*.

Se abre una ventana en la cual debe especificar el nombre de la nueva página de definición de variables. Asigne a esta nueva página que está creando el nombre *Variables auxiliares*.

Declaración e inicialización. En las Figura 7.8 y 7.9 se muestran la declaración e inicialización de las variables en las ventanas *Variables principales* y *Variables auxiliares* respectivamente. El significado de cada uno de los campos se explicó en la Sección 5.4.

El panel *Inicialización*. En el ejemplo del osciloscopio virtual, se ha asignado un valor inicial (en la columna *Valor*) a cada una de las variables . El modelo ha quedado convenientemente inicializado y por ello no es preciso introducir ninguna información en el panel *Inicialización*.

7.5. Programación del modelo

Evolución

De acuerdo con el algoritmo de la simulación del osciloscopio virtual mostrado en la Figura 7.5, sólo hay una ecuación de evolución:

$$tiempo = tiempo + deltaTiempo \tag{7.5}$$

El modelo del osciloscopio no contiene ecuaciones diferenciales, y por ese motivo el usuario debe programar explícitamente el avance en el tiempo de la simulación.

Para definir la ecuación de evolución (7.5), haga clic sobre el subpanel con el letrero *Pulse para crear una página*.

Se abre un ventana, en la cual debe especificarse el nombre que se asigna a la nueva página que se va a crear. Por ejemplo, dele a la página el nombre *Avance en el tiempo*. Escriba la Ecuación (7.5) tal como se muestra en la Figura 7.10. Observe que la ecuación finaliza con un punto y coma (;).

Deje el botón *Arranque* tal como está por defecto: activado.

Ligaduras

Las ecuaciones de ligadura del modelo del osciloscopio virtual son (7.1) y (7.2). Deben satisfacerse no sólo durante la evolución temporal de sistema, sino también en caso de que el usuario realice interactivamente (es decir, durante la ejecución de la simulación) cualquier cambio en el valor de algunos de los parámetros (la amplitud, las frecuencia, el desfase, etc.).

Para escribir los algoritmos de ligadura es preciso hacer clic sobre el botón *Ligaduras*. La interfaz de Ejs muestra un panel que contiene la frase *Pulse para crear una página*. Haga clic con el ratón sobre esta frase, con el fin de crear una página en la cual describir las ecuaciones (7.1) y (7.2). Asigne a esta página el nombre *Cálculo de la posición*. Una vez escritas las dos ecuaciones de ligadura, la ventana tiene el aspecto mostrado en la Figura 7.11.

7.6. Programación de la vista

En la Figura 7.12 se muestra el panel para la descripción de la vista. La vista se crea añadiendo elementos gráficos, los cuales pueden estar alojados unos dentro de otros, con lo cual la vista tiene estructura de árbol. Los elementos gráficos que pueden contener a otros (hasta un máximo de 5) se denominan *contenedores*. Esta estructura de árbol se define en la ventana superior izquierda del panel, que tiene el letrero *Árbol de elementos* (vea la Figura 7.12).

En la parte derecha del panel se encuentran las clases de elementos gráficos. . Se encuentran en una ventana que tiene el letrero *Elementos para la vista*, la cual, a su vez, se encuentra subdividida en tres ventanas (vea la Figura 7.12):

- *Contenedores*.
- *Básicos*.
- *Dibujo*.

Inspeccione usted mismo el contenido de estas tres ventanas. Observe que situándose con el ratón sobre cualquiera de los elementos obtiene un mensaje con información acerca de su finalidad.

Para añadir a la vista un objeto de una determinada clase debe:

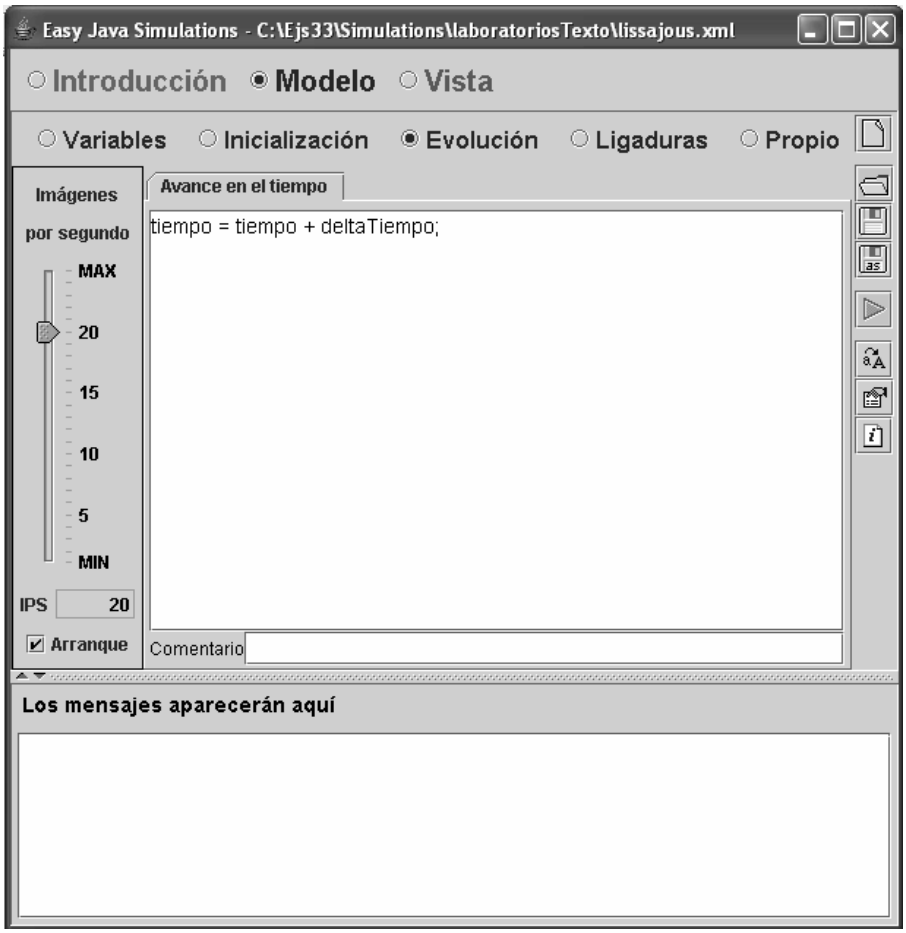


Figura 7.10: Descripción de la expresión que define el avance en el tiempo.

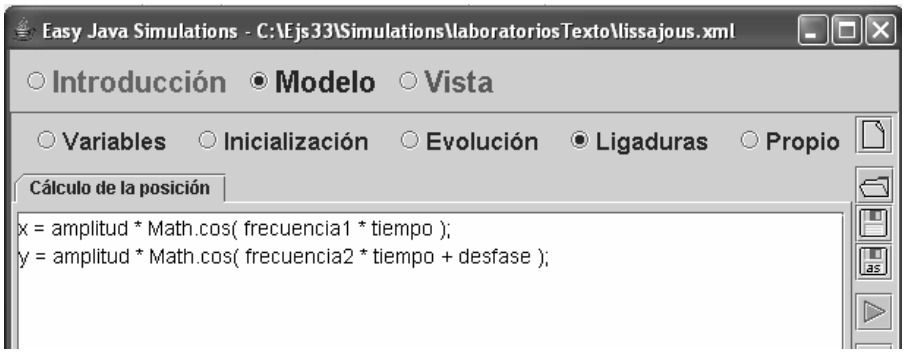


Figura 7.11: Ecuaciones de ligadura.

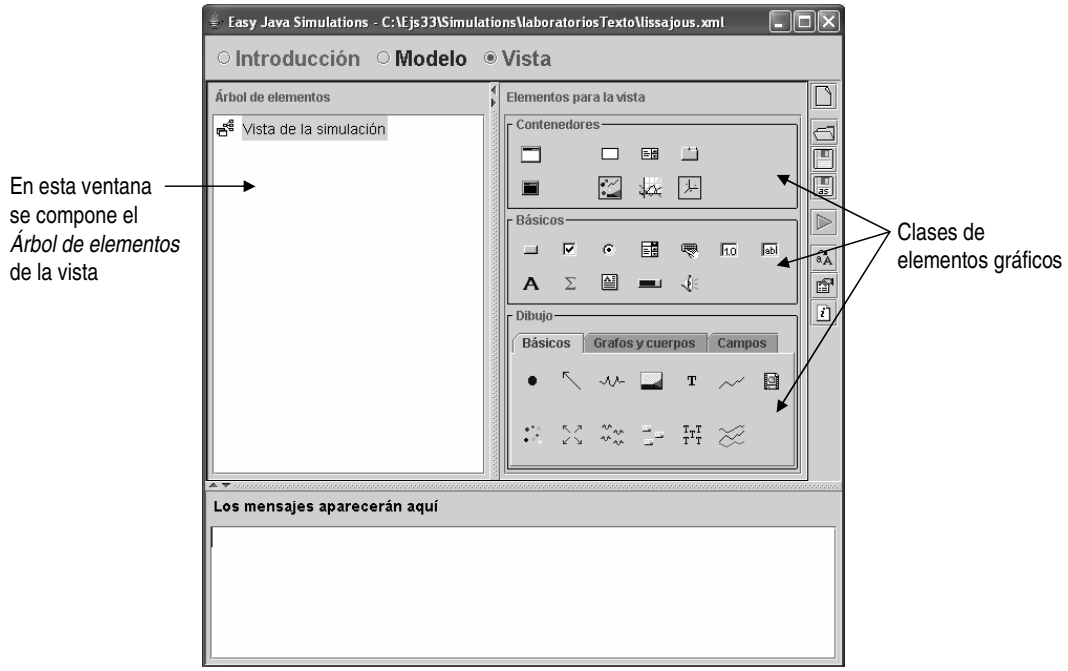


Figura 7.12: Panel para la descripción de la vista.

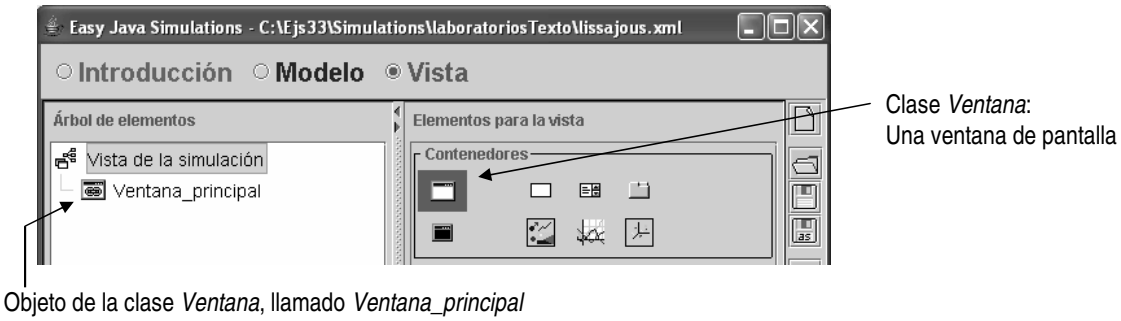


Figura 7.13: Árbol de elementos.

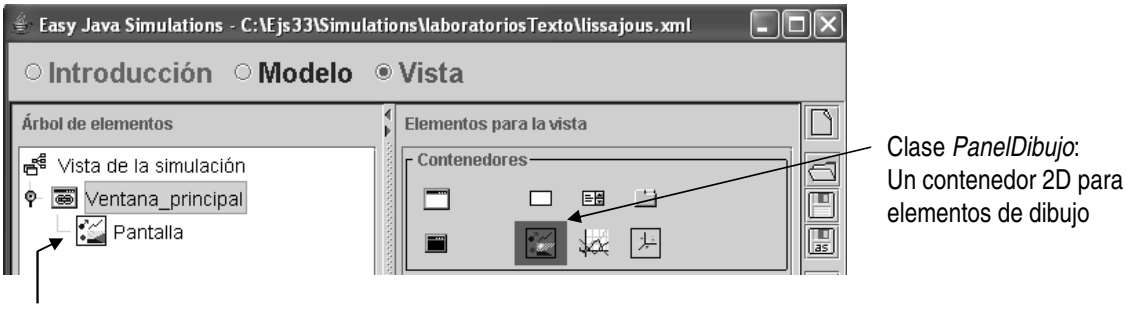


Figura 7.14: Árbol de elementos.

1. *Seleccionar la clase de objeto gráfico*. Para ello, debe hacer clic con el ratón sobre el icono de la clase de elemento gráfico en cuestión. Este estará situado dentro de la ventana *Elementos para la vista*. Al hacer esto, el icono queda inscrito en un recuadro oscuro, que señaliza que esa clase de elemento ha sido seleccionada y que a continuación van a crearse objetos gráficos de esa clase.
2. *Definir los objetos de la clase anteriormente seleccionada*. Para ello, debe hacer clic con el ratón sobre el objeto gráfico dentro del cual desee ubicar el objeto que está a punto de crear. Puede crear tantos objetos como desee. Deberá hacerlo en la ventana *Árbol de elementos*. Observe que al situar el ratón sobre cualquier punto de esta ventana, el cursor adquiere la forma de una varilla mágica.

   Al crear un nuevo objeto, debe definir el valor de determinadas propiedades, tales como su nombre y su posición espacial dentro del contenedor en el que está ubicado.

   Al definir los nombres, debe tenerse en cuenta que *los nombres de variables, métodos y elementos de la vista deben ser únicos en toda la simulación*.

A continuación, se describe cómo crear la vista del osciloscopio virtual.

Observe que en el árbol de elementos hay siempre un elemento raíz, denominado *Vista de la simulación*, dentro del cual se ubican todos los objetos que componen la vista.

La vista del osciloscopio virtual va a consistir en una única ventana, en la cual hay una pantalla (donde se representa x frente a y) y los controles para la selección de las frecuencias y el desfase.

Por tanto, el primer paso es definir esta ventana. Para ello, hay que ubicar un elemento de la clase *Ventana* dentro del objeto *Vista de la simulación* (vea la Figura 7.13). Los objetos de la clase ventana son contenedores, dentro de cada uno de los cuales se pueden ubicar hasta cinco objetos. La definición del objeto se realiza de la forma siguiente:

1. Haga clic sobre la clase *Ventana*.
2. Haga clic sobre la frase *Vista de la simulación*. Con ello está indicando que el objeto que está a punto de crear (de la clase *Ventana*) debe ubicarse dentro del objeto raíz de la vista.

   Al hacerlo, se abre una ventana, en la cual debe escribir el nombre del nuevo objeto. Dele el nombre *Ventana_principal*.

A continuación, debe añadirse un objeto de una clase que sea capaz de albergar elementos gráficos: un objeto de la clase *PanelDibujo* (vea la Figura 7.14):

1. Haga clic sobre el icono de la clase *PanelDibujo*, lo cual indica a Ejs que van a crearse objetos de esta clase.
2. A continuación, haga clic con el ratón sobre la frase *Ventana_principal*. Al hacer esto, se está ubicando un objeto del tipo *PanelDibujo* dentro del objeto contenedor *Ventana principal*.

   Es preciso asignar valor a los siguientes parámetros del objeto que está creando:
   - *Nombre del objeto*. Se abre una ventana en la cual debe escribir el nombre del objeto que está creando. Llámelo *Pantalla*.
   - *Posición del objeto Pantalla dentro del contenedor Ventana_principal*. Se abre otra pantalla, en la cual debe indicarse la posición del objeto. Se ofrecen 5 posibilidades (recuerde que un contenedor puede contener hasta 5 objetos): arriba, abajo, izquierda, derecha y centro. Acepte la opción por defecto: centro. El árbol de elementos tiene ahora el aspecto mostrado en la Figura 7.14

Situándose con el ratón sobre el objeto recién creado (es decir, sobre la palabra *Pantalla*) y pulsando el botón derecho, se despliega un menú para la configuración del objeto. Vea la Figura 7.15.

Seleccionando *Propiedades* en el *Menú para Pantalla*, se abre una pantalla como la mostrada en la Figura 7.16. Los valores de las propiedades que se muestran en la Figura 7.16 son los que aparecen por defecto.

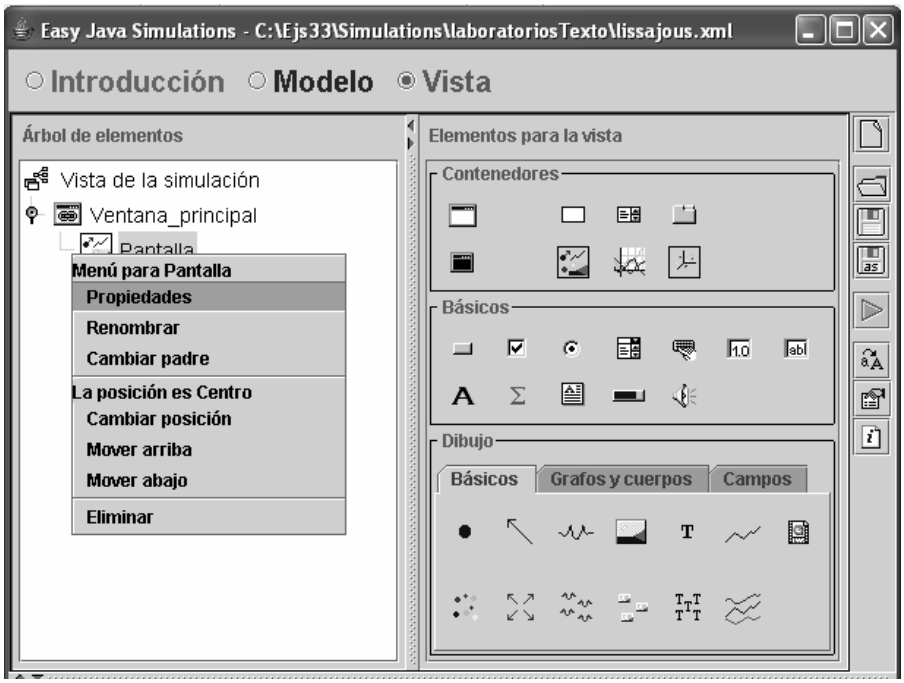


Figura 7.15: Menú del objeto *Pantalla*.

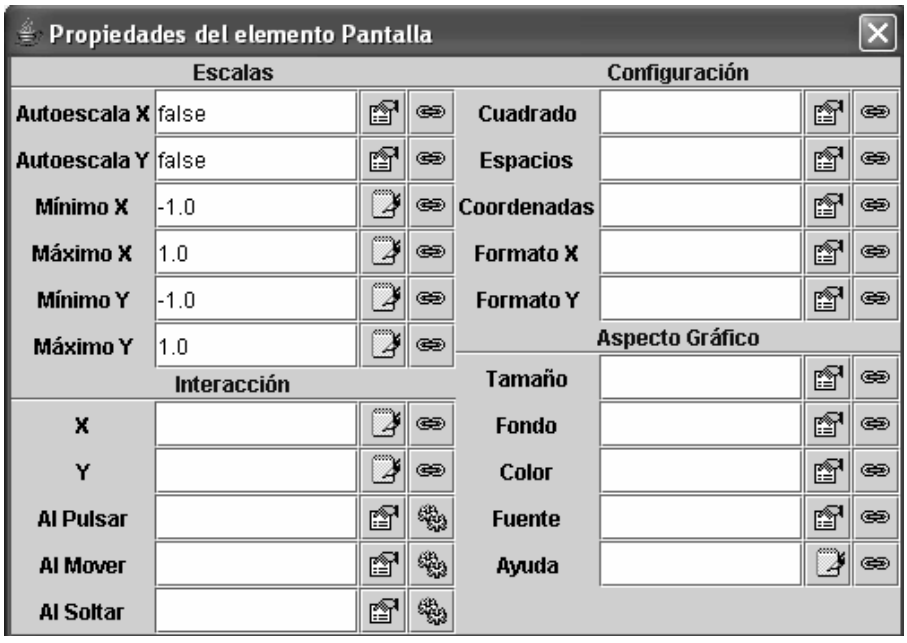


Figura 7.16: Menú de propiedades del objeto *Pantalla* (con valores por defecto).

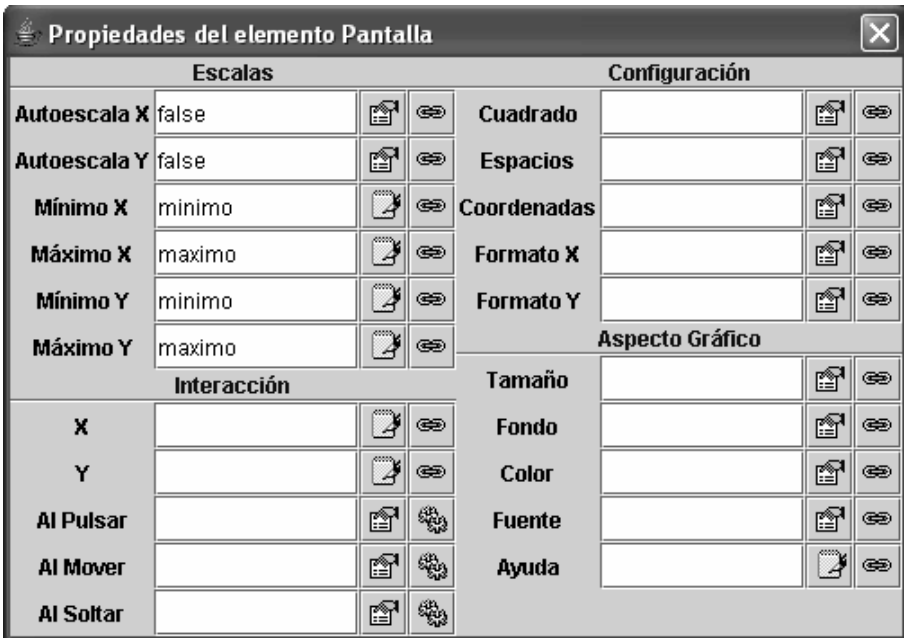


Figura 7.17: Menú de propiedades del objeto *Pantalla* una vez configurado.

Figura 7.18: Árbol de elementos.

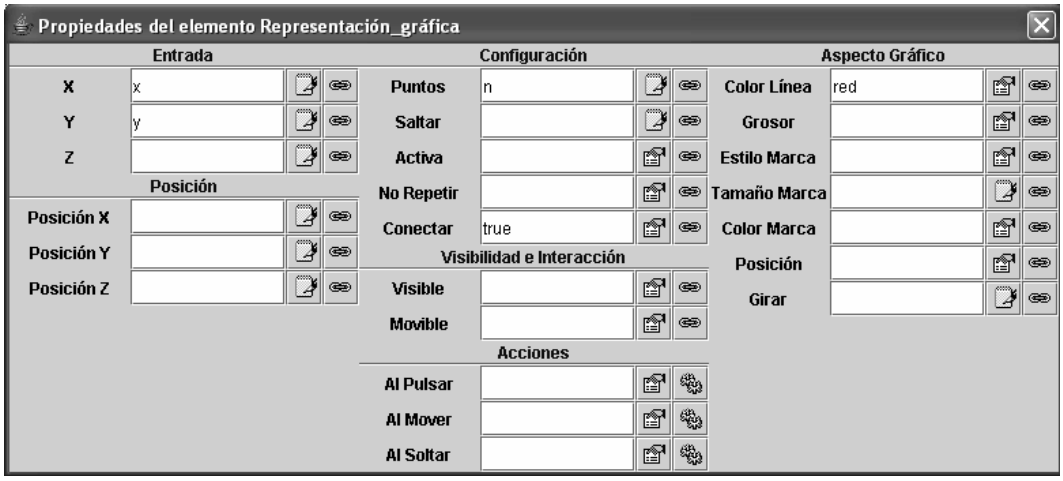


Figura 7.19: Propiedades del objeto *Representación_gráfica*.

Es posible asignar a una propiedad, o bien un determinado valor constante, o bien asociarla con una variable del modelo, de modo que en cada instante la propiedad y la variable tomen el mismo valor.

En este ejemplo, los valores máximo y mínimo de la gráfica van a asociarse con las variables del modelo maximo y minimo respectivamente. Pueden seguirse los pasos siguientes:

1. Con el fin de asociar el valor mínimo del eje horizontal de la gráfica con la variable minimo, pulse el icono con eslabones de cadena dibujados que está situado a la derecha de la propiedad *Mínimo X*. Se abre una ventana que muestra todas las variables del modelo. Haciendo doble clic sobre la variable minimo, queda asociada la propiedad *Mínimo X* con la variable minimo.
2. A continuación, proceda de la misma forma, asociando la propiedad *Mínimo Y* con la variable minimo.
3. De forma completamente análoga, asocie las propiedades *Máximo X* y *Máximo Y* con la variable maximo.

Una vez hechas las cuatro asociaciones propiedad – variable anteriormente descritas, la página de propiedades del objeto *Pantalla* tiene el aspecto mostrado en la Figura 7.17.

Finalmente, una vez definido el objeto *Pantalla* (es decir, un contenedor 2D de objetos de dibujo) es preciso ubicar en dicho contenedor un objeto de dibujo, que defina qué es lo que debe dibujarse:

1. Seleccione la clase *Traza*. Se trata del sexto icono de la primera fila del panel *Dibujo – Básicos*. Posicionando el ratón sobre él, se obtiene información acerca de su finalidad: *Traza: una secuencia de puntos*.
2. Haga clic con el ratón sobre el objeto *Pantalla*. Con ello se crea un objeto del tipo *Traza* y se ubica dentro del contenedor *Pantalla*. Dele al objeto del tipo Traza que está creando el nombre *Representación_gráfica*. Una vez hecho esto, el árbol de elementos tendrá el aspecto mostrado en la Figura 7.18.

Para asignar valor a las propiedades del objeto *Representación_gráfica*, sitúese con el ratón sobre él y pulse el botón derecho. Con ello se abre el menú para el objeto. Seleccione *Propiedades*. Realice las siguientes asociaciones entre propiedades y variables:

- *Indique qué variables deben representarse*. Para ello, enlace la propiedad *X* con la variable x y la propiedad *Y* con la variable y.
- *Indique cuántos puntos deben dibujarse*. Para ello, enlace la propiedad *Puntos* con la variable n. De esta forma, al comenzar la simulación, Ejs dibuja el valor inicial de las variables x e y, es decir, el punto $(x\,(t=0)\,, y\,(t=0))$. A continuación, avanza un paso en el tiempo, Δt, recalcula x e y, y dibuja el correspondiente punto: $(x\,(t=\Delta t)\,, y\,(t=\Delta t))$. De esta forma, dibuja en la gráfica n puntos: $(x\,(t=0)\,, y\,(t=0))$, $(x\,(t=\Delta t)\,, y\,(t=\Delta t))$, $\ldots$, $(x\,(t=(n-1)\cdot\Delta t)\,, y\,(t=(n-1)\cdot\Delta t))$. Una vez dibujados estos primeros n puntos,por cada nuevo punto que Ejs representa, borra el punto más antiguo, de tal forma que en todo momento la gráfica consta de n puntos.
- *Escoja el color de la línea*. Puede definir que el color de la línea sea rojo, haciendo clic con el ratón sobre el icono, situado a la derecha de la propiedad *Color línea*, que tiene dibujado una mano sujetando un papel.

Una vez realizados estos cambios, la ventana de propiedades tiene el aspecto mostrado en la Figura 7.19.

Ya puede ejecutar el laboratorio virtual. Para ello, haga clic sobre el botón de ejecución de la simulación (el botón con el triángulo verde dibujado). En respuesta, se abre una ventana en la que se representa la evolución temporal de una figura de Lissajous (vea la Figura 7.20).

Observará que tal como se ha definido el laboratorio virtual hasta este punto, el usuario no tiene ninguna posibilidad de interacción con la simulación. A continuación, se describe cómo añadir al laboratorio virtual algunas capacidades interactivas.

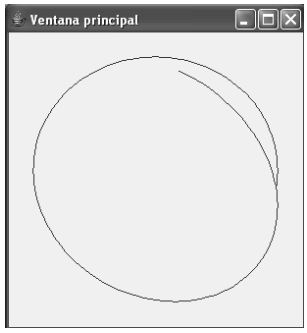


Figura 7.20: Traza de una figura de Lissajous.

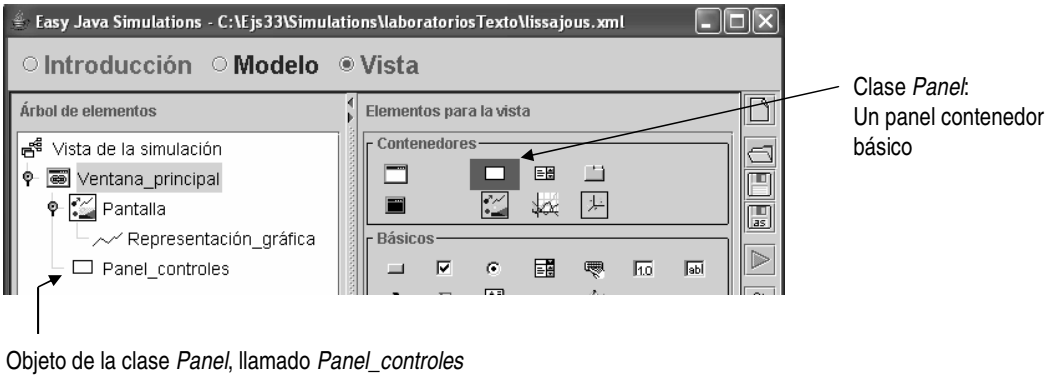


Figura 7.21: Creación del objeto *Panel_controles*.

7.7. Programación de las capacidades interactivas

En esta Sección se explica cómo dotar al osciloscopio virtual de las dos capacidades interactivas siguientes:

- Van a añadirse algunos botones que permitan al usuario seleccionar determinadas frecuencias y desfases, de entre un conjunto predeterminado de ellas, las cuales dan lugar a figuras de Lissajous vistosas. Cada botón corresponderá con una determinada selección de las frecuencias y el desfase.
- Se colocarán casillas numéricas en las cuales el usuario podrá escribir el valor de las frecuencias y del desfase de las figuras que desea visualizar.

En primer lugar, debe crear un objeto de la clase *Panel* y ubicarlo dentro de *Ventana_principal*. El propósito de este nuevo objeto de la clase *Panel* es contener en su interior:

- Los tres botones: A, B y C.
- Las casillas numéricas para seleccionar las dos frecuencias y el desfase.

Para ello:

1. Haga clic sobre el icono de la clase *Panel*.
2. Haga clic sobre la palabra *Ventana_principal*. Llame *Panel_controles* a este nuevo objeto de la clase *Panel*, y sitúelo en la posición izquierda (de este modo, los controles quedarán situados a la izquierda de la pantalla).

Figura 7.22: Ventana para la definición de métodos "propios" del usuario.

Una vez hecho esto, el árbol de elementos gráficos de la vista tiene el aspecto mostrado en la Figura 7.21.

Botones para la selección de determinadas figuras

Para añadir un botón que realice una acción, tal como asignar determinados valores a las frecuencias y al desfase, es preciso realizar dos tareas:

1. *Programar la acción a realizar cuando se pulse cada botón.* Deben programarse métodos en lenguaje Java (uno por cada botón) que realicen las acciones deseadas. La programación de estos métodos forma parte de la definición del modelo.
2. *Incluir los tres botones en la vista y asociarle a cada uno su método.* De este modo, cuando se haga clic sobre un botón se ejecutará el método asociado, con lo cual se realizará la correspondiente acción.

Se pretenden programar los tres botones mostrados a continuación:

Botón	ω_1	ω_2	Δt	n
A	0.06981	0.08744	1	2000
B	0.19198	0.24443	1	2000
C	0.54105	0.38397	1	300

La **programación de los métodos** forma parte de la definición del modelo. Pulse el botón *Modelo* y a continuación el botón *Propio*. La interfaz de Ejs adquiere al aspecto mostrado en la Figura 7.22.

Para crear una página en la que definir un método, haga clic con el ratón sobre la frase *Pulse para crear una página*. Debe asignar un nombre a la nueva página. Por defecto, este nombre asignado a la página coincidirá con el nombre del método que va a definir (salvo quizá que el nombre del método comenzará con minúsculas).

Llame *A* a la página y *a* al método. Este método asignará a las variables los valores correspondientes al Botón A. En la Figura 7.23 se muestra la definición del método.

De forma análoga, se definen otras dos páginas:

– El método *b* en la página *B*.
– El método *c* en la página *C*.

Estos dos métodos definen las acciones que serán asociadas al Botón B y al C respectivamente. En la Figura 7.24 se muestra el aspecto de la ventana tras definir los tres métodos.

Una vez definidos los métodos, puede procederse a la **definición de los botones**. Esto debe hacerse en la vista. Para ello (ver la Figura 7.25):

1. Primeramente, es preciso definir un nuevo objeto de la clase *Panel*, dentro del cual se ubicarán los tres botones. A este nuevo objeto se le asigna el nombre de *Panel botones*, y se sitúa en la posición *Arriba*. La posición de un objeto es su posición dentro del contenedor en el que se encuentra. Así pues, el objeto *Panel_botones* está situado dentro del contenedor *Panel_controles*, en la parte superior de éste.
2. Seguidamente, se definen tres objetos del tipo *Botón* y se ubican dentro del contenedor *Panel_botones*. A los objetos se les asigna los nombres *A*, *B* y *C*, y las posiciones *Arriba*, *Centro* y *Abajo* respectivamente.

Finalmente, debe realizarse la **asociación de los botones con las acciones**. Situando el ratón sobre el objeto *A* y pulsando el botón derecho del ratón, se abre la ventana de propiedades del elemento A. A la derecha de la propiedad *Acción* hay dos botones. Uno de ellos tiene dibujado dos ruedas dentadas engranadas. Haciendo clic sobre este botón se abre una ventana en la que se muestran todas las acciones que se encuentran disponibles en el modelo. Haciendo doble clic sobre *a( )*, se asocia el método *a* a la acción del botón. En la Figura 7.26 se muestra la ventana de propiedades del elemento *A*.

De forma completamente análoga, deben asociarse los métodos *b* y *c* con las acciones de los botones *B* y *C* respectivamente.

Ejecutando el modelo, se observa que la vista del osciloscopio virtual tiene la apariencia mostrada en la Figura 7.27.

Inclusión de casillas numéricas

En este apartado se explica cómo añadir a la vista del osciloscopio virtual casillas numéricas, en las cuales el usuario pueda escribir los valores de las dos frecuencias y el desfase de las figuras de Lissojous que desee visualizar.

Para ello, pueden seguirse los pasos siguientes (ver la Figura 7.28):

1. Crear un nuevo objeto de la clase *Panel*, ubicándolo dentro del contenedor *Panel_controles*, en la posición *Abajo*. A este nuevo objeto se le llama *Panel_casillas*.
2. Crear tres objetos de la clase *CampoNumerico* y ubicarlos dentro del contenedor *Panel_casillas*. Se asigna a los tres objetos los nombres *Frecuencia1*, *Frecuencia2* y *Desfase*. Son situados en las posiciones *Arriba*, *Centro* y *Abajo* respectivamente.

Finalmente, es preciso configurar los elementos *Frecuencia1*, *Frecuencia2* y *Desfase*, de modo que cada uno de ellos quede enlazado a la correspondiente variable del modelo.

Sitúese sobre *Frecuencia1* y pulse el botón derecho del ratón. Se despliega un menú (*Menú para Frecuencia1*) en el cual debe seleccionar *Propiedades*. Se abre una ventana (*Propiedades del elemento Frecuencia1*) en la cual debe introducirse la información mostrada en la Figura 7.29. Ésta es:

– El valor de la propiedad *Variable* indica qué variable del modelo está enlazada con el botón. Pulsando el botón con los eslabones dibujados, que se encuentra a la derecha de dicha propiedad, se abre una ventana en la que se muestran todas las variables del modelo. Haga doble clic sobre la variable frecuencia1, para enlazar el elemento gráfico con esta variable.

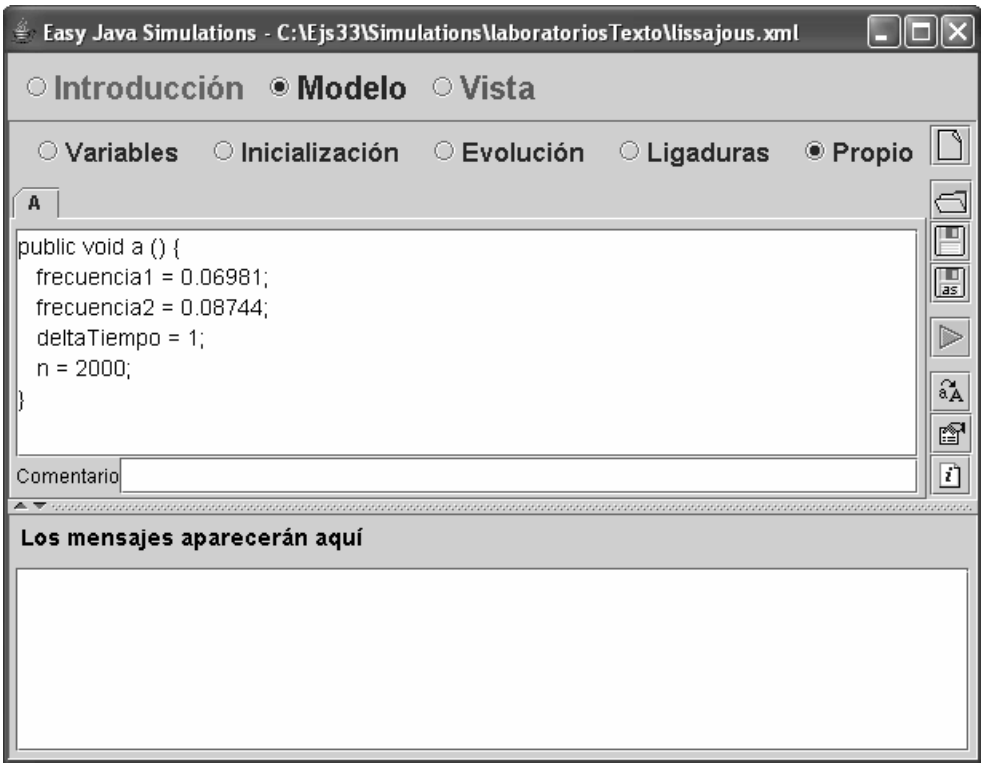


Figura 7.23: Método a, que se asociará con el Botón A.

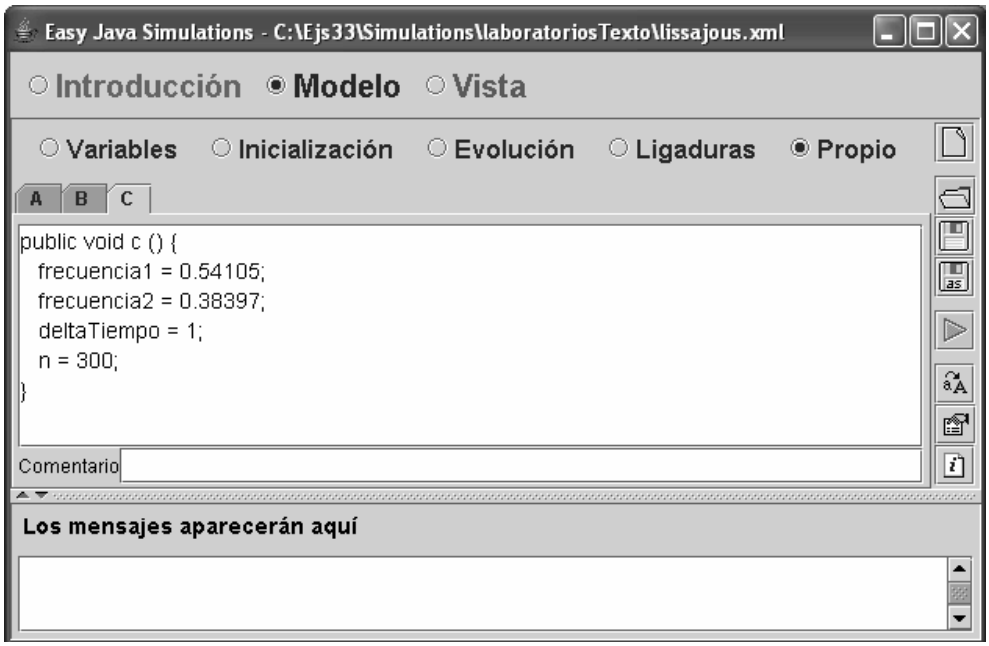


Figura 7.24: Métodos *a*, *b* y *c*, que se asociarán con los Botones *A*, *B* y *C* respectivamente.

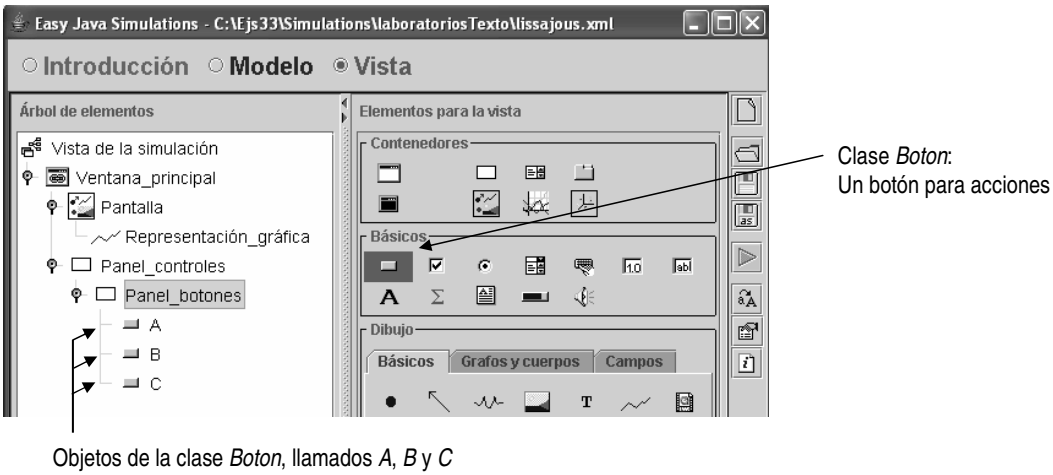


Figura 7.25: Botones *A*, *B* y *C*, ubicados dentro de *Panel_botones*.

- El valor de la propiedad *Formato* determina el formato con el que se muestra en la casilla el valor de la variable.
- La propiedad *Acción* permite especificar que se ejecute un determinado método (de los predefinidos en Ejs o de los "propios" del usuario) cuando el usuario cambie interactivamente (es decir, durante la simulación) el valor escrito en la casilla. Pulsando el botón con los eslabones dibujados, situado a la derecha de la propiedad, se abre una ventana mostrando los posibles métodos entre los cuales puede escoger. El método *_resetView()* limpia la vista.

De forma análoga pueden definirse las propiedades de los elementos *Frecuencia2* y *Desfase*.

7.8. Ejecución y distribución del laboratorio virtual

Ejecutando de nuevo la simulación, se obtiene la vista mostrada en la Figura 7.30.

Asimismo, observe que Ejs ha generado el documento lissajous.html (vea la Figura 7.31) en el directorio de trabajo (*Simulations*, por defecto).

Además de desde la ventana principal de Ejs, y como un applet incluido en una página web, el laboratorio puede ejecutarse como una aplicación Java independiente. Para ello, habilite la opción "Crear fichero BAT de ejecución" en la ventana de configuración de Ejs (vea la Figura 4.7). Si una vez habilitada esta opción vuelve a ejecutar la simulación, verá que Ejs ha creado en el directorio de trabajo el fichero *lissajous.bat*. Ejecute este fichero por lotes y observará que arranca la ejecución del laboratorio virtual, abriéndose una ventana con la vista del mismo (que es la mostrada en la Figura 7.30).

Si desea distribuir este laboratorio, debe entregar, aparte de los ficheros mencionados anteriormente, una copia del directorio *_library*, que debe situar en el mismo sitio que las simulaciones.

Figura 7.26: Propiedades del elemento *A*.

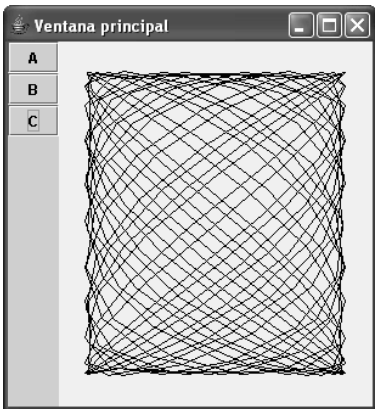


Figura 7.27: Vista del osciloscopio virtual.

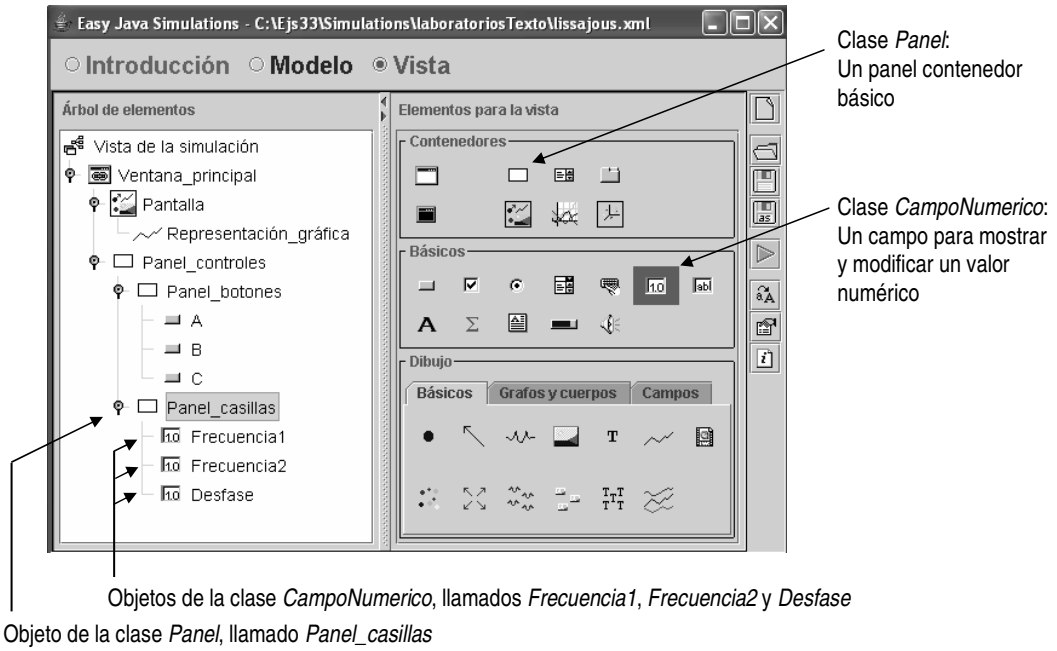


Figura 7.28: Árbol de elementos de la vista, con las tres casillas definidas.

Figura 7.29: Menú para Frecuencia1: el botón se ha asociado con la variable frecuencia1.

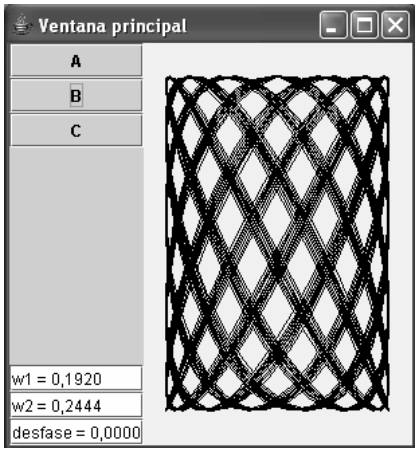


Figura 7.30: Vista del osciloscopio virtual.

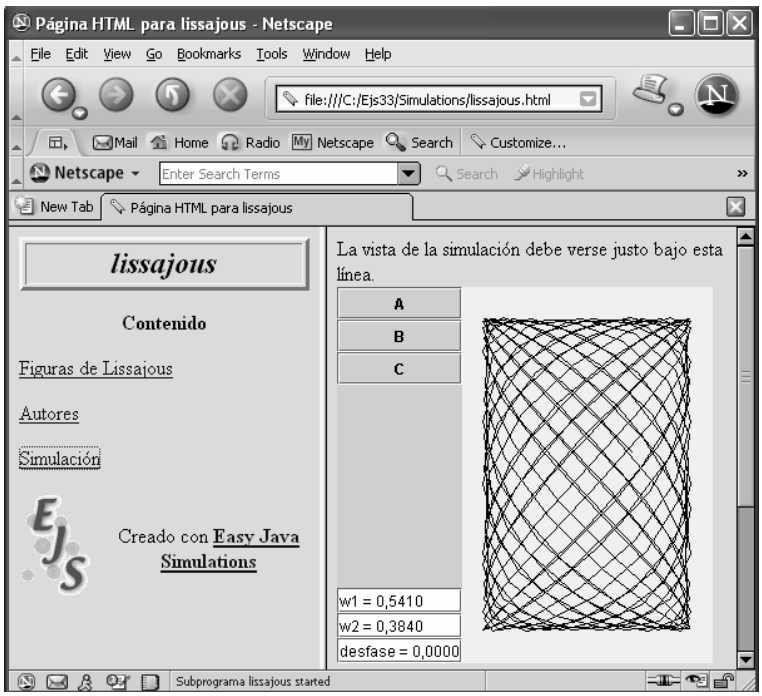


Figura 7.31: Documento lissajous.html visualizado con Netscape.

Tema 8

Un laboratorio virtual para ilustrar el concepto de ciclo límite

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Incluir imágenes en las páginas de Introducción.
- Diseñar el algoritmo de simulación de un modelo dinámico, compuesto únicamente por ecuaciones diferenciales ordinarias, y programar dicho modelo en Ejs.
- Usar las clases de elementos gráficos: PanelConEjes, Selector y Deslizador.

8.1. Modelo de un ciclo límite

En este Tema se explica la programación de un laboratorio virtual cuya finalidad es ilustrar el concepto de *ciclo límite*. Un ciclo límite en el plano XY está descrito por las ecuaciones siguientes[1]:

$$\frac{dx}{dt} = y + \frac{K \cdot x \cdot \left(1 - x^2 - y^2\right)}{\sqrt{x^2 + y^2}} \tag{8.1}$$

$$\frac{dy}{dt} = -x + \frac{K \cdot y \cdot \left(1 - x^2 - y^2\right)}{\sqrt{x^2 + y^2}} \tag{8.2}$$

donde las condiciones iniciales son:

$$x\left(0\right) = x_0 \tag{8.3}$$

$$y\left(0\right) = y_0 \tag{8.4}$$

y donde K es un parámetro del modelo, es decir, se supone que su valor no depende del tiempo.

[1] Este modelo está extraído del texto (MGA 1995)

El ciclo límite es un círculo de radio 1.0. Es decir, cualquiera que sean las condiciones iniciales para x e y (excepto $x_0 = y_0 = 0$), $\left(x^2 + y^2\right) \rightarrow 1$ cuando $t \rightarrow \infty$.

El laboratorio virtual deberá permitir al alumno modificar interactivamente:

- El valor del parámetro K.
- El valor de las variables x e y.

El laboratorio virtual deberá mostrar la evolución temporal del modelo en el plano XY (tendente hacia el círculo unidad) y la evolución de las variables x e y frente al tiempo.

Este laboratorio está disponible en el CD del curso. Se trata del fichero *cicloLimite.xml*, que está grabado en el directorio *laboratoriosTexto*.

8.2. Descripción de la introducción

En la Figura 8.1 se muestra la página de *Introducción* del laboratorio. Las fórmulas se han insertado en la página de *Introducción* como imágenes. El procedimiento seguido en este caso ha sido el siguiente:

1. Se ha escrito cada fórmula y se ha guardado como una imagen .gif. Las cuatro fórmulas empleadas en la introducción están en el directorio *Simulations/laboratoriosTexto/Imagenes*.
2. Se ha insertado cada fórmula en la página de *Introducción* seleccionando en el menú de Ejs *Insertar / Imagen...* (vea la Figura 8.2).

8.3. El algoritmo de la simulación

El paso previo a la programación del modelo en Ejs es definir el algoritmo de la simulación del laboratorio virtual. Esto es equivalente a especificar qué código deberá ejecutarse en cada uno de los paneles, teniendo en cuenta que el algoritmo de ejecución genérico de Ejs es el mostrado en la Figura 5.1.

El modelo matemático del ciclo límite en el plano XY, descrito por las Ecs. (8.1) y (8.2), está compuesto por las variables siguientes:

- Dos variables de estado: x, y.
- Un parámetro: K.

El algoritmo para la simulación de este modelo es el mostrado en la Figura 8.3. Las dos ecuaciones diferenciales se escriben en una página EDO, en el panel *Evolución*. El motivo por el cual la inicialización de la variable tiempo y de las variables de estado se realiza en el panel *Inicialización* se explicará más adelante, al describir la programación de la vista.

8.4. Declaración e inicialización de las variables

Para inicializar el modelo es preciso asignar valor a sus parámetros (K, x_0, y_0), a sus variables de estado (x, y) y a la variable tiempo (t).

En la Figura 8.4 se muestra la ventana del panel *Variables*, en la que se declaran las variables el modelo y se inicializan algunas de ellas: K, x_0 e y_0. En el panel *Inicialización* se inicializarán el resto: x, y y t.

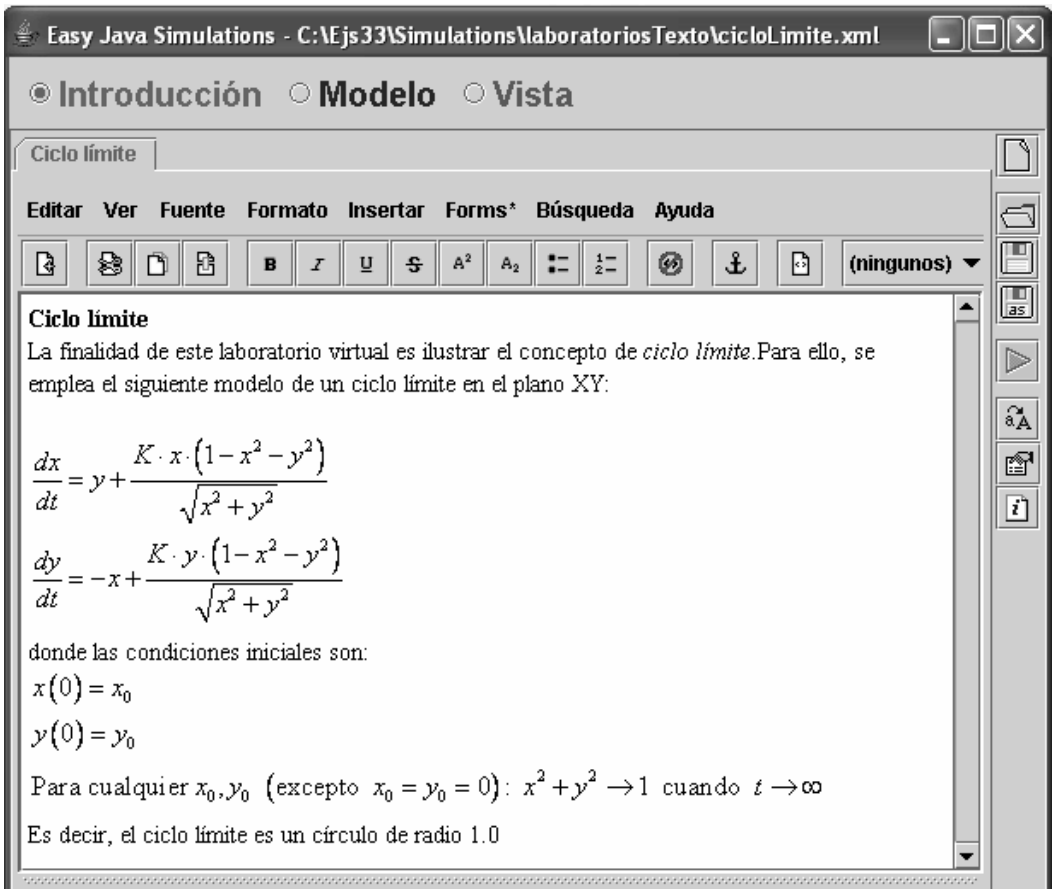


Figura 8.1: Página de *Introducción* del laboratorio virtual.

Figura 8.2: Inserción de una imagen en la página de *Introducción*.

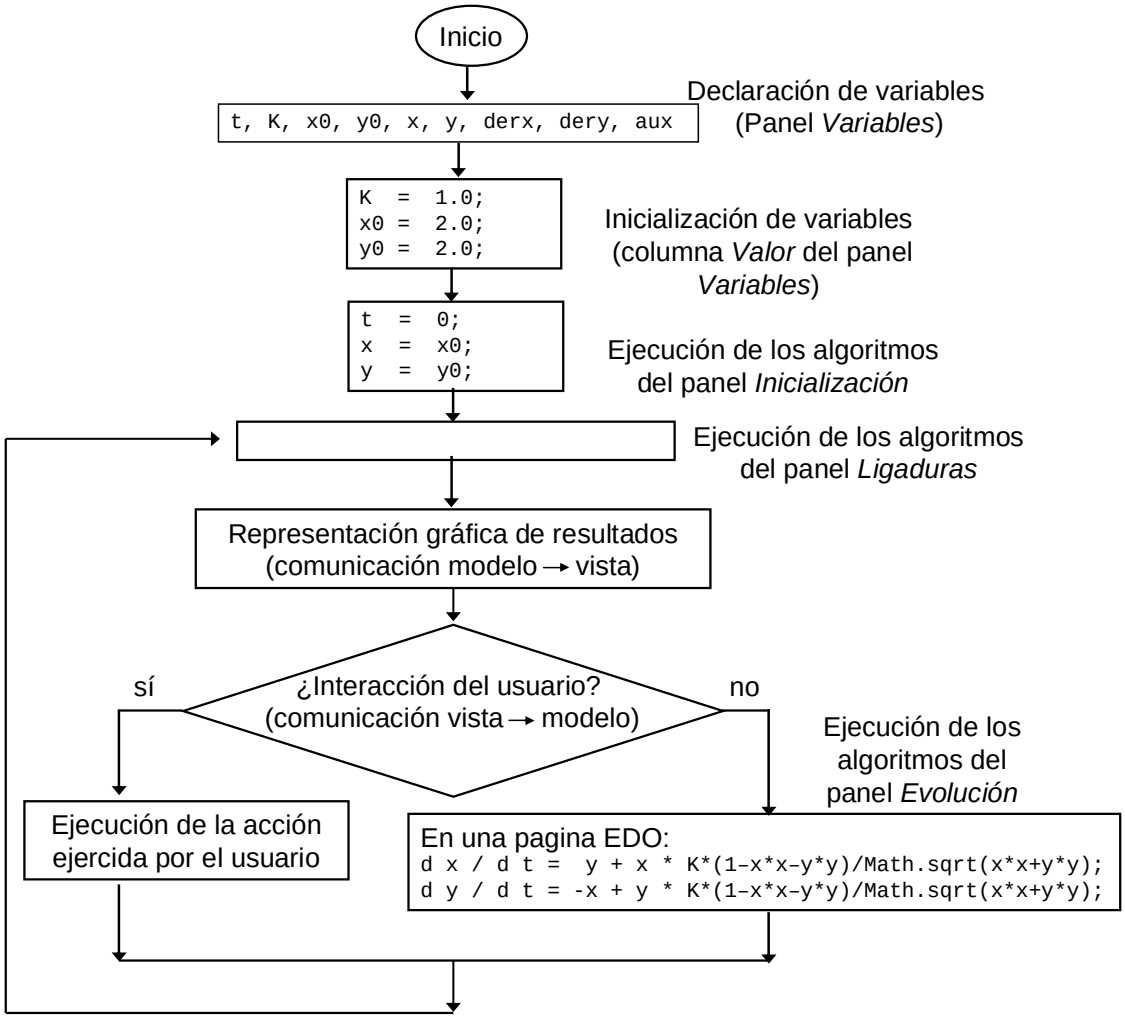


Figura 8.3: Algoritmo de la simulación del laboratorio virtual.

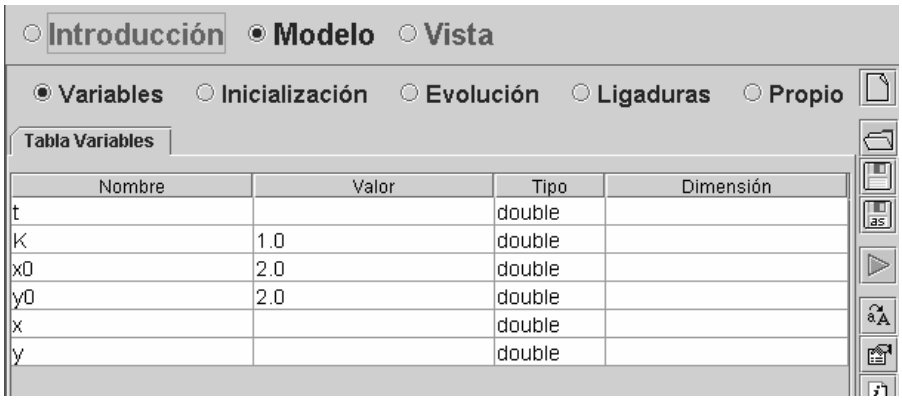


Figura 8.4: Declaración de las variables del modelo.

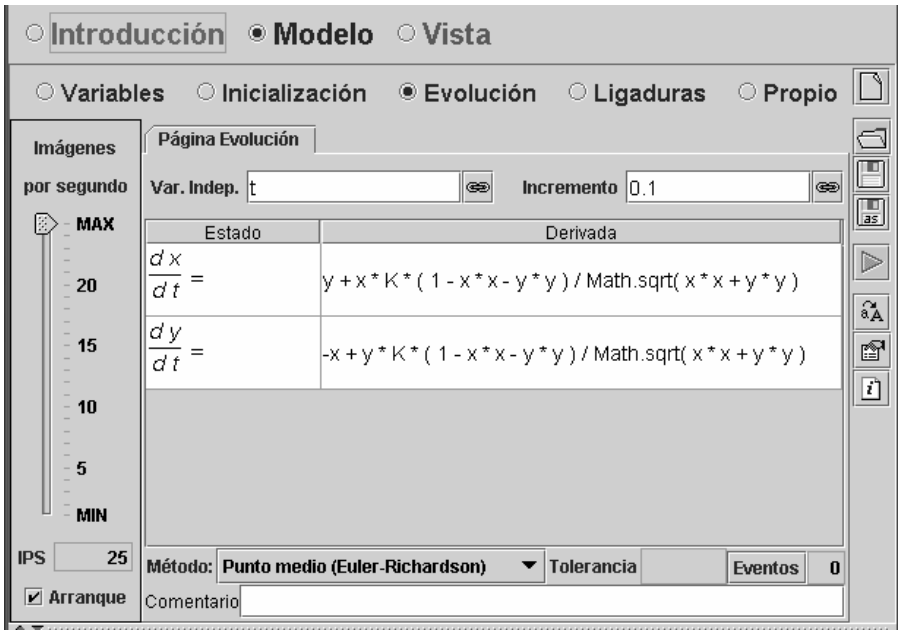


Figura 8.5: Página para la definición de Ecuaciones Diferenciales Ordinarias (EDO).

8.5. Programación de la evolución

En el panel *Evolución* deben calcularse las variables de estado (x, y) mediante la integración de sus derivadas. Para ello, va a usarse uno de los métodos de integración que proporciona Ejs: el método del punto medio.

Recuérdese que en una página EDO de Ejs es preciso expresar las derivadas en función únicamente de variables de estado, parámetros y la variable tiempo. En este modelo, tal como viene descrito mediante las Ecs. (8.1) y (8.2), se satisface esta condición.

Para definir las ecuaciones diferenciales, hay que hacer clic sobre la frase *Pulse para crear una página EDO* (vea la Figura 5.6). A continuación hay que asignar un nombre a la página EDO que acaba de crearse.

Para seleccionar la variable de estado de cada ecuación hay que situar el ratón sobre la correspondiente casilla *Estado*, pulsar el botón de la derecha y, en el menú que aparece, seleccionar la opción *Seleccionar variable de estado*. Entonces, se abre una ventana con todas las variables del modelo y debe seleccionarse una de ellas.

Una vez definidas las dos ecuaciones diferenciales del modelo matemático, la página EDO tiene el aspecto mostrado en la Figura 8.5

Puesto que el modelo contiene ecuaciones diferenciales, que han sido definidas en una página EDO, Ejs se encarga automáticamente de realizar los incrementos en la variable tiempo y no debe hacerlo el usuario (como sucedía en el modelo del osciloscopio).

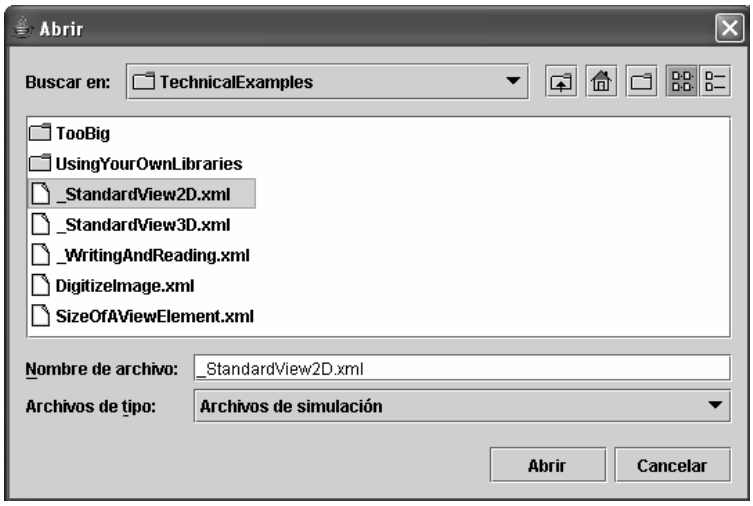
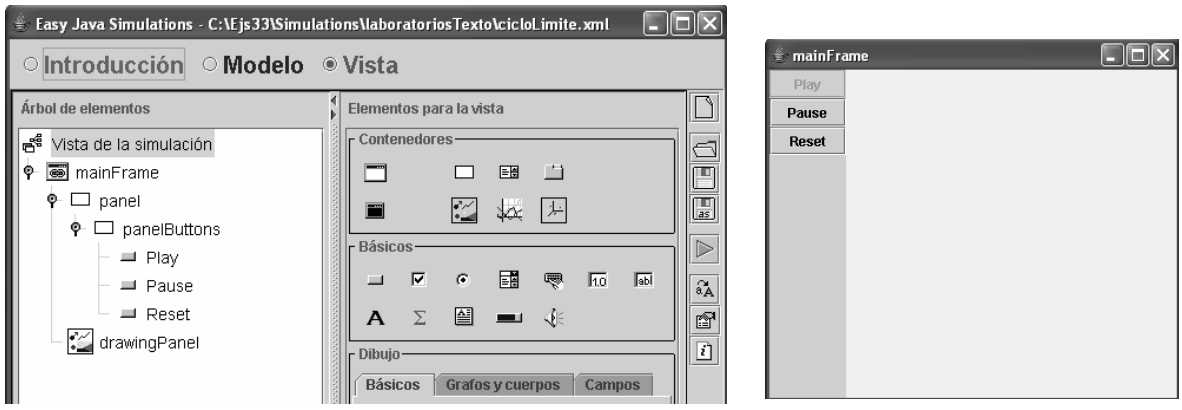


Figura 8.6: Selección del fichero *_StandardView2D.xml* para abrirlo.



a) b)

Figura 8.7: a) Árbol de elementos de la *Vista* tras abrir *_StandardView2D.xml*; b) *Vista*.

8.6. Programación de la vista

Para programar la vista, se va a emplear una vista genérica que está incluida en la distribución de Ejs[2]. Pulsando el botón *Abrir una simulación existente* (vea la Figura 3.1) seleccione el fichero *Simulations/_examples/TechnicalExamples/_StandardView2D.xml*.

Los ficheros *.xml* cuyo nombre comienza con el carácter *subrayado* (_), como por ejemplo *_StandardView2D.xml*, tienen la propiedad siguiente: cuando desde Ejs se abren estos ficheros, su contenido se añade en el entorno de Ejs a lo que hubiera definido en el entorno en ese instante. Esta capacidad de Ejs permite la *reutilización* del código para la definición de los laboratorios virtuales.

Al abrir el fichero *_StandardView2D.xml*, se añaden elementos al árbol de elementos de la *Vista*, el cual hasta este momento estaba vacío. El contenido del panel *Vista* se muestra en

[2]Las explicaciones para la programación de la *Vista* están basadas en el Tema 7 del texto (Esquembre 2002*b*).

la Figura 8.7a. Consiste en una ventana (objeto *MainFrame*), de la clase *Ventana*, dentro del cual están ubicados:

- Tres objetos (*Play*, *Pause*, *Reset*) de la clase *Botón*. Estos tres objetos están ubicados dentro de un objeto de la clase Panel (*PanelButtons*), que a su vez está ubicado dentro de otro objeto (*panel*) de la clase Panel.
- Un objeto (*DrawingPanel*) de la clase *PanelDibujo*, que es un contenedor 2D para elementos de dibujo.

Si llegado este punto se ejecuta el laboratorio virtual, la vista tiene el aspecto mostrado en la Figura 8.7b.

A continuación, se describen las operaciones realizadas sobre la vista con el fin de completar su definición.

Gráficas *x* vs *t*, *y* vs *t*

Un objeto de la clase *PanelConEjes* resulta más adecuado como contendedor de un conjunto de trazas que un objeto de la clase *PanelDibujo*. Con el fin de sustituir el objeto *DrawingPanel* por un objeto de la clase *PanelConEjes*:

1. Se elimina el objeto *DrawingPanel*. Para ello debe situarse el ratón sobre dicho objeto, pulsar el botón derecho del ratón y seleccionar *Eliminar*.
2. Se crea un objeto de la clase *PanelConEjes*. Para ello:
   - Se hace clic con el ratón sobre el icono de la clase *PanelConEjes*.
   - A continuación, se hace clic sobre el elemento contenedor de la *Vista* (*mainFrame*) en cuyo interior se desea ubicar el nuevo objeto. A este nuevo objeto, de la clase *PanelConEjes*, se le asigna el nombre: *panel_graficas_vs_tiempo*.

En la Figura 8.8 se muestra el árbol resultante de los elementos de la *Vista*.

Para modificar las propiedades del objeto *panel_graficas_vs_tiempo* hay que situar el ratón sobre dicho objeto y pulsar el botón derecho del ratón. Se abre un menú, en el que hay que seleccionar: *Propiedades*. En la Figura 8.9a se muestra la ventana de propiedades ya particularizada a los valores adecuados, y en la Figura 8.9b la vista a que dan lugar.

Para obtener la representación gráfica de x frente al tiempo e y frente al tiempo, deben crearse dos objetos de la clase *Traza*, a los que se asigna el nombre *x_vs_t* e *y_vs_t*, ubicados dentro del objeto *panel_graficas_vs_tiempo*. El árbol de elementos resultante se muestra en la Figura 8.10.

En la Figura 8.11 se muestran las propiedades de los objetos *x_vs_t* e *y_vs_t*.

Gráfica *y* vs *x*

De forma completamente análoga, se define un nuevo objeto de la clase *PanelConEjes* y se ubica dentro del contenedor *mainFrame*. A este nuevo objeto se le da el nombre: *panel_grafica_y_vs_x*.

Al igual que en el caso anterior, se crea un objeto de la clase *Traza* y se ubica dentro del objeto contenedor *panel_grafica_y_vs_x*. A este nuevo objeto se le da el nombre: *y_vs_x*.

Al haber ubicado un nuevo objeto dentro del contenedor *mainFrame*, es preciso ampliar el tamaño de la ventana de dicho contenedor. Colocando el ratón sobre *mainFrame* y pulsando el botón derecho del ratón se abre un menú, en el cual hay que seleccionar *Propiedades*. Se asigna a la propiedad *Tamaño* el valor mostrado en la Figura 8.12.

En la Figura 8.13a se muestra el árbol de los elementos de la *Vista*, y en la Figura 8.13b la *Vista* del laboratorio, obtenida ejecutando la simulación.

A continuación se explica cómo añadir capacidades interactivas al laboratorio virtual.

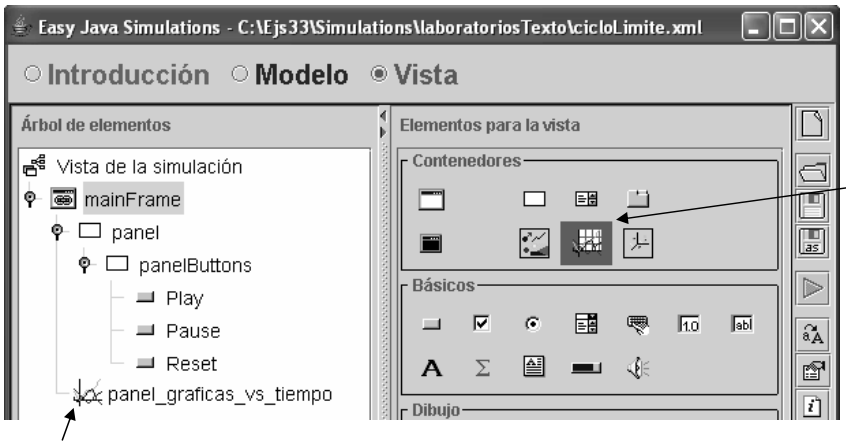


Figura 8.8: Árbol de elementos de la *Vista*.

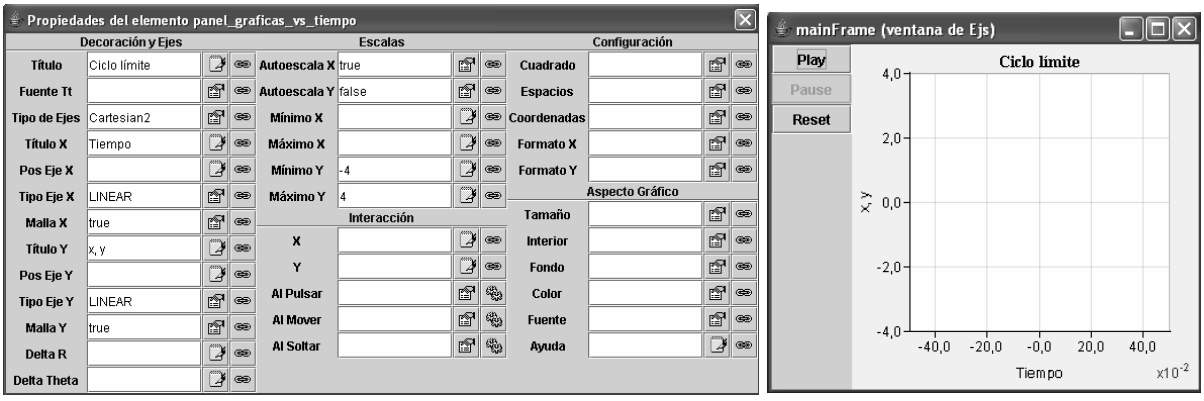


Figura 8.9: a) Propiedades del objeto *panel_graficas_vs_tiempo*; b) *Vista*.

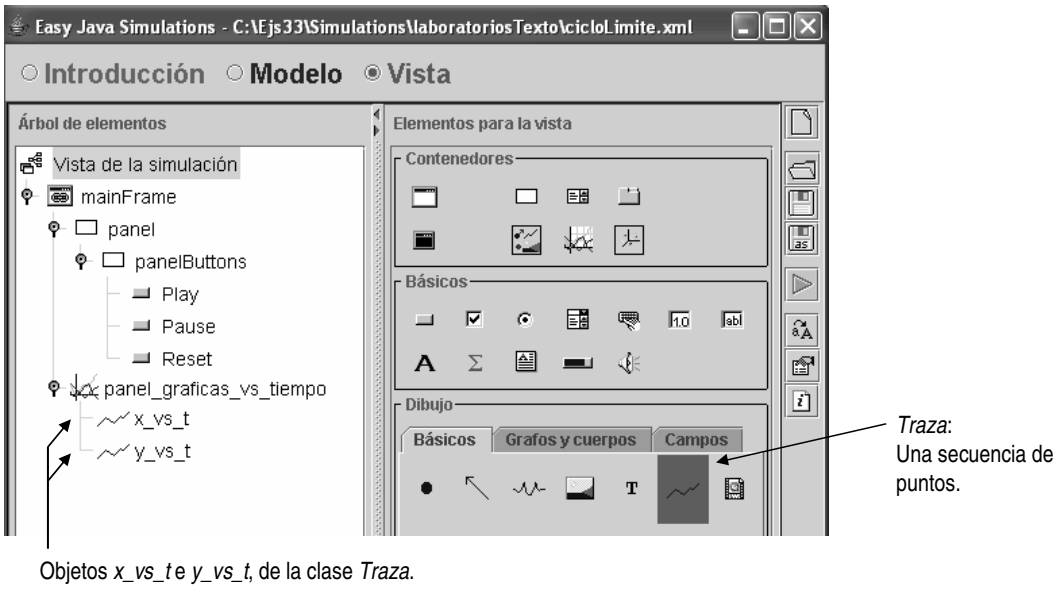
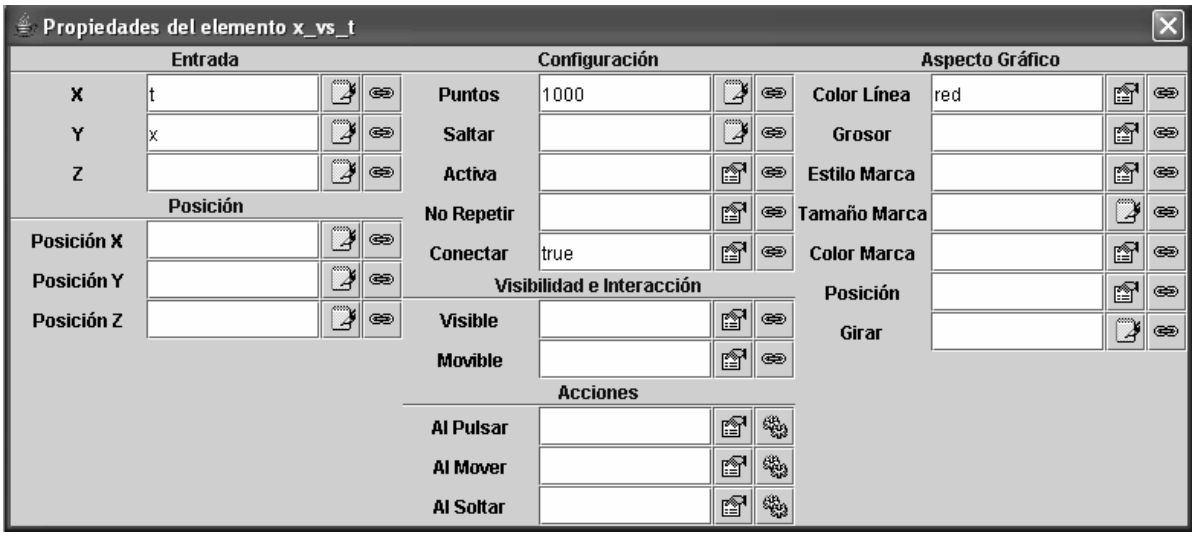
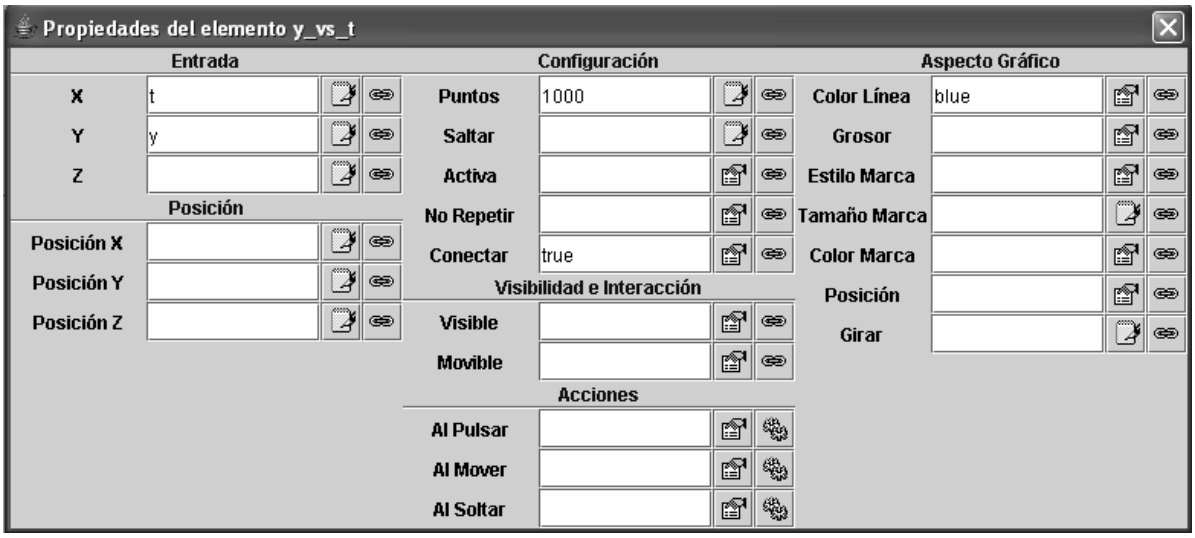


Figura 8.10: Árbol de elementos de la *Vista*.



a)



b)

Figura 8.11: Propiedades de los objetos: a) *x_vs_t*; b) *y_vs_t*.

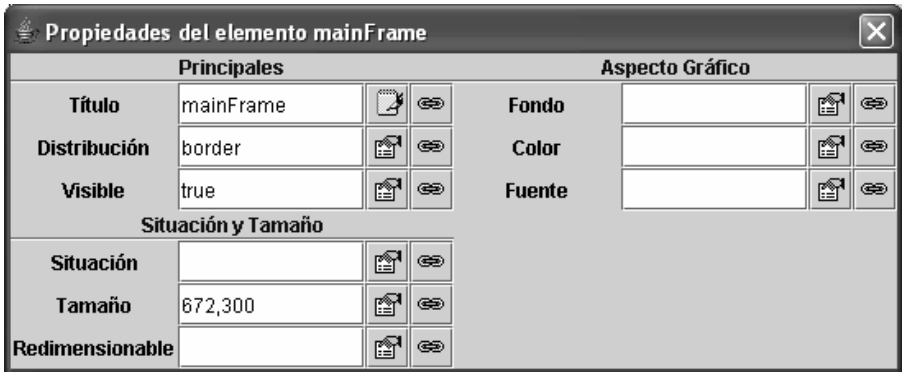


Figura 8.12: Propiedades del objeto *mainFrame*.

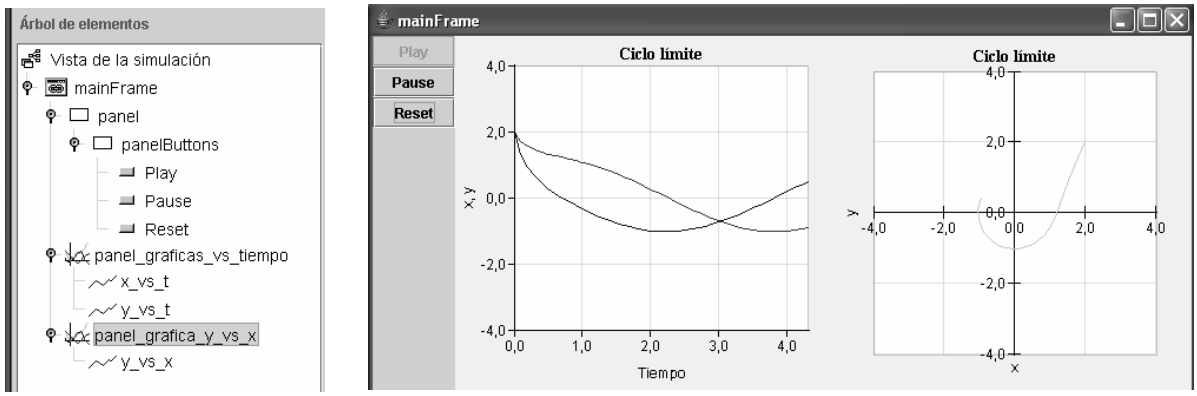


Figura 8.13: a) Árbol de elementos de la *Vista*; b) *Vista* del laboratorio.

8.7. Programación de las capacidades interactivas

El objetivo de este laboratorio es mostrar al alumno que, con independencia de los valores iniciales que asigne a x e y, la trayectoria del modelo en el plano XY tiende a describir una circunferencia de radio unidad, centrada en el origen.

Para ello, el laboratorio debe permitir que el alumno detenga la simulación, modifique los valores de x e y, y reanude la simulación.

Una forma de programar esta capacidad es:

1. Definir dos objetos de la clase *Deslizador*, que llamaremos *slider_x* y *slider_y*, de modo que los valores que hay seleccionados en estos sliders se escriban en los parámetros x_0 e y_0 respectivamente.
2. Definir un botón, que llamaremos *Inicializa*, cuya acción asociada es:
   - Limpiar la *Vista*.
   - Asignarle a la variable tiempo (t) el valor cero.
   - Escribir los valores de los parámetros x_0 e y_0 en las variables x e y respectivamente. De esta forma, cuando el usuario pulse el botón *Inicializa*, los valores fijados en los sliders *slider_x* y *slider_y* se escriben en las variables x e y respectivamente.

El procedimiento para programar en Ejs esta capacidad interactiva es el descrito a continuación.

Definición de los deslizadores (sliders). Los dos objetos de la clase *Deslizador* van a ubicarse dentro de un objeto contenedor de la clase *VentanaDialogo*, que se sitúa dentro del elemento raíz del árbol de elementos: *Vista de la simulación*. A este objeto contenedor se le da el nombre: *VentanaDialogoSliders*. De esta forma se pretende crear una ventana independiente que contenga los dos sliders. En la Figura 8.14 se muestra el árbol de elementos de la vista.

En la ventana de propiedades del objeto *VentanaDialogoSliders*, se define (vea la Figura 8.15):

- *Distribución*. La distribución de la ventana de diálogo, que en este caso es una rejilla con dos filas y una columna.
- *Visible*. La ventana es visible cuando el valor de esta propiedad es *true*. Si se escribe el nombre de una variable en la casilla de la propiedad *Visible*, la ventana de diálogo será visible cuando dicha variable valga *true*.
- *Situación, Tamaño, etc.*

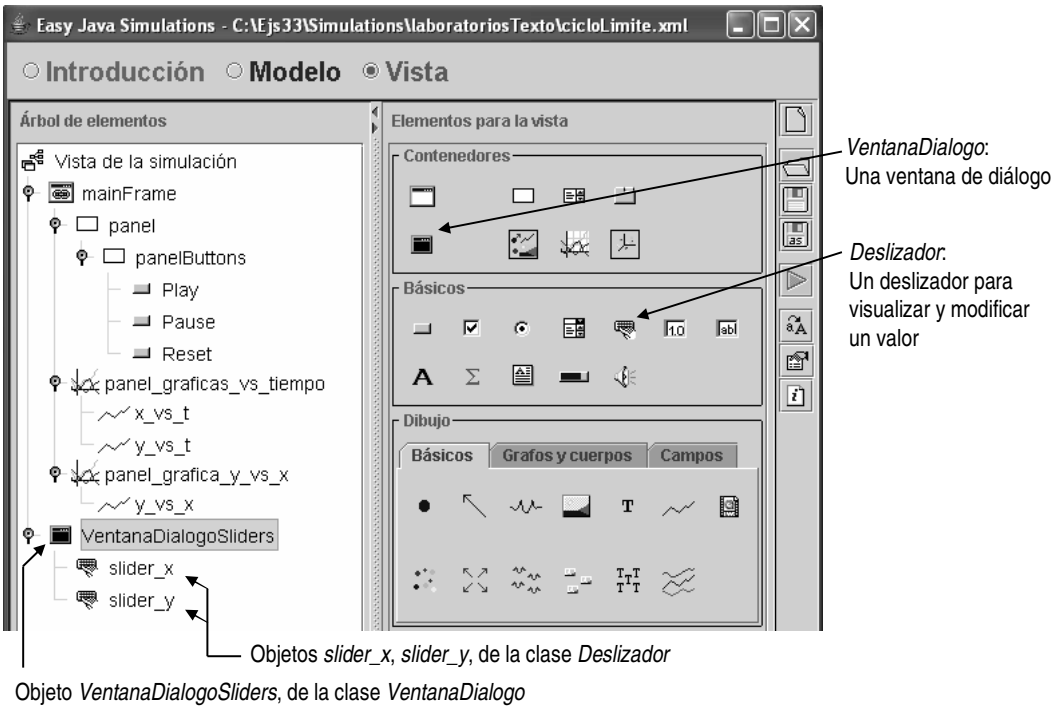


Figura 8.14: Árbol de elementos de la *Vista*.

Figura 8.15: Propiedades el objeto *VentanaDialogoSliders*, de la clase *VentanaDialogo*.

Figura 8.16: Propiedades el objeto *slider_x*, de la clase *Deslizador*.

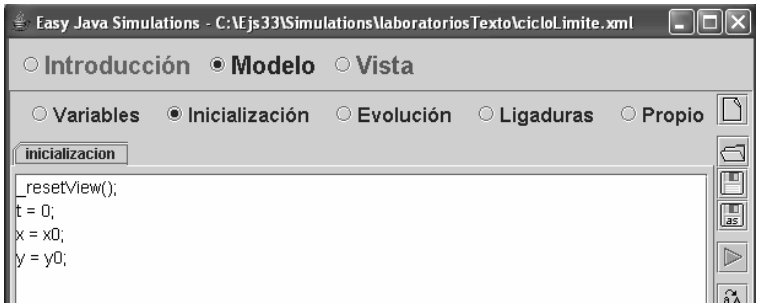


Figura 8.17: Panel *Inicialización*.

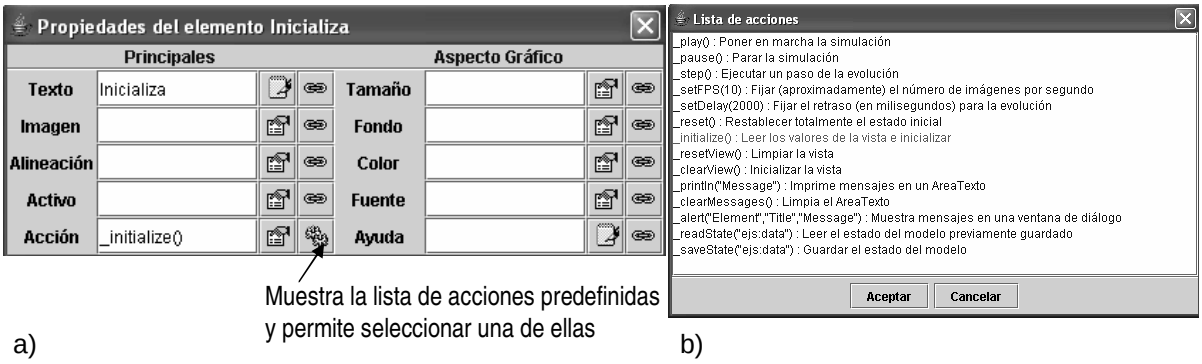


Figura 8.18: a) Propiedades el objeto *Inicializa*, de la clase *Boton*; b) Acciones predefinidas.

En la Figura 8.16 se muestran las propiedades del objeto *slider_x*, de la clase *Deslizador*. Observe que la propiedad *Variable* del objeto es la variable con la que se "enlaza" el deslizador:

- Mientras el usuario no interacciona con el deslizador, éste tiene en cada momento una posición que se corresponde con el valor que tiene la variable en ese instante. Si dicha variable es un parámetro, como es el caso de x_0 e y_0, la posición del deslizador no cambia. Si fueran variables evaluadas del modelo (como x e y), la posición del deslizador cambiaría en el tiempo siguiendo la evolución de dichas variables.
- Cuando el usuario interacciona con el deslizador, cambiando su posición, el valor correspondiente a dicha posición se escribe en la variable asociada. En este caso, cuando el usuario cambia la posición del deslizador *slider_x*, el nuevo valor se escribe en el parámetro x_0.

Las propiedades del objeto *slider_y* son análogas a las del objeto *slider_x*.

Programación del código de la acción. Cuando el usuario pulse el botón *Inicializa* deberá ejecutarse la acción:

```
_resetView();
t = 0;
x = x0;
y = y0;
```

Puesto que la acción ($t = 0$, $x = x_0$, $y = y_0$) se realiza también durante la inicialización del modelo, pueden moverse estas tres asignaciones del panel *Variables* al panel *Inicialización*. Las variables t, x e y siguen siendo declaradas en el panel *Variables*, pero no se les signa ningún valor.

La ventaja de programar el código de la acción en el panel *Inicialización*, en lugar de programarla como un método propio del usuario, es que puede usarse (para definir la acción asociada al botón *Inicializa*) uno de los métodos predefinidos de Ejs: _initialize(). Este método lee los valores de la *Vista* y ejecuta el panel *Inicialización*.

En la Figura 8.17 se muestra el código del panel *Inicialización*.

Definición del botón *Inicializa*. Se añade un nuevo objeto de la clase *Boton* y se ubica dentro del contenedor *panelButtons*. En la ventana de propiedades de este nuevo botón, que se llamará *Inicializa*, se da a la propiedad Acción el valor siguiente: *_initialize()* (vea la Figura 8.18a).

Puede obtenerse una lista de las acciones predefinidas en Ejs haciendo clic sobre el icono que está situado a la derecha de la casilla de la propiedad *Acción* y que tiene el dibujo de dos ruedas dentadas engranadas. En la Figura 8.18b se muestra esta lista de acciones.

Mostrar/Ocultar la ventana de diálogo. Finalmente, se va a añadir debajo de los botones una casilla que permita mostrar y ocultar la ventana de diálogo que contiene los dos deslizadores.

En el menú de propiedades del objeto *VentanaDialogoSliders* (vea la Figura 8.15) aparece la propiedad *Visible*, a la que se asignó el valor *true*.

Ahora va a declararse una variable boolena (*muestraVentanaSliders*), a la cual se asignará valor en la *Vista* (mediante un objeto de la clase *Selector*), y que se asignará a la propiedad *Visible* del objeto *VentanaDialogoSliders*.

Los pasos seguidos son los siguientes:

1. Definir la variable *muestraVentanaSliders* en el panel *Variables* (vea la Figura 8.19). La definición se realiza en una nueva página, llamada *Variables Vista*. Es una buena práctica realizar la definición de las variables del modelo matemático y de la *Vista* en ventanas separadas, ya que facilita la comprensión.
2. En la *Vista*, definir un objeto de la clase *Selector* y ubicarlo dentro del objeto *panelButtons*. A este nuevo objeto se le ha dado el nombre *SelectorVentanaDialogoSliders*. El árbol de elementos resultante se muestra en la Figura 8.20.
3. En las propiedades del objeto *SelectorVentanaDialogoSliders*, asignar a la propiedad *Variable* la variable *muestraVentanaSliders* (vea la Figura 8.21). Con ello, el objeto gráfico escribe en la variable el valor *true* o *false*, en función de que esté seleccionado o no.
4. Finalmente, asignar la variable *muestraVentanaSliders* a la propiedad *Visible* del objeto *VentanaDialogoSliders*.

Finalmente, la vista de la simulación tiene el aspecto mostrado en la Figura 8.22.

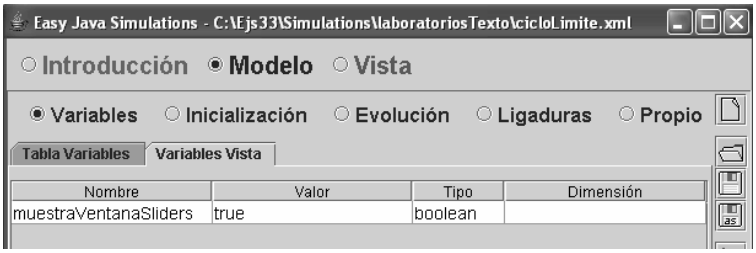


Figura 8.19: Declaración e inicialización de la variable *muestraVentanaSliders*.

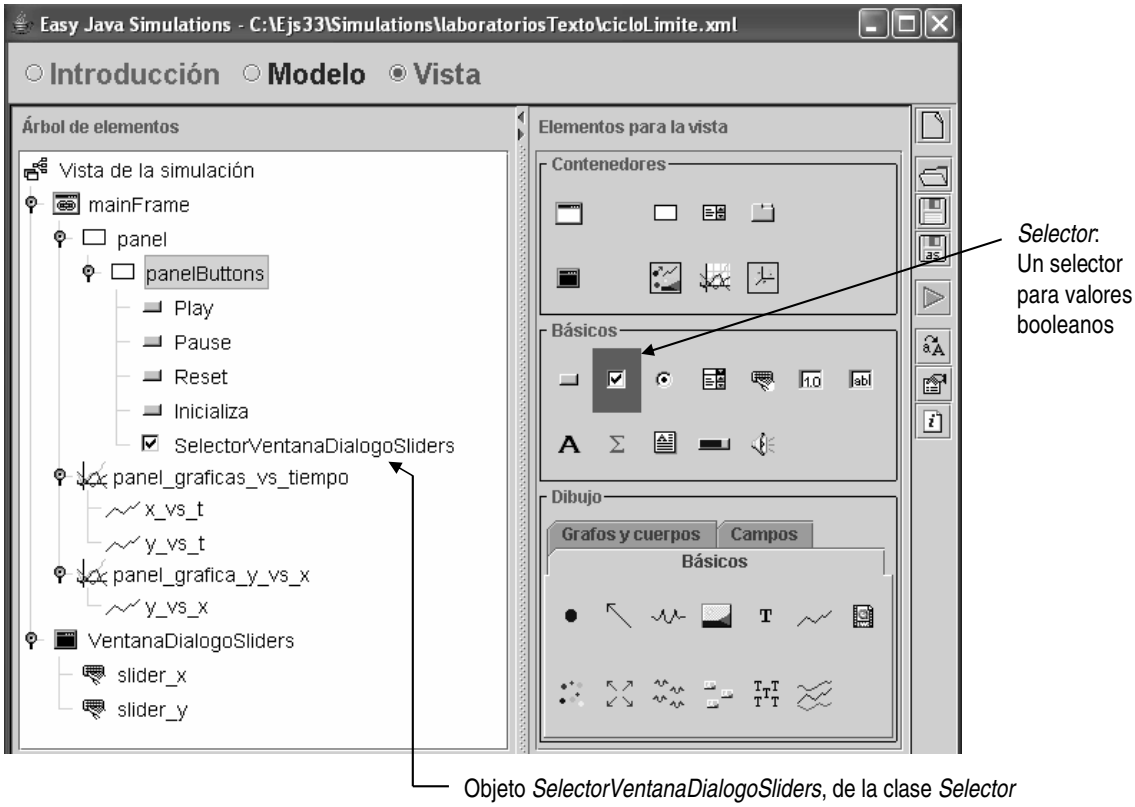


Figura 8.20: Árbol de elementos.

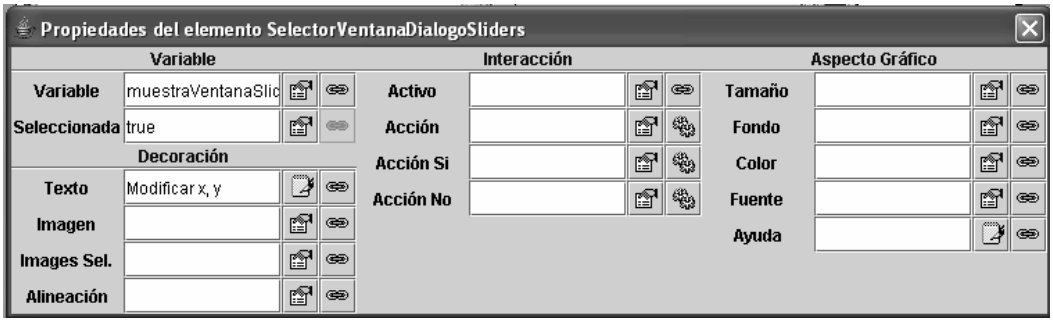


Figura 8.21: Propiedades del objeto *SelectorVentanaDialogoSliders*, de la clase *Selector*.

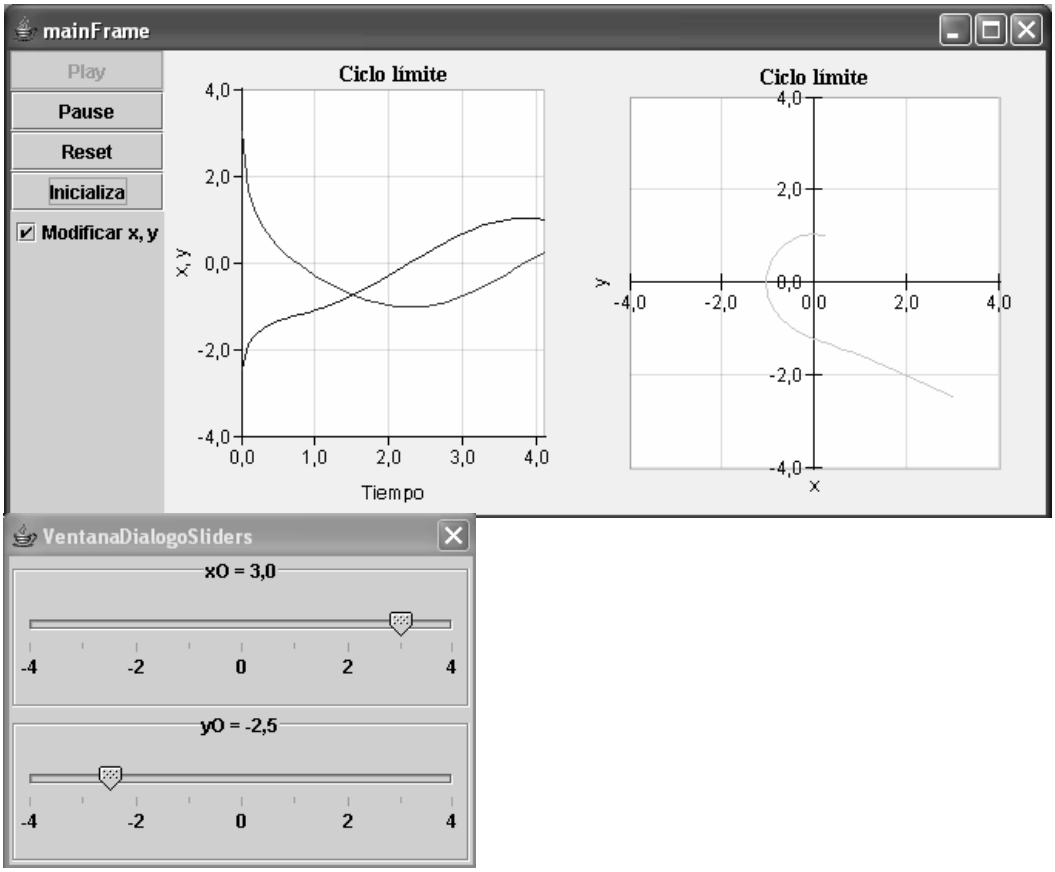


Figura 8.22: *Vista* del laboratorio virtual del ciclo límite.

Tema 9

Principio de Arquímedes

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Diseñar el algoritmo de simulación de un modelo estático y programar el modelo usando Ejs.
- Usar el elemento gráfico Poligono.

Este laboratorio virtual está disponible en el CD del curso. Se trata de *Arquimedes.xml*, y se encuentra en el directorio *laboratoriosTexto*.

9.1. Descripción del modelo matemático

Se pretende modelar el comportamiento de un objeto cúbico, de lado L y masa m sumergido parcialmente en un líquido de densidad ϱ_L. En este modelo se van a despreciar las variaciones en la altura del líquido que se producen al variar la proporción del objeto que se encuentra sumergida. Las fuerzas ejercidas sobre el objeto cúbico se representan en la Figura 9.1. Estas son:

$$Peso = m \cdot g \tag{9.1}$$

$$Empuje = L^2 \cdot h \cdot \varrho_L \cdot g \tag{9.2}$$

Siendo g la aceleración de la gravedad y h la longitud del lado del cubo que se encuentra sumergido en el líquido. Para el cálculo del empuje se ha empleado el principio de Arquímedes.

En equilibrio la suma de fuerzas sobre el objeto es nula, es decir, el empuje es igual al peso del objeto. La condición de equilibrio es pues:

$$m \cdot g = L^2 \cdot h \cdot \varrho_L \cdot g \tag{9.3}$$

El máximo valor del empuje se produce cuando el cuerpo esté totalmente sumergido en el líquido ($h = L$). Si el peso del objeto es mayor que el valor de empuje para $h = L$, el objeto se hundirá sometido a una aceleración. Si el peso es menor que el valor de empuje para $h = L$ el cuerpo flotará. El cuerpo flotará mientras se cumpla la siguiente condición:

$$m \leq \varrho_L \cdot L^3 \tag{9.4}$$

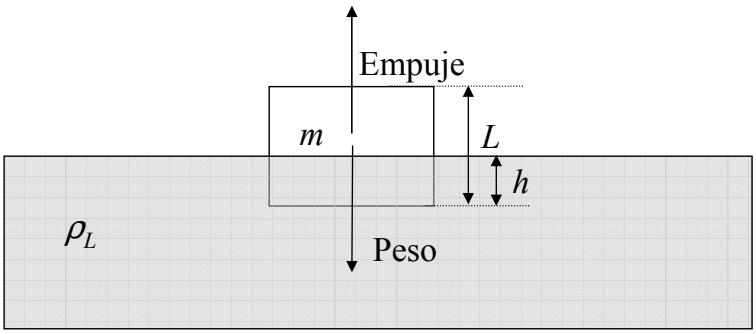


Figura 9.1: Diagrama del modelo.

Ecuaciones del modelo:

Se ha elegido como variable interactiva del modelo la masa del sólido (m). Al variar la masa del sólido variará el volumen del sólido sumergido y, por tanto, el valor de h. Despejamos de la Ec. (9.3) el valor de h:

$$h = \frac{m}{\varrho_L \cdot L^2} \tag{9.5}$$

Además, se ha de satisfacer la condición de la Ec. (9.4) para que el cuerpo flote. Para que se cumpla esta condición se va a limitar el valor máximo de la masa (m_{max}) que puede introducir el usuario. Este valor máximo se calcula aplicando la siguiente expresión:

$$m_{max} = \varrho_L \cdot L^3 \tag{9.6}$$

9.2. El algoritmo de la simulación

Antes de programar el modelo, vamos a agrupar las variables del modelo en parámetros, variables de estado y algebraicas. De este modo obtenemos la siguiente clasificación:

- Parámetros del modelo que no vamos a permitir que el usuario varíe en tiempo de simulación: longitud del lado del cubo (L), densidad del líquido (ϱ_L). A partir de estos dos parámetros se calcula un tercer parámetro: la masa máxima del objeto (m_{max}). Para ello se emplea la Ec. (9.6).
- Parámetros del modelo que vamos a permitir que el usuario varíe en tiempo de simulación: masa del sólido (m).
- Variable algebraica del modelo: h. Esta variable algebraica se calcula a partir de la Ec. (9.5).

Hay que inicializar, dando un valor adecuado, los parámetros siguientes: L, ϱ_L y m. El resto de variables se calculan a partir de las anteriores variables y no precisan que se les de un valor inicial.

En la Figura 9.2 se muestra una posible forma de programar el laboratorio.

9.3. Declaración e inicialización de las variables

Hemos creado una página de variables que hemos llamado variablesModelo donde se declaran e inicializan todas las variables del modelo matemático. El valor de m_{max} se calcula

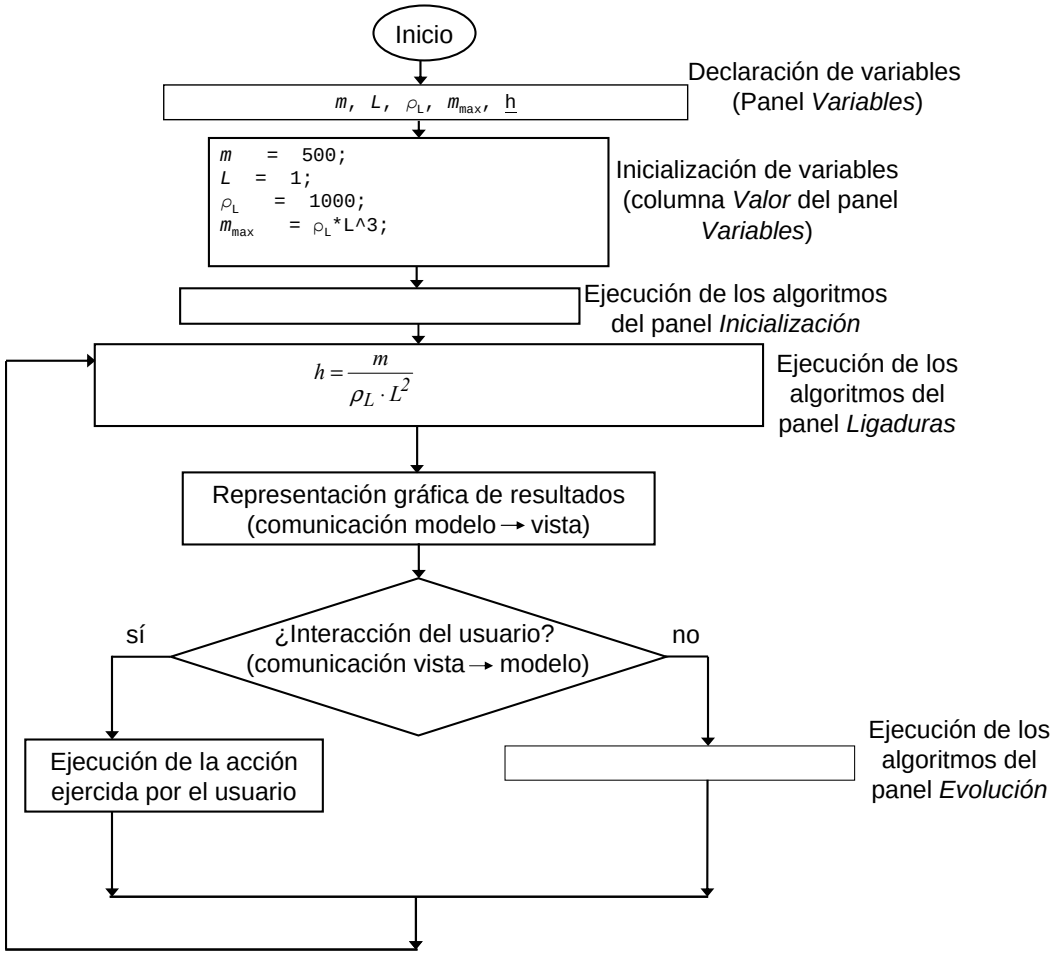


Figura 9.2: Algoritmo de simulación.

empleando para ello la Ec. (9.6). Sólo se necesita calcular este valor una vez ya que depende de parámetros que no van a variar durante la simulación. En la Figura 9.3 se muestra la apariencia final de la ventana variablesModelo.

9.4. Programación de las ligaduras

Vamos a crear una página de Ligaduras donde se va a calcular el valor de h empleando para ello la Ec. (9.5). El aspecto final de la página se muestra en la Figura 9.4.

9.5. Programación de la vista

Elementos del árbol de la vista del modelo:

Vamos a realizar en la vista una representación del modelo en dos dimensiones similar a la de la Figura 9.1. El objeto cúbico y el líquido se van a representar empleando dos cuadrados.

El árbol de la vista del modelo va a contener los siguiente elementos (ver Figura 9.5):

- Elemento de la clase “Ventana”.
- Elemento de la clase “Panel” que contiene a su vez un elemento de la clase “Deslizador” para permitir al usuario modificar el valor de la densidad de la masa del objeto.

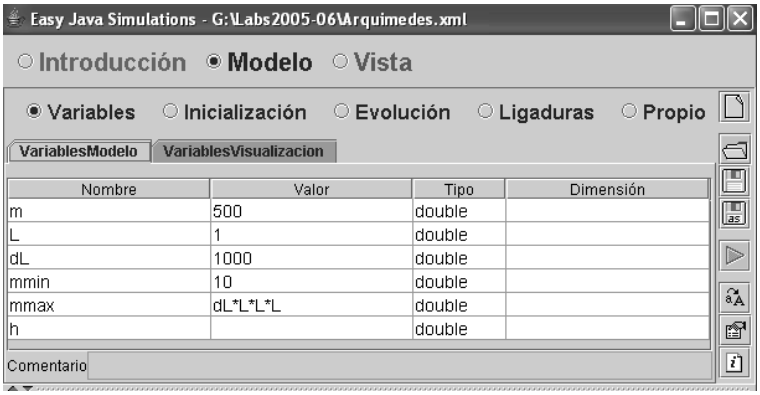


Figura 9.3: Ventana *VariablesModelo*, del panel Variables.

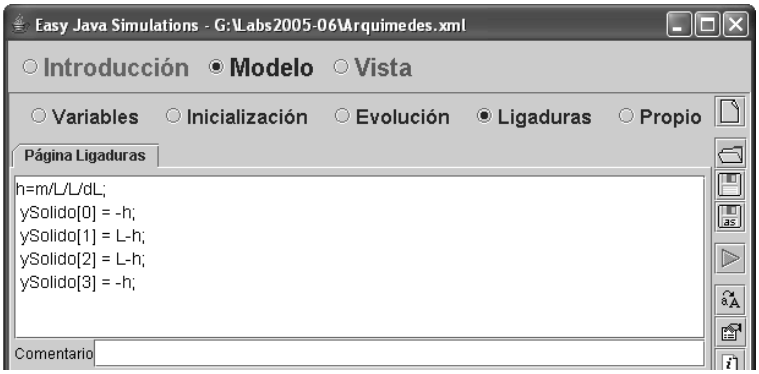


Figura 9.4: Ventana *Página Ligaduras*, del panel Ligaduras.

– Elemento de la clase "PanelDibujo" que contiene dos elementos de la clase "Poligono" para representar el objeto cúbico y el líquido.

A continuación, se describen brevemente algunas de las propiedades del elemento gráfico Poligono.

Clase de elemento gráfico *Poligono*. Este elemento de la *Vista* se encuentra en el panel de dibujo *Grafos y cuerpos*. Se trata de un polígono cerrado, que se especifica mediante las coordenadas de sus vértices: (x, y, z) o (x, y), según se esté trabajando en tres o en dos dimensiones.

Se han empleado dos polígonos usando dos objetos de la clase *Poligono* que se han ubicado dentro del objeto *DrawingPanel*.

Para cada uno de estos polígonos, se han especificado las propiedades siguientes:

– El número de sus vértices.

– Dos vectores, con las posiciones x e y de los vértices del polígono.

En la Figura 9.8 se muestran las propiedades del poligono *Liquido*:

– En la propiedad *Puntos* se ha introducido el número de vértices del polígono.

– En las propiedades *X* e *Y*, se han escrito las coordenadas x e y respectivamente de los vértices del polígono.

Conjunto de variables necesarias para la definición de la Vista del modelo:

Se ha creado una nueva página de Variables para definir las variables que necesitamos emplear para confeccionar la vista del modelo. A esta página le hemos dado el nombre de VariablesVisualización (ver Figura 9.6).

Figura 9.5: Árbol de Elementos de la *Vista*.

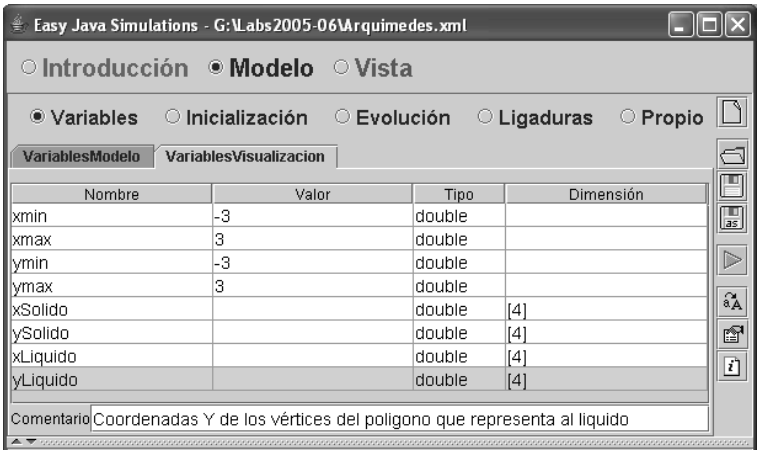


Figura 9.6: Ventana *variablesVisualizacion*, del panel Variables.

Para definir los vértices del cuadrado que representa al objeto necesitamos declarar dos vectores de dimensión 4. Usaremos un vector para especificar las coordenadas x de los vértices del polígono y otro para las coordenas y. Del mismo modo, necesitamos dos vectores de dimensión 4 para especificar las coordenadas x e y de los vértices del cuadrado que representa el líquido.

Para inicializar estos vectores creamos una nueva página Inicialización en la sección Modelo que hemos llamado "Página Inicio" (ver Figura 9.7).

En la ventana *Página Ligaduras* del panel Ligaduras añadimos el código necesario para calcular el valor de la coordenada y de los vértices del polígono que representa el objeto en función de la variable h (ver Figura 9.4).

La vista del laboratorio se muestra en la Figura 9.9.

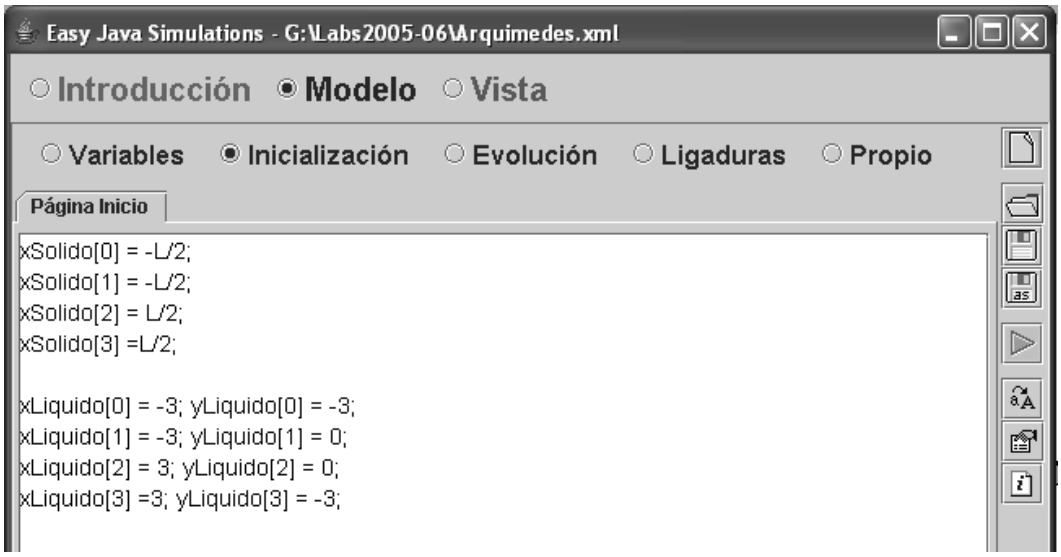


Figura 9.7: Ventana *Página Inicio*, del panel Inicialización.

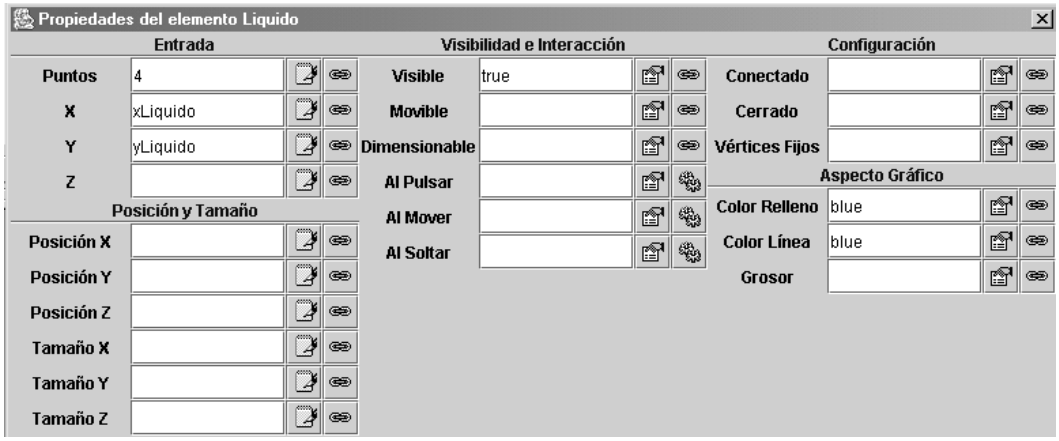


Figura 9.8: Propiedades objeto *Liquido*, de la clase Poligono.

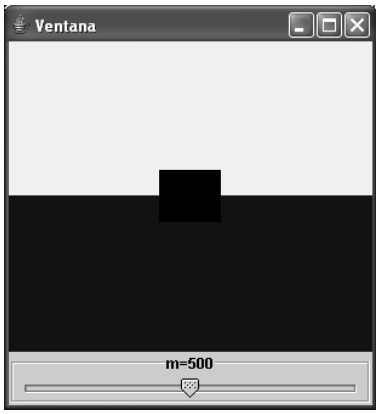


Figura 9.9: Vista del laboratorio virtual.

Tema 10

Péndulo simple

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Diseñar el algoritmo de simulación de un modelo dinámico que contiene ecuaciones diferenciales ordinarias y ecuaciones algebraicas y programar el modelo usando Ejs.
- Usar las clases de elementos gráficos: Etiqueta, Flecha, Particula.

Este laboratorio virtual está disponible en el CD del curso. Se trata de *PenduloSimple.xml*, y se encuentra en el directorio *laboratoriosTexto*.

10.1. Descripción del modelo matemático

El movimiento armónico simple (M.A.S.) en una dimensión viene descrito matemáticamente por la siguiente ecuación diferencial:

$$\frac{d^2x}{dt^2} = -K^2 \cdot x \tag{10.1}$$

Este tipo de movimiento se produce siempre que una fuerza sea proporcional al desplazamiento y de sentido opuesto. Ejemplos de sistemas que describen este movimiento son el formado por una masa unida a un resorte o un péndulo simple siempre que el ángulo que forma la lenteja del péndulo con la vertical no sea demasiado grande.

En este laboratorio virtual se estudia el movimiento de un péndulo simple como ejemplo de M.A.S. La lenteja del péndulo tiene masa m y la longitud de la cuerda es L.

Para describir correctamente el modelo matemático establecemos un sistema de referencia y un criterio de signos. Escogemos el eje y como el vertical y con su sentido positivo hacia abajo y el eje x como horizontal como se muestra en la Figura 10.1. El modelo matemático del péndulo simple linealizado en torno a la posición de equilibrio viene descrito por la siguiente ecuación:

$$\frac{d^2\theta}{dt^2} = -K^2 \cdot \theta \tag{10.2}$$

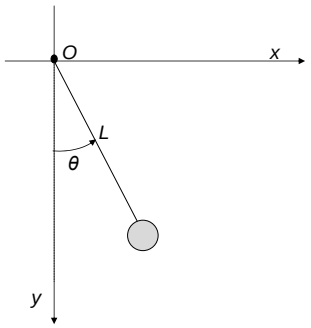


Figura 10.1: Diagrama péndulo simple.

La posición de la lenteja del péndulo en coordenadas cartesianas conocido θ se calcula empleando las siguientes ecuaciones:

$$x = L \cdot \cos(\theta) \quad (10.3)$$
$$y = L \cdot \sin(\theta) \quad (10.4)$$

10.2. Descripción de la introducción

En la Figura 10.2 se muestra la página de introducción del laboratorio.

10.3. El algoritmo de la simulación

El modelo matemático está compuesto por las ecuaciones 10.2, 10.3 y 10.4. Se van a emplear los algoritmos numéricos de resolución de ecuaciones diferenciales ordinarias (EDO) de Ejs. Para ello hay que transformar la ecuación diferencial de 2º orden 10.2 en dos ecuaciones diferenciales de primer orden.

$$\frac{d\theta}{dt} = \omega \quad (10.5)$$
$$\frac{d\omega}{dt} = -K^2 \cdot \theta \quad (10.6)$$

Este modelo consta de las siguientes variables:

- Variables de estado, es decir, aquellas variables que aparecen derivadas y que, por tanto, necesitan inicialización: θ y ω. El valor de estas variables se calcula mediante integración.
- Parámetros, es decir, variables cuyo valor permanece constante durante la simulación a menos que sea modificado por el usuario: K, g y L. El parámetro K se calcula a partir de los parámetros g y L.
- Variables algebraicas x e y. Las variables algebraicas son aquellas que, no apareciendo derivadas, dependen de su cálculo en el instante de evaluación del valor de otras variables.

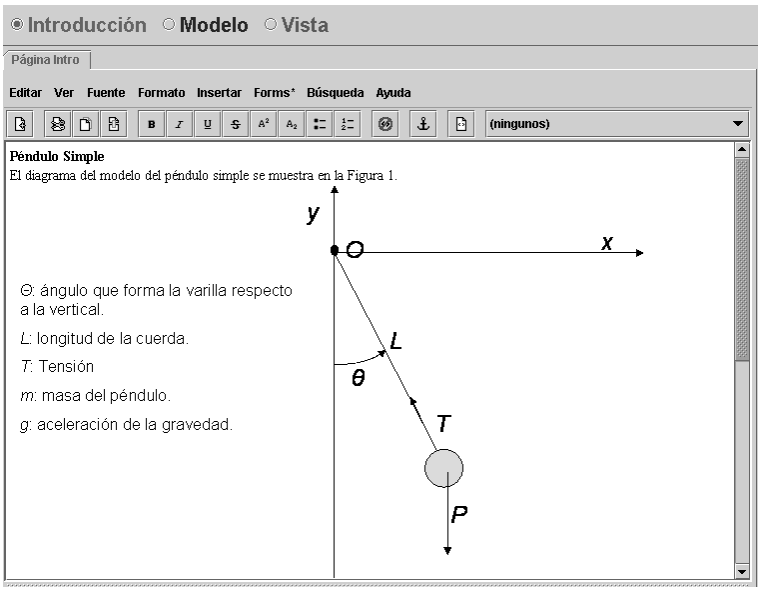


Figura 10.2: Página de Introducción.

- Variable que representa el tiempo: $time$. En este laboratorio se usan los métodos de resolución de ODE de Ejs para realizar la integración del sistema de EDO y obtener así el valor de θ y ω en cada instante de tiempo. Al emplear los métodos de resolución de ODE de Ejs, la gestión de la variable tiempo es llevada a cabo por éste.
- Parámetro dt: paso de integración que será utilizado por el algoritmo de resolución de ODE de Ejs.

El algoritmo de la simulación se muestra en la Figura 10.3.

10.4. Declaración e inicialización de variables

Hemos seguido la metodología de distinguir entre las variables empleadas en el modelo matemático y el resto. De este modo, la declaración e inicialización de las variables se ha realizado en dos ventanas del panel Variables:

- *VariablesModelo*, en la que se han declarado las variables del modelo matemático (vea la Figura 10.4).
- *VariablesVisualizacion*, en la que se han declarado las variables empleadas para la definición de la vista.

10.5. Programación del modelo

El modelo matemático está formado por un sistema de dos ecuaciones diferenciales y dos ecuaciones algebraicas. El sistema de ecuaciones diferenciales se introduce en el panel Evolución. En el panel Ligaduras introducimos las ecuaciones algebraicas (10.3) y (10.4).

Panel Evolución

En el panel Evolución deben calcularse las variables de estado mediante la integración de sus derivadas. Para ello, va a usarse uno de los métodos de integración que proporciona Ejs:

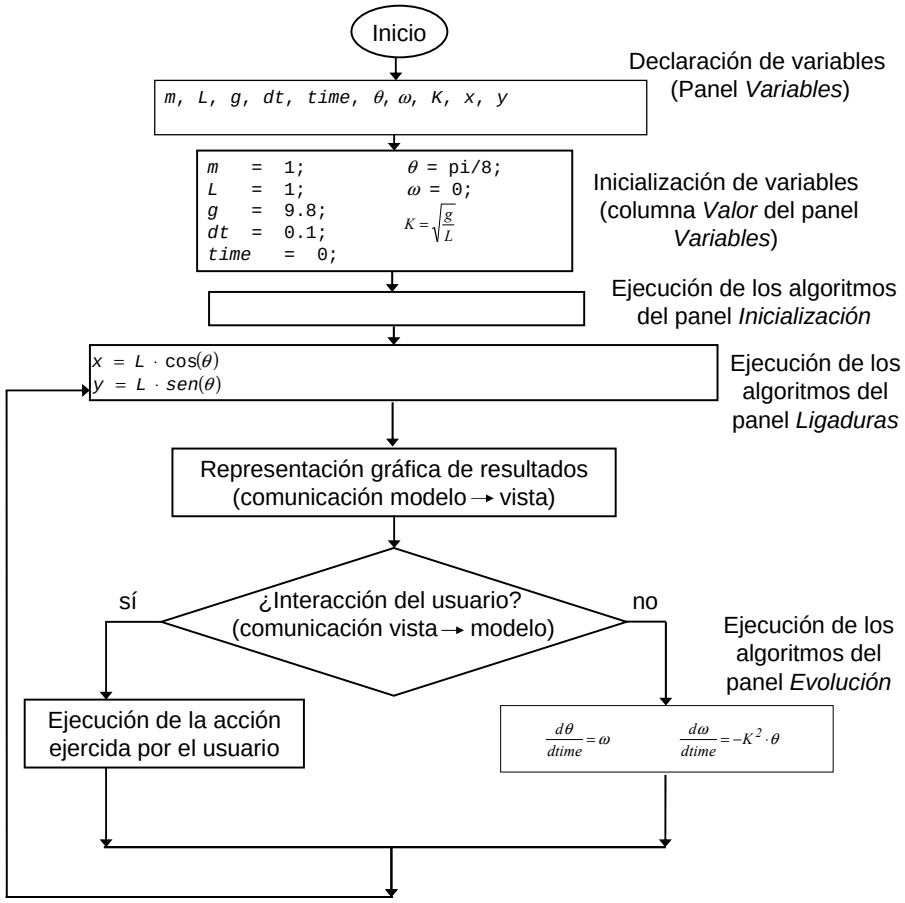


Figura 10.3: Algoritmo de simulación.

el método de Runge-Kutta de cuarto orden. La página EDO tiene el aspecto mostrado en la Figura 10.5.

Puesto que el modelo contiene ecuaciones diferenciales, que han sido definidas en una página EDO, Ejs se encarga automáticamente de realizar los incrementos en la variable tiempo.

Panel Ligaduras

Una vez escritas las ecuaciones de ligadura, la ventana tiene el aspecto mostrado en la Figura 10.6.

10.6. Programación de la vista

La vista del laboratorio virtual diseñada se compone de dos ventanas: una ventana principal y una ventana secundaria (vea Figura 10.7).

La parte central de la ventana principal contiene una representación gráfica del sistema. La parte inferior de dicha ventana contiene elementos gráficos que permiten al usuario controlar la simulación e interactuar con el modelo. La ventana secundaria contiene un gráfico mostrando la evolución temporal de θ.

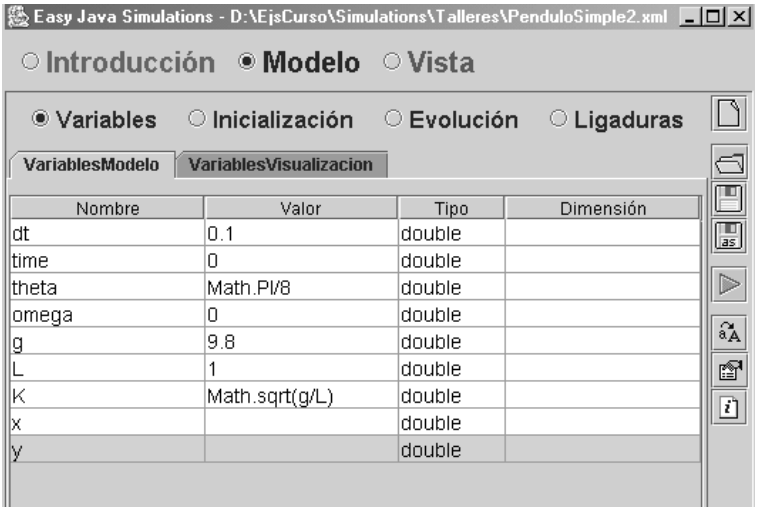


Figura 10.4: Página VariablesModelo, del panel *Variable*.

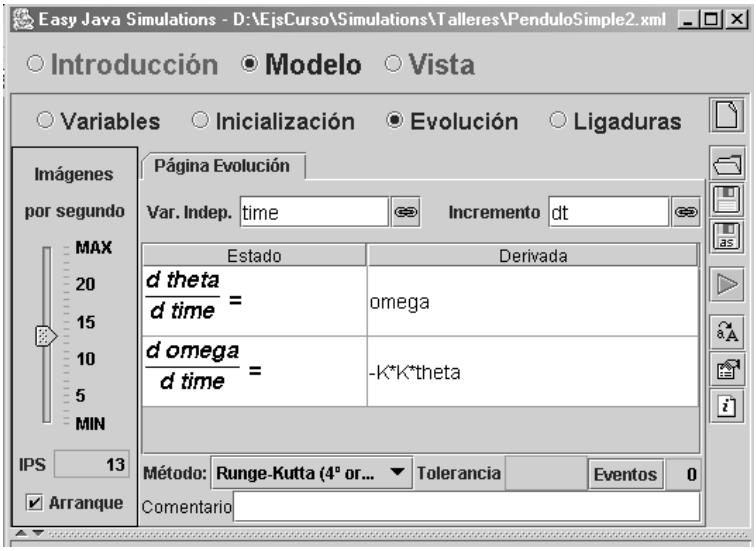


Figura 10.5: Página de *Evolución*.

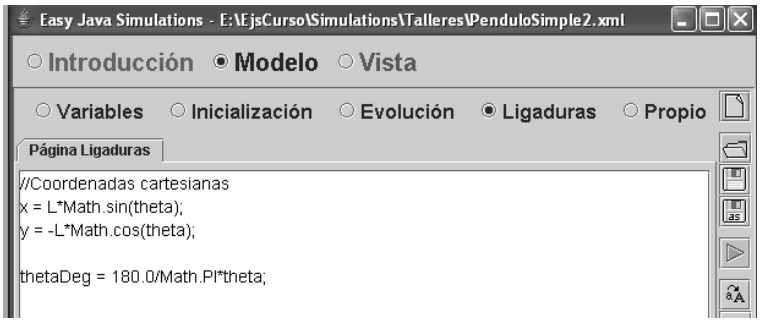


Figura 10.6: Página de *Ligaduras*.

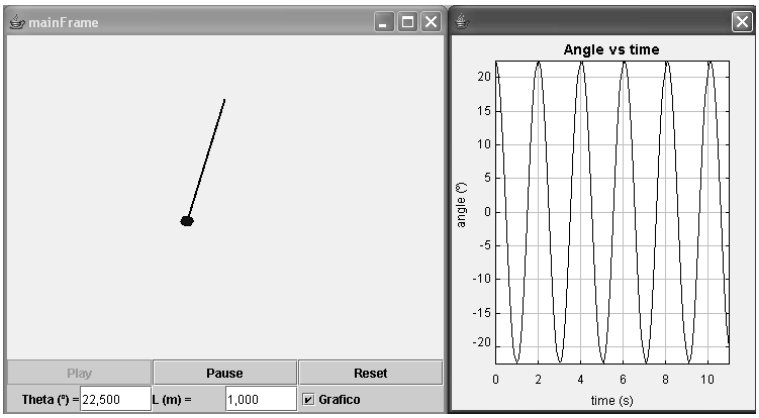


Figura 10.7: *Vista* del laboratorio del péndulo simple.

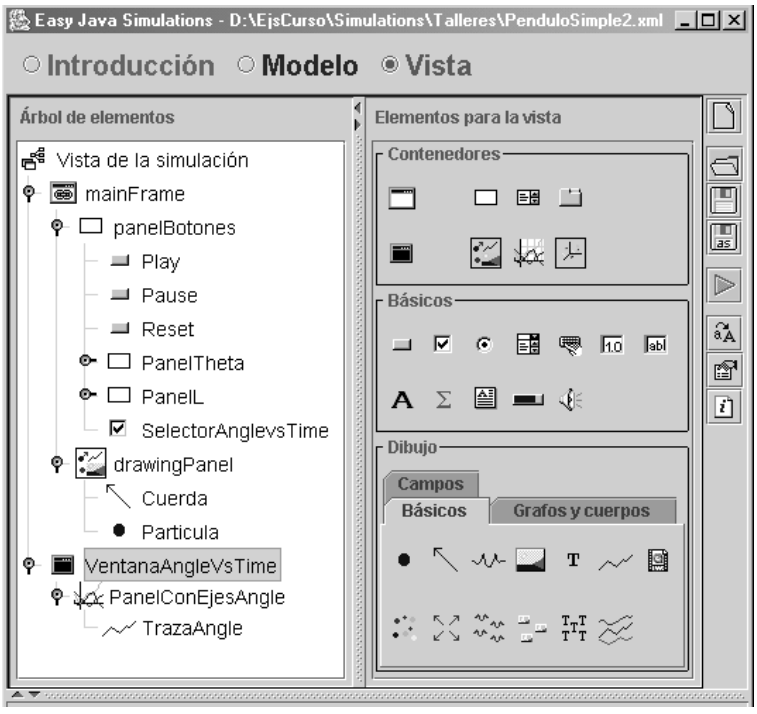


Figura 10.8: Árbol de elementos de la vista

En la Figura 10.8 se muestra el árbol de elementos correspondiente a la vista del laboratorio virtual.

Los elementos de la clase Boton nos permiten controlar la simulación (parar, reanudar y volver la simulación a su estado inicial). Los elementos de la clase CampoNumerico permiten al usuario introducir nuevos valores de parámetros y variables. Los elementos de la clase Selector se usan para fijar el valor de una variable lógica a true o false según estén o no seleccionados. Al introducir dicha variable en la propiedad Visible de un elemento gráfico se muestra u oculta dicho elemento según la variable esté a true o false.

En la programación del laboratorio virtual se han usado tres elementos nuevos: Particula, Flecha y Texto. A continuación se van a describir sus características fundamentales así como las propiedades de estos elementos que hemos modificado para programar la vista.

Clase de elemento gráfico *Etiqueta*.

Se han empleado los objetos *EtiquetaL* y *EtiquetaTheta*, de la clase *Etiqueta*. Una *Etiqueta* es un elemento básico que se usa para presentar un texto o una imagen (o ambos) con propósito decorativo o informativo.

La clase *Etiqueta* se encuentra en el panel de dibujo llamado *Básicos*.

En la Figura 10.9 se muestran las propiedades del objeto *EtiquetaL*. A continuación, se describen algunas propiedades del objeto *EtiquetaL*:

- *Texto*: texto mostrado por el elemento. Puede ser una constante o cualquier variable de tipo String.
- *Imagen*: ruta donde se localiza la imagen que deseamos que muestre el elemento. La imagen debe tener formato gif o gif animado. La ruta puede ser una dirección de internet o la relativa a un directorio de trabajo.
- *Alineación*: indica la forma de alinear el texto en el elemento gráfico. Puede ser left (izquierda), center (centro) o right (derecha).

Clase de elemento gráfico *Flecha*.

Se ha empleado el objeto *Cuerda*, de la clase *Flecha*, para representar la cuerda del péndulo. La clase *Flecha* se encuentra en el panel de dibujo llamado *Grafos y cuerpos*.

En la Figura 10.10 se muestran las propiedades del objeto *Cuerda*. A continuación, se describen las propiedades del objeto *Cuerda* que se han especificado:

- *X*: coordenada x del origen del vector.
- *Y*: coordenada y del origen del vector.
- *Tamaño X*: tamaño del vector según el eje x.
- *Tamaño Y*: tamaño del vector según el eje y.
- *Activo*: indica mediante un valor booleano si este componente es interactivo o no.
- *Estilo*: tipo de vector a dibujar. Puede ser una flecha (*flecha*), un segmento (*segmento*) o un segmento con un cuadrado en el punto final (*cajita*).

Clase de elemento gráfico *Particula*.

Se ha empleado el objeto *Particula*, de la clase *Particula*, para representar la lenteja del péndulo. La clase *Particula* se encuentra en el panel de dibujo llamado *Básicos*. Una Particula es un elemento de dibujo que dibuja una forma geométrica sencilla, un rectángulo o una elipse, en unas determinadas coordenadas del padre. La ubicación de este elemento en el padre se especifica dando las coordenadas de su centro y el tamaño del objeto según cada uno de los ejes coordenados.

En la Figura 10.11 se muestran las propiedades del objeto *Particula*. A continuación, se describen las propiedades del objeto *Particula* que se han especificado:

- *X*: coordenada x del centro del elemento.
- *Y*: coordenada y del centro del elemento.
- *Tamaño X*: tamaño del vector según el eje x.
- *Tamaño Y*: tamaño del vector según el eje y.
- *Activo*: indica mediante un valor booleano si este componente es interactivo o no.

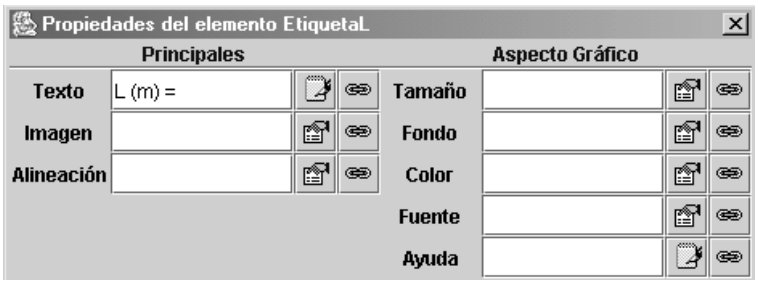


Figura 10.9: Propiedades del objeto *EtiquetaL*, de la clase Etiqueta.

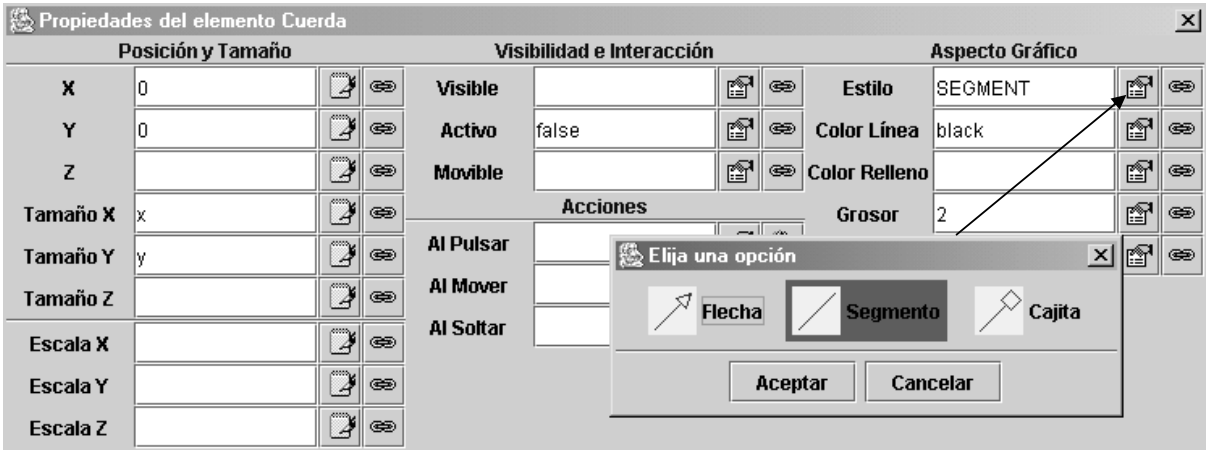


Figura 10.10: Propiedades del objeto *Cuerda*, de la clase Flecha.

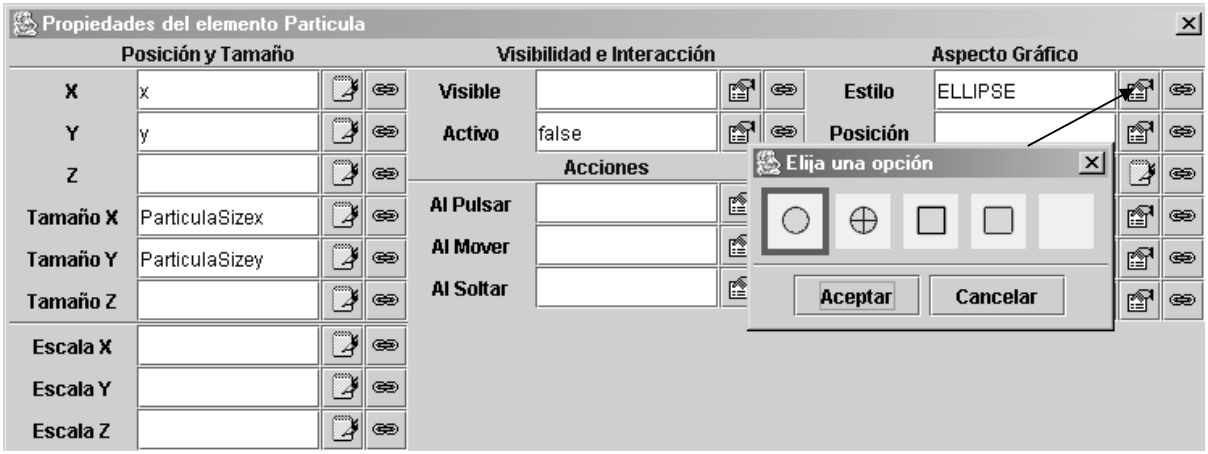


Figura 10.11: Propiedades del objeto *Particula*, de la clase Particula.

– *Estilo*: forma gráfica de dibujo del elemento. Haciendo clic sobre el botón señalado en la Figura 10.11 aparece una ventana que nos permite seleccionar entre 5 formas gráficas de dibujo para el elemento: elipse, elipse con ejes, rectángulo, rectángulo con bordes redondeados y un punto.

10.7. Algunas actividades propuestas

A continuación se describen alguna de las actividades que puede realizar el alumno usando el laboratorio.

- Describe cómo depende la frecuencia de oscilación del péndulo de la masa y longitud del mismo.
- ¿Depende la frecuencia de oscilación de la amplitud del movimiento del péndulo?
- Modifica las ecuaciones del modelo para tener en cuenta las no linealidades del modelo del péndulo simple. Para ello sustituya en la ecuación (6) θ por $\sin\theta$. ¿Depende ahora la frecuencia de oscilación de la amplitud del movimiento del péndulo?
- Añade una ventana con un gráfico donde se visualice la trayectoria del péndulo simple.

Tema 11

Conducción de calor a través de una pared múltiple

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Diseñar el algoritmo de simulación de un modelo estático y programar el modelo usando Ejs.
- Usar las clases de elementos gráficos: Poligono, Flecha y Texto.

Este laboratorio virtual está disponible en el CD del curso. Se trata de *ParedMulticapa.xml*, y se encuentra en el directorio *laboratoriosTexto*.

11.1. Descripción del modelo matemático

El objetivo de este laboratorio virtual es ilustrar el cálculo del flujo de calor por conducción a través de la pared múltiple de una cámara frigorífica[1]. Se considera la conducción de calor únicamente en una dimensión y en el estado estacionario.

La pared de la cámara frigorífica está compuesta de tres capas de diferente material, de espesores L_A, L_B y L_C. La temperatura de la superficie interior de la pared de la cámara se denomina T_1 y la temperatura de la superficie exterior T_4 (vea la Figura 11.1).

Se supone que la transferencia de calor se produce en el estacionario y se desea calcular el flujo de calor por unidad de superficie (q_x) y la temperatura en las interfaces entre los diferentes materiales (T_2, T_3). En la Figura 11.1 se representa el circuito térmico equivalente.

En primera aproximación, se supone que las conductividades térmicas de los materiales (κ_A, κ_B, κ_C) son independientes de las demás variables del modelo. Son parámetros, a los que se asignan los valores siguientes al inicializar el modelo:

[1] Este modelo matemático, correspondiente a la pared de una cámara frigorífica, está extraído del texto (Cutlip & Shacham 1999).

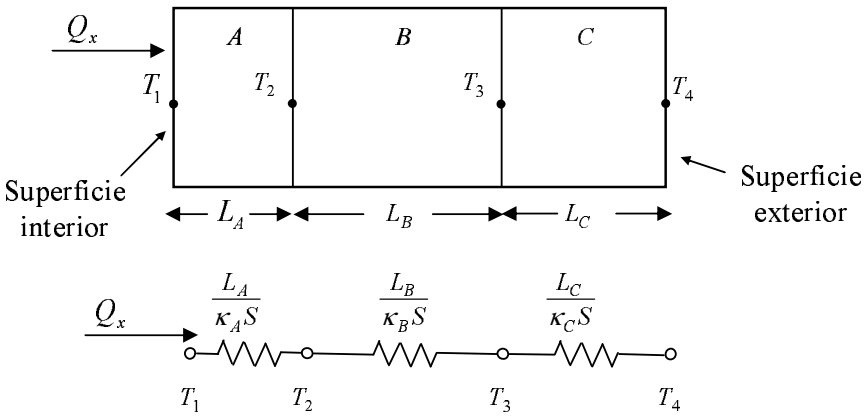


Figura 11.1: Transferencia de calor a través de una pared múltiple.

$$\kappa_A = 0.1510 \quad W \cdot m^{-1} \cdot K^{-1} \tag{11.1}$$
$$\kappa_B = 0.0433 \quad W \cdot m^{-1} \cdot K^{-1} \tag{11.2}$$
$$\kappa_C = 0.7620 \quad W \cdot m^{-1} \cdot K^{-1} \tag{11.3}$$

Los espesores de las capas que componen la pared también son parámetros del modelo. Se les asignan los valores siguientes:

$$L_A = 0.015 \quad m \tag{11.4}$$
$$L_B = 0.100 \quad m \tag{11.5}$$
$$L_C = 0.075 \quad m \tag{11.6}$$

La superficie de la pared (S) se considera constante. Puesto que se desea calcular el flujo de calor por unidad de superficie (q_x), el parámetro S no interviene en las ecuaciones del modelo, ya que

$$q_x = \frac{Q_x}{S} \tag{11.7}$$

El modelo matemático está descrito por las ecuaciones siguientes:

$$T_1 - T_2 = \frac{L_A}{\kappa_A} \cdot q_x \tag{11.8}$$
$$T_2 - T_3 = \frac{L_B}{\kappa_B} \cdot q_x \tag{11.9}$$
$$T_3 - T_4 = \frac{L_C}{\kappa_C} \cdot q_x \tag{11.10}$$

11.2. Descripción de la introducción

En la Figura 11.2 se muestra la introducción del laboratorio virtual. La figura y las fórmulas que han sido incluidas en la introducción son imágenes, en formato *.gif*, que están guardadas en el directorio *Simulations/laboratoriosTexto/Imagenes*.

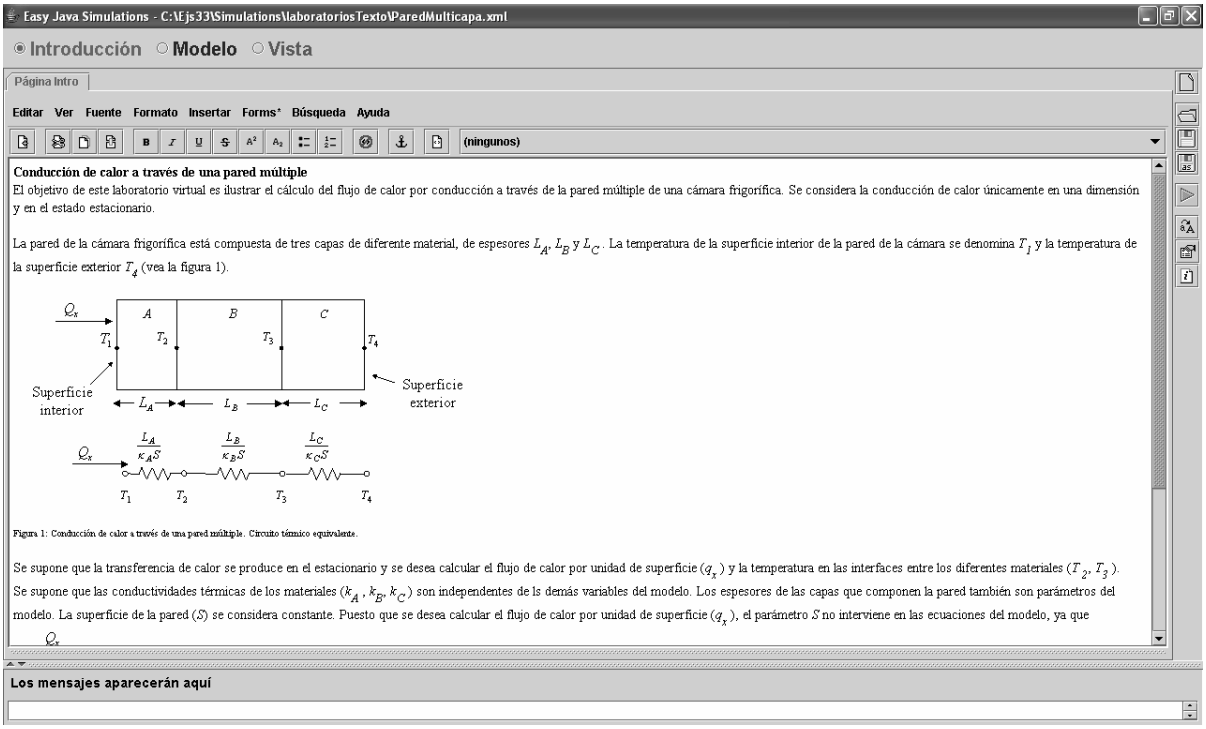


Figura 11.2: *Introducción* del laboratorio virtual.

11.3. El algoritmo de la simulación

El modelo matemático está compuesto por las Ecuaciones (11.8) – (11.10). Sus variables se clasifican de la forma siguiente:

– *Variables conocidas*: los parámetros del modelo (T_1, T_4, L_A, L_B, L_C, κ_A, κ_B, κ_C).

– *Variables desconocidas*: las variables algebraicas (T_2, T_3, q_x).

Por tratarse de un modelo en el estado estacionario, la variable tiempo no juega ningún papel, y por ello no se incluye en el modelo.

Se comprueba que el sistema no es singular, ya que puede asociarse una incógnita con cada ecuación:

$$T_1 - T_2 = \frac{L_A}{\kappa_A} \cdot q_x \quad \rightarrow \quad T_2 \tag{11.11}$$

$$T_2 - T_3 = \frac{L_B}{\kappa_B} \cdot q_x \quad \rightarrow \quad T_3 \tag{11.12}$$

$$T_3 - T_4 = \frac{L_C}{\kappa_C} \cdot q_x \quad \rightarrow \quad q_x \tag{11.13}$$

Al asignar la causalidad computacional, se observa que cada ecuación contiene dos incógnitas, es decir, que el modelo es un sistema de tres ecuaciones con tres incógnitas que debe ser resuelto. Resolviendo y ordenando las tres ecuaciones, de modo que las tres incógnitas puedan calcularse secuencialmente, una tras otra, se obtiene:

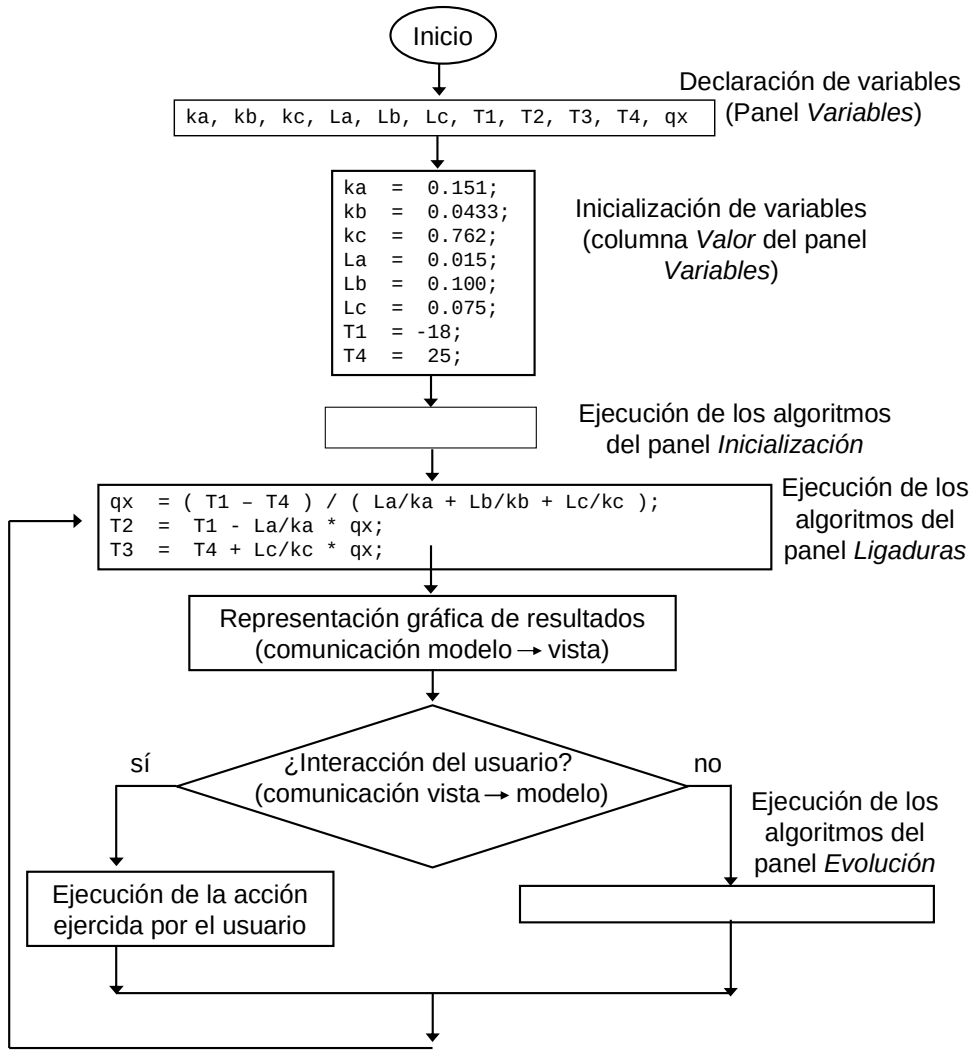


Figura 11.3: Algoritmo de la simulación.

$$[q_x] = \frac{T_1 - T_4}{\frac{L_A}{\kappa_A} + \frac{L_B}{\kappa_B} + \frac{L_C}{\kappa_C}} \tag{11.14}$$

$$[T_2] = T_1 - \frac{L_A}{\kappa_A} \cdot q_x \tag{11.15}$$

$$[T_3] = T_4 + \frac{L_C}{\kappa_C} \cdot q_x \tag{11.16}$$

La Ecuación (11.14) se obtiene de forma sencilla sumando las tres Ecuaciones (11.8) – (11.10). Se ha adoptado la notación habitual: se escribe entre corchetes la variable a evaluar de cada ecuación.

Las Asignaciones (11.14) – (11.16) deben escribirse, en el modelo en Ejs, en el panel *Ligaduras*. El algoritmo de la simulación se muestra en la Figura 11.3.

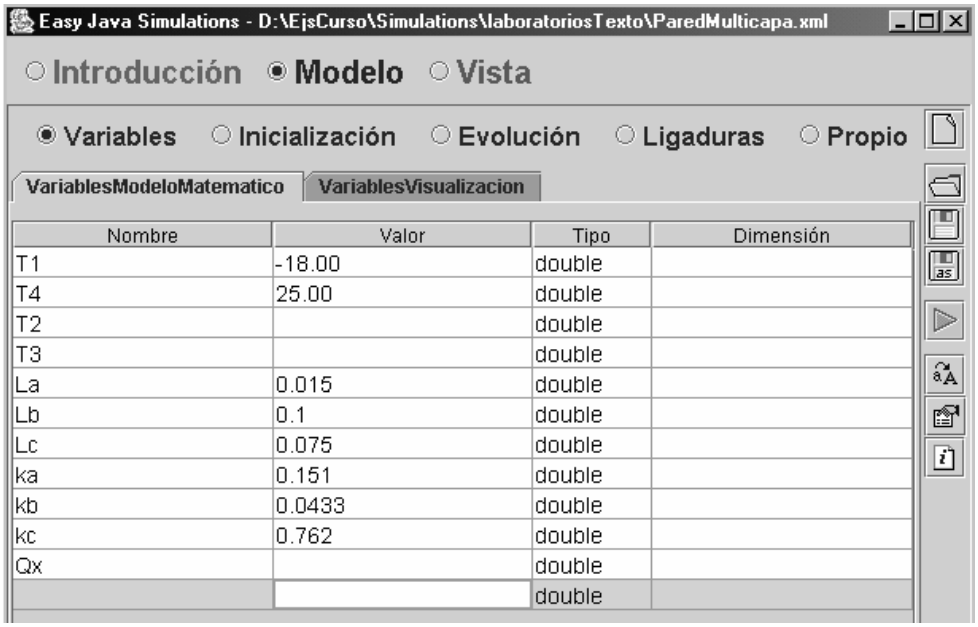


Figura 11.4: Ventana *VariablesModeloMatematico*, del panel *Variables*.

11.4. Declaración e inicialización de las variables

La declaración e inicialización de las variables se ha realizado en dos ventanas del panel *Variables*:

- *VariablesModeloMatematico*, en la que se han declarado las variables del modelo matemático (vea la Figura 11.4).
- *VariablesVisualización*, en la que se han declarado las variables empleadas para la definición de la vista.

Es una práctica muy recomendable separar la definición de ambos tipos de variables, ya que facilita la comprensión del modelo.

11.5. Programación de la vista

En la Figura 11.5 se muestra la vista del laboratorio virtual. A continuación, se explica qué elementos gráficos componen la vista.

En la Figura 11.6 puede verse el árbol de los elementos de la vista. Se compone de:

- La ventana *mainFrame* (ventana principal), que contiene un panel con botones para reinicializar la simulación, y cambiar la temperatura interior (T_1) y exterior de la pared multicapa (T_4). También, contiene elementos que permiten mostrar y ocultar las dos ventanas de diálogo *VentanaParam* y *VentanaPlots*.

  En la parte central de la ventana principal está representada esquemáticamente la pared y se muestran los valores de la temperatura de las caras interior (T_1) y exterior (T_4) de la pared, así como el flujo de calor por unidad de superficie (q_x).
- Dos ventanas de dialogo:
  - La ventana de diálogo *VentanaParam*, que permite modificar los valores de los parámetros del modelo (conductividades térmicas y espesores de las capas de la pared).

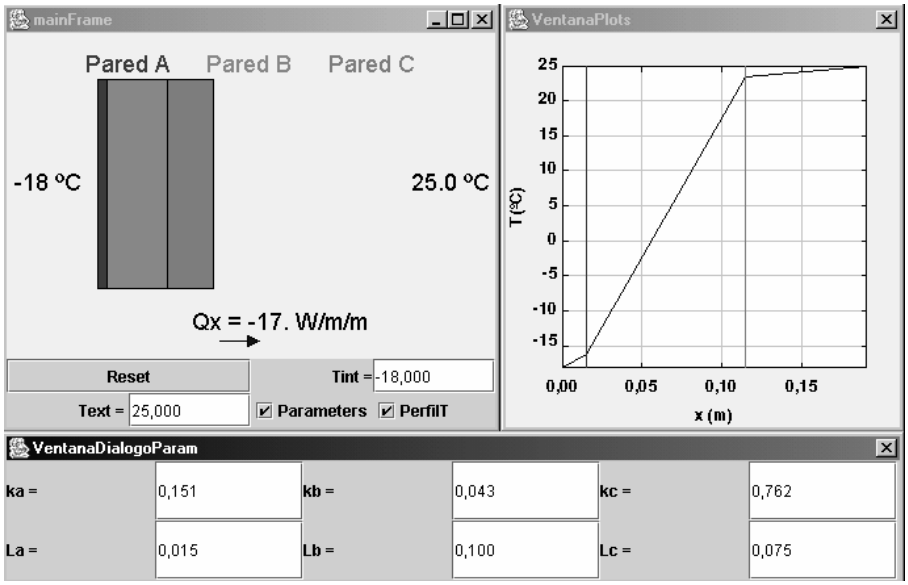


Figura 11.5: *Vista* del laboratorio virtual.

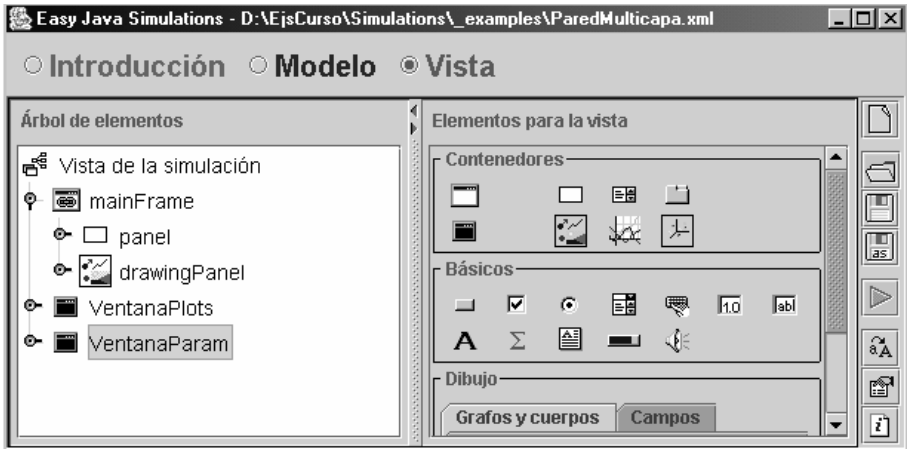


Figura 11.6: Árbol de elementos de la vista.

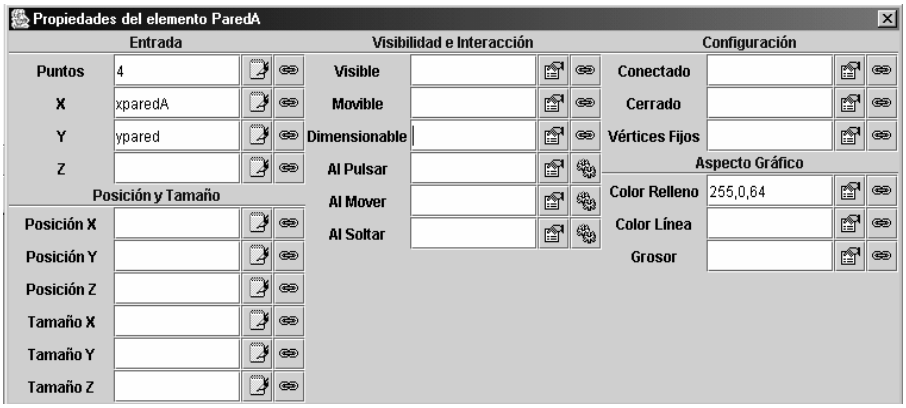


Figura 11.7: Propiedades del objeto *ParedA*, de la clase *Poligono*.

– La ventana de diálogo *VentanaPlots*, que muestra un gráfico del perfil de temperaturas a lo largo de la pared.

A continuación, se explican los aspectos más relevantes de la programación de cada una estas ventanas. Las explicaciones se centrarán en las clases de elementos gráficos que no se han empleado en los laboratorios virtuales explicados en los temas precedentes.

Programación de la ventana *mainFrame*

En la programación de la ventana *mainFrame* se han empleado tres clases de elementos gráficos de la vista que no han sido explicados hasta el momento: *Poligono*, *Flecha*, y *Texto*.

Clase de elemento gráfico *Poligono*. Este elemento de la vista se encuentra en el panel de dibujo *Grafos y cuerpos*. Se trata de un polígono cerrado, que se especifica mediante las coordenadas de sus vértices: (x, y, z) o (x, y), según se esté trabajando en tres o en dos dimensiones.

Se han empleado tres polígonos con el fin de dibujar las tres capas que forman la pared. Estos tres objetos de la clase *Poligono* se han ubicado dentro del objeto *DrawingPanel*.

Para cada uno de estos polígonos, se han especificado las propiedades siguientes:

– El número de sus vértices.
– Dos vectores, con las posiciones x e y de los vértices del polígono.

En la Figura 11.7 se muestran las propiedades del poligono *ParedA*:

– En la propiedad *Puntos* se ha introducido el número de vértices del polígono.
– En las propiedades *X* e *Y*, se han escrito las coordenadas x e y respectivamente de los vértices del polígono.

Clase de elemento gráfico *Flecha*. Se ha empleado el objeto *Flechax*, de la clase *Flecha*, para representar el sentido del flujo de calor cuando el signo de éste es positivo. La clase *Flecha* se encuentra en el panel de dibujo llamado *Grafos y cuerpos*.

En la Figura 11.11 se muestran las propiedades del objeto *Flechax*. A continuación, se describen las propiedades del objeto *Flechax* que se han especificado:

– *X*: coordenada x del origen del vector.
– *Y*: coordenada y del origen del vector.
– *Tamaño X*: tamaño del vector según el eje x.
– *Tamaño Y*: tamaño del vector según el eje y.
– *Activo*: indica mediante un valor booleano si este componente es interactivo o no.
– *Estilo*: tipo de vector a dibujar. Puede ser una flecha (*flecha*), un segmento (*segmento*) o un segmento con un cuadrado en el punto final (*cajita*).

Clase de elemento gráfico *Texto*. Esta clase de elemento se encuentra en el panel de dibujo *Básicos*. Los objetos de esta clase muestran un conjunto de caracteres en una posición que se especifica al definir el objeto. Dicha cadena de caracteres puede ser una constante o una variable, pero en cualquier caso debe ser del tipo *String*.

Los objetos de la clase *Texto* deben ubicarse en el interior de un contenedor. Por ejemplo, de la clase *PanelDibujo* o *PanelConEjes*.

Para mostrar el valor de la variable q_x, se ha creado un objeto de la clase *Texto*, al que se ha dado el nombre *TextoQ* y se ha ubicado dentro del objeto *drawingPanel*.

En la Figura 11.8 se muestran las propiedades de este elemento de la vista:

– Las propiedades *X* e *Y* indican las coordenadas x e y respectivamente donde se va a mostrar la cadena de caracteres. Estos campos se han enlazado con variables definidas en la ventana de variables *VariablesVisualización*.
– La propiedad *Texto* contiene la variable de tipo String (*QxText*) que ha sido definida en la ventana de variables VariablesDibujo.

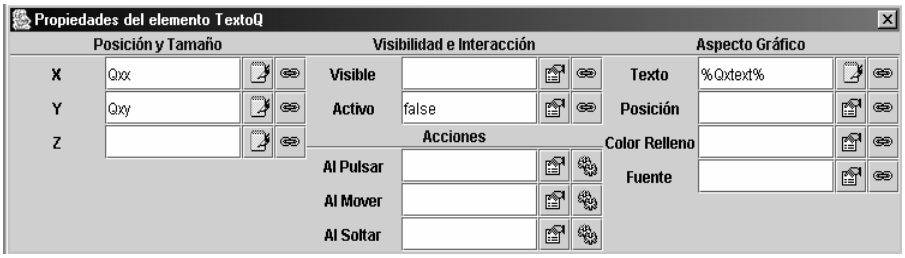


Figura 11.8: Propiedades del objeto *TextoQ*, de la clase *Texto*.

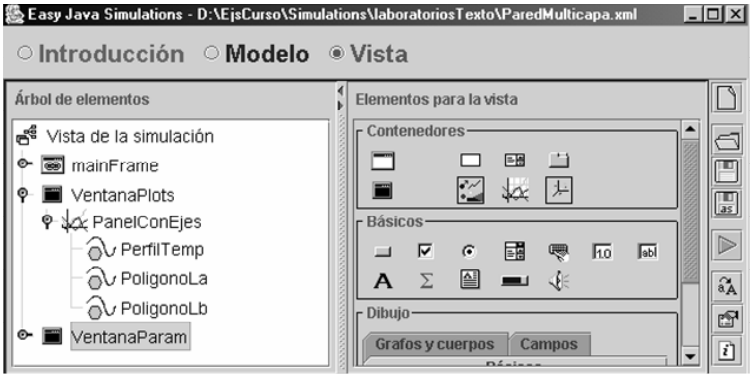


Figura 11.9: Árbol de elementos de la vista.

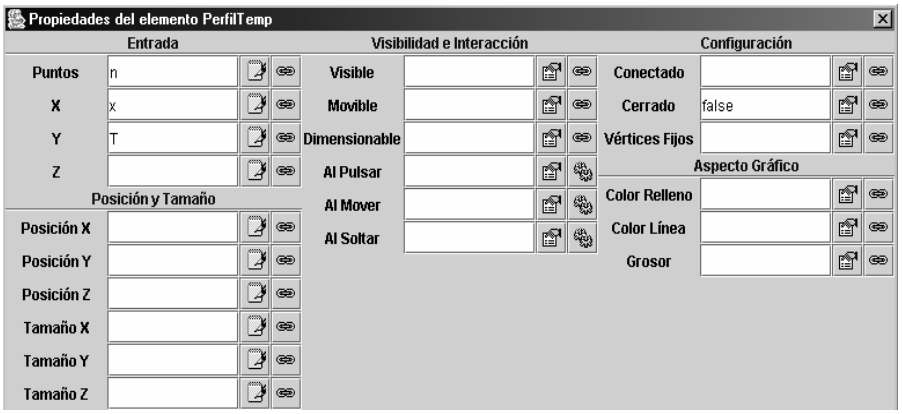


Figura 11.10: Propiedades del objeto *PerfilTemp*, de la clase *Poligono*.

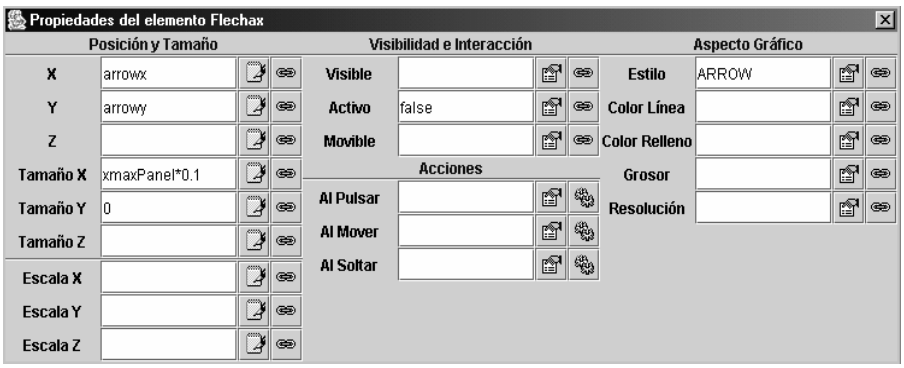


Figura 11.11: Propiedades del objeto *Flechax*, de la clase *Flecha*.

Programación de la ventana de diálogo *VentanaPlots*

La venta de diálogo *VentanaPlots* muestra un gráfico con el perfil de temperatura (vea la Figura 11.5).

Para crear este gráfico se han incorporado en la vista los elementos que se muestran en la Figura 11.9.

Como el perfil de temperaturas no depende del tiempo de evolución, se ha empleado el objeto *PerfilTemp*, de la clase *Poligono*, para realizar la gráfica. Las propiedades de dicho objeto se muestran en la Figura 11.10:

- En la propiedad *Puntos* se ha introducido el número de vértices del polígono, que es igual al número de capas más uno.
- En las propiedades *X* e *Y*, se han escrito las coordenadas x e y respectivamente de los vértices del polígono. En este caso, dos vectores (*x*, *T*) que contienen las posiciones de inicio y finalización de cada capa y la temperatura en dichas posiciones.
- Se ha asignado a la propiedad *Cerrado* el valor false, para que Ejs no conecte el primer punto del polígono con el último.

Tema 12

Laboratorio virtual de un sistema mecánico

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Realizar modelos matemáticos sencillos con estructura variable y analizar su causalidad computacional.
- Declarar e inicializar variables de tipo *array*.
- Discutir algunos conceptos fundamentales acerca de la detección de eventos en la simulación de modelos híbridos.
- Discutir el concepto de modelo con *índice superior*.
- Definir elementos en la vista de las clases *Cubo, Muelle, Partícula, Polígono* y *Flecha*.
- Definir métodos propios sencillos.

Los laboratorios virtuales explicados en este tema están disponibles en el CD del curso. Se trata de los ficheros *SistemaMecanicoSimplificado.xml* y *SistemaMecanicoCompleto.xml*, que se encuentran en el directorio *laboratoriosTexto*.

12.1. Descripción del sistema e hipótesis de modelado

En la Figura 12.1a se muestra la representación esquemática del sistema mecánico a modelar. Esta compuesto por los componentes siguientes:

- Un muelle de masa despreciable, *muelle1*, que está unido por su extremo izquierdo a una pared inmóvil y por su extremo derecho al objeto *masa1*.
- Un amortiguador: *amortiguador*.
- Dos objetos con masa:
  - Un objeto, *masa1*, de masa constante M_1, que desliza sobre el suelo sin rozamiento.
  - Un segundo objeto, *masa2*, de masa constante M_2, que desliza con rozamiento sobre la superficie superior el objeto *masa1*.

El objetivo planteado en este tema es el modelado del sistema mecánico mostrado en la Figura 12.1a, y la programación de un laboratorio virtual que ilustre su funcionamiento.

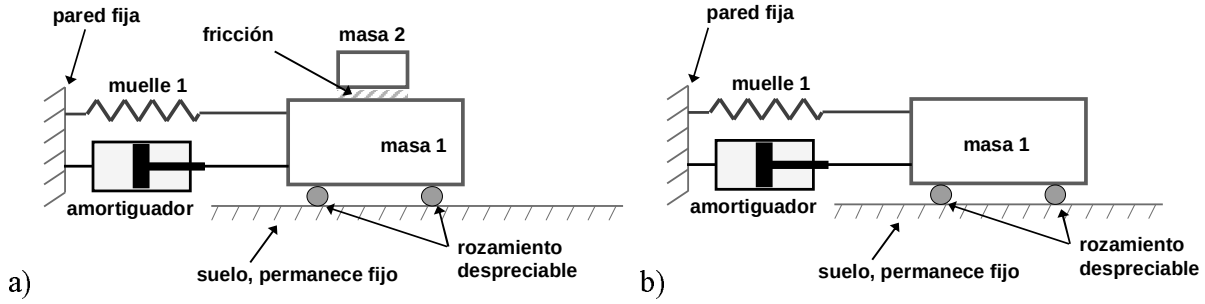


Figura 12.1: a) Sistema mecánico a modelar; b) Un subsistema de éste.

Para hacer más sencillas las explicaciones, en primer lugar se va a estudiar una versión simplificada del sistema, que es la mostrada en la Figura 12.1b. Obsérvese que este subsistema del sistema original no contiene el objeto *masa2*.

12.2. Modelado físico de los componentes

En esta sección se describe el modelado de los componentes del sistema mostrado en la Figura 12.1b. Estos son: la pared, el muelle, el amortiguador y el objeto *masa1*.

Modelado del muelle

Se supone que el muelle tienen masa despreciable, y que se comporta de forma elástica, obedeciendo la *Ley de Hooke*. Esta ley establece que:

> la fuerza (F) que ejerce el muelle sobre cualquiera de sus extremos es igual a una constante de proporcionalidad, k, multiplicada por la diferencia entre su elongación (x) y su elongación natural (x_0).

$$F = k \cdot (x - x_0) \qquad \text{(Ley de Hooke)} \tag{12.1}$$

La constante de proporcionalidad, k, se denomina *constante del muelle*, y tiene unidades de fuerza dividida por distancia: N/m.

La *elongación natural* (x_0) del muelle es su longitud cuando la fuerza ejercida es cero.

Obsérvese, que la fuerza que ejerce el muelle tiene el sentido tendente a restablecer su elongación natural (vea la Figura 12.2).

El muelle es un componente almacenador de energía:

- Cuando su elongación se aleja de la elongación natural, acumula energía.
- Cuando su elongación tiende a la elongación natural, cede energía.

Modelado del amortiguador

Se supone que el amortiguador no tiene masa y que presenta un comportamiento lineal:

- La magnitud de la fuerza ejercida por el amortiguador (F) en cada uno de sus extremos es proporcional a la velocidad relativa entre ambos extremos, es decir, a la derivada de la elongación ($v = \frac{dx}{dt}$).

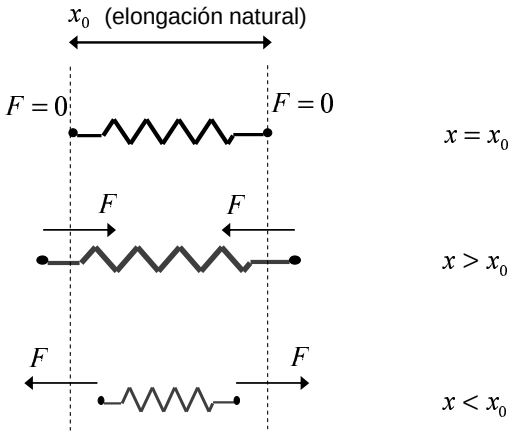


Figura 12.2: La fuerza ejercida por el muelle tiende a restablecer su elongación natural.

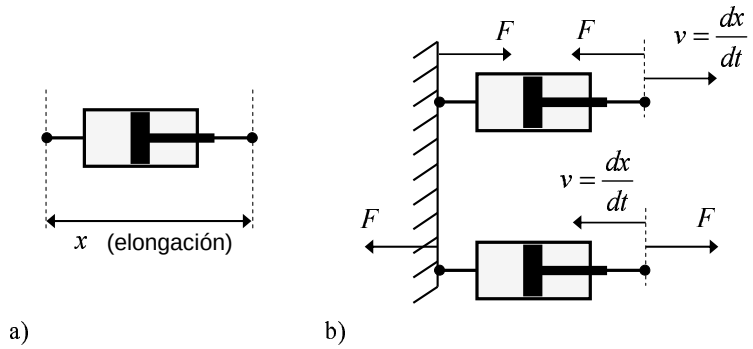


Figura 12.3: Amortiguador: a) elongación; b) fuerza ejercida sobre sus extremos.

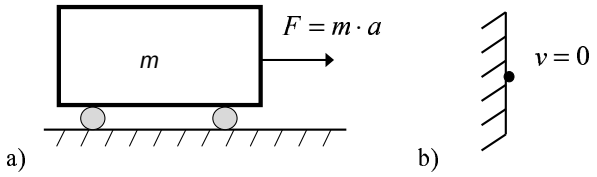


Figura 12.4: Relación constitutiva de: a) el objeto *masa1*; b) la pared.

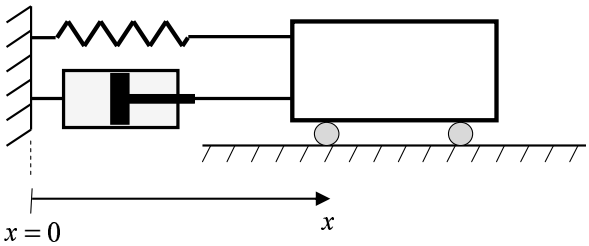


Figura 12.5: Sistema de referencia y criterio de signos.

– El sentido de la fuerza es aquel que se opone a que se produzca el cambio en la elongación.

La relación constitutiva del amortiguador es:

$$F = b \cdot \frac{dx}{dt} \tag{12.2}$$

La elongación del amortiguador (x) es la distancia que existe en cada instante entre sus extremos (vea la Figura 12.3a).

Para ilustrar el comportamiento del amortiguador, consideremos que tiene uno de sus extremos unido a una pared fija. En la Figura 12.3b se muestra el sentido de la fuerza ejercida por el amortiguador sobre sus extremos. Obsérvese que:

– En el extremo libre, la velocidad del mismo y la fuerza aplicada por el amortiguador llevan sentido opuesto.
– La fuerza que ejerce el amortiguador sobre la pared tiene la misma magnitud, pero sentido opuesto, a la fuerza que ejerce el amortiguador sobre su extremo libre.
– Por otra parte, la fuerza que ejerce la pared sobre el extremo del amortiguador es igual, y de sentido opuesto, a la que ejerce el extremo fijo del amortiguador sobre la pared.

Al contrario de lo que sucede con el muelle, que es un elemento almacenador de energía potencial, el amortiguador es un componente que disipa energía, es decir, transforma la energía cinética en calor.

Modelado del objeto *masa1*

La masa del objeto permanece constante. En consecuencia, la fuerza neta aplicada sobre el objeto (F) es igual a su masa (m) multiplicada por la aceleración (a) que adquiere (ver la Figura 12.4a):

$$F = m \cdot a \tag{12.3}$$

Modelado de la pared

La fuerza que ejerce la pared sobre el punto de conexión a ella de cualquier componente es la necesaria para que éste permanezca en reposo (ver la Figura 12.4b). Por tanto, este componente se describe mediante la ecuación:

$$v = 0 \tag{12.4}$$

12.3. Modelo matemático del sistema simplificado

En primer lugar, se va a establecer el sistema de referencia y el criterio de signos para la posición, la velocidad, la aceleración y la fuerza. A continuación, se formulará el modelo matemático del sistema.

Referencia y criterio de signos

Debe fijarse el sistema de referencia para la coordenada espacial: se considera que el origen de coordenadas se encuentra en la pared (vea la Figura 12.5).

La coordenada espacial de los puntos situados a la derecha de la pared tiene signo positivo, mientras que los puntos situados a la izquierda de la pared tienen signo negativo.

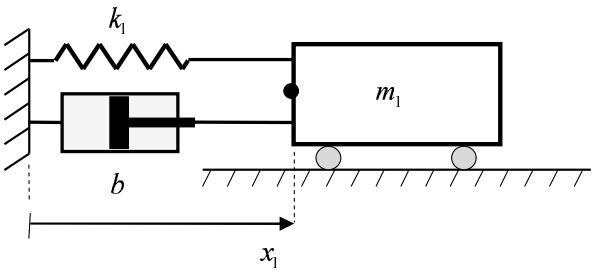


Figura 12.6: La coordenada del punto de conexión se denomina x_1.

La velocidad y la fuerza se considerarán positivas cuando lleven el sentido creciente de la coordenada espacial.

Modelo matemático

Sean x_1, v_1 y a_1 la posición, la velocidad y la aceleración respectivamente del punto de conexión entre el muelle, el amortiguador y el objeto *masa1*. Este punto se encuentra señalado mediante un círculo negro en la Figura 12.6.

En el punto de conexión entre el muelle, el amortiguador y el objeto *masa1*, se verifica que la fuerza ejercida sobre *masa1* (F_{masa1}) es igual a la suma de las fuerzas ejercidas por el muelle ($F_{muelle1}$) y por el amortiguador ($F_{amortiguador}$) sobre dicho punto de conexión:

$$F_{masa1} = F_{muelle1} + F_{amortiguador} \quad (12.5)$$
$$F_{muelle1} = -k_1 \cdot (x_1 - x_{0,muelle1}) \quad (12.6)$$
$$F_{amortiguador} = -b \cdot v_1 \quad (12.7)$$

Asimismo, la fuerza ejercida sobre el objeto *masa1* y la aceleración que adquiere éste, están relacionadas de la forma siguiente:

$$F_{masa1} = m_1 \cdot a_1 \quad (12.8)$$

El modelo se completa, teniendo en cuenta que la velocidad es la derivada de la posición, y que la aceleración es la derivada de la velocidad.

El modelo matemático del sistema mostrado en la Figura 12.1a es el siguiente:

$$F_{masa1} = F_{muelle1} + F_{amortiguador} \quad (12.9)$$
$$F_{muelle1} = -k_1 \cdot (x_1 - x_{0,muelle1}) \quad (12.10)$$
$$F_{amortiguador} = -b \cdot v_1 \quad (12.11)$$
$$F_{masa1} = m_1 \cdot a_1 \quad (12.12)$$
$$\frac{dx_1}{dt} = v_1 \quad (12.13)$$
$$\frac{dv_1}{dt} = a_1 \quad (12.14)$$

Las variables de este modelo pueden clasificarse de la manera siguiente:

– Parámetros: m_1, k_1, b, $x_{0,muelle1}$.

– Variables de estado: x_1, v_1.
– Variables algebraicas: a_1, F_{masa1}, $F_{muelle1}$, $F_{amortiguador}$.

Obsérvese que el modelo está compuesto por 6 ecuaciones y 6 incógnitas: las 4 variables algebraicas más las 2 derivadas de los estados ($derx1$ y $derv1$).

12.4. El algoritmo de la simulación

El primer paso para plantear el algoritmo de la simulación del modelo es asignar su causalidad computacional. Esto permitirá determinar en qué orden debe resolverse el modelo.

Asignación de la causalidad computacional

La asignación de la causalidad computacional consiste en decidir qué incógnita se evalúa de cada ecuación del modelo, y en qué orden deben resolverse las ecuaciones del modelo.

La causalidad computacional de un modelo se decide siguiendo las dos reglas siguientes:

– **Regla 1:** Si una ecuación posee una única incógnita, debe emplearse para calcularla.
– **Regla 2:** Si una incógnita aparece sólo en una ecuación, debe calcularse de ella.

Si el modelo matemático está correctamente planteado (es decir, no es singular), entonces aplicando estas dos reglas se llega a una de estas dos posibles situaciones:

1. Todas las incógnitas pueden ser resueltas una tras otra, en secuencia.
2. Se llega a un punto en que todas las ecuaciones tienen al menos 2 incógnitas, y todas las incógnitas aparecen al menos en dos ecuaciones. Es decir, el modelo contiene sistemas de ecuaciones que deben resolverse simultáneamente. Estos sistemas de ecuaciones se denominan *lazos algebraicos*.

Asignemos la causalidad computacional al modelo formado por las Ecs. (12.9) – (12.14). Para ello, en primer lugar, sustituimos las derivadas de las variables de estado por variables auxiliares:

– Sustituimos $\frac{dx_1}{dt}$ por la variable $derx_1$.
– Sustituimos $\frac{dv_1}{dt}$ por la variable $derv_1$.

Realizada esta sustitución, se obtiene el modelo siguiente:

$$F_{masa1} = F_{muelle1} + F_{amortiguador} \quad (12.15)$$
$$F_{muelle1} = -k_1 \cdot (x_1 - x_{0,muelle1}) \quad (12.16)$$
$$F_{amortiguador} = -b \cdot v_1 \quad (12.17)$$
$$F_{masa1} = m_1 \cdot a_1 \quad (12.18)$$
$$derx_1 = v_1 \quad (12.19)$$
$$derv_1 = a_1 \quad (12.20)$$

Las incógnitas a calcular son las variables: a_1, F_{masa1}, $F_{muelle1}$, $F_{amortiguador}$, $derx_1$, $derv_1$.

A continuación, se aplican las dos reglas anteriormente descritas a las Ecs. (12.15) – (12.20). A medida que se asigna la causalidad computacional, se va señalando la variable a evaluar de cada ecuación incluyéndola entre corchetes.

1. La variable $F_{muelle1}$ es la única incógnita de la Ec. (12.16). Por tanto:

$$[F_{muelle1}] = -k_1 \cdot (x_1 - x_{0,muelle1}) \quad (12.21)$$

2. La variable $F_{amortiguador}$ es la única incógnita de la Ec. (12.17):

$$[F_{amortiguador}] = -b \cdot v_1 \quad (12.22)$$

3. La variable $derx_1$ es la única incógnita de la Ec. (12.19):

$$[derx_1] = v_1 \tag{12.23}$$

4. Una vez $F_{muelle1}$ y $F_{amortiguador}$ han sido calculadas de las Ecs. (12.16) y (12.17) respectivamente, la única incógnita que hay en la Ec. (12.15) es F_{masa1}:

$$[F_{masa1}] = F_{muelle1} + F_{amortiguador} \tag{12.24}$$

5. Una vez calculada F_{masa1}, la única incógnita de la Ec. (12.18) es a_1:

$$F_{masa1} = m_1 \cdot [a_1] \tag{12.25}$$

6. Una vez calculada a_1, la única incógnita de la Ec. (12.20) es $derv_1$.

$$[derv_1] = a_1 \tag{12.26}$$

Así pues, despejando al lazo izquierdo de la igualdad la variable a evaluar de cada ecuación, y ordenando las ecuaciones de modo que puedan ser resueltas en secuencia, se obtiene el modelo ordenado y resuelto:

$$\begin{aligned}
[F_{muelle1}] &= -k_1 \cdot (x_1 - x_{0,muelle1}) && (12.27)\\
[F_{amortiguador}] &= -b \cdot v_1 && (12.28)\\
[derx_1] &= v_1 && (12.29)\\
[F_{masa1}] &= F_{muelle1} + F_{amortiguador} && (12.30)\\
[a_1] &= \frac{F_{masa1}}{m_1} && (12.31)\\
[derv_1] &= a_1 && (12.32)
\end{aligned}$$

El algoritmo de la simulación

Para escribir las ecuaciones diferenciales (12.29) y (12.32) en una página EDO del panel *Evolución* de Ejs, es preciso expresar las derivadas en función de las variables de estado, los parámetros y el tiempo.

A partir del modelo ordenado y resuelto (es decir, de las Ecs. (12.27) – (12.32)), es muy sencillo expresar las derivadas de la forma anteriormente indicada. Se obtiene:

$$\begin{aligned}
[derx_1] &= v_1 && (12.33)\\
[derv_1] &= \frac{-k_1 \cdot (x_1 - x_{0,muelle1}) - b \cdot v_1}{m_1} && (12.34)
\end{aligned}$$

El algoritmo para la simulación del modelo empleando Ejs es el mostrado en la Figura 12.7.

Obsérvese que las variables $derx_1$ y $derv_1$, que se han introducido sustituyendo a las derivadas, han resultado útiles para realizar la asignación de la causalidad computacional y para expresar las derivadas en función de los estados, los parámetros y el tiempo. Sin embargo, puesto que en este ejemplo estas dos variables no son necesarias para calcular ninguna de las variables relevantes del sistema, las Ecs. (12.29) y (12.32) no se incluyen en el modelo en Ejs (ver las ecuaciones del panel *Ligaduras* en la Figura 12.7).

La variable $incremento\_t$ es el tamaño del paso en el tiempo del método de integración. Esta variable debe ser definida (en el panel *Variables* e introducida en la casilla *Incremento* de la página EDO (vea la Figura 12.9.b).

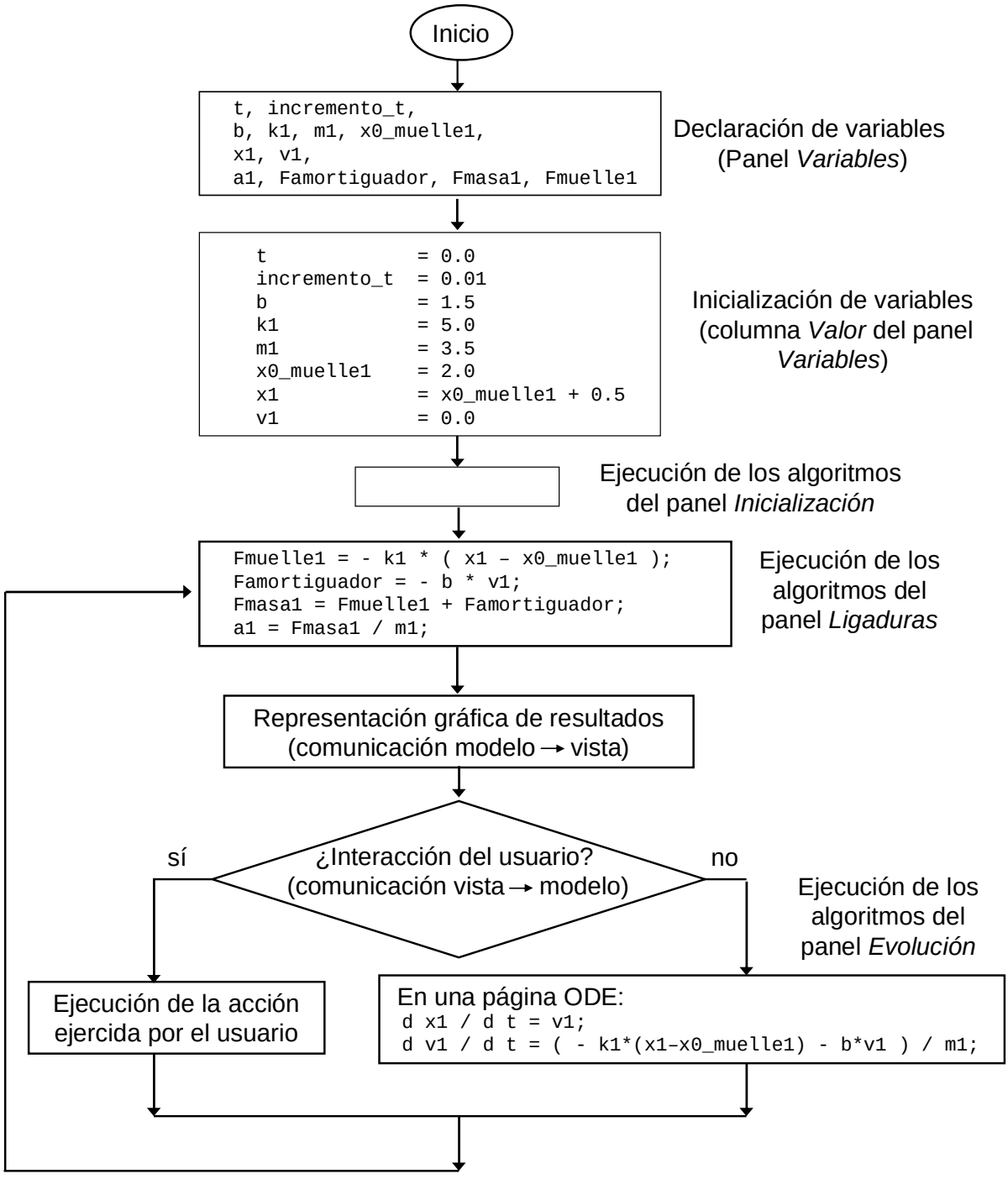


Figura 12.7: Algoritmo de la simulación del sistema mostrado en la Figura 12.1b.

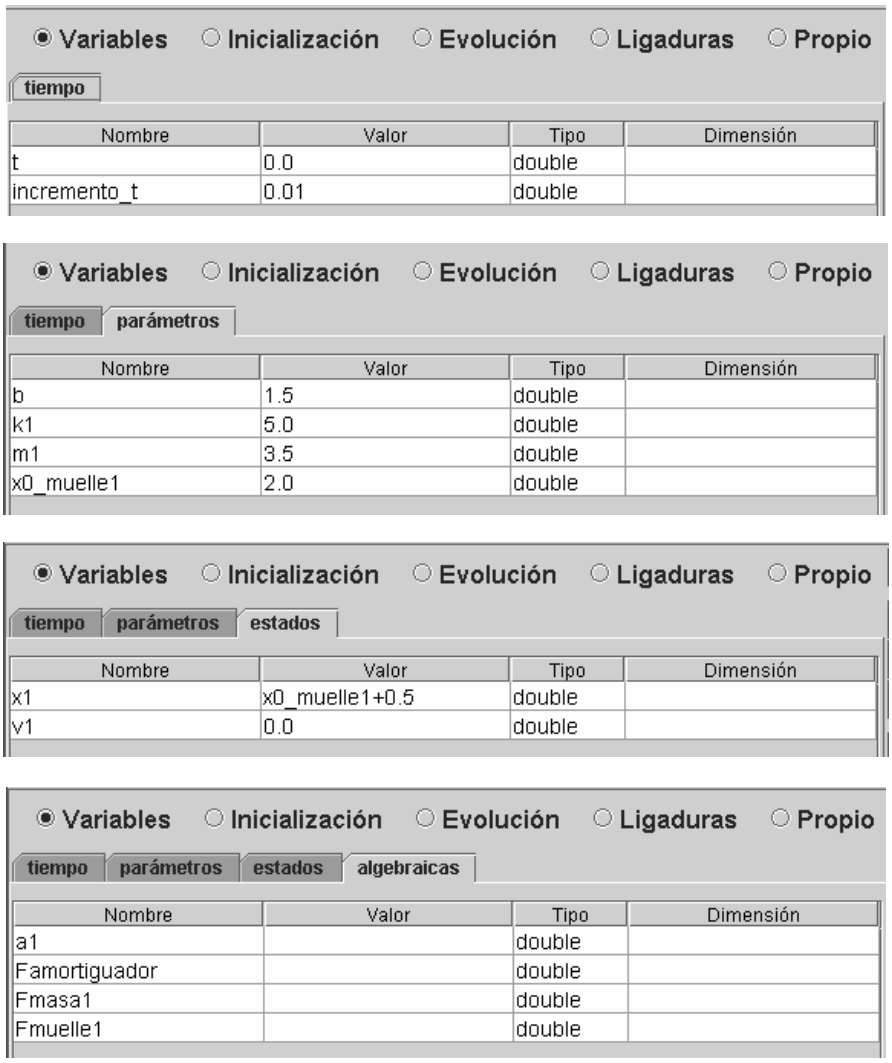


Figura 12.8: Páginas de definición e inicialización de las variables del modelo en Ejs.

Una vez planteado el algoritmo de la simulación, la programación del modelo en Ejs es inmediata. En las siguientes secciones se describe cómo hacerlo.

12.5. Declaración e inicialización de las variables en Ejs

Existen múltiples formas de definir las variables en el panel *Variables* de Ejs. Una de ellas es definir las páginas de modo que agrupen los diferentes tipos de variables:

- tiempo
- parámetros
- estados
- variables algebraicas

En la Figura 12.8 se muestran las páginas con la definición de las variables del modelo. Obsérvese que se ha asignado valor inicial a la variable tiempo, a los parámetros y a las variables de estado.

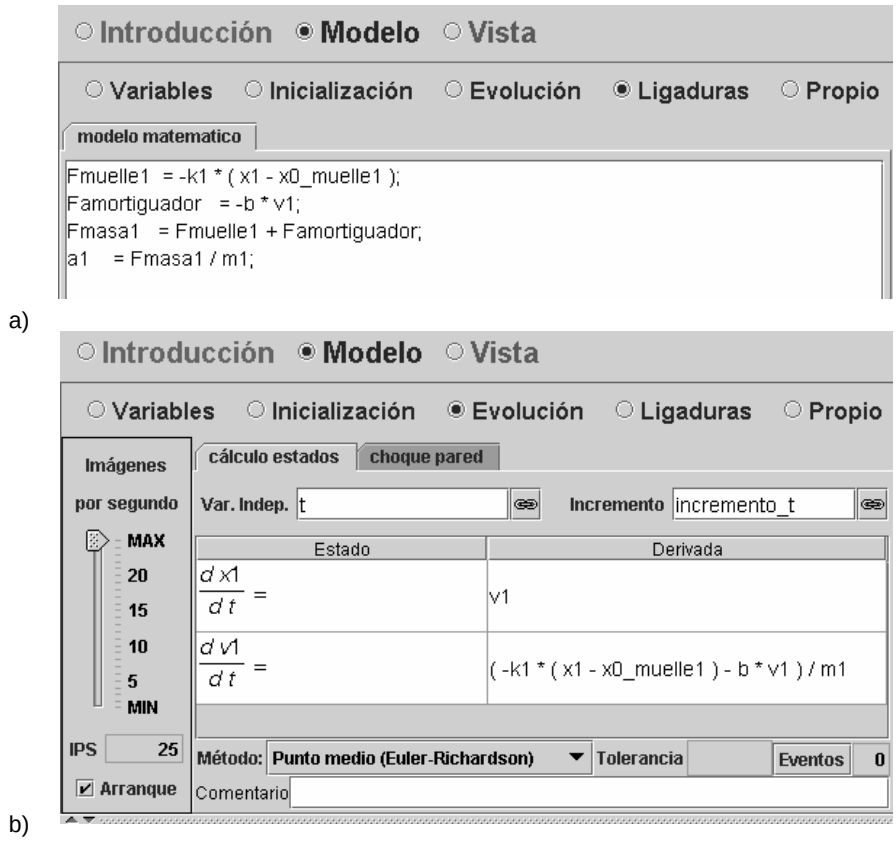


Figura 12.9: Paneles: a) *Ligaduras*; b) *Evolución*.

El valor inicial de las variables algebraicas se calculará al ejecutar el algoritmo del panel *Ligaduras*. Por ello, no es preciso escribirlo en los paneles *Variables* o *Inicialización*.

12.6. Definición del modelo en Ejs

Como se ha indicado anteriormente, una vez que se ha planteado el algoritmo de la simulación para el modelo (vea la Figura 12.7), la definición del modelo en Ejs se realiza de forma inmediata. En la Figura 12.9 se muestra el contenido de los paneles *Ligaduras* y *Evolución*.

12.7. Definición de la vista

En esta sección se describen los pasos a seguir para la programación de la vista del laboratorio virtual, que es mostrada en la Figura 12.13.

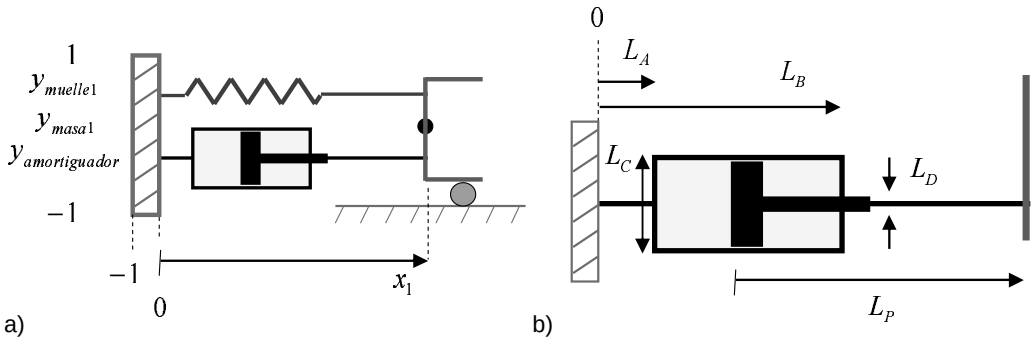
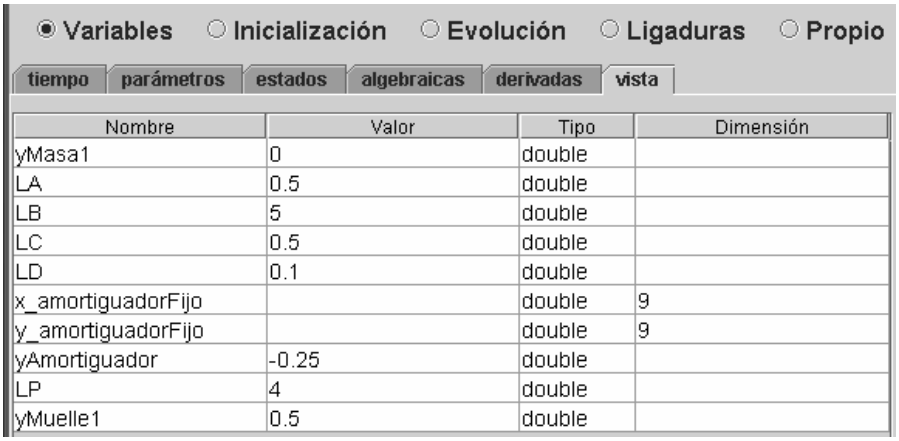


Figura 12.10: Variables necesarias para la definición de la vista.



Nombre	Valor	Tipo	Dimensión
yMasa1	0	double	
LA	0.5	double	
LB	5	double	
LC	0.5	double	
LD	0.1	double	
x_amortiguadorFijo		double	9
y_amortiguadorFijo		double	9
yAmortiguador	-0.25	double	
LP	4	double	
yMuelle1	0.5	double	

Figura 12.11: Definición de las variables de la vista.

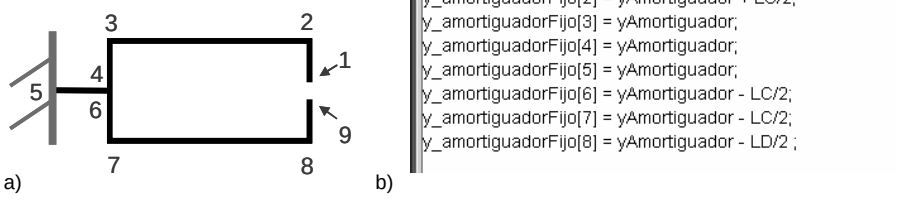


Figura 12.12: Variables para la definición de la parte fija del amortiguador.

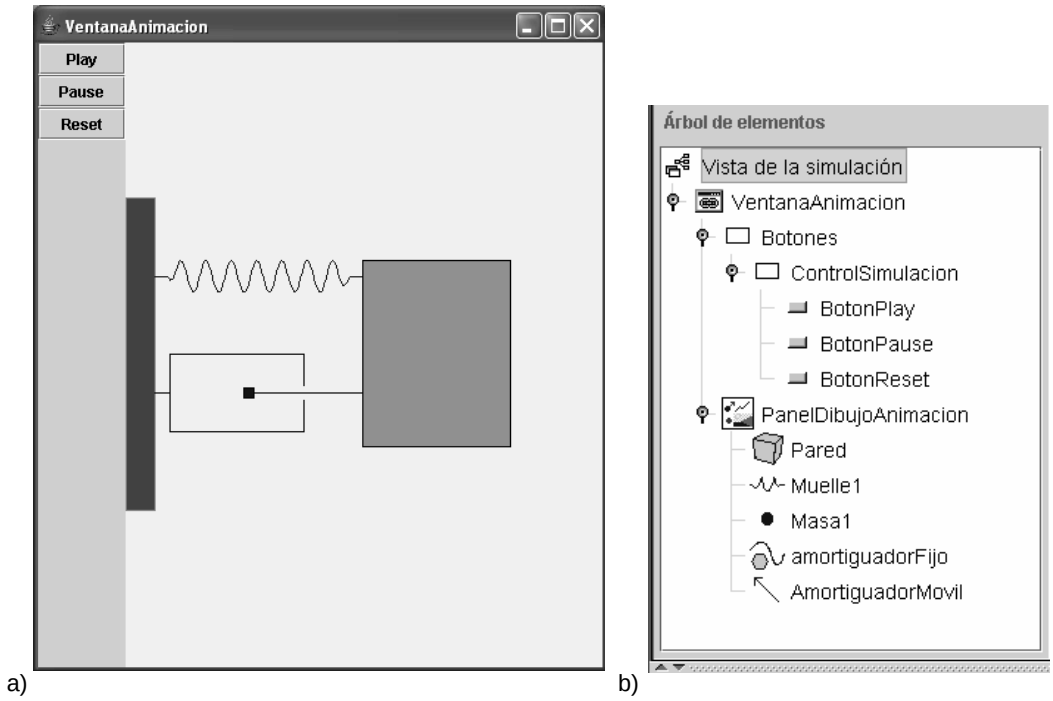


Figura 12.13: a) Vista del laboratorio virtual; b) Árbol de elementos de la vista.

Variables de la vista

Frecuentemente, la programación de la vista requiere del empleo de variables que no intervienen en el modelo matemático. Estas variables se introducen específicamente para definir la posición y las dimensiones de los elementos gráficos que componen la vista.

En este caso, la programación de la vista requiere de la definición de las variables mostradas en la Figura 12.10.

En la Figura 12.10a se muestran las variables que definen la posición vertical del amortiguador ($y_{amortiguador}$), de la masa (y_{masa1}) y del muelle ($y_{muelle1}$). También, se establecen las dimensiones de la pared, y la posición horizontal del punto de unión de los tres elementos (x_1).

En la Figura 12.10b se muestran las variables empleadas para la definición de la vista del amortiguador.

Finalmente, en la Figura 12.11 se muestra el panel de Ejs en el que se definen e inicializan las variables descritas anteriormente. Obsérvese que se han definido dos variables vectoriales: $x_{amortiguadorFijo}$ e $y_{amortiguadorFijo}$. En la columna *Dimensión* se ha escrito 9. Esto indica que cada una de estas variables vectoriales está compuesta de 9 valores. Estos valores definen las coordenadas de polígono que representa la parte fija del amortiguador. La inicialización de las variables vectoriales debe realizarse en el panel *Inicialización* (vea la Figura 12.12).

Componentes de la vista

En la Figura 12.13b se muestra el árbol de elementos de la vista. A continuación, se describen las propiedades de los elementos que lo componen.

En la Figura 12.14 se muestran las propiedades de los elementos *VentanaAnimacion*, *Botones* y *ControlSimulacion*. El elemento *VentanaAnimacion* contiene a todos los demás elementos de la vista (vea la Figura 12.13a), los cuales a su vez están distribuidos en los dos elementos siguientes:

- *Botones*: que da lugar a la regleta con los botones situada en la parte izquierda de la vista.
- *PanelDibujoAnimación*: que da lugar al diagrama animado del sistema, mostrado en la parte central de la vista.

Control de la simulación

El elemento *ControlSimulacion* contiene los elementos *BotonPlay*, *BotonPause* y *BotonReset*, que están situados en la posición arriba, centro y abajo respectivamente. Estos tres elementos pertenecen a la clase *Botón*. Sus propiedades se muestran en la Figura 12.15.

Componentes de la animación

En la Figura 12.16 se muestran las propiedades del elemento *PanelDibujoAnimacion*, que es de la clase *PanelDibujo*. Como se indica en la figura, los campos *Mínimo X*, *Máximo X*, *Mínimo Y* y *Máximo Y* determinan las coordenadas del extremo inferior izquierdo (punto x) y del extremo superior derecho (punto Y) del área de dibujo del elemento *PanelDibujo*.

Obsérvese que el elemento *Botones* está situado en la posición izquierda, dentro del elemento *VentanaAnimación*, y que elemento *PanelDibujoAnimación* está situado en el centro.

El elemento *Pared*

En las Figura 12.17 se muestran las propiedades del elemento *Pared*, de la clase *Cubo*. Los campos *Origen X* y *Origen Y* definen las coordenadas horizontal y vertical respectivamente del vértice inferior izquierdo del cubo. Los campos *Tamaño X* y *Tamaño Y* determinan la anchura y la altura del cubo respectivamente.

El elemento *Muelle1*

En la Figura 12.18 se muestran las propiedades del elemento *Muelle1*, que es de la clase *Muelle*. Los campos X e Y especifican las coordenadas de uno de los extremos del muelle (en este caso, del extremo unido a la pared). Los campos *Tamaño X* y *Tamaño Y* especifican la coordenada horizontal y vertical de la distancia entre el otro extremo del muelle (el que está unido a la *masa1*) y el extremo definido anteriormente (unido a la pared).

El elemento *Masa1*

En la Figura 12.19 se muestran las propiedades del elemento *Masa1*, de la clase *Partícula*. Las coordenadas horizontal y vertical de la posición del elemento, vienen determinadas por las variables x_1 e y_{masa1} respectivamente, según se especifica en los campos X e Y de la ventana de propiedades. El punto del objeto al que hace referencia su posición se define en el campo *Posición*. En este caso, el punto empleado para determinar la posición del objeto es el punto central de su lado izquierdo (vea la parte superior derecha de la Figura 12.19).

Se pretende que el objeto *Masa1* sea interactivo, es decir, que en cualquier instante de la simulación pueda cambiarse su posición pinchando sobre él y arrastrándolo con el ratón. Para ello, debe escribirse el valor *true* en el campo *Activo* (vea la Figura 12.19).

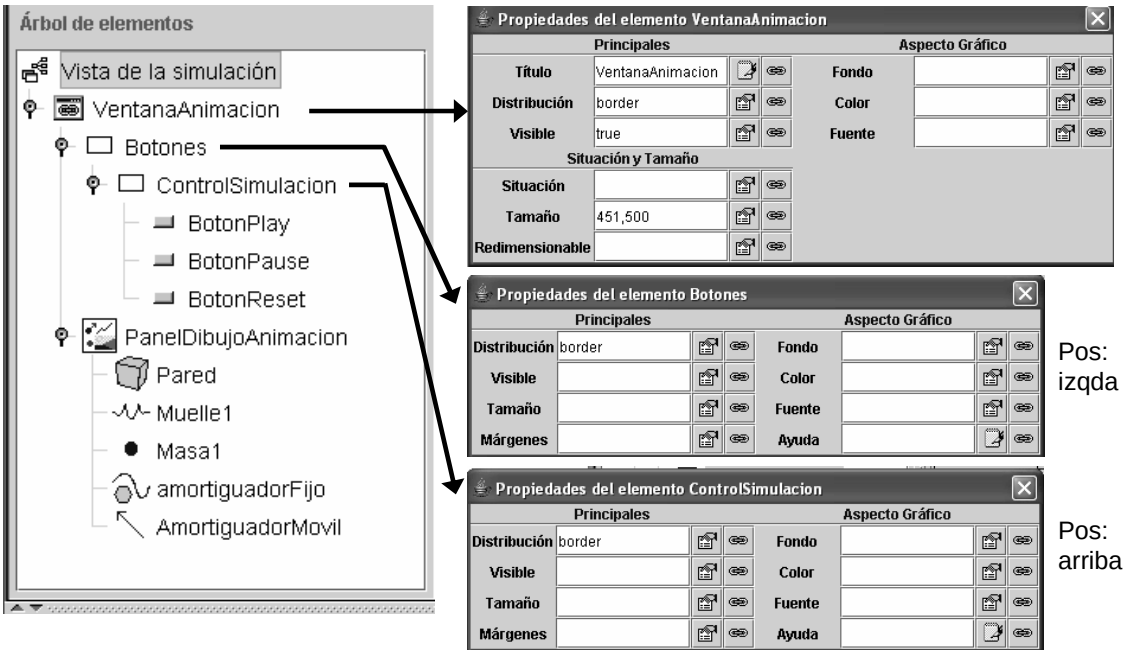


Figura 12.14: Propiedades de los elementos *VentanaAnimacion*, *Botones*, *ControlSimulacion*.

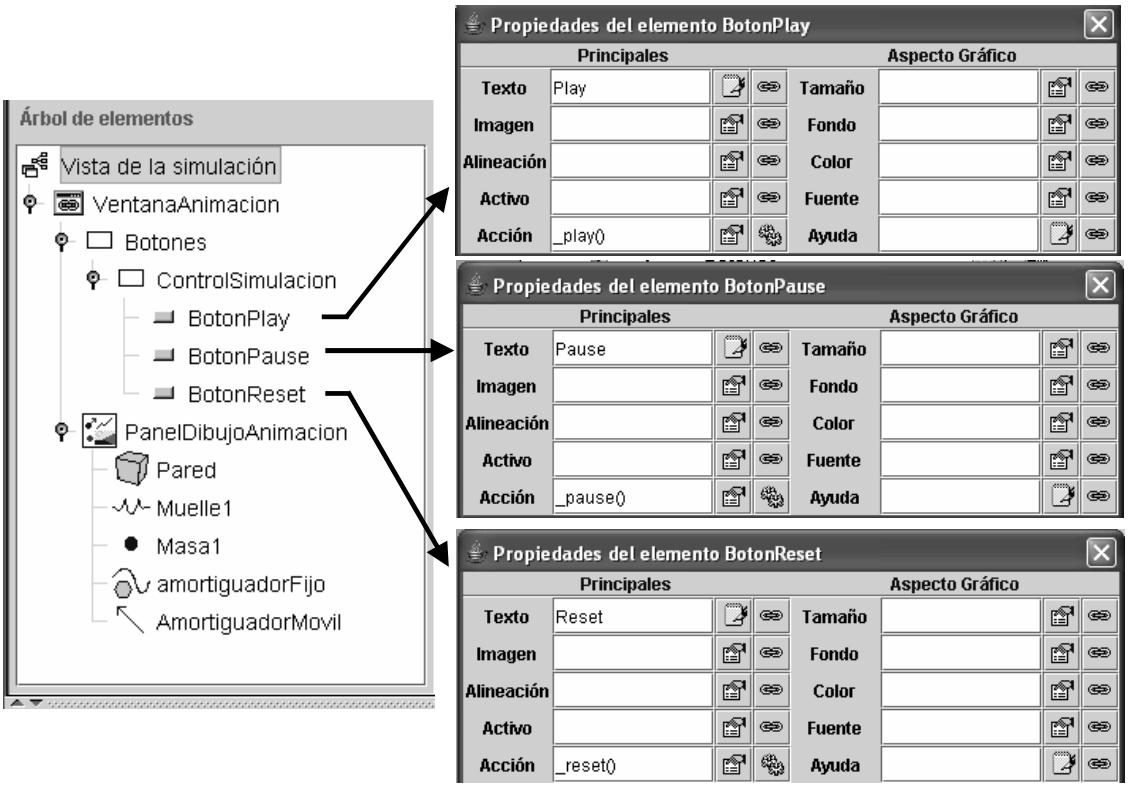


Figura 12.15: Propiedades de los elementos *BotonPlay*, *BotonPause* y *BotonReset*.

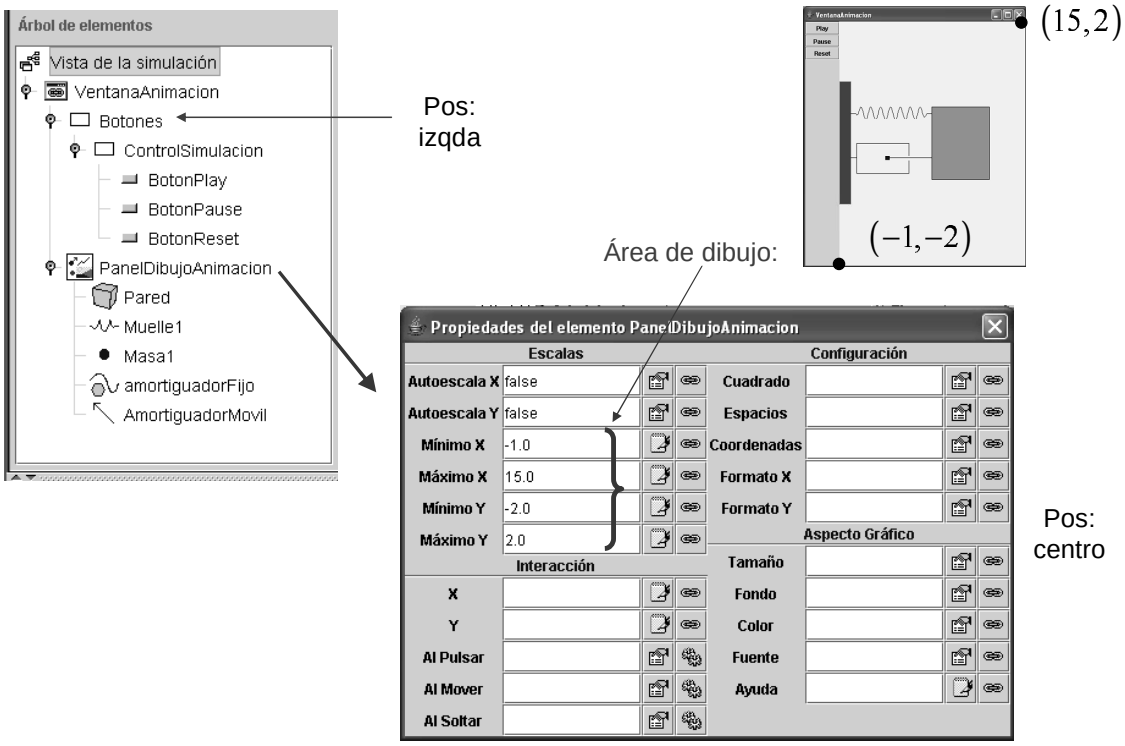


Figura 12.16: Propiedades del elemento *PanelDibujoAnimacion*.

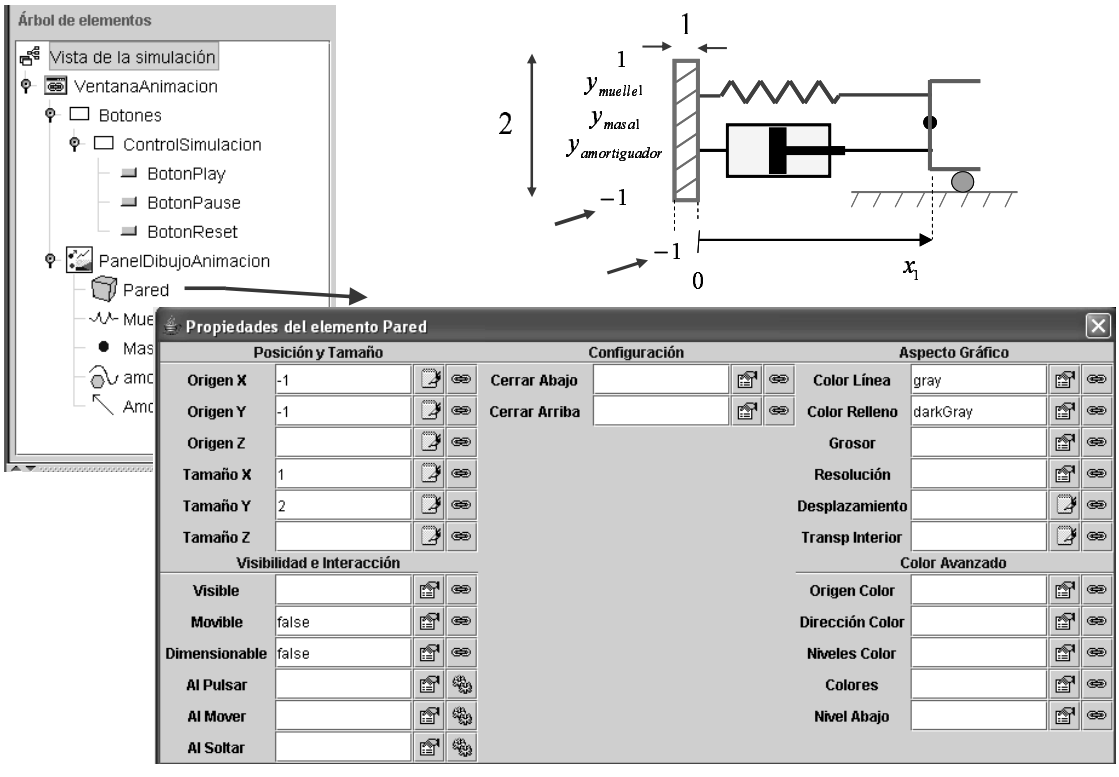


Figura 12.17: Propiedades del elemento *Pared*, de la clase *Cubo*.

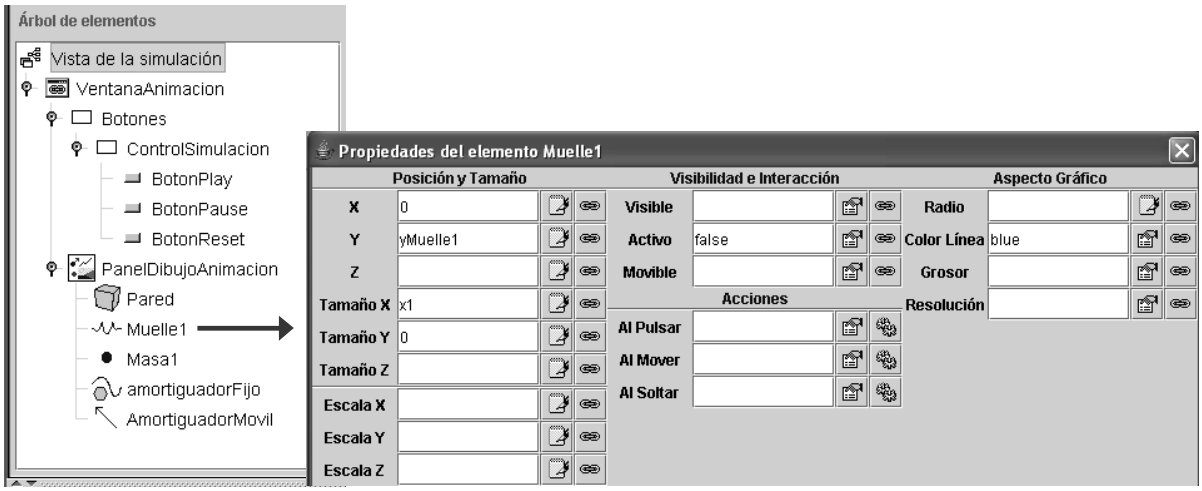


Figura 12.18: Propiedades del elemento *Muelle1*, de la clase *Muelle*.

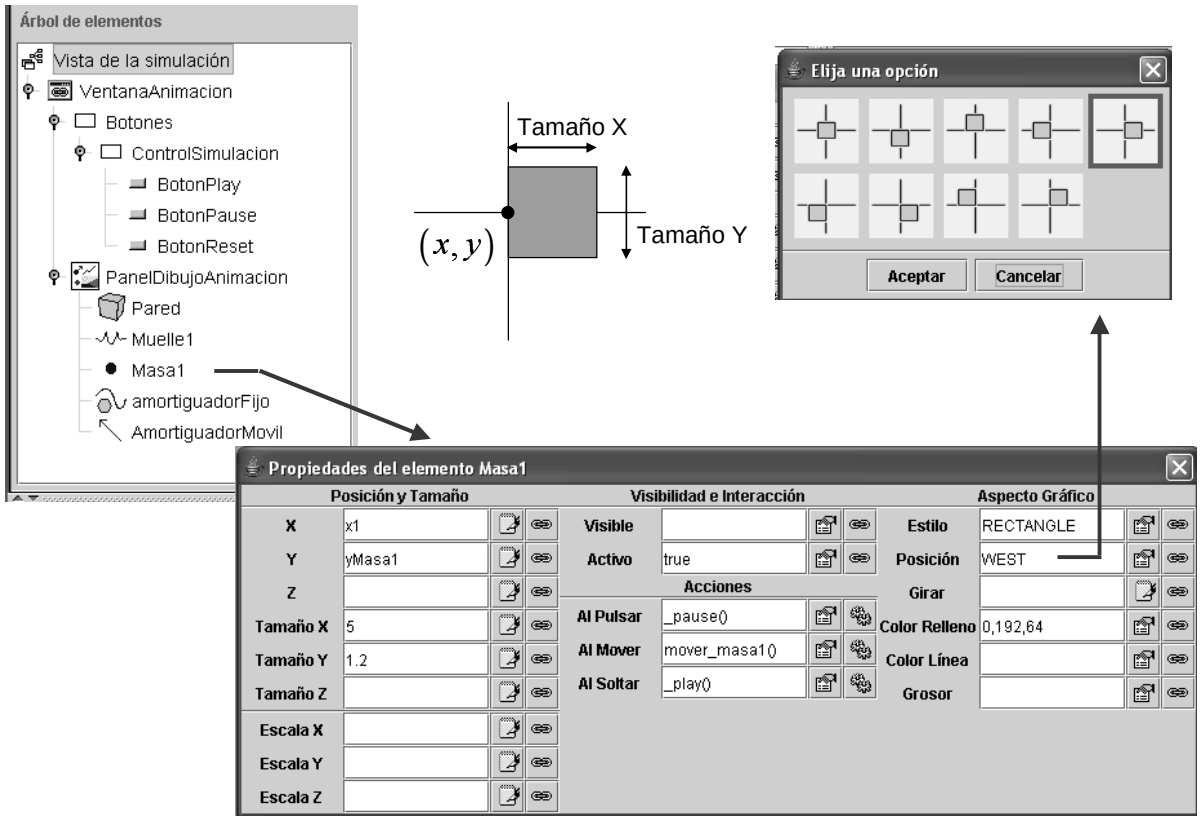
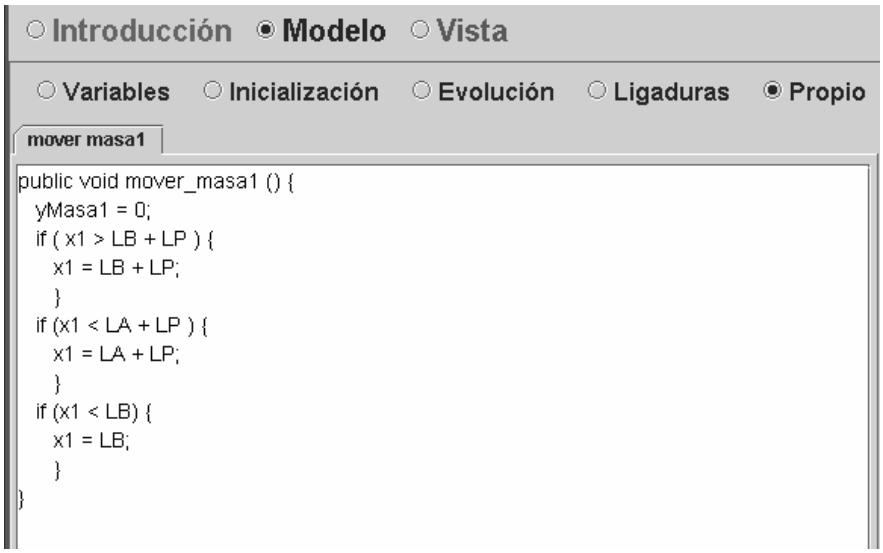


Figura 12.19: Propiedades del elemento *Masa1*, de la clase *Partícula*.

a)



b)

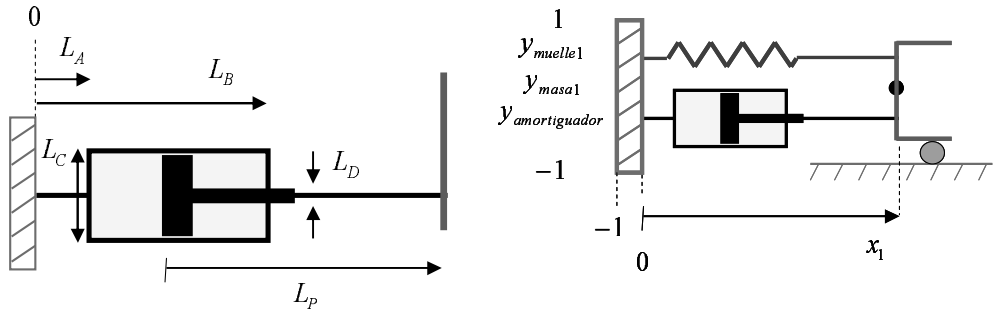


Figura 12.20: a) Método *mover_masa1()*, definido en el panel *Propio*; b) Significado de las variables empleadas.

Las acciones a realizar cuando el usuario pulsa sobre el objeto, lo arrastra con el ratón, y deja de pulsar sobre el objeto se indican en los campos *Al Pulsar*, *Al Mover* y *Al Soltar* respectivamente. En estos campos se escribe el nombre del método de Java que debe ejecutarse en cada caso. Ejs proporciona algunos métodos predefinidos, como son *_pause()*, que detiene la simulación, y *_play()*, que la continua partiendo del punto en el cual se detuvo. Puede obtenerse una lista de los métodos proporcionados por Ejs pulsando sobre el icono que representa dos engranajes y que está situado a la derecha del campo correspondiente.

Además de los métodos proporcionados por Ejs, puede asociarse a una acción (pulsar, mover o soltar) cualquier método que haya sido definido en el panel *Propio*. En la Figura 12.20a se muestra el código del método *mover_masa1()*, que será ejecutado cada vez que se arrastre el objeto *Masa1* con el ratón. Para facilitar la comprensión del código, en la Figura 12.20b se muestra el significado de las variables que definen las dimensiones físicas y la posición de los elementos del sistema.

El código del método *mover_masa1()* es el mostrado a continuación (vea la Figura 12.20). Su propósito es limitar el tipo de desplazamientos del objeto *masa1* que pueden realizarse arrastrándolo con el ratón. Se han numerado las líneas de código con el fin de facilitar la referencia a las mismas en las explicaciones siguientes.

```
1   public void mover_masa1 () {
2      yMasa1 = 0;
```

```
3       if ( x1 > LB + LP ) { x1 = LB + LP; }
4       if ( x1 < LA + LP ) { x1 = LA + LP; }
5       if ( x1 < LB      ) { x1 = LB;      }
6   }
```

En la línea 2 se establece que la coordenada vertical del objeto *Masa1* siempre sea cero. Con ello se impide que el objeto se desplace verticalmente al ser arrastrado con el ratón. Es decir, únicamente se permiten desplazamientos horizontales.

La finalidad de las líneas 3 y 4 es impedir que el émbolo del amortiguador quede fuera de la caja en la que está contenido. Finalmente, la línea 5 del código impide que la masa penetre dentro de la caja del amortiguador.

El amortiguador: elementos *AmortiguadorFijo* y *AmortiguadorMovil*

En las Figuras 12.21 y 12.22 se muestran las propiedades de los dos elementos empleados para representar el amortiguador. El elemento *AmortiguadorFijo* representa la caja, que se encuentra unida solidariamente a la pared, y el elemento *AmortiguadorMovil*, que representa el émbolo móvil.

El elemento *AmortiguadorFijo* es de la clase *Polígono*. Se trata de un polígono no cerrado. Las coordenadas de los puntos de los vértices del polígono están definidas por los vectores *x_amortiguadorFijo* e *y_amortiguadorFijo*. Recuérdese que estos dos vectores se definieron en el panel *Variables* (vea la Figura 12.11) y se les asignó valores en el panel *Inicialización* (ver la Figura 12.12).

El elemento *AmortiguadorMovil* es de la clase *Flecha*. En la Figura 12.22 se muestran las propiedades del elemento. Estas incluyen las coordenadas horizontal y vertical del punto base de la flecha (campos X e Y), la dimensión horizontal y vertical de la flecha (campos *Tamaño X* y *Tamaño Y*), el estilo de la flecha (campo Estilo) y sus propiedades de interactividad.

12.8. Modelo matemático del sistema completo

El sistema mecánico a modelar es el mostrado en la Figura 12.1a. El objeto *masa2* desliza sobre el objeto *masa1*, existiendo una fuerza de fricción entre ambos.

Modelo de la fricción

En la Figura 12.23a se muestra el modelo de la fricción[1]. Se observa que en el modelo intervienen las dos variables siguientes: la velocidad relativa entre los cuerpos (v) y la fuerza de fricción (F_f).

El modelo contempla las dos situaciones siguientes:

– *Fricción dinámica*. Cuando la velocidad relativa entre ambos cuerpos es diferente de cero ($v \neq 0$), se produce una fuerza de fricción dinámica que viene dada por la expresión siguiente:

$$F_f = \begin{cases} R_v \cdot v + R_m & \text{si } v > 0 \\ R_v \cdot v - R_m & \text{si } v < 0 \end{cases} \tag{12.35}$$

– *Fricción estática*. Cuando la velocidad relativa entre ambas superficies es cero ($v = 0$), la fuerza de fricción estática se opone a que la velocidad se haga diferente de cero, siempre que para ello sea preciso una fuerza de magnitud menor o igual que R_0.

[1] El modelo de la fricción está extraído de (Elmqvist, Cellier & Otter 1993).

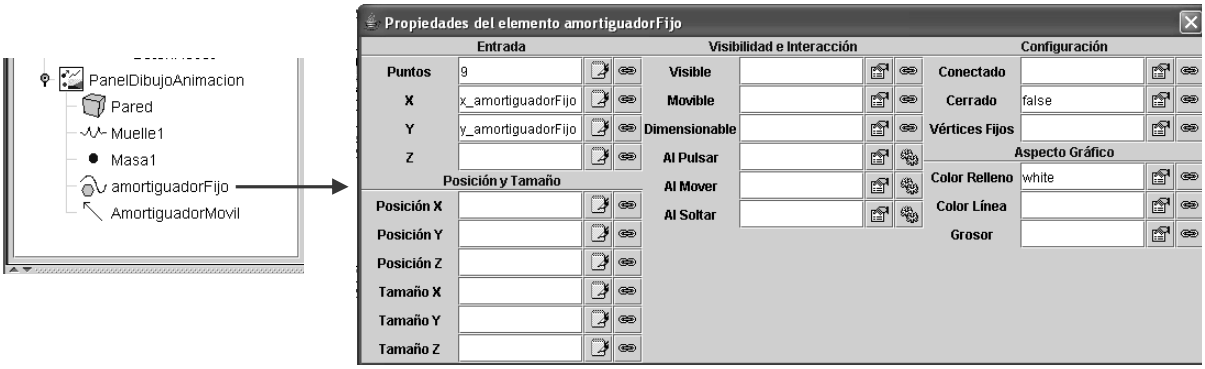


Figura 12.21: Propiedades del elemento *AmortiguadorFijo*.

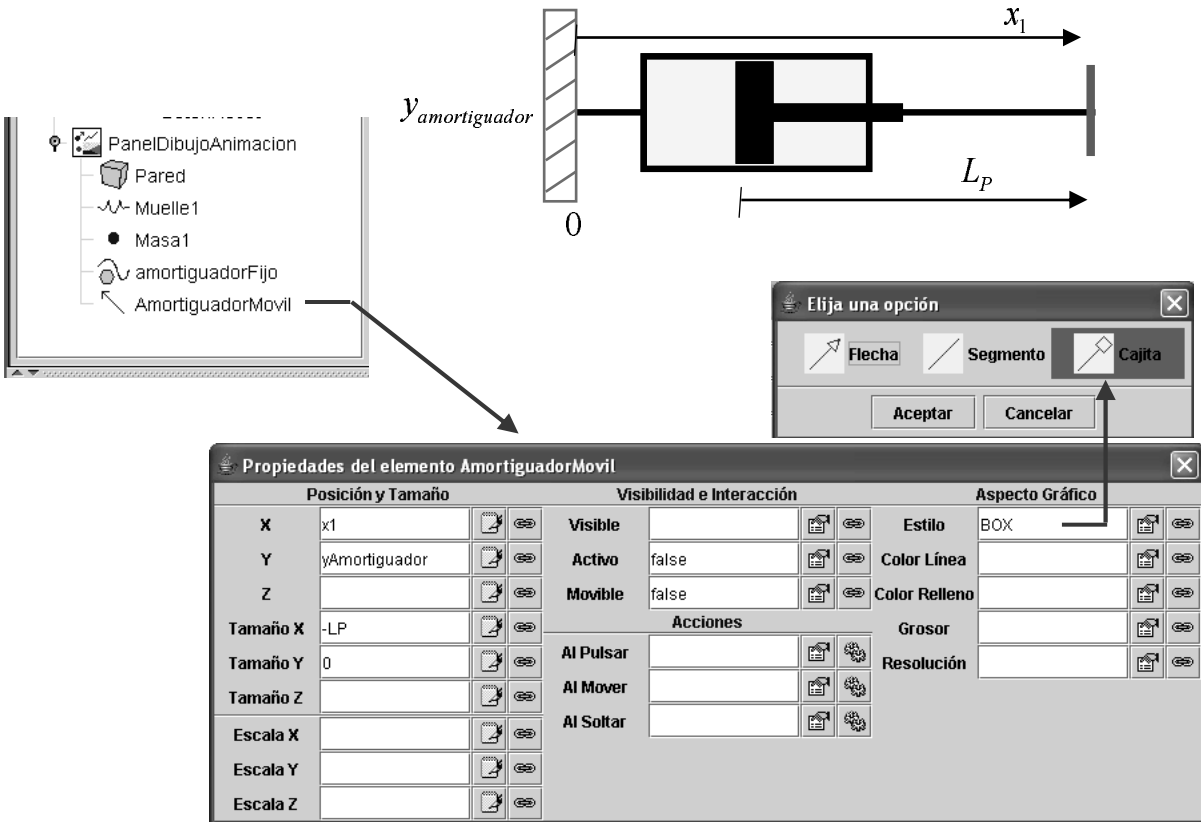


Figura 12.22: Propiedades del elemento *AmortiguadorMovil*.

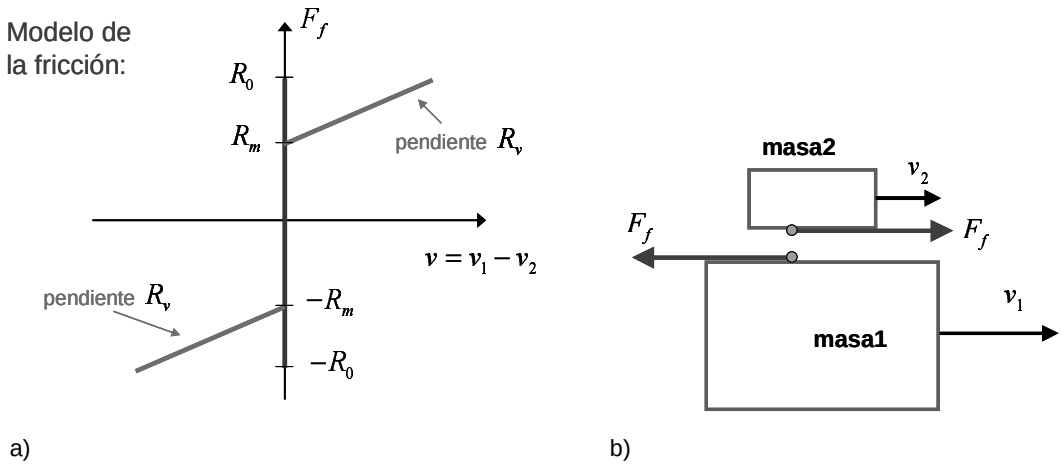


Figura 12.23: a) Relación constitutiva de la fricción; b) Ejemplo.

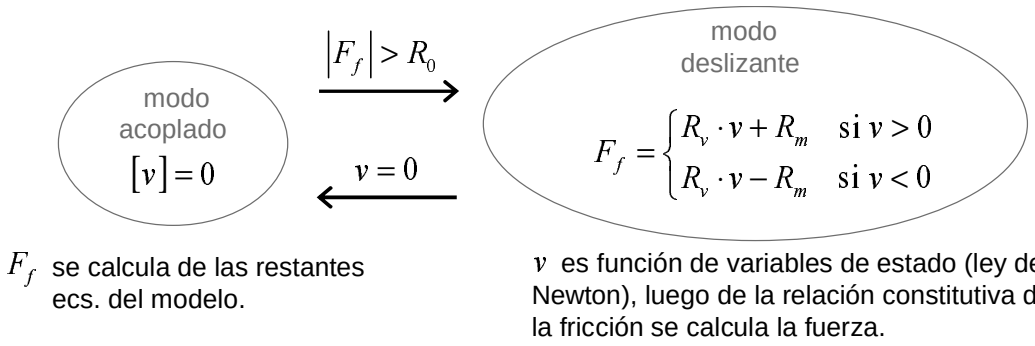


Figura 12.24: Modelo de la fricción como un autómata con dos modos.

Ejemplo 12.8.1. *En la Figura 12.23b se muestra un ejemplo de fricción dinámica. La velocidad del objeto masa1 es mayor que la velocidad del objeto masa2. Es decir:* $v_1 > v_2$*. Tal como se muestra en la Figura 12.23b, ambas velocidades tienen la misma dirección y sentido.*

Puesto que la velocidad relativa es mayor que cero ($v = v_1 - v_2 > 0$*), aparece una fuerza de fricción entre los objetos masa1 y masa2. La fuerza ejercida sobre masa1 tiene la misma magnitud, pero sentido opuesto, a la fuerza ejercida sobre masa2.*

Estas fuerzas de fricción tienden a reducir la velocidad relativa entre los objetos. Es decir, la fuerza de fricción que actúa sobre masa1 tiene a reducir la velocidad de masa1, mientras que la fuerza que actúa sobre masa2 tiende a aumentar la velocidad de masa2. □

Obsérvese que, cuando se produce la transición entre el reposo relativo y el movimiento relativo de los dos cuerpos, se produce un cambio discontinuo en la magnitud de la fuerza de fricción, que pasa de valer R_0 a valer R_m.

La fricción puede modelarse como se muestra en la Figura 12.24. Se distinguen dos modos diferentes de operación del sistema:

– *Modo acoplado*. En este caso, la fuerza de fricción entre *masa1* y *masa2* es tal que la velocidad relativa entre ambos objetos es cero. Así pues, el fenómeno físico de la fricción se modela mediante la ecuación:

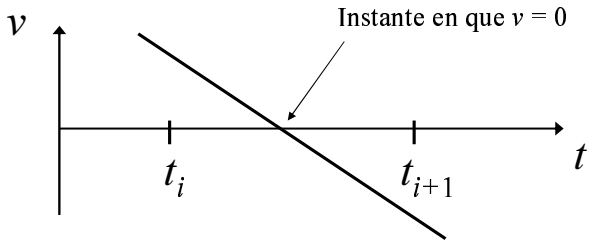


Figura 12.25: La variable v pasa por cero, sin valer exactamente cero en ninguno de los instantes de evaluación del modelo: t_i y t_{i+1}.

$$v = 0 \tag{12.36}$$

En esta ecuación sólo interviene una variable: la velocidad relativa (v). Por tanto, la Ec. (12.36) se empleará para calcular v. Este hecho se representa en la Figura 12.24 incluyendo la variable v entre corchetes cuadrados.

Puesto que de la ecuación constitutiva de la fricción, la Ec. (12.36), se calcula v, la fuerza de fricción (F_f) deberá calcularse de las restantes ecuaciones del modelo.

– *Modo deslizante*. Cuando existe movimiento relativo entre *masa1* y *masa2*, la fuerza de fricción viene dada por la Ec. (12.35).

En la Figura 12.24 se muestran las condiciones de conmutación entre ambos modos de funcionamiento:

– Cuando el sistema está en *modo acoplado*, y el módulo de la fuerza de fricción supera el valor R_0, entonces se produce la conmutación al *modo deslizante*.
– Cuando el sistema está en *modo deslizante* y la velocidad relativa entre los objetos se hace cero, entonces se produce la conmutación al *modo acoplado*.

El modelo de la fricción descrito anteriormente representa satisfactoriamente la física del problema. Sin embargo, no es un modelo válido para simulación, ya que presenta los dos problemas descritos a continuación.

Problema I: Detección de los eventos

La condición de conmutación entre el modo deslizante y el modo acoplado es que la velocidad relativa entre *masa1* y *masa2* valga cero ($v = 0$). En general, al describir un modelo para simulación, una condición del tipo "*cuando* v *valga cero, entonces* ..." no debe ser escrita empleando expresiones de igualdad.

Esto es debido a que la condición $v = 0$ se comprobará en los instantes de tiempo en los que se evalúa el modelo, los cuales vienen determinados por el tamaño del paso del método de integración. Así pues, si el corte por cero de la variable v se produce en un instante de tiempo que no coincide con un instante de evaluación, entonces no se detectará el evento. En la Figura 12.25 se ilustra esta situación.

Para evitar que no se detecte el corte por cero de la variable, es preciso expresar adecuadamente la condición, empleando para ello expresiones de desigualdad, en lugar de expresiones de igualdad. Las condiciones del tipo "*cuando* v *valga cero, entonces* ..." deben expresarse de la forma siguiente: "*cuando* v *pase de ser positiva a ser menor o igual que cero, o cuando* v *pase de ser negativa a ser mayor o igual que cero, entonces* ...".

Volviendo al ejemplo mostrado en la Figura 12.25, obsérvese que esta forma de expresar la condición permite detectar correctamente el cruce por cero: en el instante t_i, la variable v

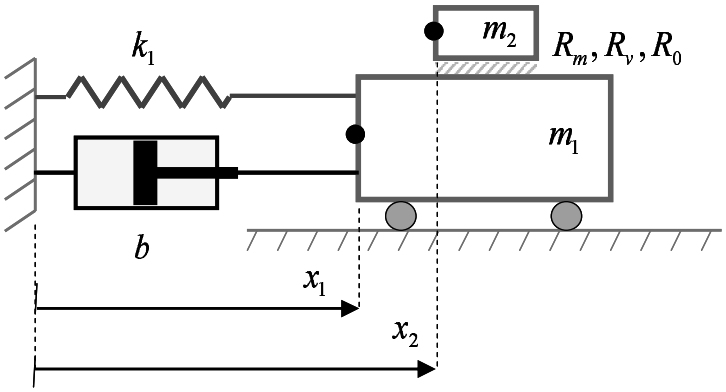


Figura 12.26: Representación esquemática del sistema.

es mayor que cero, y en el instante t_{i+1} es menor que cero, con lo cual, en el instante t_{i+1} se verifica la condición de disparo del evento.

Problema II: Índice superior

El segundo problema que presenta el modelo mostrado en la Figura 12.24 está originado por la formulación del modelo de la fricción cuando las dos masas están acopladas. En esa situación, la fricción viene descrita mediante la ecuación:

$$v = 0 \tag{12.37}$$

Esta ligadura reduce en uno el número de grados de libertad del modelo, ya que impone que las velocidades de los dos objetos (que son variables de estado) sean iguales. Veamos esto más detenidamente. Para ello, a continuación se escriben las ecuaciones que describen cada uno de los componentes del sistema. En la Figura 12.26 se muestra el sistema, con los parámetros asociados a cada componente, y con el significado de las variables x_1 y x_2 indicado.

Ecuación dinámica de masa1:

$$F_{masa1} = m_1 \cdot a_1 \tag{12.38}$$

$$\frac{dv_1}{dt} = a_1 \tag{12.39}$$

$$\frac{dx_1}{dt} = v_1 \tag{12.40}$$

Fuerza que actúa sobre la masa1:

$$F_{masa1} = F_{muelle1} + F_{amortiguador} - F_f \tag{12.41}$$

Ecuación dinámica de masa2:

$$F_{masa2} = m_2 \cdot a_2 \tag{12.42}$$
$$\frac{dv_2}{dt} = a_2 \tag{12.43}$$
$$\frac{dx_2}{dt} = v_2 \tag{12.44}$$

Fuerza que actúa sobre la masa2:

$$F_{masa2} = F_f \tag{12.45}$$

Ecuación dinámica del muelle:

$$F_{muelle1} = -k_1 \cdot (x_1 - x_{0,muelle1}) \tag{12.46}$$

Ecuación dinámica del amortiguador:

$$F_{amortiguador} = -b \cdot v_1 \tag{12.47}$$

Definición de la velocidad relativa entre las masas:

$$v = v_1 - v_2 \tag{12.48}$$

Modelo de la fuerza de fricción:

$$\begin{cases} \text{Modo deslizante:} & F_f = \begin{cases} R_v \cdot v + R_m & \text{si } v > 0 \\ R_v \cdot v - R_m & \text{si } v < 0 \end{cases} \\ \text{Modo acoplado:} & v = 0 \end{cases} \tag{12.49}$$

El modelo está compuesto por 12 ecuaciones, Ecs. (12.38) – (12.49), y tiene las 12 incógnitas siguientes:

$$F_{masa1}, F_{masa2}, F_{amortiguador}, F_{muelle1}, F_f, a_1, a_2, v, derx_1, derx_2, derv_1, derv_2 \tag{12.50}$$

donde puede observarse que las derivadas han sido sustituidas por las variables auxiliares $derx_1$, $derx_2$, $derv_1$ y $derv_2$.

Las variables

$$x_1, x_2, v_1, v_2 \tag{12.51}$$

aparecen derivadas en el modelo, con lo cual son variables de estado. Esto implica que el valor inicial de cada una de ellas puede especificarse independientemente del valor inicial de las demás, y que la evolución de cada una de ellas se calcula mediante integración numérica de su derivada (es decir, de $derx_1$, $derx_2$, $derv_1$ y $derv_2$ respectivamente).

En la Figura 12.27 se muestra las ecuaciones del modelo para cada uno de los dos modos, con la causalidad computacional señalada (se incluye entre corchetes la variable a evaluar de cada ecuación), y con las ecuaciones ordenadas y resueltas (es decir, con la incógnita a evaluar despejada al lado izquierdo de la igualdad).

Para realizar la asignación de la causalidad computacional se parte de que los parámetros del modelo y las variables de estado son conocidas. A continuación, se aplican las dos reglas enunciadas en la Sección 12.4.

Como se observa en la parte izquierda de la Figura 12.27, las ecuaciones del modelo, cuando se emplean las ecuaciones de la fricción en modo deslizante, pueden ser resueltas una tras otra y permiten evaluar todas las incógnitas.

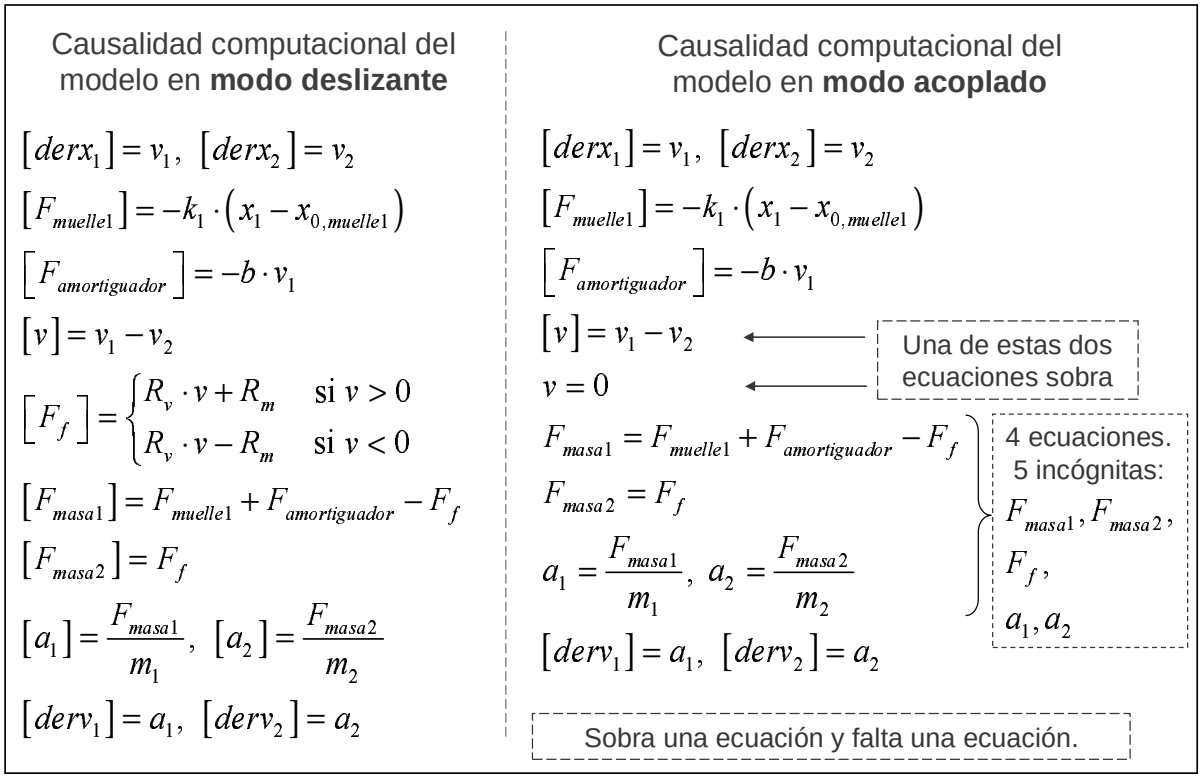


Figura 12.27: Asignación de la causalidad computacional del modelo en modo deslizante (izquierda) y acoplado (derecha).

$$[derx_1] = v_1,\ [derx_2] = v_2$$
$$[F_{muelle1}] = -k_1 \cdot (x_1 - x_{0,muelle1})$$
$$[F_{amortiguador}] = -b \cdot v_1$$
$$[v] = v_1 - v_2$$
$$a_1 = a_2$$
$$F_{masa1} = F_{muelle1} + F_{amortiguador} - F_f$$
$$F_{masa2} = F_f$$
$$a_1 = \frac{F_{masa1}}{m_1},\ a_2 = \frac{F_{masa2}}{m_2}$$

5 ecuaciones. 5 incógnitas: $F_{masa1}, F_{masa2}, F_f, a_1, a_2$

$$[derv_1] = a_1,\ [derv_2] = a_2$$

Figura 12.28: Asignación de la causalidad computacional del modelo modificado para el modo acoplado.

Sin embargo, al realizar la asignación de la causalidad computacional empleando la ecuación de la fricción para el modo acoplado (es decir, $v = 0$), se llega a la conclusión de que una de las ecuaciones es redundante y, a la vez, falta una ecuación para poder calcular las incógnitas F_{masa1}, F_{masa2}, F_f , a_1 y a_2. Véase la parte derecha de la Figura 12.27.

Esta situación está motivada porque:

– Por una parte, v_1 y v_2 aparecen derivadas en el modelo. Esto implica que estas dos variables son variables de estado, con lo cual su causalidad computacional debe ser:

$$\frac{d[v_1]}{dt} = derv_1 \tag{12.52}$$

$$\frac{d[v_2]}{dt} = derv_2 \tag{12.53}$$

– Por otra lado, en la ecuación que describe la fricción

$$v_1 = v_2 \tag{12.54}$$

sólo intervienen las variables v_1 y v_2, con lo cual debe emplearse esta ecuación para calcular una de ellas. Es posible una de las dos causalidades computacionales siguientes:

$$[v_1] = v_2 \qquad\qquad v_1 = [v_2] \tag{12.55}$$

En consecuencia, en el modelo existen tres ecuaciones, Ecs. (12.52), (12.53) y (12.54), para calcular dos incógnitas, v_1 y v_2, con lo que se produce una sobredeterminación. Asimismo, data esta sobredeterminación y puesto que el modelo tiene el mismo número de ecuaciones que de incógnitas, no es posible calcular una de las variables del modelo.

Otra forma de entender el problema es darse cuenta de que al tener 4 variables de estado (x_1, x_2, v_1 y v_2) el modelo tiene 4 grados de libertad. Sin embargo, al imponer la ligadura $v_1 = v_2$ se está eliminando un grado de libertad, con lo cual el sistema queda con sólo 3 grados de libertad. La consecuencia de ello es que el modelo no puede tener 4 variables de estado, sino sólo 3. Cuando se produce esta circunstancia se dice que el modelo tiene *índice superior*, y debe ser reformulado antes de simularlo con Ejs.

Revisión del modelo de la fricción: reducción del índice

Una solución para el Problema II (es decir, para reducir el índice del modelo) consiste en reemplazar la ecuación

$$v = 0 \tag{12.56}$$

por la ecuación

$$a_1 = a_2 \tag{12.57}$$

y además, cuando se produce la conmutación del modo deslizante al modo acoplado, imponer que en ese instante la velocidad relativa entre ambos cuerpos sea cero: $v = 0$. Esto último es necesario, ya que, en general, en el instante en que se produce la conmutación, el valor de v estará próximo a cero, pero no valdrá exactamente cero.

En la Figura 12.28 se muestra la causalidad computacional del modelo, para el modo acoplado, modificado según se ha indicado anteriormente. Las incógnitas F_{masa1}, F_{masa2}, F_f, a_1 y a_2 deben ser calculadas de un sistema simultáneo de ecuaciones. Despejando estas incógnitas, se obtiene que la causalidad computacional del sistema completo es la mostrada en la Figura 12.29.

Modelo ordenado y resuelto, válido para ambos modos:

$$[derx_1] = v_1,\ [derx_2] = v_2$$
$$[F_{muelle1}] = -k_1 \cdot (x_1 - x_{0,muelle1})$$
$$[F_{amortiguador}] = -b \cdot v_1$$
$$[v] = v_1 - v_2$$

modo deslizante

$$[F_f] = \begin{cases} R_v \cdot v + R_m & \text{si } v > 0 \\ R_v \cdot v - R_m & \text{si } v < 0 \end{cases}$$

modo acoplado

$$[F_f] = \frac{m_2}{m_1 + m_2}(F_{muelle1} + F_{amortiguador})$$

$$[F_{masa1}] = F_{muelle1} + F_{amortiguador} - F_f$$
$$[F_{masa2}] = F_f$$
$$[a_1] = \frac{F_{masa1}}{m_1},\ [a_2] = \frac{F_{masa2}}{m_2}$$
$$[derv_1] = a_1,\ [derv_2] = a_2$$
$$cuando\ (\text{deslizante} \rightarrow \text{acoplado})\ \{v = 0\}$$

Figura 12.29: Asignación de la causalidad computacional del modelo completo modificado.

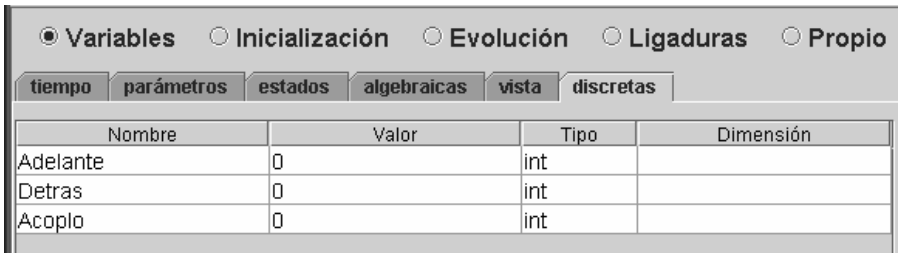


Figura 12.30: Definición e inicialización de las variables *Detras*, *Adelante* y *Acoplo*.

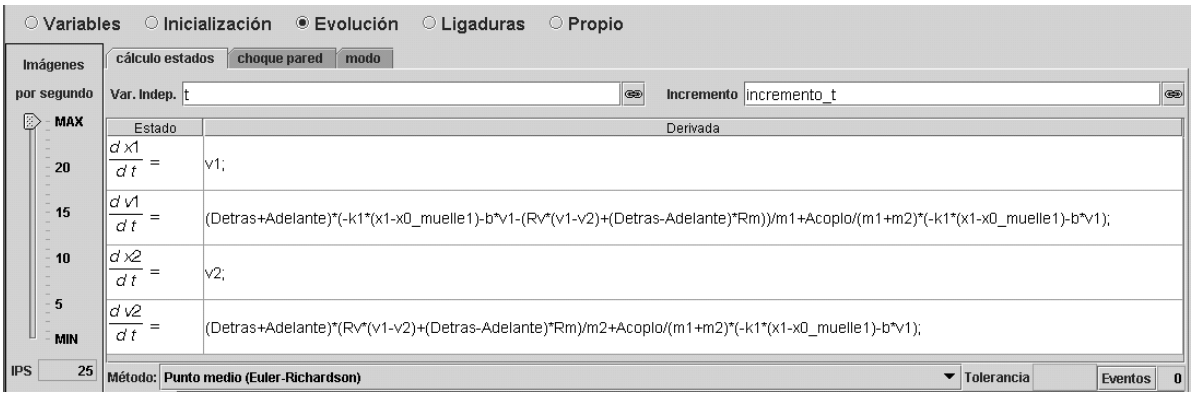


Figura 12.31: Ecuaciones para el cálculo de las variables de estado.

En la página EDO de Ejs, las derivadas de las variables de estado deben expresarse únicamente en función de parámetros, variables de estado y la variable tiempo. Una vez asignada la causalidad computacional del modelo, expresar las derivadas de la forma anteriormente indicada es sencillo. Para ello, basta con tener en cuenta que:

$$\frac{dx_1}{dt} = derx_1 \tag{12.58}$$
$$\frac{dx_2}{dt} = derx_2 \tag{12.59}$$
$$\frac{dv_1}{dt} = derv_1 \tag{12.60}$$
$$\frac{dv_2}{dt} = derv_2 \tag{12.61}$$

Realizando las sustituciones, se obtiene:

$$\frac{dx_1}{dt} = v_1 \tag{12.62}$$
$$\frac{dx_2}{dt} = v_2 \tag{12.63}$$
$$\frac{dv_1}{dt} = \begin{cases} \frac{-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1-(R_v\cdot(v_1-v_2)+R_m)}{m_1} & \text{si “deslizante” y } v>0 \\ \frac{-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1-(R_v\cdot(v_1-v_2)-R_m)}{m_1} & \text{si “deslizante” y } v<0 \\ \frac{1}{m_1+m_2}\cdot(-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1) & \text{si “acoplado”} \end{cases} \tag{12.64}$$
$$\frac{dv_2}{dt} = \begin{cases} \frac{R_v\cdot(v_1-v_2)+R_m}{m_2} & \text{si “deslizante” y } v>0 \\ \frac{R_v\cdot(v_1-v_2)-R_m}{m_2} & \text{si “deslizante” y } v<0 \\ \frac{1}{m_1+m_2}\cdot(-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1) & \text{si “acoplado”} \end{cases} \tag{12.65}$$

Es posible escribir la ecuación para $\frac{dv_1}{dt}$ como una única expresión, e igualmente para $\frac{dv_2}{dt}$. Para ello, se definen tres variables, *Detras*, *Adelante* y *Acoplo*, las cuales sólo pueden tomar el valor 0 ó 1. En cada instante, sólo una de estas tres variables tomará el valor 1, y será aquella que se corresponda con el modo en el que se encuentra el sistema:

- *Detras* valdrá 1 si y sólo si el sistema está en el modo “deslizante” y $v_1 > v_2$.
- *Adelante* valdrá 1 si y sólo si el sistema está en el modo “deslizante” y $v_1 < v_2$.
- *Acoplo*, valdrá 1 si el sistema está en el modo “acoplado”.

Con ayuda de estas tres variables, puede escribirse:

$$\begin{aligned} \frac{dv_1}{dt} &= Detras\cdot\frac{-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1-(R_v\cdot(v_1-v_2)+R_m)}{m_1} \\ &+ Adelante\cdot\frac{-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1-(R_v\cdot(v_1-v_2)-R_m)}{m_1} \\ &+ Acoplo\cdot\frac{1}{m_1+m_2}\cdot(-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1) \end{aligned} \tag{12.66}$$
$$\begin{aligned} \frac{dv_2}{dt} &= Detras\cdot\frac{R_v\cdot(v_1-v_2)+R_m}{m_2} \\ &+ Adelante\cdot\frac{R_v\cdot(v_1-v_2)-R_m}{m_2} \\ &+ Acoplo\cdot\frac{1}{m_1+m_2}\cdot(-k_1\cdot(x_1-x_{0,muelle1})-b\cdot v_1) \end{aligned} \tag{12.67}$$

Estas dos expresiones, pueden simplificarse de la forma siguiente:

$$\frac{dv_1}{dt} = (Detras + Adelante) \cdot \frac{-k_1 \cdot (x_1 - x_{0,muelle1}) - b \cdot v_1 - (R_v \cdot (v_1 - v_2) + (Detras - Adelante) \cdot R_m)}{m_1}$$
$$+ \quad Acoplo \cdot \frac{1}{m_1 + m_2} \cdot (-k_1 \cdot (x_1 - x_{0,muelle1}) - b \cdot v_1) \tag{12.68}$$

$$\frac{dv_2}{dt} = (Detras + Adelante) \cdot \frac{R_v \cdot (v_1 - v_2) + (Detras - Adelante) \cdot R_m}{m_2}$$
$$+ \quad Acoplo \cdot \frac{1}{m_1 + m_2} \cdot (-k_1 \cdot (x_1 - x_{0,muelle1}) - b \cdot v_1) \tag{12.69}$$

En la Figura 12.30 se muestra cómo se realiza la definición de las variables *Detras*, *Adelante* y *Acoplo*, en el panel *Variables*. Las tres variables son inicializadas al valor 0. Se trata de *variables discretas*, ya que su valor sólo se modifica en determinados instantes de tiempo: cuando se produce la transición entre los modos del sistema.

Las transiciones en el modo del sistema sólo se producen en los instantes de evaluación del modelo. Es decir, en los instantes del tipo t_i y t_{i+1} mostrados en la Figura 12.25. Esto implica que, en el intervalo de tiempo entre instantes de evaluación consecutivos, el valor de estas variables permanece constante. A efectos del método de integración, estas variables pueden considerarse constantes dentro de cada paso de integración, y por ello pueden incluirse en la expresión de la derivada en el panel EDO de Ejs (vea la Figura 12.31).

El modelo debe describir el cambio de valor de estas tres variables (*Detras*, *Adelante* y *Acoplo*), en función de las transiciones en el modo del sistema. Asimismo, las condiciones de disparo de estas transiciones deben estar formuladas en términos de desigualdades, de modo que no se obtenga un modelo erróneo como el descrito al comentar el Problema I. A continuación, se propondrá una manera de hacerlo.

Revisión del modelo de la fricción: los modos y su conmutación

Una posible solución consiste en modelar el funcionamiento del sistema como un autómata determinista, tal como se muestra en la Figura 12.32. En ella, se indica en qué panel se codifica cada una de las condiciones de transición entre modos. A continuación, se realiza una explicación detallada de ello.

Al comienzo de la simulación, el modelo se encuentra en el modo *Inicio*. En ese mismo instante inicial, y en función del valor inicial que se haya asignado a la velocidad de cada una de las masas ($v1$ y v_2), el modo del sistema cambia a:

- *Detrás*, si $v_1 > v_2$.
- *Adelante*, si $v_1 < v_2$.
- *Acoplo*, si no se satisface ninguna de las dos condiciones anteriores. Es decir, si $v_1 = v_2$.

Estas condiciones de conmutación entre modos, que han de ejecutarse en el instante inicial de la simulación, se codifican en el panel *Inicialización* de Ejs (vea la Figura 12.33).

Como se muestra en la Figura 12.32, la conmutación desde los modos *Detras* y *Adelante* al modo *Acoplo* se produce en función del valor de las variables de estado v_1 y v_2. Así pues, esas condiciones de conmutación son comprobadas una vez es calculado el valor de las variables de estado, en el panel *Evolución* (vea la Figura 12.34).

Finalmente, la conmutación desde el modo *Acoplo* a los modos *Detras* y *Adelante* se produce en función del valor que tenga la fuerza de fricción. Tanto la fuerza de fricción, que es una variable algebraica, como estas condiciones de conmutación se calculan en el panel *Ligaduras* (vea la Figura 12.35).

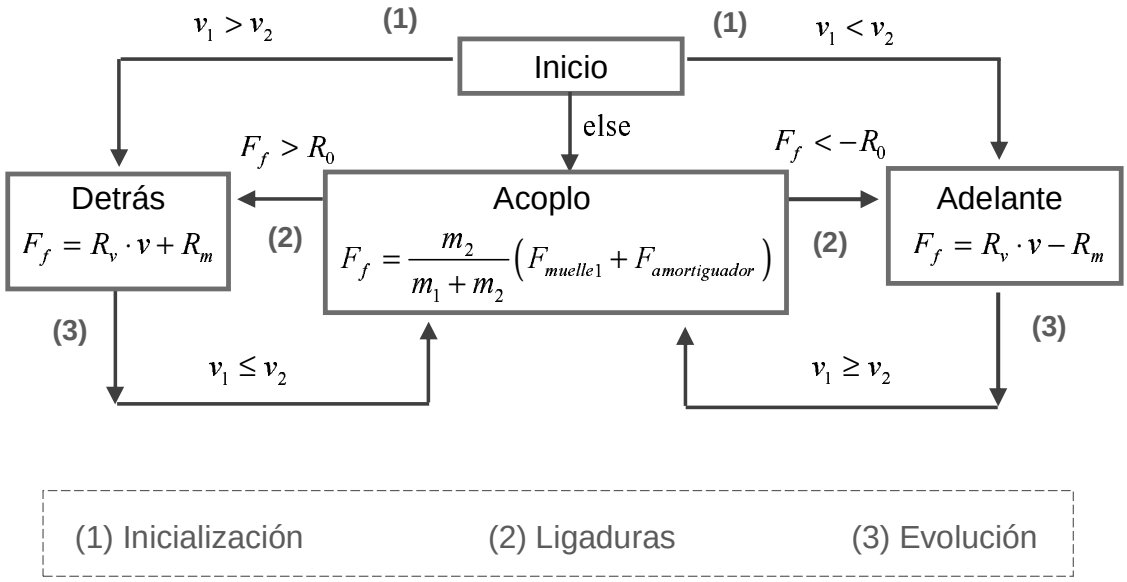


Figura 12.32: Conmutación entre los modos de funcionamiento del sistema. Se indica el panel en el cual se programa la conmutación entre modos.

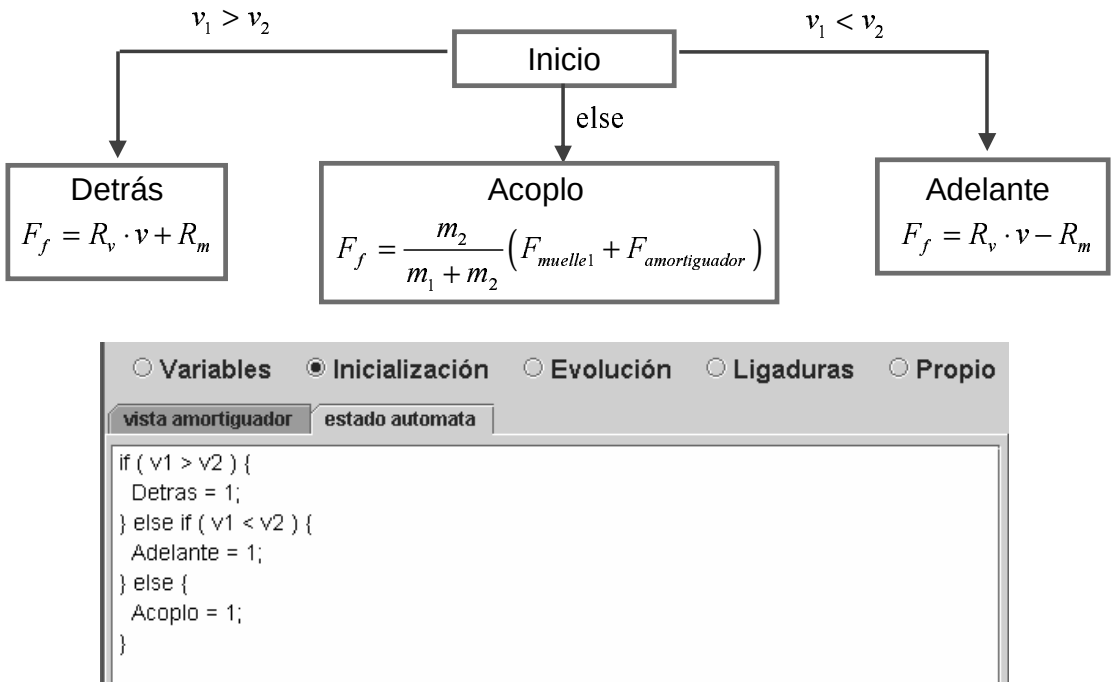


Figura 12.33: Modo inicial del sistema, en función de la velocidad relativa entre las masas.

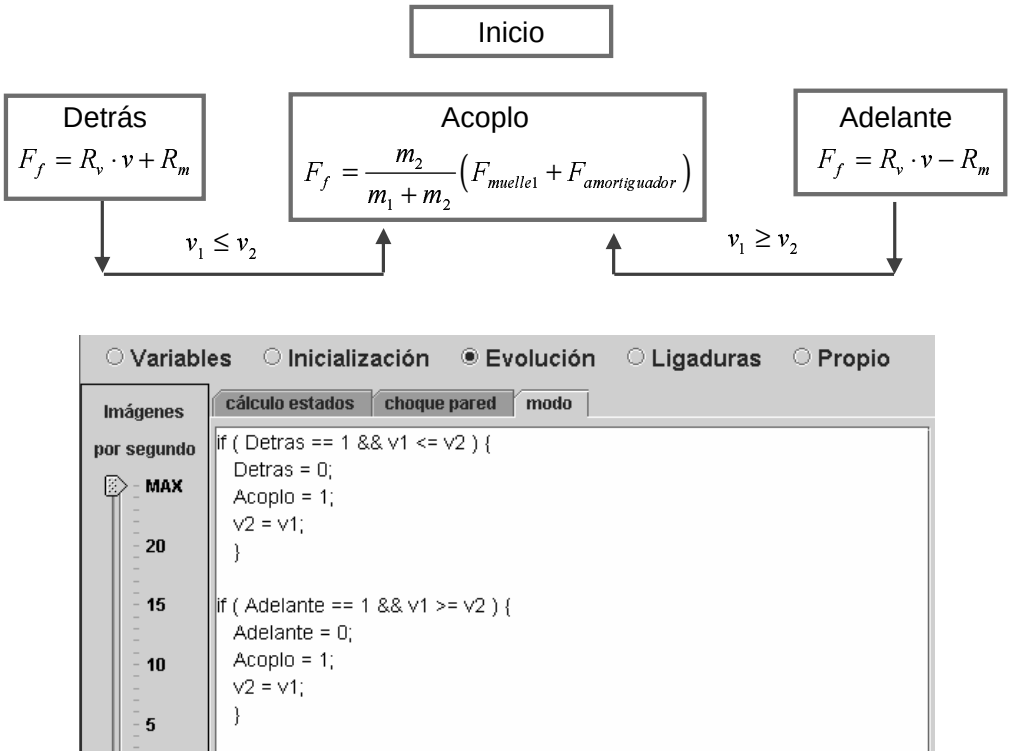


Figura 12.34: Conmutación entre los modos tras el cálculo de las variables de estado.

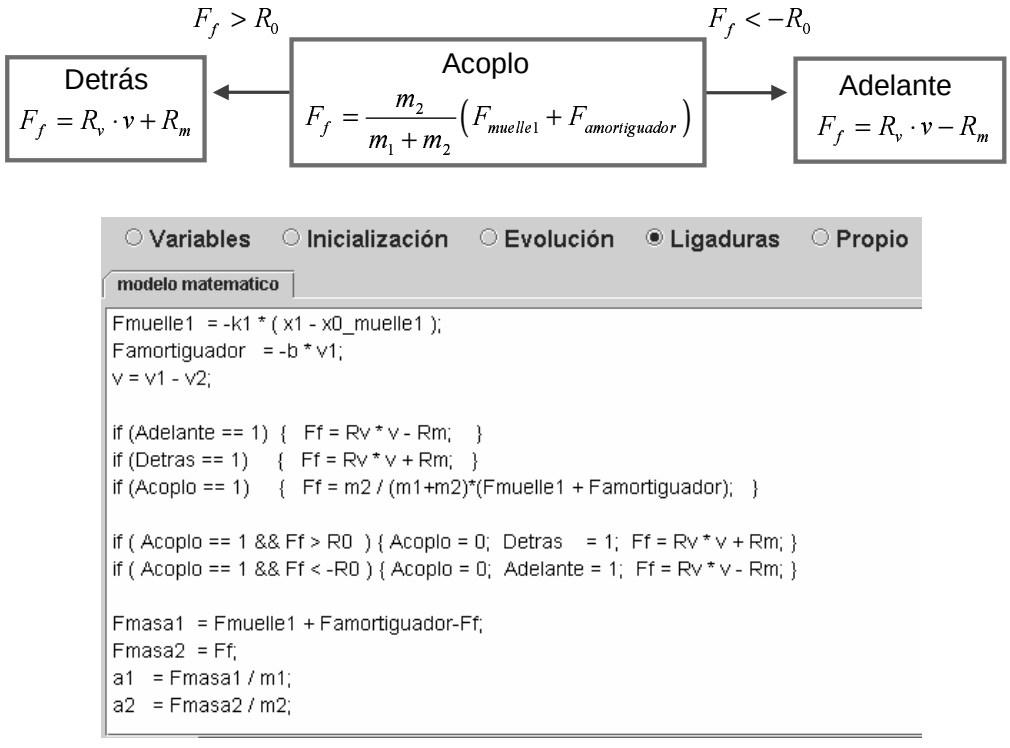


Figura 12.35: Conmutación entre los modos tras el cálculo de la fuerza de fricción.

12.9. Inclusión de *masa2* en el panel de animación

La inclusión del objeto *masa2* en la ventana de animación se realiza añadiendo un nuevo elemento al árbol de la vista: el elemento *Masa2*, de la clase *Particula* (vea la Figura 12.36). Las propiedades de este elemento, cuya posición puede ser cambiada interactivamente, son completamente análogas a las del elemento *Masa1*:

- Las variables *yMasa2* y *LMasa2* representan la coordenada vertical de la posición del objeto y el tamaño horizontal del mismo respectivamente. Son definidos e inicializados en el panel *Variables*, de forma completamente análoga a *yMasa1* y *LMasa1*. Su valor permanece constante durante la simulación.
- El método *mover_masa2()* se muestra en la Figura 12.37. Su finalidad es doble. Por una parte, garantizar que la coordenada vertical de la posición del objeto no varía cuando se modifica interactivamente la posición del mismo. Por otra, impedir que interactivamente pueda situarse el objeto *masa2* fuera de la superficie superior del objeto *masa1*.

En la Figura 12.38 se muestra la ventana de animación del modelo del sistema completo.

12.10. Ligadura en la posición de *masa2*

El método *mover_masa2()* impide que, producto de un cambio interactivo en la posición de *masa2*, ésta quede fuera de la superficie de *masa1*. Sin embargo, no hay nada en el modelo que impida que durante su simulación se produzca esta situación. Es decir, que la posición del objeto *masa2* se mueva fuera de la superficie del objeto *masa1* y, en consecuencia, quede "flotando" en el aire.

Para evitar esta situación, una vez se ha calculado el valor de x_2, se comprueba si se encuentra fuera del rango $[x1, x1 + Lmasa1]$. Si se verifica que $x_2 < x_1$, entonces se modifica el valor de x_2, igualándolo a x_1. Igualmente, si $x_2 > x1 + Lmasa1$, entonces se modifica el valor de x_2, igualándolo a $x1 + Lmasa1$. Estas acciones se programan en el panel *Evolución*, tal y como se muestra en la Figura 12.39.

12.11. Visualización del modo del sistema

Ejs permite, de manera sencilla, representar la evolución temporal de las variables del sistema. En esta sección se describe cómo añadir a la vista una ventana que contenga tres gráficas, en las que se represente la evolución temporal de las variables *Detras*, *Adelante* y *Acoplo*.

Para ello, se añade al árbol de elementos de la vista un objeto de la clase *Ventana*, al que se asigna el nombre *Modos*. Colgando de él se añaden tres elementos de la clase *PanelConEjes*, situados en la posición arriba, centro y abajo respectivamente (*Modo_Detras*, *Modo_Adelante* y *Modo_Acoplo*). Con ello, queda definida la ventana que contendrá las gráficas y los ejes coordenados de las tres gráficas.

Finalmente, se añade un elemento de la clase *Traza* colgando de cada elemento de la clase *PanelConEjes*. El elemento de la clase *Traza* define cuál es la variable a representar.

En la Figura 12.40 se muestra el árbol de elementos de la vista y las propiedades de los elementos *Modos*, *Modo_Detras* y *traza_Detras*. De forma análoga se definen las propiedades de los restantes elementos. La vista del laboratorio virtual es mostrada en la Figura 12.41.

12.12. Añadiendo interactividad al laboratorio

Llegado este punto, el laboratorio virtual permite modificar de forma interactiva la posición de los objetos *masa1* y *masa2*. En esta sección se explica cómo añadir a la vista del laboratorio

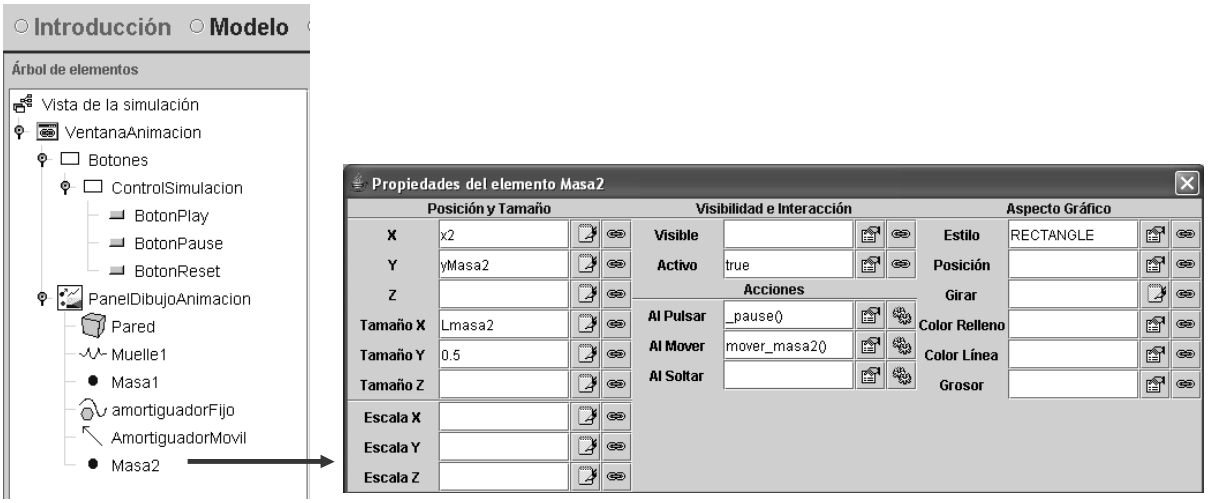


Figura 12.36: Elemento *Masa2*.

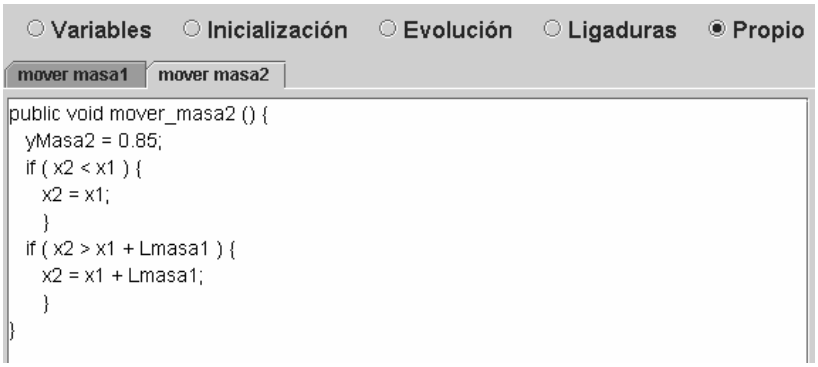


Figura 12.37: Método *mover_masa2()*.

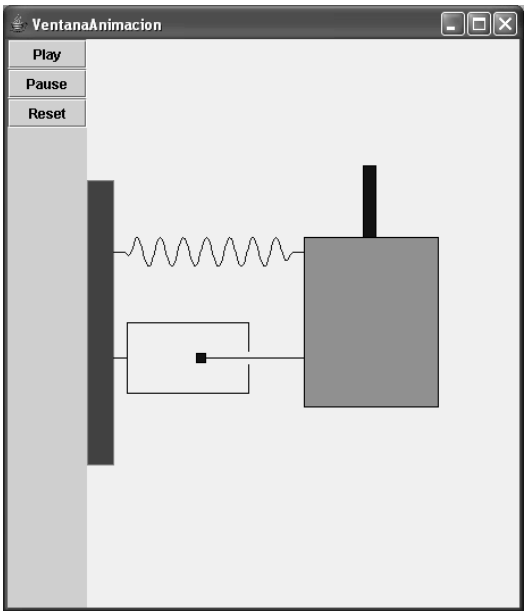


Figura 12.38: Ventana de animación de la vista del sistema completo.

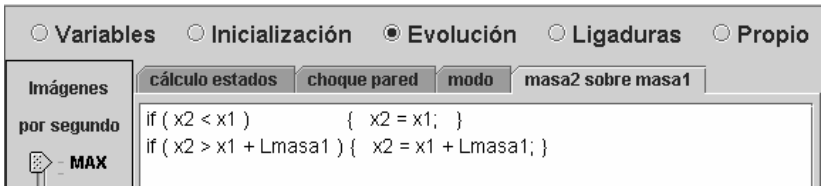


Figura 12.39: Ligadura en la posición de *Masa2*.

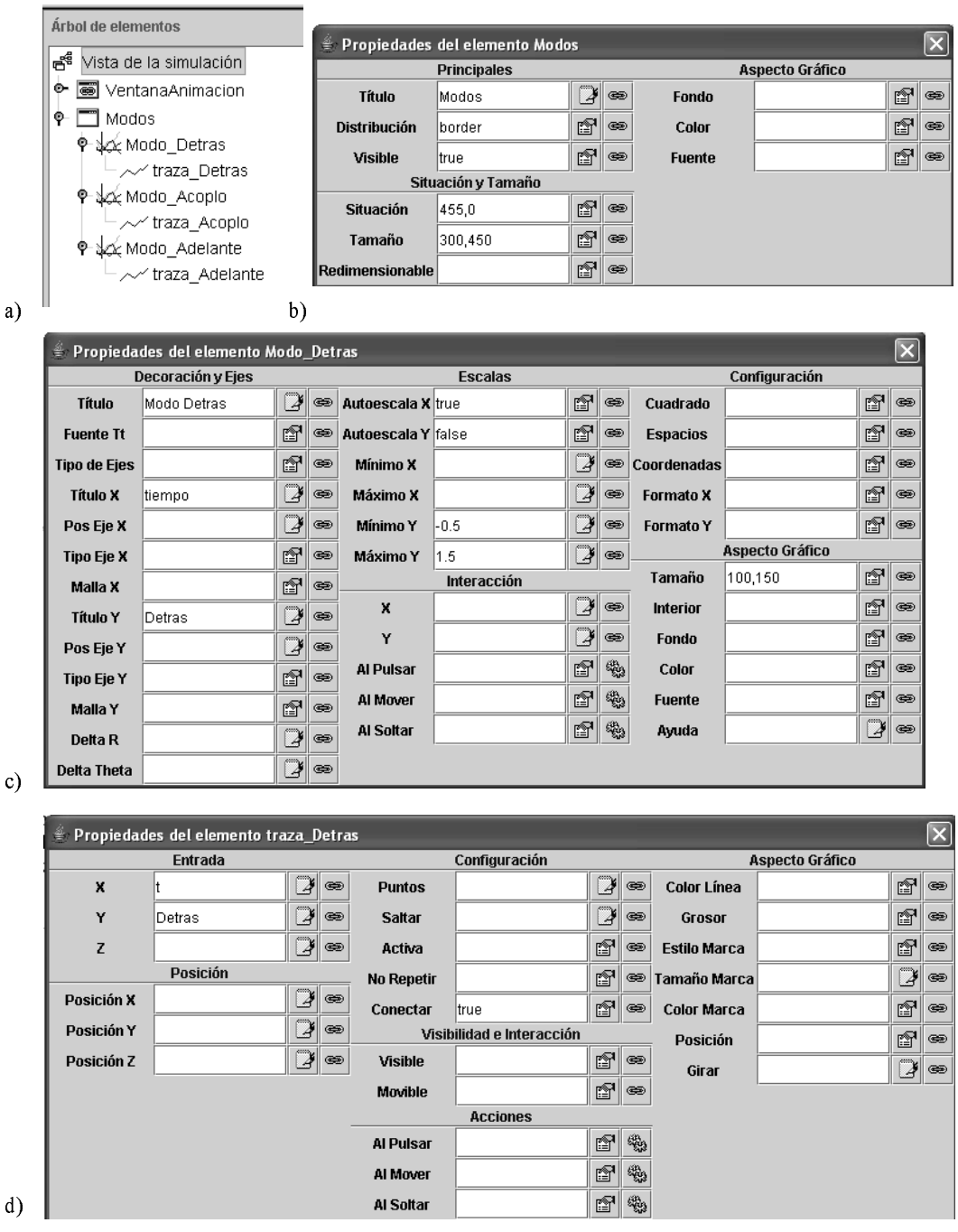


Figura 12.40: a) Árbol de elementos de la vista. Propiedades del elemento: b) *Modos*; c) *Modo_Detras*; y d) *traza_Detras*;

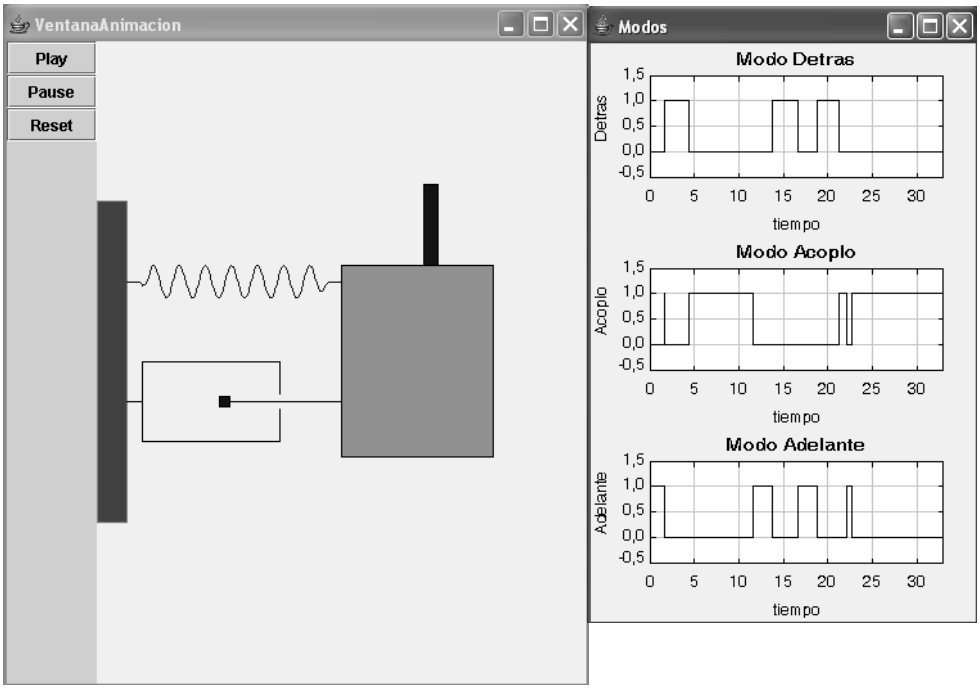


Figura 12.41: Vista del laboratorio virtual.

deslizadores que permitan visualizar y modificar el valor de los siguientes parámetros del modelo:

- Constante de proporcionalidad (k_1) y elongación natural ($x_{0,muelle1}$) del muelle.
- Constante de proporcionalidad (b) del amortiguador.
- Masas (m_1 y m_2).
- Parámetros característicos de la fuerza de fricción (R_m, R_0 y R_v).

Para ello, se añade al árbol de la vista un nuevo objeto de la clase *Ventana* (llamado *Deslizadores*), dentro del cual se incluyen tres elementos de la clase *Panel*: *Panel1*, *Panel2* y *Panel3*, cuya posición es arriba, centro y abajo respectivamente. Dentro de cada elemento de la clase *Panel* se incluyen elementos de la clase *Deslizador*.

El árbol de la vista se muestra en la parte izquierda de la Figura 12.42. También se muestran las propiedades de los elementos *Panel1* y *deslizador_k1*. Obsérvese que:

- En la casilla *Variable* se indica qué variable del sistema se enlaza con el elemento *Deslizador*.
- Las casillas *Mínimo* y *Máximo* permiten definir el intervalo de valores que puede tomar la variable.
- En la casilla *Marcas* se indica el número de marcas que se dibujarán en el deslizador.
- En la casilla *Formato M* se indica el formato del número que se escribe bajo cada marca. Por ejemplo, si se escribe 0 en la casilla *Formato M*, entonces el número se escribe sin cifras decimales. Si se escribe 0.0 en la casilla *Formato M*, entonces el número se escribe con una cifra decimal. Así sucesivamente.
- En la casilla *Formato* se indica cómo debe visualizarse el valor actual de la variable. Al escribir k1=0.0 se está indicando que debe escribirse k1= y a continuación el valor de la variable con una cifra decimal.

En la Figura 12.43 se muestra la vista completa del laboratorio.

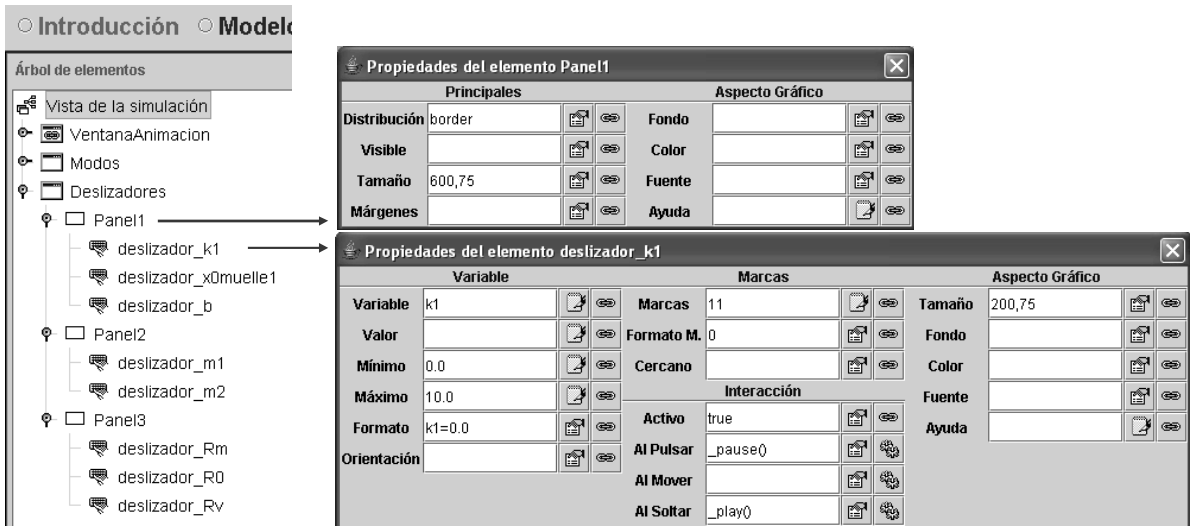


Figura 12.42: Árbol de la vista y propiedades de los elementos *Panel1* y *deslizador_k1*.

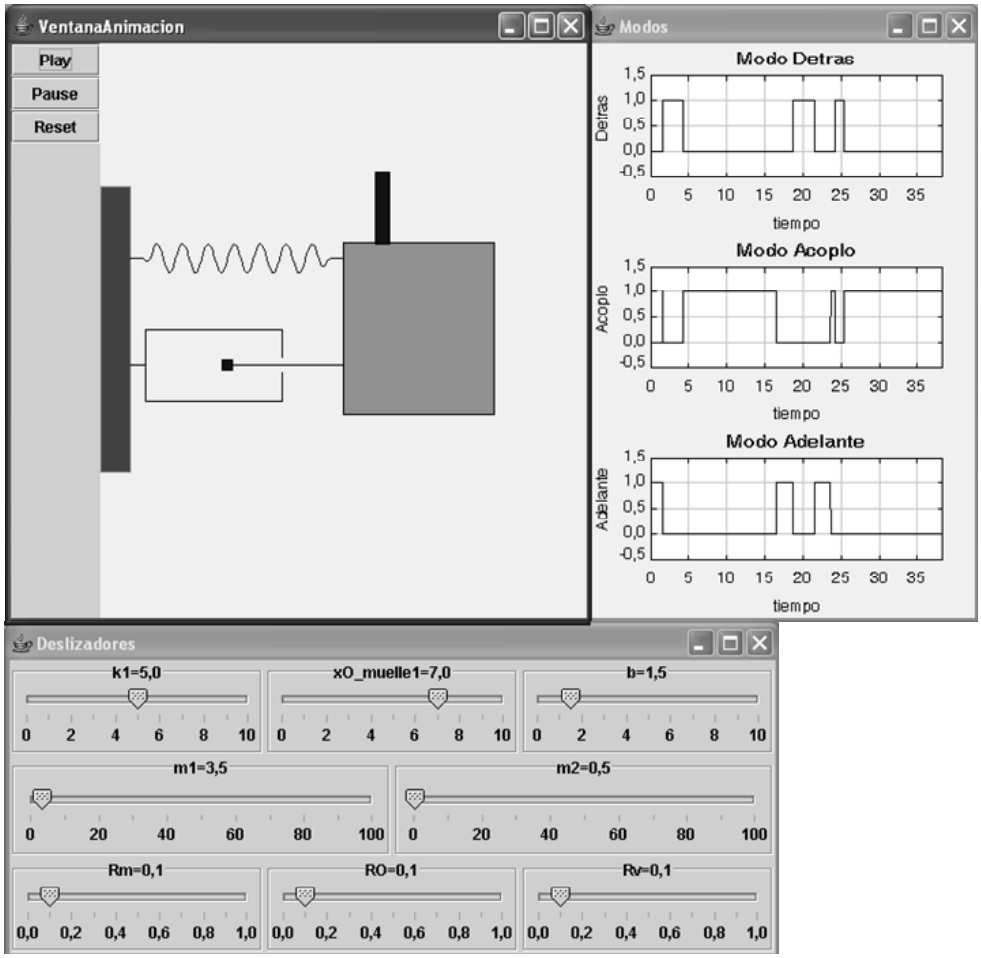


Figura 12.43: Vista del laboratorio virtual.

Tema 13

Cálculo del número π por el método de Monte Carlo

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Diseñar el algoritmo de simulación de un modelo estático y estocástico, y programarlo usando Ejs.
- Programar métodos propios, que son invocados desde los paneles Inicialización y Vista.
- Usar las clases de elementos gráficos *ConjuntoParticulas* y *CampoNumerico*.

13.1. Simulaciones de Monte Carlo

Bajo el nombre genérico de "*simulaciones de Monte Carlo*" suelen englobarse todas las simulaciones que emplean números aleatorios para resolver problemas estáticos, es decir, problemas en los cuales el tiempo no juega un papel relevante.

El término "Monte Carlo" proviene del nombre en clave de un trabajo secreto en el que von Neumann y Ulam emplearon esta técnica matemática (que ya era conocida anteriormente). Este trabajo se realizó en Los Álamos, durante el conocido proyecto para la fabricación de la bomba atómica.

En esta técnica se generan artificialmente datos mediante el uso de un generador de números aleatorios y la función de probabilidad de interés. El generador de números aleatorios se emplea para generar variables seudo aleatorias distribuidas uniformemente en el intervalo $[0, 1]$. La función de probabilidad a muestrear puede ser una distribución teórica (normal, exponencial, etc.) o puede ser empírica, es decir, puede estar basada en observaciones del sistema real.

La técnica de Monte Carlo se emplea no sólo para simular sistemas estocásticos, sino también para resolver problemas completamente deterministas (típicamente aquellos que no pueden ser resueltos analíticamente). Un ejemplo es la estimación del valor del número π mediante el procedimiento descrito a continuación.

13.2. Estimación del valor de π

Suponga que se lanza un dardo al azar sobre una superficie cuadrada, cuyo lado tiene dos metros. Inscrito en el cuadrado, hay un círculo de un metro de radio. Si el dardo se lanza de forma aleatoria, la probabilidad de dar dentro del círculo (acierto) es igual a $\pi/4$.

Así pues, puede obtenerse una estimación del número π siguiendo el procedimiento siguiente:

1. Se lanza el dardo de forma aleatoria un número n_{total} de veces. Sea $n_{aciertos}$ el número de veces que el dardo ha quedado dentro del círculo.
2. El número π se estima de la forma siguiente:

$$\pi \approx 4 \cdot \frac{n_{aciertos}}{n_{total}} \tag{13.1}$$

A mayor número de tiradas se espera una mejor aproximación.

A continuación se describe el algoritmo de la simulación del laboratorio para la estimación del valor de π y cómo se ha programado usando Ejs.

13.3. El algoritmo de la simulación

En la Figura 13.1 se muestra el algoritmo de la simulación.

Inicialmente se asigna al número de lanzamientos (n_{total}) el valor 100. Empleando el método *calculaPI* (definido en el panel *Propio*), que se muestra en la Figura 13.2, se realiza una estimación del valor de π:

1. Se realizan n_{total} lanzamientos aleatorios.
2. Se calcula $n_{aciertos}$.
3. Se estima el valor del número π aplicando la Ecuación (13.1).

Mientras el usuario no cambie el valor de n_{total}, la simulación progresa sin realizar nuevos cálculos. Cuando el usuario cambia n_{total}, vuelve a ejecutarse el método *calculaPI*, con lo que se obtiene una nueva estimación del valor de π.

Puesto que cada vez que se realiza una réplica de la simulación se emplea una semilla diferente para el generador de números aleatorios, se obtienen diferentes estimaciones del número π aun si se escoge para las diferentes réplicas un mismo valor de n_{total}.

13.4. Programación de la vista

En la Figura 13.3 se muestra la *Vista* del laboratorio virtual, que está compuesta de una única ventana. En la parte inferior de esta ventana hay una botonera, con una casilla para introducir el valor de n_{total} y un botón (*Reset*) para reinicializar la simulación.

Cada vez que el usuario introduce un nuevo valor de n_{total} y pulsa *Enter* se realiza una nueva estimación del valor de π: se ejecuta el método *calculaPI*.

En el centro de la ventana se muestra la mitad superior del círculo inscrito en el cuadrado y se señalan mediante puntos las posiciones en las cuales se ha caído el dardo. Para ello, se ha empleado el objeto *Puntos*, de la clase *ConjuntoParticulas*. El árbol de elementos de la *Vista* se muestra en la Figura 13.4.

Los valores de las propiedades más relevantes del objeto *Puntos* son las siguientes (vea la Figura 13.5):

- Propiedad *Elementos*, que es el número de partículas (en este caso n_{total}).
- Propiedades *X* e *Y*, que son vectores que almacenan las coordenadas (x, y) de las partículas.

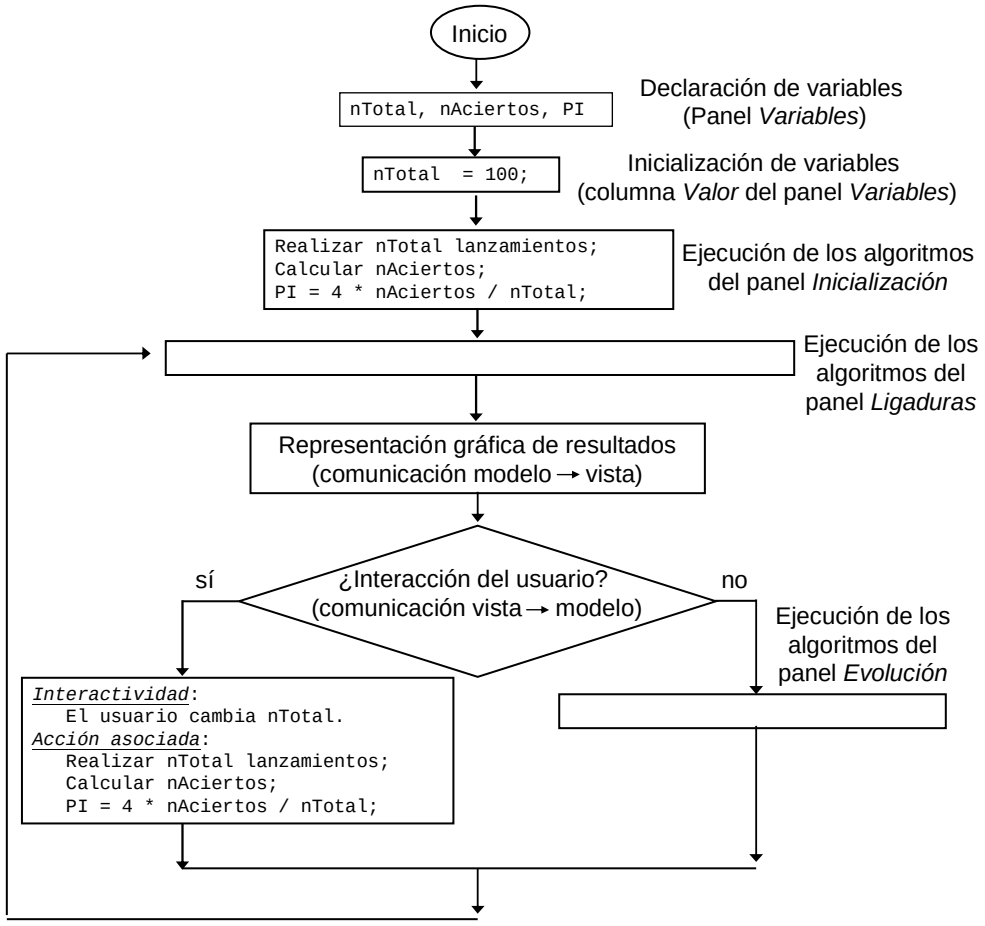


Figura 13.1: Algoritmo de la simulación.

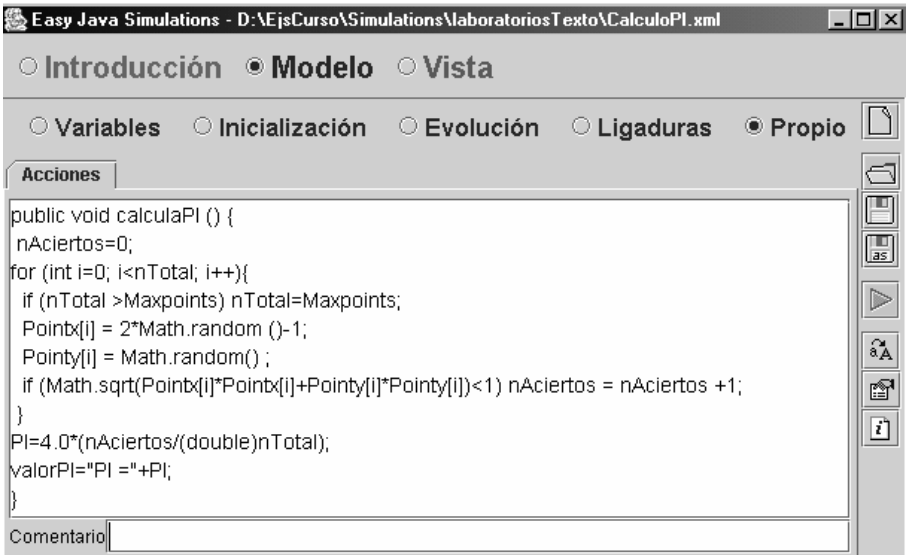


Figura 13.2: Método *calculaPI*. Estima el valor de π para un determinado n_{total}.

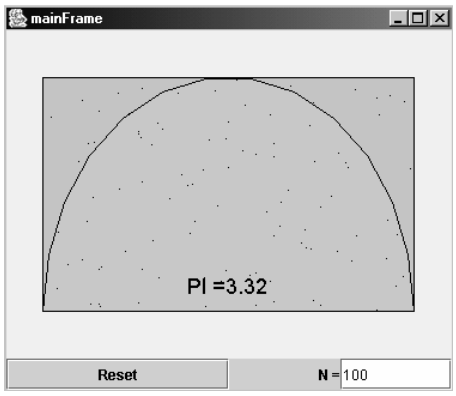


Figura 13.3: *Vista* del laboratorio virtual.

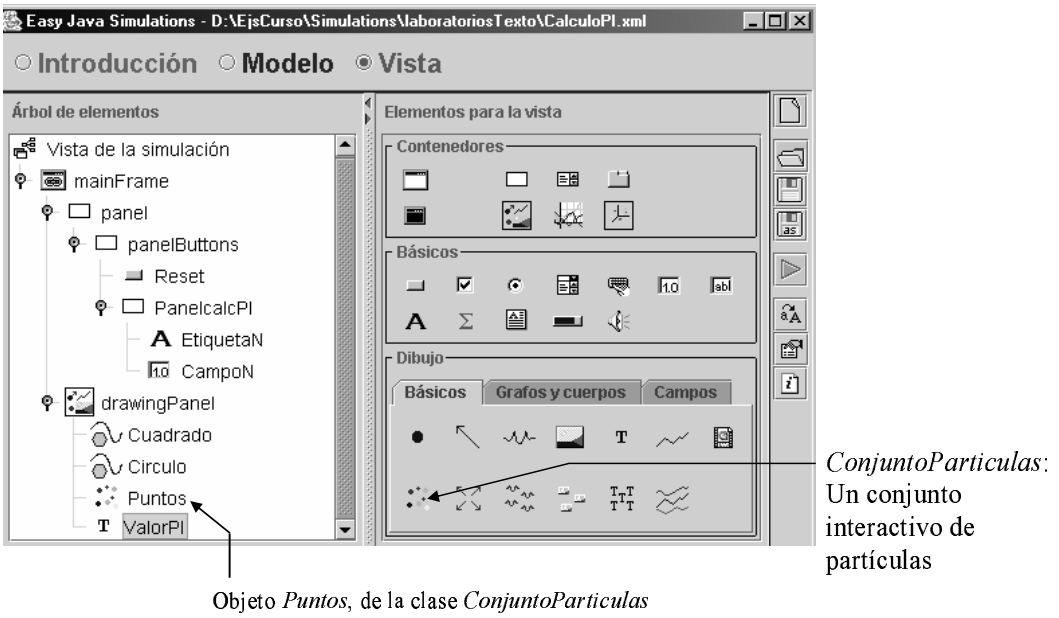


Figura 13.4: Árbol de elementos de la *Vista*.

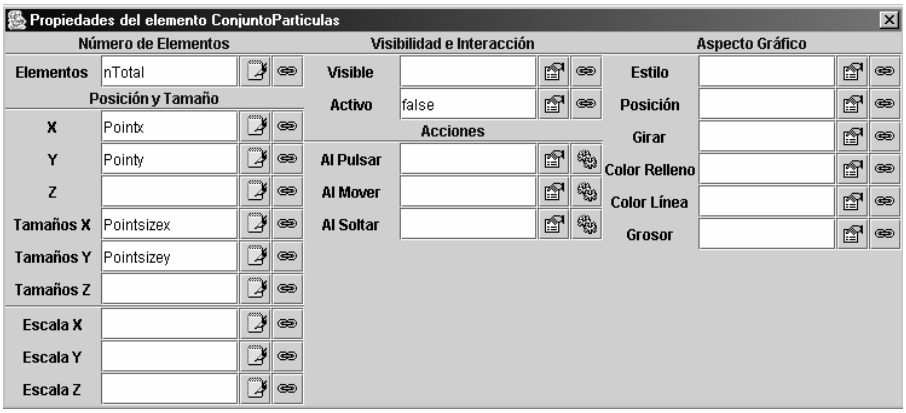


Figura 13.5: Propiedades del elemento *Puntos*, de la clase *ConjuntoParticulas*.

- Propiedad *Tamaños X*, que es tamaño en la dirección del eje X de cada partícula. La propiedad *Tamaños Y* es tamaño en la dirección del eje Y de cada partícula.
- Propiedad *Activo*, que es una variable booleana que indica si el elemento es interactivo.

Tema 14

Simulación interactiva de un globo aerostático

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Diseñar el algoritmo de simulación de un modelo dinámico que contiene ecuaciones diferenciales ordinarias y ecuaciones algebraicas, y programarlo usando Ejs.
- Programar métodos propios, que son invocados desde el panel Evolución.
- Usar la clase de elementos gráficos *Imagen*.

El objetivo de este laboratorio virtual es ilustrar el funcionamiento de un globo aerostático y constituye un bonito ejemplo de aplicación del principio de Arquímedes.

Este laboratorio virtual está incluido en el CD del curso. Se trata del fichero *GloboAerostatico.xml*, que está grabado en el directorio *laboratoriosTexto*.

14.1. Modelo de un globo aerostático

El globo aerostático obtiene su fuerza de sustentación mediante el calentamiento de aire dentro de una cavidad. La diferencia de densidades entre el aire caliente dentro de la cavidad y el aire frío del exterior origina una fuerza debida al empuje de Arquímedes que compensa el peso total de globo (teniendo en cuenta el peso de los ocupantes y el lastre).

Sobre el globo actúan dos fuerzas de sentido contrario:

– La fuerza del flotación (F_l).

$$F_l = \rho \cdot V \cdot g \tag{14.1}$$

donde V es el volumen del globo, ρ es la densidad del aire atmosférico y g es la aceleración gravitatoria.

– El peso total del globo (w) es la suma de tres términos:

1. El peso del gas del globo ($\rho_{dg} \cdot V \cdot g$).
2. El peso total (w_s) de la tela del globo, la barquilla del globo y los pasajeros.
3. El peso del lastre (w_l).

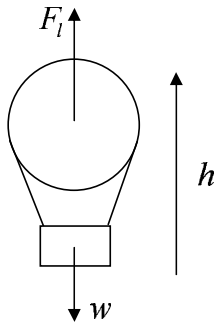


Figura 14.1: Diagrama de cuerpo libre del globo aerostático.

Es decir, el peso total del globo es:

$$w = \rho_{dg} \cdot V \cdot g + w_s + w_l \tag{14.2}$$

donde ρ_{dg} es la densidad del aire contenido dentro del globo.

Dado que la fuerza neta que actúa sobre el globo es $F_l - w$, aplicando la Ley de Newton se obtiene:

$$F_l - w = m \cdot \frac{dv}{dt} \tag{14.3}$$

$$v = \frac{dh}{dt} \tag{14.4}$$

donde h es la posición vertical del globo respecto al nivel del mar, v es la velocidad de ascenso del globo y m es el peso total del globo:

$$m = \frac{w}{g} \tag{14.5}$$

Para obtener la fuerza de sustentación (F_l) se ha aplicado el principio de Arquímedes (vea la Ecuación (14.1)), considerando que el volumen del globo (V) es constante y teniendo en cuenta la dependencia de la densidad del aire (ρ) con la presión (P) y la temperatura (T).

Se ha modelado el aire como un gas perfecto, por lo que se puede calcular su densidad aplicando la ecuación de estado de los gases perfectos. Así pues, la densidad del aire atmosférico (ρ) y la la densidad del aire contenido dentro del globo (ρ_{dg}) se calculan de la forma siguiente:

$$\rho = \frac{P \cdot M}{R \cdot T} \tag{14.6}$$

$$\rho_{dg} = \frac{P \cdot M}{R \cdot T_{dg}} \tag{14.7}$$

siendo R la constante de los gases ideales y M el peso molecular del aire.

Obsérvese que se ha considerado que la presión del aire dentro del globo es la misma en todo momento que la presión del aire atmosférico.

La dependencia de la temperatura y de la presión del aire de la atmósfera con la altura se ha obteniendo considerando el modelo de la atmósfera estándar hasta los 11 Km de altura, que establece que la temperatura (T) desciende 6.5 grados cada kilómetro:

$$T = T_0 - \left(6.5 \cdot 10^{-3} \text{ grados/metro}\right) \cdot h \tag{14.8}$$

y que la presión (P) obedece a la fórmula siguiente:

$$P = P_0 \cdot \left(\frac{T_0}{T}\right)^{-5.256} \tag{14.9}$$

donde T_0 y P_0 son la temperatura y la presión al nivel del mar respectivamente.

Finalmente, la variación en la temperatura del aire contenido dentro del globo (T_{dg}) depende de que el quemador esté apagado ($apagado = 1$) o encendido ($apagado = 0$). Cuando el quemador está apagado, existe un flujo de calor por convección entre el aire del globo y el aire atmosférico. Cuando el quemador está encendido, el flujo de calor procedente del mismo hace que aumente la temperatura del aire contenido dentro del globo.

La ecuación que describe la variación de la temperatura T_{dg} en ambas situaciones es:

$$\frac{dT_{dg}}{dt} = apagado \cdot K \cdot (T - T_{dg}) + (1 - apagado) \cdot q \tag{14.10}$$

14.2. El algoritmo de la simulación

Resumiendo la descripción del modelo realizada en la Sección 14.1, el modelo consta de las diez ecuaciones siguientes:

$$\begin{aligned}
F_l &= \rho \cdot V \cdot g && (14.11)\\
w &= \rho_{dg} \cdot V \cdot g + w_s + w_l && (14.12)\\
F_l - w &= m \cdot \frac{dv}{dt} && (14.13)\\
v &= \frac{dh}{dt} && (14.14)\\
m &= \frac{w}{g} && (14.15)\\
\rho &= \frac{P \cdot M}{R \cdot T} && (14.16)\\
\rho_{dg} &= \frac{P \cdot M}{R \cdot T_{dg}} && (14.17)\\
T &= T_0 - 6.5 \cdot 10^{-3} \cdot h && (14.18)\\
P &= P_0 \cdot \left(\frac{T_0}{T}\right)^{-5.256} && (14.19)\\
\frac{dT_{dg}}{dt} &= apagado \cdot K \cdot (T - T_{dg}) + (1 - apagado) \cdot q && (14.20)
\end{aligned}$$

Las variables se clasifican de la forma siguiente:

Variables conocidas. Parámetros:

- Temperatura al nivel del mar (T_0).
- Presión al nivel del mar (P_0).
- Volumen del globo (V).
- Peso molecular del aire (M).
- Aceleración gravitatoria (g).
- Peso de la tela del globo, la barquilla y los ocupantes (w_s).
- Peso del lastre (w_l).

– Constante de los gases ideales (R).
– Estado del calentador ($apagado$).
– Variación de la temperatura cuando el calentador está encendido (q).
– Coeficiente de transferencia de calor entre el aire del globo y la atmósfera (K).

Variables conocidas. Tiempo y variables de estado:

– Tiempo (t).
– Altura a la que está situado el globo (h), tomando como referencia el nivel del mar.
– Velocidad ascendente del globo (v).
– Temperatura del gas contenido dentro del globo (T_{dg}).

Variables desconocidas. Variables algebraicas del modelo:

– Presión del aire atmosférico (P).
– Temperatura del aire atmosférico (T).
– Densidad del aire atmosférico a la altura a la que está situado el globo (ρ).
– Densidad del aire contenido dentro del globo (ρ_{dg}).
– Fuerza de sustentación (F_l).
– Peso total del globo (w).
– Masa total del globo (m).

Variables desconocidas. Variables auxiliares que sustituyen a las derivadas de las variables de estado:

– Derivada de la altura del globo ($derh$).
– Derivada de la velocidad de ascenso del globo ($derv$).
– Derivada de la temperatura del aire contenido dentro del globo ($derT_{dg}$).

Asignando la causalidad computacional a las Ecuaciones (14.11) – (14.20), ordenándolas y despejando la variable a evaluar de cada una de ellas (que se señala incluyéndola entre corchetes) se obtiene:

$$[T] = T_0 - 6.5 \cdot 10^{-3} \cdot h \tag{14.21}$$

$$[P] = P_0 \cdot \left(\frac{T_0}{T}\right)^{-5.256} \tag{14.22}$$

$$[\rho] = \frac{P \cdot M}{R \cdot T} \tag{14.23}$$

$$[F_l] = \rho \cdot V \cdot g \tag{14.24}$$

$$[\rho_{dg}] = \frac{P \cdot M}{R \cdot T_{dg}} \tag{14.25}$$

$$[w] = \rho_{dg} \cdot V \cdot g + w_s + w_l \tag{14.26}$$

$$[m] = \frac{w}{g} \tag{14.27}$$

$$[derv] = \frac{F_l - w}{m} \tag{14.28}$$

$$[derh] = v \tag{14.29}$$

$$[derT_{dg}] = apagado \cdot K \cdot (T - T_{dg}) + (1 - apagado) \cdot q \tag{14.30}$$

En una página de EDO, en el panel *Evolución*, deben calcularse las variables de estado (h, v, T_{dg}) en función de sus derivadas. Para ello, se va a usar uno de los métodos de integración que proporciona Ejs: el método Runge-Kutta de cuarto orden.

Para el cálculo de las derivadas se han programado dos métodos propios en la página *Propio* de Ejs (ver Figura 14.4). Estos métodos propios son invocados desde el panel *Evolución* de Ejs para calcular las derivadas (ver Figura 14.5).

Como estos métodos son invocados para el cálculo de las derivadas desde una página Edo de Ejs, únicamente pueden ser función de variables de estado, parámetros y de la variable tiempo.

A continuación, se describen brevemente los dos métodos programados:

- Método *Flcalc*: Devuelve el valor de F_l y tiene como argumentos h, P_0 y T_0. El valor de F_l es calculado en función de los valores de h, P_0, T_0 y otros parámetros que no varían durante la simulación.

  La expresión para el cálculo de F_l se obtiene combinando las Ecs. (14.21) – (14.24):

$$F_l = \frac{P_0 \cdot \left(\frac{T_0}{(T_0 - 6.5 \cdot 10^{-3} \cdot h)}\right)^{-5.256} \cdot M}{R \cdot (T_0 - 6.5 \cdot 10^{-3} \cdot h)} \cdot V \cdot g \tag{14.31}$$

- Método *wcalc*: Devuelve el valor de w y tiene como argumentos P_0, T_0, h, T_{dg} y w_l. El valor de w es calculado en función de los valores de P_0, T_0, h, T_{dg}, w_l y otros parámetros que no varían durante la simulación.

  La expresión para el cálculo de w se obtiene combinando las Ecs. (14.21), (14.22), (14.25) y (14.26):

$$w = \frac{P_0 \cdot \left(\frac{T_0}{(T_0 - 6.5 \cdot 10^{-3} \cdot h)}\right)^{-5.256} \cdot M}{R \cdot T_{dg}} \cdot V \cdot g + w_s + w_l \tag{14.32}$$

Dentro del panel *Evolución*, se crea una segunda página, que se sitúa a la derecha de la página EDO (es decir, se ejecuta a continuación de la página de EDO). Esta segunda página tiene como finalidad evitar que el globo descienda por debajo de $h = 0$.

El código de esta página de evolución es el siguiente:

```
if ( h < 0 && ( Fl - w ) < 0 ) {
   h = 0;
   v = 0;
   }
```

14.3. Programación de la vista

En la Figura 14.2 se muestra la vista del laboratorio virtual.

La vista se compone de una ventana principal (*mainFrame*) y una ventana de diálogo (*VentanaDialogoPlot*). Esta última contiene gráficas de la evolución temporal de ciertas variables de interés: la altura del globo (h), la presión y la temperatura del aire a dicha altura (P, T) , y la temperatura del gas del globo (T_{dg}).

En la parte inferior de la ventana principal hay botones para controlar la simulación (*Pause*, *Reset* y *Play*), un selector que permite mostrar y ocultar la ventana de diálogo (*VentanaDialogoPlot*), botones que permiten encender y apagar el quemador, y casillas que permiten introducir la temperatura y la presión al nivel del mar (T_0, P_0), y la masa del lastre (m_l).

El árbol de elementos de la vista se muestra en la Figura 14.3. Se ha introducido en la vista un elemento de la clase *Imagen*, que se explica a continuación al ser la primera vez que aparece en este texto.

La clase *Imagen* es un elemento gráfico que muestra una imagen *.gif* o *.jpg* con el tamaño y posición especificados en sus propiedades. Las propiedades más relevantes de la clase *Imagen* son:

- *X*: coordenada X de la posición de la imagen.
- *Y*: coordenada Y de la posición de la imagen.
- *Tamaño X*: tamaño de la imagen según el eje X.
- *Tamaño Y*: tamaño de la imagen según el eje Y.
- *Activo*: indica si el elemento es interactivo.
- *Posicion*: indica la situación del "punto sensitivo" de la imagen respecto de las coordenadas especificadas para dicha imagen.

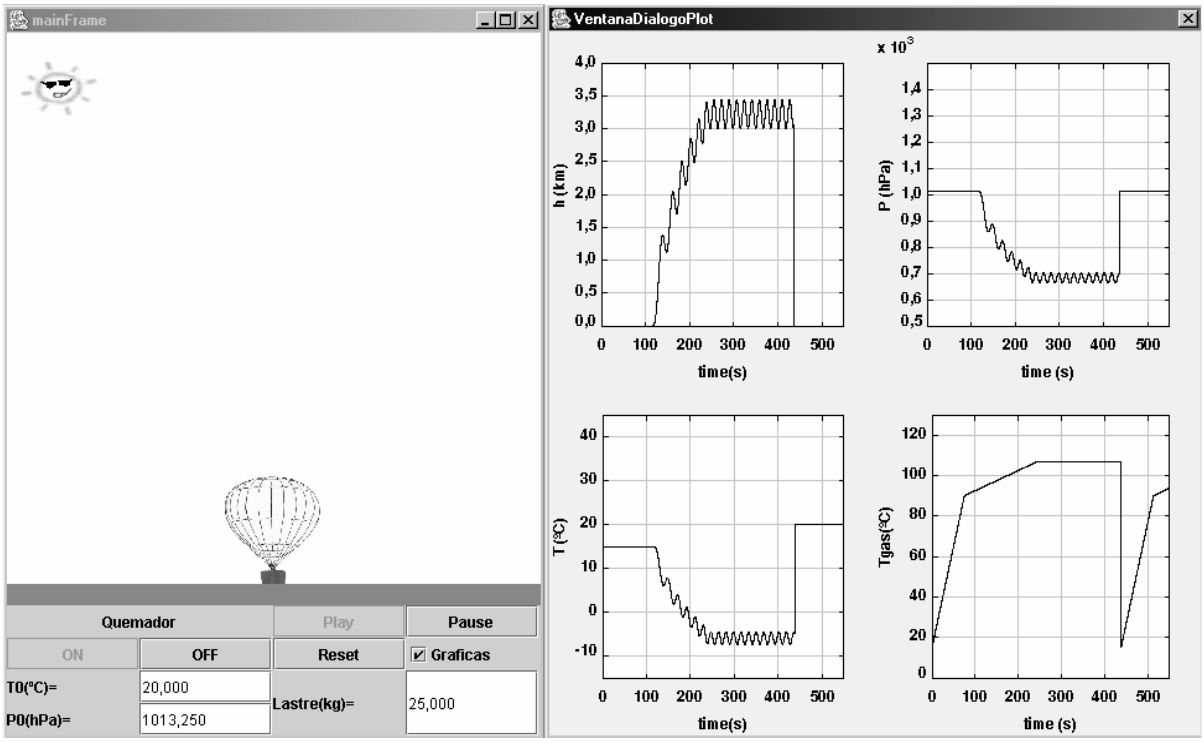


Figura 14.2: *Vista* del laboratorio virtual.

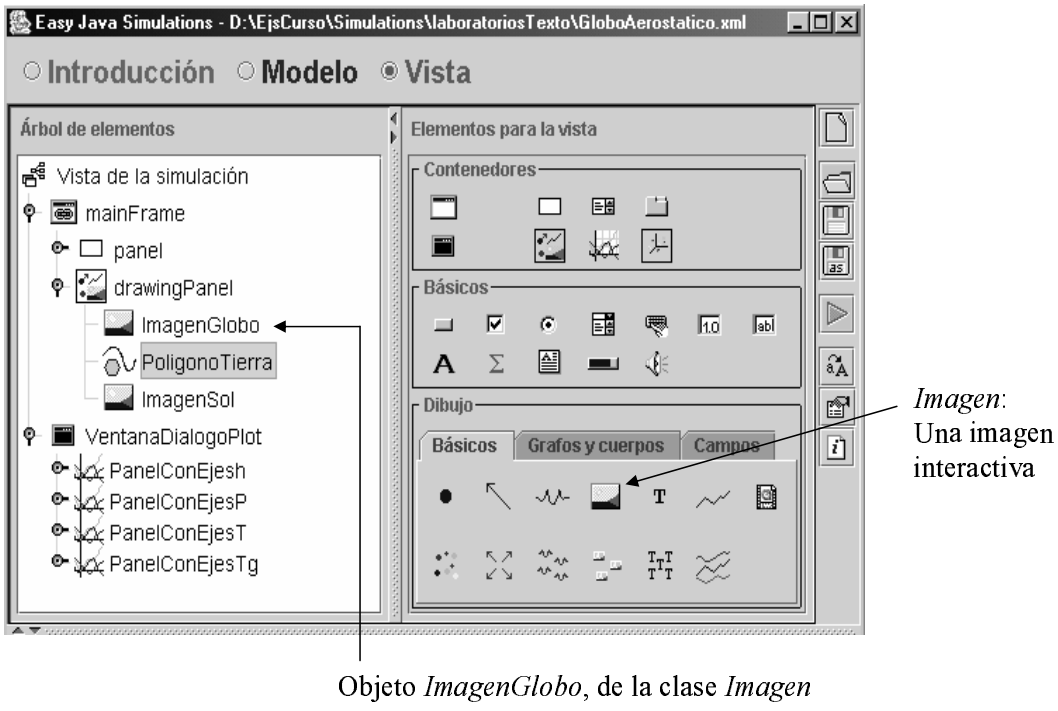


Figura 14.3: Árbol de elementos de la vista.

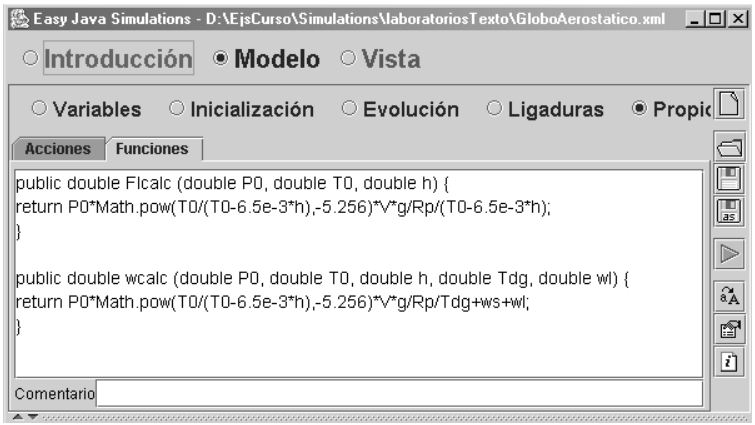


Figura 14.4: Panel *Propio*.

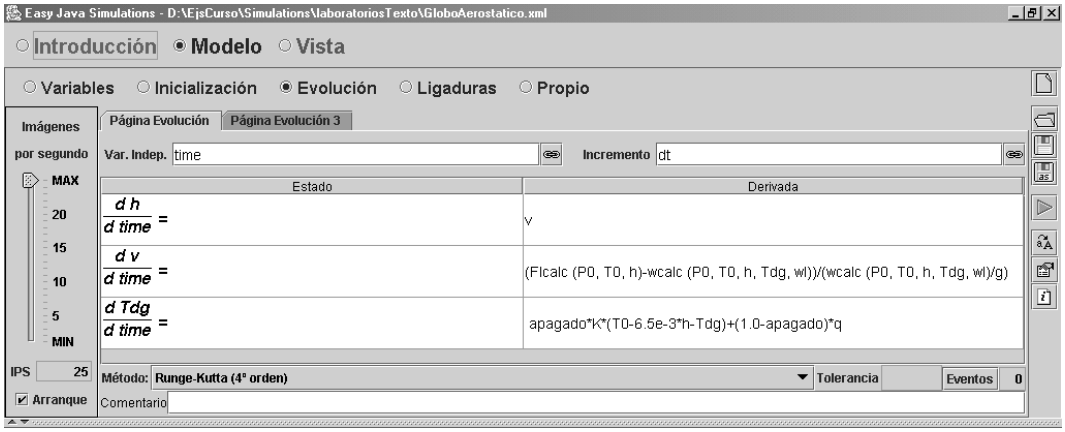


Figura 14.5: Panel *Evolución*.

Tema 15

Laboratorio virtual del sistema bola y varilla

Objetivos: Una vez estudiado el contenido del tema debería saber:

- Diseñar el algoritmo de simulación de un modelo con una parte de tiempo continuo y otra de tiempo discreto, y programarlo usando Ejs.
- Usar las clases de elementos gráficos: Particula y BotonRadio.

Este laboratorio virtual está grabado en el CD del curso. Se trata del fichero *Balland-Beam.xml*, que está en el directorio *laboratoriosTexto*.

15.1. Modelo del sistema bola y varilla

Este sistema está compuesto de una varilla, un motor y una bola. La varilla está unida a un motor que permite variar el ángulo θ que forma la varilla con respecto a la horizontal. La bola está situada sobre la varilla y es libre de rodar bajo la acción de la gravedad (un pequeño canal en la varilla impide que la bola ruede hacia un lado).

El modelo que describe el movimiento de la bola, considerando que la bola rueda sobre la varilla, es:

$$\frac{dx}{dt} = v \tag{15.1}$$

$$\frac{dv}{dt} = -\frac{5}{7} \cdot g \cdot \sin(\theta) \tag{15.2}$$

donde x es la posición de la bola (vea la Figura 15.1), v es su velocidad, θ es el ángulo de la varilla y g la aceleración gravitatoria.

Este modelo es inestable y constituye un buen ejemplo para aprender a aplicar técnicas de control automático de sistemas. El objetivo de control es mover la bola a la posición deseada, cambiando el ángulo de la varilla. Así pues, la variable controlada es la posición de la pelota (x) y la variable manipulada es el ángulo de la varilla (θ).

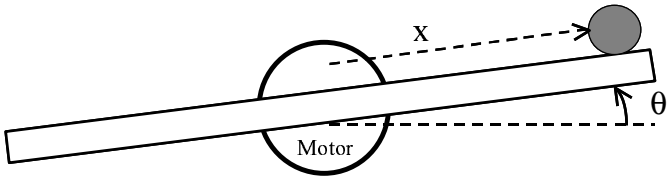


Figura 15.1: Diagrama del sistema bola y varilla.

En este laboratorio virtual el sistema se ha controlado mediante un controlador PID discreto, cuyos parámetros pueden ser modificados interactivamente durante la simulación. También pueden ser modificados interactivamente la posición de la bola (x), la inclinación de la varilla (θ) y el valor de consigna para la posición de la bola. A continuación se describe el modelo del controlador PID empleado.

Controlador PID discreto

Se ha programado un controlador PID discreto con dos grados de libertad y anti-windup.

En la descripción del algoritmo del controlador se emplea la notación siguiente:

- El valor de la consigna, en el instante de tiempo t_n, es la posición en la que se desea situar la bola. Se denomina ref_n.
- El valor de la salida del sistema en el instante t_n es la posición de la bola en dicho instante, que se denomina x_n.
- La salida del controlador, en el instante de tiempo t_n, es el ángulo de la varilla en dicho instante: θ_n.

El algoritmo que describe el controlador es el siguiente[1]:

1. Inicialización del controlador:

$$n = 0 \tag{15.3}$$
$$I_0 = 0 \tag{15.4}$$
$$D_0 = 0 \tag{15.5}$$

   Al inicializar el modelo de la planta debe asignarse valor a x_0.
2. $n = n + 1$
3. Leer el valor de la referencia (ref_n) y de la variable controlada (x_n).
4. Cálculo de P_n:
$$P_n = K_p \cdot (b \cdot ref_n - x_n) \tag{15.6}$$
5. Cálculo de D_n:
$$D_n = A_d \cdot D_{n-1} - B_d \cdot (x_n - x_{n-1}) \tag{15.7}$$
6. Cálculo de u_n:
$$u_n = P_n + I_n + D_n \tag{15.8}$$
7. Cálculo del valor saturado de la señal de control:
$$\theta_n = \text{mín}\left(u_{\text{máx}}, \text{máx}\left(u_{\text{mín}}, u_n\right)\right) \tag{15.9}$$
8. Cálculo del término integral en el instante t_{n+1}:
$$I_{n+1} = I_n + B_i \cdot (ref_n - x_n) + A_r \cdot (\theta_n - u_n) \tag{15.10}$$

[1]El modelo de este controlador está extraído del texto (Astrom & Wittenmark 1997).

9. Saltar al Paso 2.

Como puede observarse en el algoritmo anterior, el controlador tiene los parámetros siguientes:

- La constante proporcional (K_p).
- La fracción del valor de consigna en el término proporcional (b).
- Valor mínimo de la señal de control ($u_{\text{mín}}$).
- Valor máximo de la señal de control ($u_{\text{máx}}$).
- Tiempo de muestreo (T_s).
- Constante de tiempo integral (T_i).
- Constante de tiempo derivativa (T_d).
- Constante de tiempo de reinicio (T_t).
- Ganancia máxima derivativa (N).

Asimismo, los parámetros B_i, A_r, A_d y B_d se calculan, a partir de los parámetros anteriores, de la forma siguiente:

$$B_i = \frac{K_p \cdot T_s}{T_i} \tag{15.11}$$

$$A_r = \frac{T_s}{T_t} \tag{15.12}$$

$$A_d = \frac{T_d}{T_d + N \cdot T_s} \tag{15.13}$$

$$B_d = K_p \cdot N \cdot A_d \tag{15.14}$$

15.2. El algoritmo de la simulación

En la Figura 15.2 se muestra el algoritmo de la simulación del laboratorio virtual.

El cálculo de las variables de estado continuas (x, v) y de la variable de estado discreta (I) se realiza en el panel *Evolución*.

El periodo de muestreo del controlador debe ser un múltiplo entero del tamaño del paso de integración de las ecuaciones diferenciales ordinarias. Para sincronizar la solución de la parte discreta del modelo con la integración de la parte continua, se ha definido la variable *tSampleCounter*:

- Cada vez que el método de integración avanza un paso, se incrementa el valor de *tSampleCounter*.
- Cuando *tSampleCounter* alcanza el valor *tSample* (T_s), se ejecuta la parte discreta del modelo y a continuación se pone a cero el valor de *tSampleCounter*.

Las incógnitas algebraicas, tanto de tiempo continuo ($derx$, $dery$) como de tiempo discreto (P, D, u, θ) de calculan en el panel *Ligaduras*.

En las Figura 15.3 se muestra la ventana del panel *Ligaduras* en la que se calculan las variables algebraicas de tiempo discreto.

En las Figuras 15.4 y 15.5 se muestran las ventanas de Evolución para el cálculo de las variables de estado discretas y continuas del modelo respectivamente.

15.3. Programación de la vista

La *Vista* del laboratorio virtual, que se muestra en la Figura 15.6, se compone de:

- Una ventana principal (*mainFrame*).

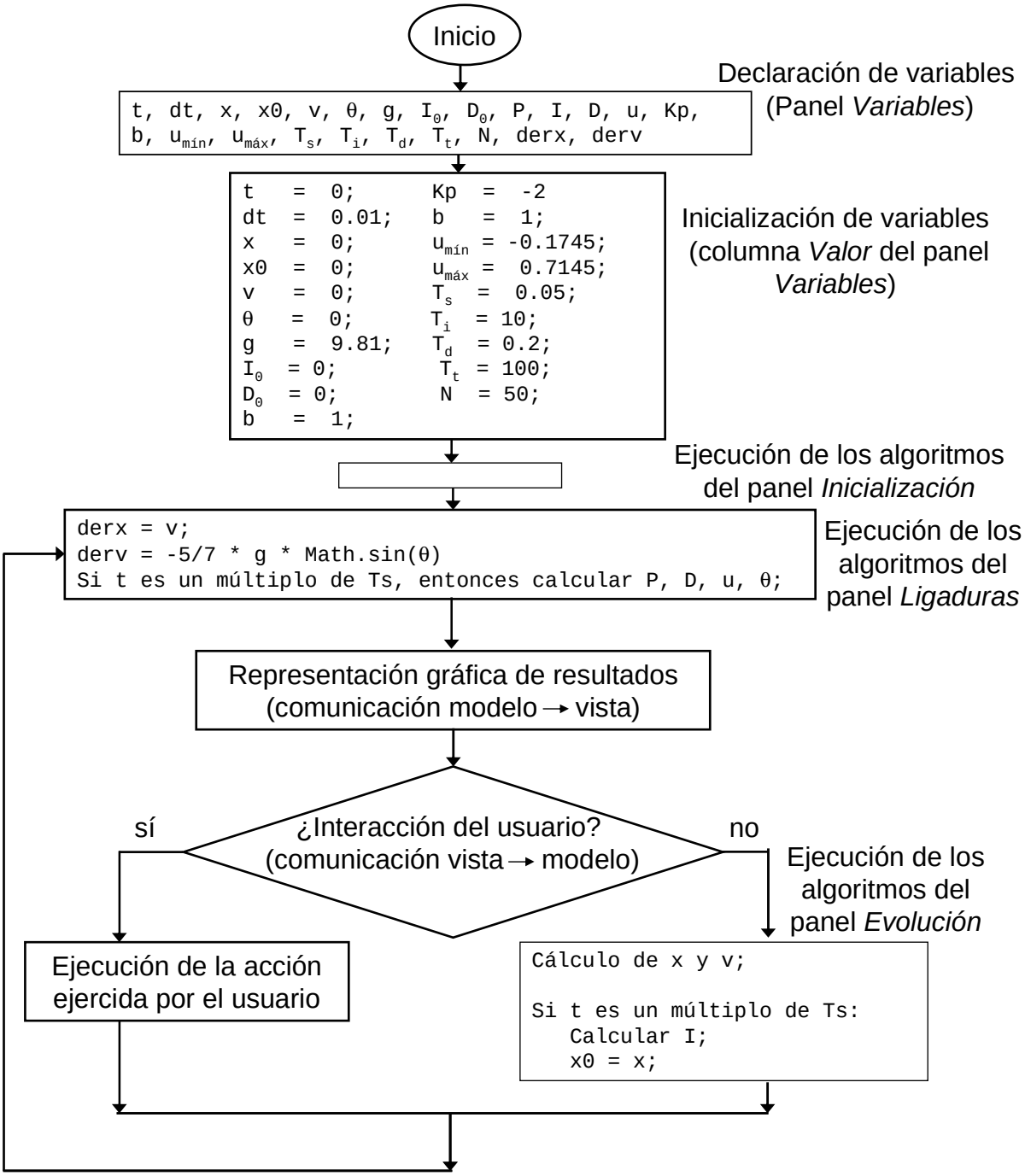


Figura 15.2: El algoritmo de la simulación.

Figura 15.3: Cálculo de las variables algebraicas de tiempo discreto.

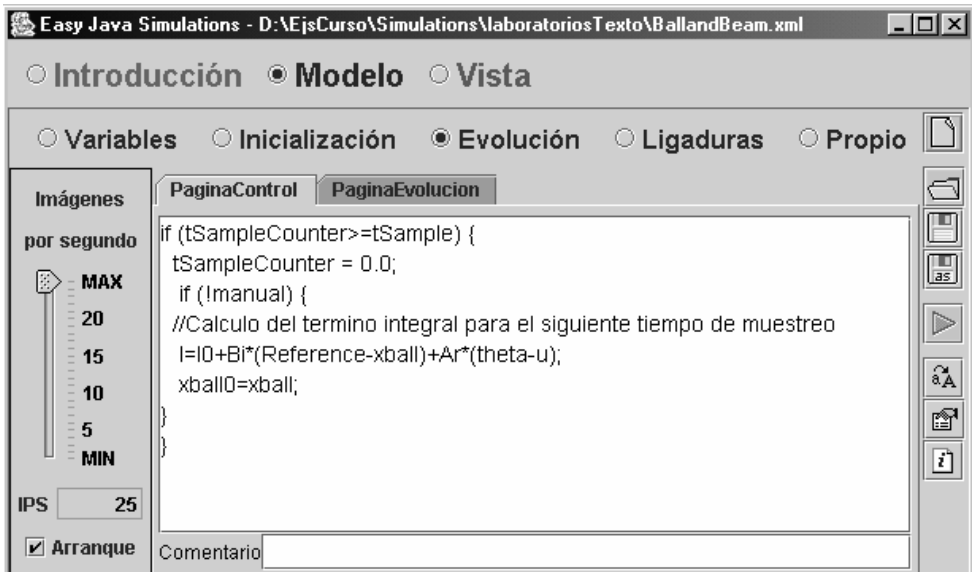


Figura 15.4: Cálculo de la variable de estado de tiempo discreto.

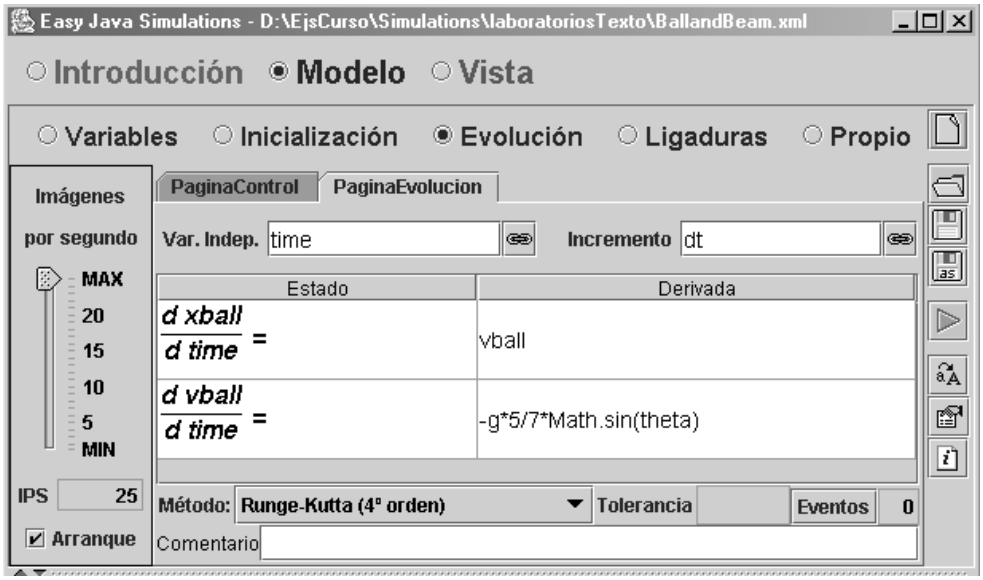


Figura 15.5: Cálculo de las variables de estado de tiempo continuo.

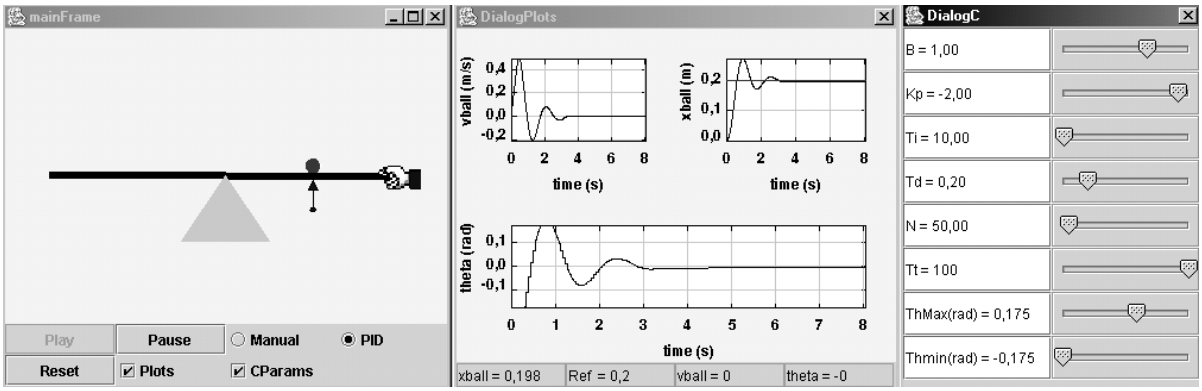


Figura 15.6: *Vista* del laboratorio virtual.

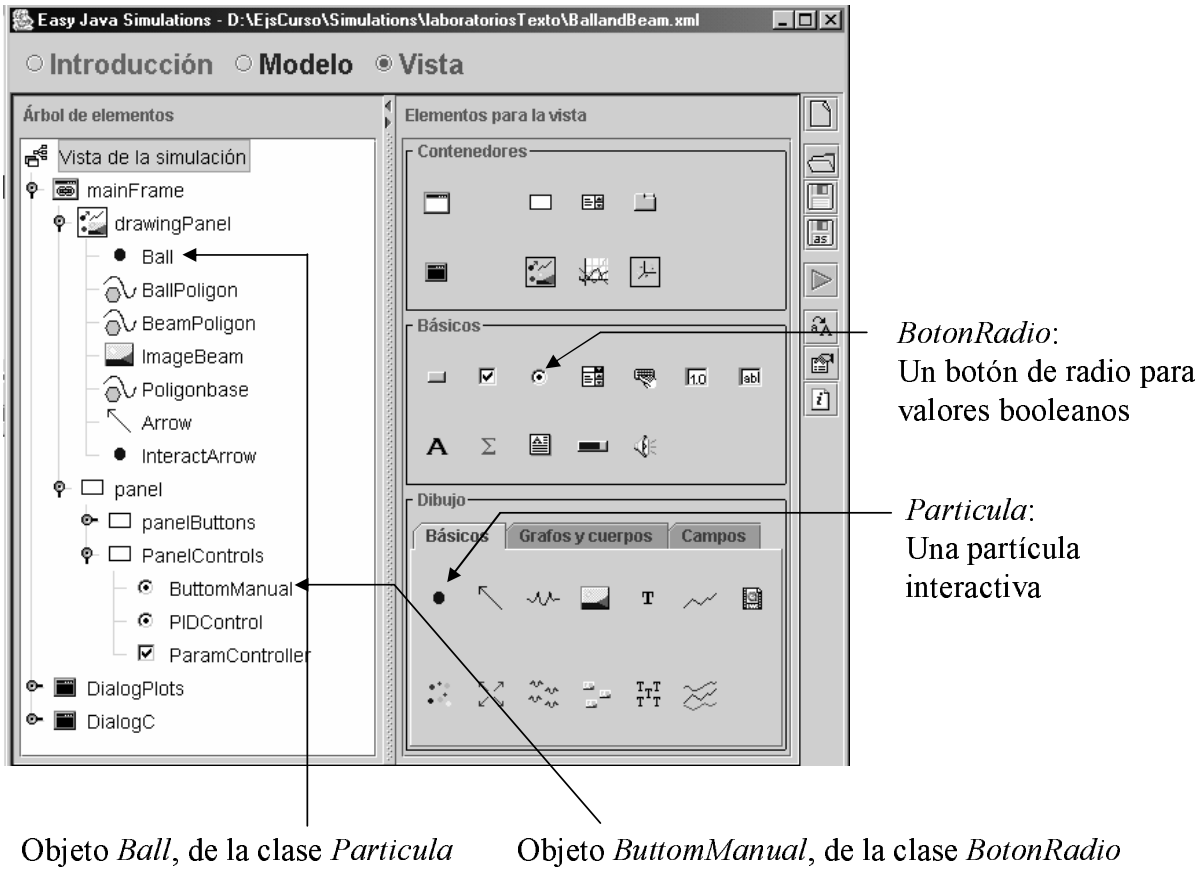


Figura 15.7: Árbol de elementos de la *Vista* del laboratorio virtual.

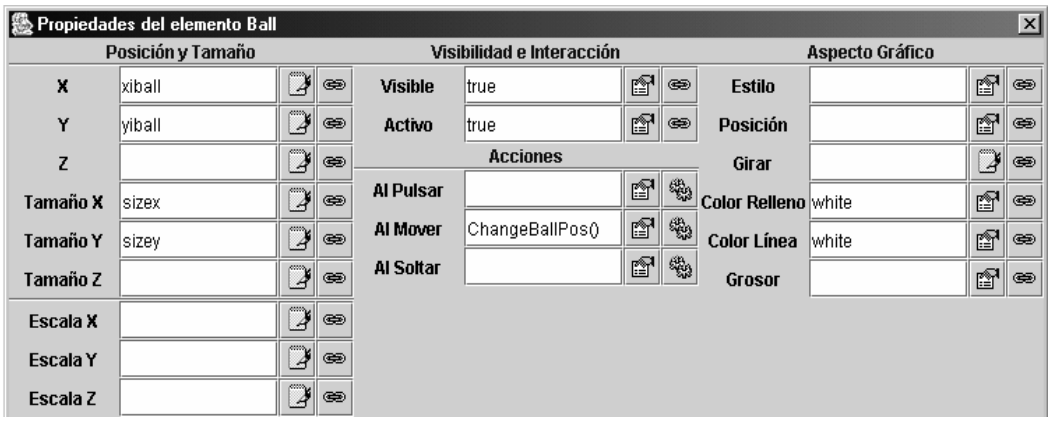


Figura 15.8: Propiedades del objeto *Ball*, de la clase *Particula*.

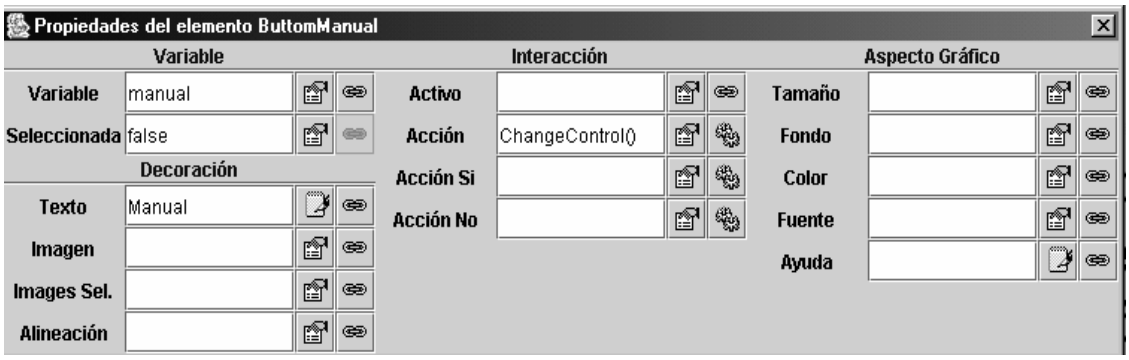


Figura 15.9: Propiedades del objeto *ButtonManual*, de la clase *BotonRadio*.

- La ventana de diálogo *dialogPlots*, donde se muestra la evolución temporal de las variables relevantes del sistema.
- La ventana de diálogo *dialogC*, donde se encuentran casillas y deslizadores (sliders) que permiten modificar los valores de los parámetros del controlador.

El árbol de la *Vista* de este laboratorio se muestra en la Figura 15.7. La ventana principal contiene en el centro una representación esquemática del sistema bola y varilla. En la parte inferior de esta ventana hay botones que permiten controlar la simulación (*Play*, *Pause* y *Reset*), radio-botones que permiten pasar de control manual a automático o viceversa, y selectores que permiten mostrar y ocultar las dos ventanas de diálogo.

En la programación de la *Vista* se han empleado dos nuevas clases de elementos gráficos que no habían sido usadas anteriormente en este texto:

- Particula.
- Radioboton.

A continuación se describen algunas propiedades relevantes de cada una de ellas.

La clase de elemento gráfico *Particula*

Las propiedades del objeto *Ball*, de la clase Partícula se muestran en la Figura 15.8. Las propiedades más relevantes de esta clase son:

- *X*: coordenada X de la posición de la partícula.
- *Y*: coordenada Y de la posición de la partícula.
- *Tamaño X*: tamaño de la partícula según el eje X.
- *Tamaño Y*: tamaño de la partícula según el eje Y.
- *Visible*: indica si el elemento es visible o no.
- *Activo*: indica si el elemento es interactivo.
- *Al pulsar*: acción que se produce cuando el botón del ratón se presiona sobre el objeto.
- *Al mover*: acción que se produce cuando se mueve el objeto con el ratón.
- *Al soltar*: acción que se produce cuando se deja de presionar el botón del ratón.

La clase de elemento gráfico *BotonRadio*

Las propiedades del objeto *ButtonManual*, de la clase *BotonRadio*, se muestran en la Figura 15.9. La clase *BotonRadio* se encuentra en el panel *Básicos*. Esta clase se usa para mostrar y modificar un valor booleano.

Cuando más de un objeto de la clase *BotonRadio* está en el mismo contenedor, sólo uno puede ser seleccionado en cada momento, aunque es posible que no haya ninguno seleccionado.

Las propiedades más relevantes de esta clase son las siguientes:

- *Variable*: valor que se muestra y modifica al accionar el botón.
- *Seleccionada*: valor inicial de la variable.
- *Acción*: acción que se produce cuando se pulsa el botón.
- *Acción Si*: acción que se produce cuando se selecciona el elemento.
- *Acción No*: acción que se produce cuando se deselecciona el elemento.

Parte IV

Apéndices

Apéndice A

Java para el desarrollo de laboratorios virtuales en Ejs

Objetivos: Una vez estudiado el contenido de este apéndice, debería conocer:

- Cuáles son los tipos de datos del lenguaje Java y cuáles encuentran aplicación en Ejs.
- Cómo se declaran e inicializan las variables, vectores y matrices en Ejs.
- Cuáles son los operadores de Java más comúnmente empleados en la descripción del modelo en Ejs.
- Las sentencias **if** y **for** de Java.
- Algunos fundamentos de la programación de métodos en Java.
- Algunos de los métodos de la clase **Math** de Java y cómo emplearlos.

A.1. Introducción

Para poder sacar el máximo partido a Ejs es necesario tener algunos conocimiento básicos del lenguaje de programación Java. Esto es debido a que el programador del laboratorio virtual debe describir en lenguaje Java el modelo matemático, y la interacción entre la vista y el modelo.

Este apéndice pretende servir como introducción y como guía rápida de referencia de aquellos aspectos del lenguaje Java que resultan de utilidad a la hora de realizar laboratorios virtuales usando Ejs.

En Internet está disponible abundante información acerca del lenguaje Java: cursos, documentación, ejemplos, foros, páginas de preguntas frecuentes, etc. También existe un gran número de libros[1] y manuales de Java, algunos de los cuales se encuentran disponibles gratuitamente en la red[2]. Todo ello puede servir para profundizar en el tema, a aquellas personas que estén interesadas en ello.

[1] En la escritura de este Apéndice se ha empleado como referencia el texto *"Java 2. Manual de Referencia. Cuarta Edición"*, escrito por Herbert Schildt y publicado en 2001 por la editorial McGraw-Hill. En nuestra opinión se trata de un magnífico libro, que recomendamos a todo aquel que esté interesado en aprender Java desde cero.

[2] Por ejemplo, el libro "Thinking in Java", de Bruce Eckel.

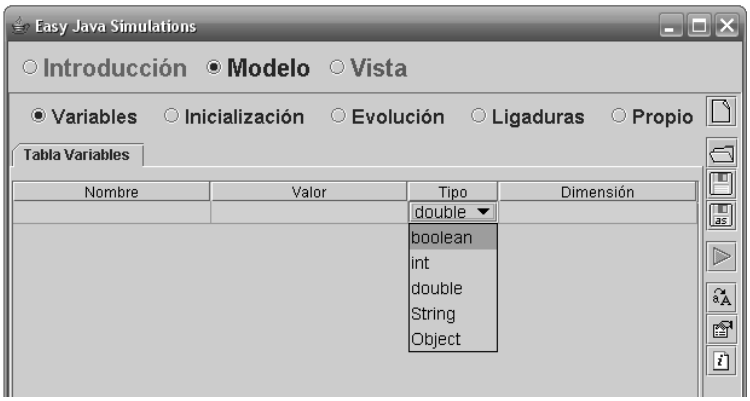


Figura A.1: Opciones para la declaración del tipo de las variables.

A.2. Tipos de datos

Al definir las variables del laboratorio virtual es preciso especificar a qué tipo pertenece cada una de ellas. Esto se hace seleccionado el tipo en la columna *Tipo* del panel *Variables* (véase la Figura A.1). Ejs ofrece cinco opciones: **boolean**, **int**, **double**, **String** y **Object**. Las tres primeras (**boolean**, **int**, **double**) corresponden a tipos simples de datos de Java. La cuarta, **String**, es el tipo empleado para definir cadenas de caracteres. En esta sección se explicarán cuáles son los tipos simples de datos y qué son las cadenas de caracteres en Java.

A.2.1. Tipos simples de datos

Java posee ocho tipos simples de datos, que se definen mediante las palabras reservadas **byte**, **short**, **int**, **long**, **char**, **float**, **double** y **boolean**. Estos ocho tipos pueden clasificarse en:

- *Tipos enteros*: **byte**, **short**, **int** y **long**. La diferencia entre estos cuatro tipos es el rango de valores admisibles. Por ejemplo, una variable del tipo **byte** sólo puede tomar valores enteros comprendidos entre -128 y 127, ambos inclusive. En la Tabla A.1 se muestra el rango de valores que pueden tomar las variables de cada tipo. El tipo entero utilizado en Ejs es **int**.
- *Tipos en coma flotante*: **float** y **double**. Representan números en simple y doble precisión respectivamente. El rango de valores para números positivos se muestra en la Tabla A.2. Además, ambos tipos comprenden los correspondientes números negativos y el cero. Por motivo de precisión en los cálculos, cuando se emplea Java para la simulación por ordenador de modelos matemáticos, el tipo **double** es la mejor elección para representar números en coma flotante. Por otra parte, todas las funciones matemáticas transcendentes (por ejemplo, sin, cos, sqrt) devuelven valores del tipo **double**. El tipo en coma flotante que usa Ejs es **double**.
- *Tipo carácter*: **char**. Éste es el tipo de datos que utiliza Java para almacenar caracteres. Java emplea *Unicode*[3], que es un juego de caracteres que comprende todos los caracteres que se pueden encontrar en todas las lenguas de la humanidad. Puesto que las variables en Ejs no pueden ser del tipo **char**, las variables de tipo carácter pueden definirse en Ejs como cadenas de caracteres, es decir, como variables del tipo **String**.
- *Tipo booleano*: **boolean**. Las variables de este tipo pueden tomar dos valores: *true* y *false*. Estos valores no se convierten a ninguna representación numérica. Es decir, el literal

[3]Se puede obtener más información sobre *Unicode* en http://www.unicode.org/

true en Java no es igual a 1, ni el literal *false* es igual a 0. En Java estos dos literales sólo pueden asignarse a las variables declaradas como **boolean** y ser empleadas en expresiones con operadores booleanos.

Tabla A.1: Rango de los tipos enteros.

Tipo de variable	Rango de valores		
byte	-128	a	127
short	-32,768	a	32,767
int	-2,147,483,648	a	2,147,483,547
long	-9,223,372,036,854,775,808	a	9,223,372,036,854,775,807

Tabla A.2: Rango de los tipos en coma flotante.

Tipo de variable	Rango de valores		
float	$3.4 \cdot 10^{-38}$	a	$3.4 \cdot 10^{+38}$
double	$1.7 \cdot 10^{-308}$	a	$1.7 \cdot 10^{-308}$

A.2.2. Cadenas de caracteres

Las *cadenas de caracteres* no son tipos simples de Java, sino objetos de la clase **String**.

Las cadenas de caracteres se especifican en Java encerrando la secuencia de caracteres entre dobles comillas. Por ejemplo:

```
"Hola mundo"
```

Algunos caracteres especiales, que pueden ser incluidos en las cadenas de caracteres, son los dos siguientes:

\n	(salto de línea)
\t	(tabulador)

Por ejemplo:

```
"Primera linea\nSegunda linea"
```

Por otra parte, como la comilla doble señala en principio y el fin de la cadena de caracteres, si se desea incluir el carácter comilla doble en la cadena de caracteres no puede incluirse tal cual, sino que debe escribirse:

\"

Algo similar sucede con la comilla simple: si se desea incluir un carácter comilla simple en la cadena de caracteres debe anteponerse el carácter barra inclinada, es decir:

\'

Si se desea incluir un carácter barra inclinada en la cadena de caracteres debe escribirse:

\\

A.3. Variables

En esta sección se explican las restricciones que impone el lenguaje Java respecto a la forma de nombrar las variables. Asimismo, se explica brevemente cómo se definen e inicializan en Ejs las variables multidimensionales (vectores y matrices). No entraremos en detalles acerca de cómo habría que realizar esta definición directamente en Java, puesto que no es necesario conocerlo para programar laboratorios virtuales en Ejs.

A.3.1. Nombre de las variables

El nombre de una variable puede ser cualquier *identificador* válido en Java. En Java, un identificador comienza con una letra, un subrayado (_) o un símbolo de dólar ($). Los siguientes caracteres pueden ser letras o dígitos. Se distinguen las mayúsculas de las minúsculas, y no hay longitud máxima. Por ejemplo, dos identificadores válidos son: `longPendulo2` y `long_Pendulo_2`.

Las palabras clave de Java no pueden ser usadas como identificadores. Éstas son:

abstract	continue	for	new	switch
boolean	default	goto	null	synchronized
break	do	if	package	this
byte	double	implements	private	threadsafe
byvalue	else	import	protected	throw
case	extends	instanceof	public	transient
catch	false	int	return	true
char	final	interface	short	try
class	finally	long	static	void
const	float	native	super	while

Además de estas palabras clave, hay palabras reservadas que tampoco pueden emplearse como identificadores. Éstas son:

cast	future	generic	inner
operator	outer	rest	var

A.3.2. Vectores y matrices

En la columna *Dimension* del panel *Variables* de Ejs puede especificarse la dimensión de las variables. En la Figura A.2 se muestra un ejemplo. La variable `posicion_planeta` es un vector de tres componentes de tipo **double**. La variable `nombres` es un vector de cuatro componentes de tipo **String**.

La inicialización de estas variables vectoriales puede hacerse en el panel *Inicialización* de Ejs. En la Figura A.3 se muestra un ejemplo. Obsérvese que el índice de los vectores comienza en cero. Por ejemplo, el vector `nombres`, de dimensión 4, posee los componentes siguientes:

```
nombres[0]    nombres[1]    nombres[2]    nombres[3]
```

A.4. Operadores

La mayor parte de los operadores de Java pueden clasificarse en los cuatro grupos siguientes: aritméticos, a nivel de bit, relacionales y lógicos. En esta sección se explicarán algunos de los operadores aritméticos, relacionales y booleanos más comúnmente usados.

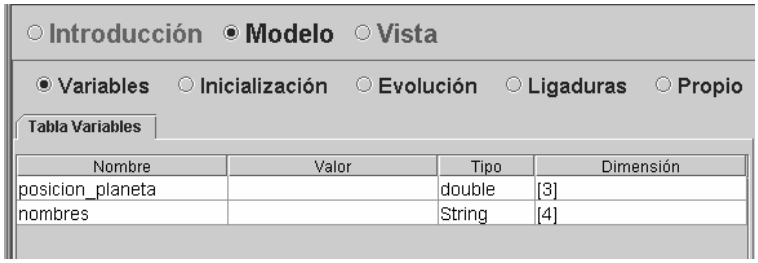


Figura A.2: Declaración del nombre, tipo y dimensión de variables vectoriales.

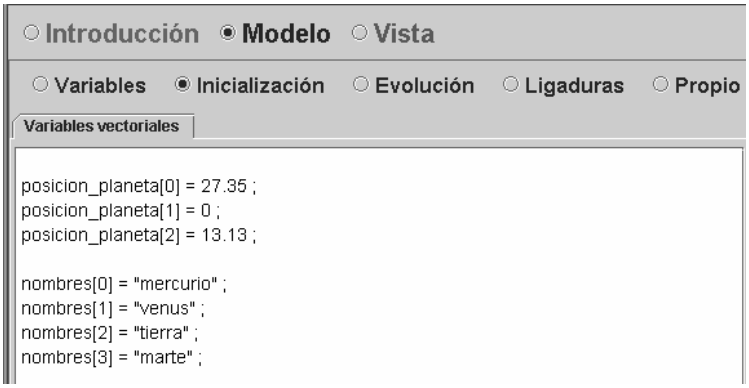


Figura A.3: Inicialización de variables vectoriales.

A.4.1. Operadores aritméticos

Los operadores aritméticos se utilizan el expresiones matemáticas, debiéndose aplicar sobre operandos de tipo numérico (es decir, en el caso de Ejs, sobre variables de los tipos **int** y **double**). A continuación, se muestran los más comúnmente empleados:

`+`	Suma
`-`	Resta
`*`	Multiplicación
`/`	División
`++`	Incremento
`--`	Decremento
`+=`	Suma y asignación
`-=`	Resta y asignación
`*=`	Producto y asignación
`/=`	División y asignación

Los operandos aritméticos básicos (suma, resta, multiplicación y división) se comportan de la forma habitual. No obstante, hay que tener en cuenta que cuando se opera el operador división a un tipo entero se pierde la componente decimal del resultado.

Java proporciona operadores especiales que permiten combinar una operación aritmética con una asignación. Se denominan *operadores de asignación*. Por ejemplo, una sentencia como la siguiente:

```
a = a + 5;
```

puede escribirse de la forma siguiente:

```
a += 5;
```

Ambas sentencias realizan la misma acción: incrementar el valor de a en 5. Análogamente, existen operadores de asignación para todos los operadores que se aplican sobre dos operandos. Cualquier sentencia de la forma:

```
variable = variable operador expresión;
```

puede escribirse de la forma siguiente:

```
variable operador= expresión;
```

Otro tipo de operadores aritméticos son los operadores *incremento* (--) y *decremento* (++). El operador incremento incrementa en una unidad su operando, mientras que el operador decremento lo reduce en una unidad. Por ejemplo, esta sentencia:

```
a = a + 1;
```

puede escribirse de la forma siguiente:

```
a++;
```

Análogamente, esta sentencia:

```
a = a - 1;
```

puede escribirse de la forma siguiente:

```
a--;
```

A.4.2. Operadores relacionales

Los *operadores relacionales* determinan la relación que un operando tiene con otro, dando como resultado un booleano. A continuación se muestran los operadores relacionales:

==	Igual a
!=	Distinto que
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

La aplicación más frecuente de estos operandos es es la obtención de expresiones que controlan la sentencia **if** y las sentencias de bucles (**for**, **while** y **do-while**).

A.4.3. Operadores lógicos booleanos

Los operadores lógicos que se muestran a continuación sólo operan sobre operandos del tipo **boolean**. Todos ellos combinan dos valores **boolean** para dar como resultado un valor **boolean**.

&	AND lógico
&&	AND en cortocircuito
|	OR lógico
||	OR en cortocircuito
!	NOT lógico unario
==	Igual a
!=	Distinto de

El operador NOT lógico (!) invierte el estado booleano:

```
!true = false                    !false = true
```

Con el fin de definir el significado de ambos operadores, en la siguiente tabla se muestra el resultado obtenido de realizar la operación lógica OR y la operación lógica AND entre dos operandos booleanos A y B.

A	B	A OR B	A AND B
false	false	false	false
false	true	true	false
true	false	true	false
true	true	true	true

Java proporciona dos versiones del operador AND: el AND lógico (&) y el AND en cortocircuito (&&). Lo mismo sucede con el operador OR. Existen dos versiones en Java: el OR lógico (|) y el OR en cortocircuito (||).

En la tabla anterior puede observarse que el operador OR da como resultado `true` cuando el operando A vale `true`, con independencia del valor que tome B. Del mismo modo, el operador AND da como resultado `false` cuando el operando A vale `false`, con independiencia del valor que tome B.

Cuando se utilizan las formas && y || de estos operadores, en lugar de & y |, Java no evalúa el operando de la derecha si el resultado de la operación ya queda determinado por el valor que toma el operando de la izquierda.

Los operadores en cortocircuito (&& y ||) resultan útiles en situaciones como la que se muestra en la expresión lógica siguiente:

```
denom != 0 && num/denom >10
```

Si la variable `denom` vale cero, la expresión `num/denom` produce un error en tiempo de ejecución. Para evitar que esto suceda, se emplea el operador AND en cortocircuito. Con ello, en primer lugar se evalúa la expresión lógica `denom != 0`, y sólo en caso de que sea `true` se evalúa la expresión `num/denom >10`.

A.4.4. Operador de asignación

El operador de asignación es un solo signo igual (=). Se emplea en sentencias del tipo siguiente:

```
variable = expresión;
```

donde el tipo de la variable situada al lado izquierdo de la igualdad debe ser compatible con el tipo resultante de evaluar la expresión escrita en el lazo derecho. El efecto de una sentencia como la anterior es evaluar la expresión del lado derecho y asignar el resultado obtenido a la variable del lado izquierdo de la igualdad.

A.5. Control del flujo del programa

Java proporciona dos sentencias de selección: **if** y **switch**. Mediante estas sentencias se controla el flujo del programa en función de condiciones conocidas durante el tiempo de ejecución.

Además, Java proporciona sentencias de iteración: **for**, **while** y **do-while**. Mediante estas sentencias se crean lo que comúnmente se denomina *bucles*. Un *bucle* ejecuta repetidas veces un mismo conjunto de sentencias hasta que se satisface una determinada condición de finalización.

En esta sección se explicará únicamente el funcionamiento de las sentencias **if** y **for**, por ser las más frecuentemente empleadas.

A.5.1. if

La sentencia **if-else** es la sentencia de bifurcación condicional en Java. La forma general de la sentencia es:

```
if ( condición booleana ) {
    sentencias1;
}
else {
    sentencias2;
}
```

donde la cláusula else el opcional. Es decir, es válido escribir:

```
if ( condición booleana ) {
    sentencias1;
}
```

La sentencia **if** funciona de la forma siguiente. Si el resultado de evaluar la expresión booleana es `true`, entonces se ejecutan las sentencias `sentencias1`. Si el resultado es `false`, entonces se ejecutan las sentencias `sentencias2` (si es que existen). En ningún caso se ejecutarán ambos grupos de sentencias.

Por ejemplo, el funcionamiento de la sentencia:

```
if ( a >= b ) {
    a = 0;
} else {
    b = 0;
}
```

es el siguiente. Si `a` es mayor o igual que `b`, entonces se asigna el valor cero a la variable `a`. Por el contrario, si `a` no es mayor o igual que `b`, entonces se asigna el valor cero a la variable `b`. En ningún caso se asignará el valor cero a ambas variables.

A.5.2. for

La forma general de la sentencia **for** es la siguiente:

```
for(expresión inicialización; expresión condición; expresión iteración) {
    sentencias;
}
```

Si solamente se repite una sentencia en el bucle, no es necesario utilizar las llaves. En este caso, se escribiría:

```
for(expresión inicialización; expresión condición; expresión iteración)
    sentencia;
```

El bucle **for** funciona como se describe a continuación.

- En primer lugar, se ejecuta la `expresión de inicialización`. Generalmente, esta expresión declara e inicializa la variable de control del bucle.
- A continuación, se evalúa la `expresión de condición`, que debe ser una expresión booleana. Si la expresión es verdadera, se ejecutan las sentencias del bucle. Si es falsa, el bucle finaliza. Generalmente, en esta expresión se compara la variable de control del bucle con algún valor en concreto.
- En caso de que el bucle no haya finalizado, se ejecuta la `expresión de iteración`. Habitualmente, ésta es la expresión en la que se incrementa o se reduce el valor de la variable de control.
- Cada vez que se recorre el bucle, en primer lugar se vuelve a evaluar la expresión condicional, a continuación se ejecuta las sentencias del cuerpo del bucle y después la expresión de iteración. El proceso se repite hasta que la `expresión de condición` sea falsa.

Por ejemplo, el siguiente bucle **for** calcula la suma de los cinco primeros números enteros y almacena el resultado en la variable `n`. Una vez finalizada la ejecución del bucle, `n` guarda el valor obtenido de evaluar $1+2+3+4+5$.

```
for (int i = 0; i < 6; i++)
   n += i;
```

Obsérvese que en el ejemplo anterior, la variable que controla el bucle (`i`) es declarada e inicializada en la `expresión de inicialización`. Esto se hace frecuentemente en aquellos casos en los cuales la variable que controla el bucle sólo se usa en el bucle. En el ejemplo anterior, se supone que la variable `n` ha sido previamente declarada.

A.6. Comentarios

Es una buena práctica introducir comentarios en el código de cualquier programa, con el fin de facilitar su comprensión. En el código Java que se escribe en los paneles de Ejs también es conveniente introducir comentarios. Pueden escribirse los dos tipos siguientes de comentarios:

```
// comentarios para una sola línea

/*
   comentarios de una o
   más líneas
*/
```

A.7. Métodos

En el panel *Propio* de Ejs es posible definir *métodos* de Java. La forma general de un *método* es la siguiente:

```
tipo nombre_del_método ( lista de parámetros ) {
   cuerpo del método
}
```

donde:

- `tipo` especifica el tipo de dato que devuelve el método. En concreto, `tipo` puede ser cualquier tipo básico de Java (por ejemplo, **int** y **double**), el nombre de una clase de objetos (por ejemplo, **String**), o el tipo **void** en caso de que no devuelva ningún dato.
- El nombre del método puede ser cualquier identificador válido que sea distinto de los que ya están siendo utilizados por otros elementos del programa.
- La lista de parámetros es una lista de pares de tipo e identificador separados por comas. Si el método no tiene parámetros, la lista de parámetros estará vacía.

Los métodos que devuelven un tipo diferente del tipo **void** devuelven el control a la rutina llamante mediante la siguiente forma de la sentencia **return**:

```
return valor;
```

Por ejemplo, el método siguiente calcula el cuadrado de un número entero:

```
int cuadrado ( int i ) {
   return i * i;
}
```

Para obtener el cuadrado del número 5, basta con realizar la llamada al método de la forma siguiente:

```
cuadrado ( 5 );
```

Por ejemplo, puede programarse el método siguiente para calcular el volumen de un cubo:

```
double volumen(double ancho, double alto, double largo) {
   double volumenCubo;
   volumenCubo = ancho * alto * largo;
   return volumenCubo;
}
```

o de forma más concisa:

```
double volumen(double ancho, double alto, double largo) {
   return ancho * alto * largo;
}
```

A.8. String

Una de las clases incluidas en la biblioteca de Java es **String**, la cual está destinada al almacenamiento y manipulación de cadenas de caracteres. De hecho, cualquier cadena de caracteres, incluso aunque sea una constante (por ejemplo, la cadena `"Hola"`), es un objeto del tipo **String**. Esto implica que pueden aplicarse a dicha cadena todos los métodos definidos en la clase **String**.

Los objetos de la clase **String** (es decir, las cadenas de caracteres) pueden concatenarse empleando el operador +. Por ejemplo, el fragmento de código siguiente:

```
String   str1  = "Primera cadena";
String   str2  = "Segunda cadena";
String   str3  = str1 + " y " str2;
```

declara tres objetos de la clase **String**: `str1`, `str2` y `str3`. La cadena de caracteres de la variable `str3` es:

```
"Primera cadena y Segunda cadena"
```

La clase **String** contiene varios métodos. Entre los más frecuentemente usados se encuentran los tres siguientes:

- El método **equals()** comprueba la igualdad entre dos cadenas de caracteres. Por ejemplo, puede compararse las cadenas de caracteres de `str1` y `str2` de la forma siguiente:
  `str1.equals(str2)`
- El método **length()** devuelve la longitud de la cadena de caracteres. Por ejemplo,
  `str1.length()`
  devuelve el número de caracteres de la cadena de `str1`.
- Se puede obtener el carácter que ocupa una posición determinada en la cadena de caracteres llamando al método **charAt()**. Por ejemplo,
  `str3.charAt(3)`
  devuelve el carácter situado en la posición 3 de la cadena de caracteres de la variable `str3`.

A.9. La clase Math de Java

La clase **Math** de Java posee métodos que implementan las funciones matemáticas más comúnmente usadas, tales como la función raíz cuadrada (**sqrt**), valor absoluto (**abs**), funciones trigonométricas (**sin**, **cos**, **tan**), trigonométricas inversas (**asin**, **acos**, **atan**), exponencial (**exp**), potencia (**pow**), logaritmo (**log**), etc.

Por ejemplo, el método siguiente calcula la hipotenusa de un triángulo rectángulo a partir de la longitud de sus dos catetos.

```
double pitagoras(double cateto1, double cateto2) {
   return Math.sqrt( Math.pow(cateto1,2) + Math.pow(cateto2,2) );
}
```

A.10. Ejemplos

En esta sección se muestran dos ejemplos. El primero de ellos ilustra el uso de la sentencia **if** y de variables del tipo **String**. El segundo ilustra el uso de la sentencia **for** y el empleo de la clase **Math** de Java.

A.10.1. Ejemplo 1

La vista del laboratorio virtual que va a desarrollarse en este ejemplo se muestra en la Figura A.4. Consta de una ventana donde se visualiza sólo un objeto, que puede ser o bien de la clase *Flecha* o bien de la clase *Particula*, según seleccione el usuario del laboratorio virtual en una lista desplegable situada en la parte inferior de la vista. En concreto, el objeto de la clase *Flecha* se hace visible si la palabra seleccionada en la lista desplegable es Flecha. Análogamente sucede para el objeto de la clase *Particula*.

En la Figura A.5 se muestra la ventana de variables del laboratorio.

En la Figura A.6 se muestra el método que permite seleccionar que se muestre el objeto de la clase *Flecha* o de la clase *Particula*. Según el valor de la variable `seleccion`, que es de la

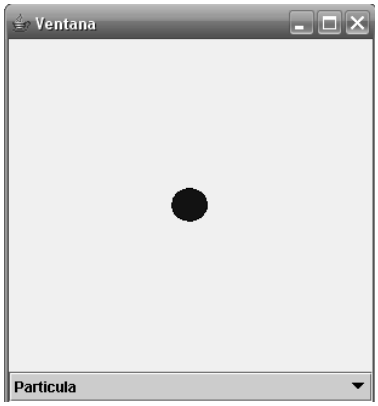


Figura A.4: Vista del laboratorio virtual explicado en el Ejemplo 1.

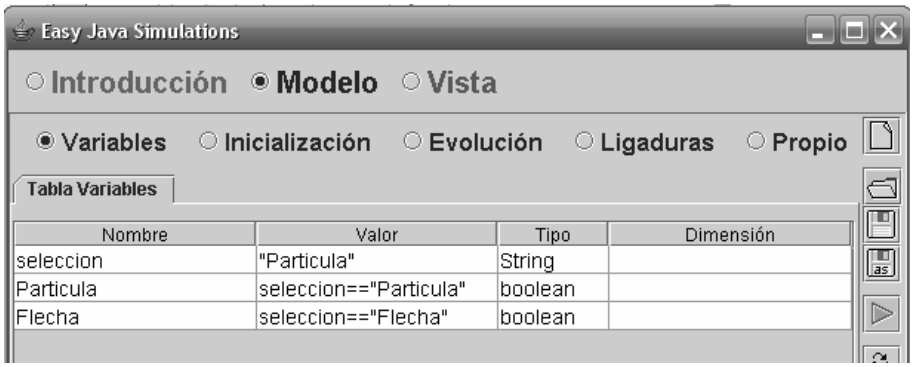


Figura A.5: Ventana para la definición de las variables.

Figura A.6: Definición del método propio.

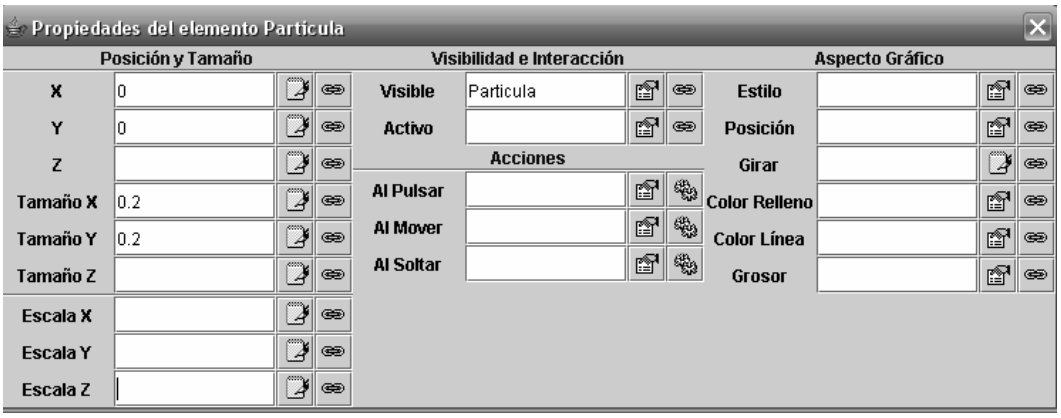


Figura A.7: Ventana de propiedades del objeto `Particula`.

Figura A.8: Ventana de propiedades del objeto `Flecha`.

clase **String**, sea `"Particula"` o `"Flecha"`, se asigna el correspondiente valor a las variables booleanas `Particula` y `Flecha`.

En este método se muestra un ejemplo de uso de la sentencia if, donde la variable de control es la variable denominada `seleccion`. Así, si la variable `seleccion` es igual a `"Particula"`, entonces se pone la variable `Particula` a `true` y la variable `Flecha` a `false`. Las variables `Particula` y `Flecha` están asociadas respectivamente a la propiedad *Visible* de los objetos `Particula` y `Flecha` (ver las Figuras A.7 y A.8).

A.10.2. Ejemplo 2

En la Figura A.9 se muestra la vista de un laboratorio virtual, que consiste en un polígono con n vértices. En la Figura A.10 se muestra el árbol de elementos de la vista.

En la Figura A.11 se muestra la declaración de las variables del laboratorio. Las variables son las siguientes: una variable entera (`n`) y dos vectores de n componentes (`VerticesX` y `VerticesY`).

Las variables `n`, `VerticesX` y `VerticesY` están enlazadas, respectivamente, con las propiedades *Puntos*, *X* e *Y* del objeto *Poligono* (véase la Figura A.12).

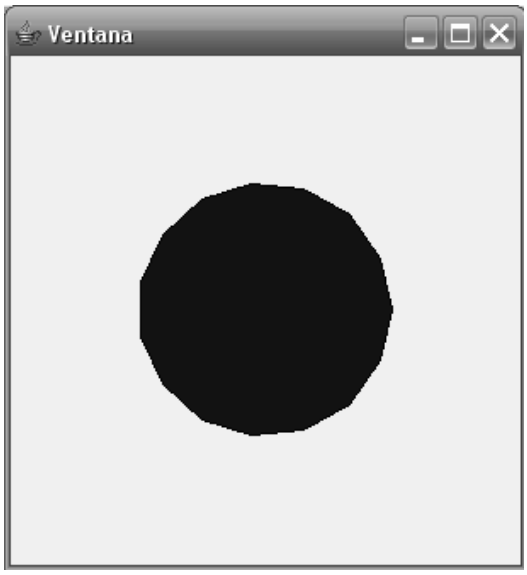


Figura A.9: Vista del laboratorio virtual explicado en el Ejemplo 2.

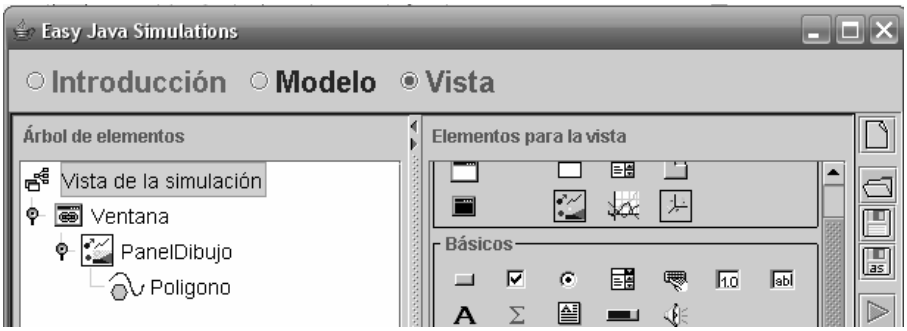


Figura A.10: Árbol de elementos de la vista.

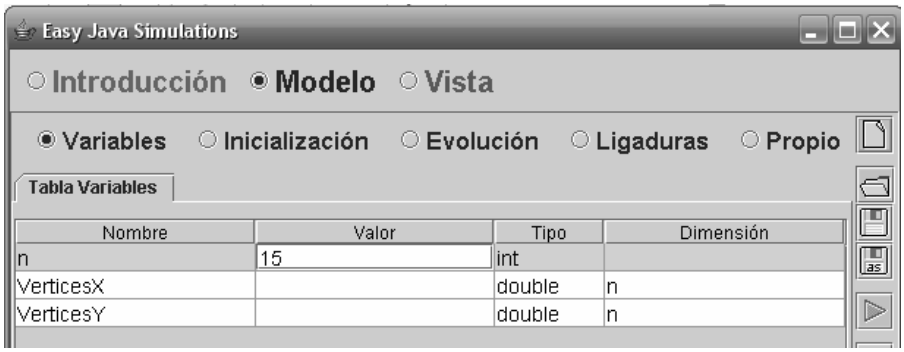


Figura A.11: Declaración de las variables.

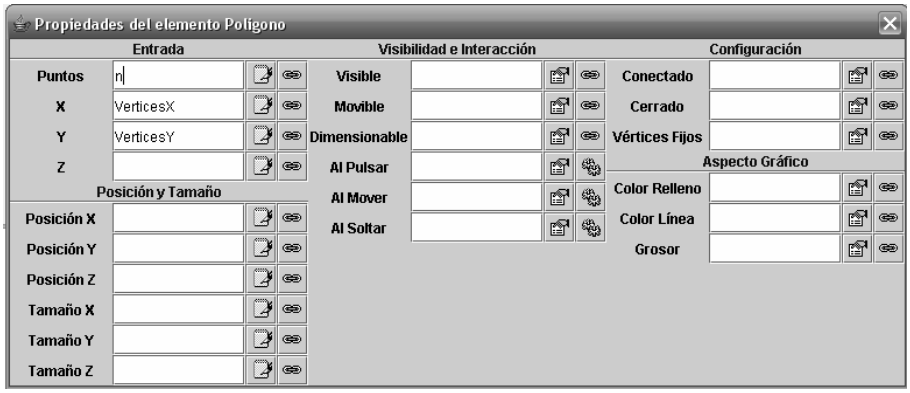


Figura A.12: Propiedades del objeto de la clase *Poligono*.

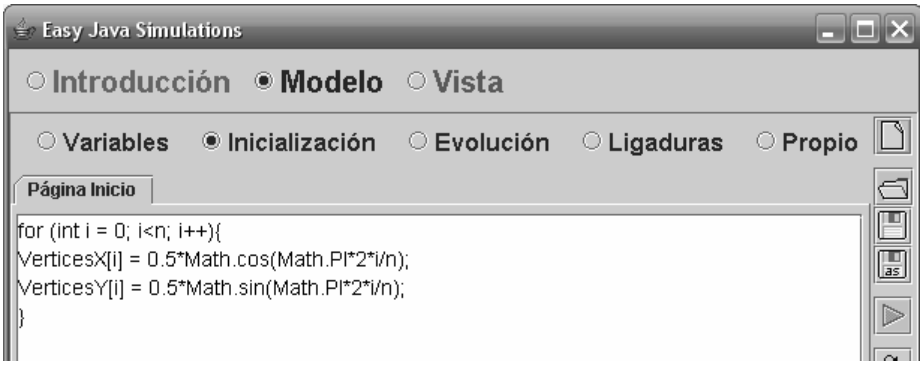


Figura A.13: Ventana de inicialización de las variables.

En la Figura A.13 se muestra la página de inicialización, donde se usa un bucle **for** para inicializar los valores de las componentes de los vectores `VerticesX` y `VerticesY`.

Bibliografía

Astrom, K. J. & Wittenmark, B. (1997), *Computer Controlled Systems. Theory and Design*, Prentice Hall.

Cellier, F. C. (1991), *Continuous System Modeling*, Springer-Verlag.

Christian, W. & Belloni, M. (2004), *Physlet Physics*, Pearson Education.

Cutlip, M. B. & Shacham, M. (1999), *Problem Solving in Chemical Engineering with Numerical Methods*, Prentice Hall.

Dormido, S. & Esquembre, F. (2003), The quadruple-tank process: An interactive tool for control education, *in* 'Procc. 2003 European Control Conference'.

Elmqvist, H. (1978), A Structured Model Language for Large Continuous Systems, PhD thesis, Lund Institute of Technology. Suecia.

Elmqvist, H., Cellier, F. E. & Otter, M. (1993), Object-oriented modeling of hybrid systems, *in* 'ESS'93, European Simulation Symposium', Delft, The Netherlands, pp. xxxi–xli. *citeseer.ist.psu.edu/elmqvist93objectoriented.html

Esquembre, F. (2002*a*), *Easy Java Simulations. Apendices for the manual, for version 3.1*. Disponible en http://www.fem.um.es/Ejs y en el CD del curso (documento Appendices3.2.pdf).

Esquembre, F. (2002*b*), *Easy Java Simulations. The manual for version 3.1*. Disponible en http://www.fem.um.es/Ejs y en el CD del curso (documento EjsManual3.1.pdf).

Esquembre, F. (2004*a*), *Creación de Simulaciones Interactivas en Java. Aplicación a la Enseñanza de la Física*, Prentice Hall.

Esquembre, F. (2004*b*), *Creation of Interactive Simulations in Java. Application to the teaching of physics*. Disponible en http://www.fem.um.es/Ejs y en el CD del curso (documento EjsManual3.3beta040918.pdf).

Esquembre, F. (2004*c*), 'Easy Java Simulations: a software tool to create scientific simulations in Java', *Computer Physics Communications* **156**, 199–204.

Esquembre, F. & Sánchez, J. (2004), *How to use Ejs with Matlab and Simulink, for version 3.3*. Disponible en http://www.fem.um.es/Ejs y en el CD del curso (documento EjsManual3.3beta040918.pdf).

Johansson, K. H. (2000), 'The quadruple-tank process: A multivariable laboratory process with an adjustable zero', *IEEE Transactions on Control Systems Technology* **8**(3), 456–465.

Ljung, L. & Torkel, G. (1994), *Modeling of Dynamic Systems*, Prentice-Hall.

Martin, C., Urquia, A. & Dormido, S. (2004), JARA 2i - A Modelica library for interactive simulation of physical-chemical processes, *in* 'European Simulation and Modelling Conference', pp. 128–132.

Martin, C., Urquia, A., Sanchez, J., Dormido, S., Esquembre, F., Guzman, J. L. & Berenguel, M. (2004), Interactive simulation of object-oriented hybrid models, by combined use of Ejs, Matlab/Simulink and Modelica/Dymola, *in* 'Proc. 18th European Simulation Multiconference', pp. 210–215.

MGA (1995), *ACSL Reference Manual*, MGA Software, Conconrd, MA, USA.

Schildt, H. (2001), *Java 2. Manual de Referencia. Cuarta Edición*, McGraw-Hill.

Urquia, A. (2000), Modelado Orientado a Objetos y Simulación de Sistemas Híbridos en el Ámbito del Control de Procesos Químicos, PhD thesis, Dept. Informática y Automática, UNED, Madrid, Spain.

van den Bosch, P. P. J. & van der Klauw, A. C. (1994), *Modeling, Identification and Simulation of Dynamical Systems*, CRC Press.

Índice alfabético