Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias

Alfonso Urquía Carla Martín-Villalba

Departamento de Informática y Automática, UNED
Juan del Rosal 16, 28040 Madrid, España
{aurquia,carla}@dia.uned.es
http://www.euclides.dia.uned.es/

Monterrey, octubre de 2007

Materiales y contenido del taller

Materiales

CD-ROM

- Ejs 3.4 y manuales
- Java DK (jdk-1_5_0_09-windows-i586-p.exe)
- Libro sobre el desarrollo de laboratorios virtuales con Ejs

  Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
  Alfonso Urquía, Carla Martín-Villalba

- Código de los laboratorios virtuales explicados en el libro y usados en el taller
- Transparencias

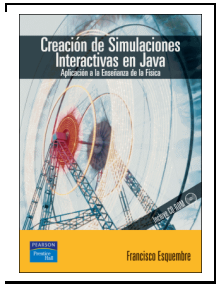
Copia impresa de las transparencias

Índice del texto "*Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias*"

Contenido del taller

Material complementario

Libro sobre Ejs



Creación de Simulaciones Interactivas en Java
Francisco Esquembre

Editorial Pearson Prentice-Hall

Sitios web

- `http://www.um.es/fem/Ejs/Ejs_es/`
- `http://www.euclides.dia.uned.es/simulab-pfp/`

Simulación de Modelos de Tiempo Continuo

Motivación y contenido

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.

Contenido

1. Variables y ecuaciones
2. Parámetros, variables de estado y variables algebraicas
3. Algoritmo de la simulación
4. Causalidad computacional
5. Variables conocidas y desconocidas
6. Asignación de la causalidad computacional

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

Sobre un objeto de masa constante (m) actúan dos fuerzas:

1. La fuerza de la gravedad ($m \cdot g$), donde g se considera constante
2. Una fuerza armónica de amplitud (F_0) y frecuencia (ω) constantes

La fuerza total (F) aplicada sobre el objeto

$$F = m \cdot g + F_0 \cdot \sin(\omega \cdot t)$$

hace que éste adquiera una aceleración (a):

$$m \cdot a = F$$

Además, la posición (x) y la velocidad (v) pueden calcularse de: $\frac{dx}{dt} = v$, $\frac{dv}{dt} = a$

Variables y ecuaciones

Ejemplo (cont.)

El modelo está compuesto de cuatro ecuaciones:

$$\begin{aligned} F &= m \cdot g + F_0 \cdot \sin(\omega \cdot t) \\ m \cdot a &= F \\ \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= a \end{aligned}$$

Estas ecuaciones 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

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.

Parámetros, variables de estado y variables algebraicas

Ejemplo

$$\begin{aligned} F &= m \cdot g + F_0 \cdot \sin(\omega \cdot t) \\ m \cdot a &= F \\ \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= a \end{aligned}$$

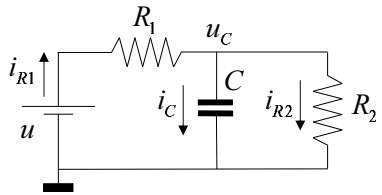
- Cuatro parámetros: g, m, F_0, ω
- Dos variables de estado: x, v
- Dos variables algebraicas: a, F

La variable tiempo (t) no se incluye en la clasificación.

Parámetros, variables de estado y variables algebraicas

Ejemplo

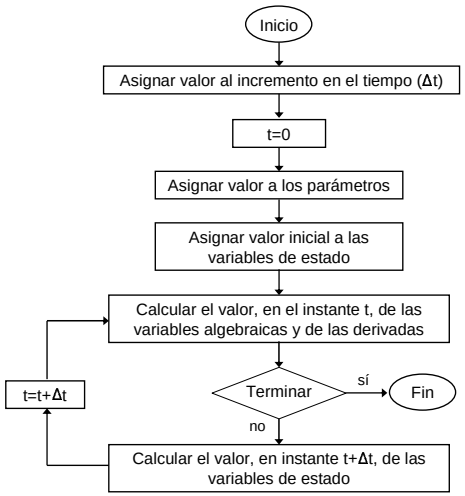
$$
\begin{aligned}
u &= u_0 \cdot \sin(\omega \cdot t) \\
i_{R1} &= i_{R2} + i_C \\
u - u_C &= R_1 \cdot i_{R1} \\
C \cdot \frac{du_C}{dt} &= i_C \\
u_C &= i_{R2} \cdot R_2
\end{aligned}
$$



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

Un algoritmo para la simulación



Un algoritmo para la simulación

Ejemplo

$$
\begin{aligned}
F &= m \cdot g + F_0 \cdot \sin(\omega \cdot t) \\
m \cdot a &= F \\
\frac{dx}{dt} &= v \\
\frac{dv}{dt} &= a
\end{aligned}
$$

Parámetros: g, m, F_0, ω
Variables de estado: x, v
Variables algebraicas: a, F

Se sustituyen las derivadas por variables auxiliares:

$$
\begin{aligned}
\frac{dx}{dt} &\rightarrow derx \\
\frac{dv}{dt} &\rightarrow derv
\end{aligned}
$$

x, v variables de estado

$$
\begin{aligned}
[F] &= m \cdot g + F_0 \cdot \sin(\omega \cdot t) \\
[a] &= \frac{F}{m} \\
[derx] &= v \\
[derv] &= a
\end{aligned}
$$

Un algoritmo para la simulación

Ejemplo (cont.)

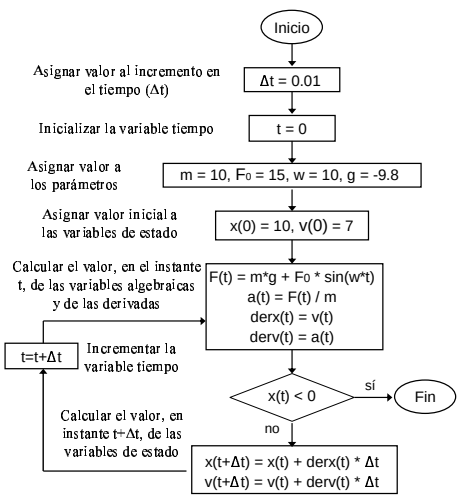
$x(0)$, $v(0)$ conocidos

$$
\begin{aligned}
F(t) &= m \cdot g + F_0 \cdot \sin(\omega \cdot t) \\
a(t) &= \frac{F(t)}{m} \\
derx(t) &= v(t) \\
derv(t) &= a(t)
\end{aligned}
$$

Método Euler 1^{er} orden explícito:

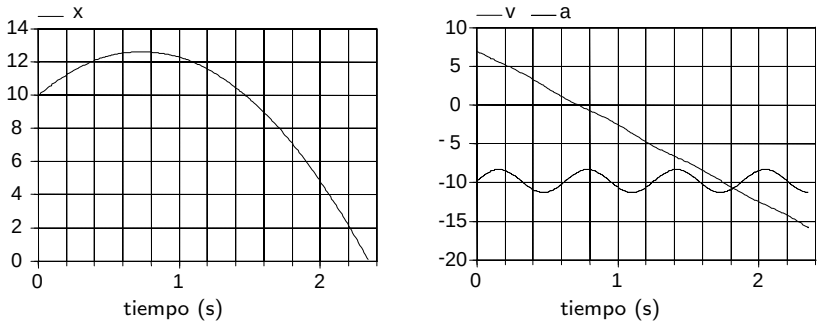
$$
\frac{dx}{dt} = derx(t) \approx \frac{x(t + \Delta t) - x(t)}{\Delta t}
$$

$$
\begin{aligned}
x(t + \Delta t) &= x(t) + derx(t) \cdot \Delta t \\
v(t + \Delta t) &= v(t) + derv(t) \cdot \Delta t
\end{aligned}
$$



Un algoritmo para la simulación

Ejemplo (cont.)



Un algoritmo para la simulación

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

Un procedimiento (existen otros) para estimar el error cometido al escoger un determinado valor de Δt es:

Comparar los resultados obtenidos usando Δt y los obtenidos usando un valor menor, por ejemplo, $\frac{\Delta t}{2}$.

- Si la diferencia entre ambos resultados es aceptable, 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, se 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.

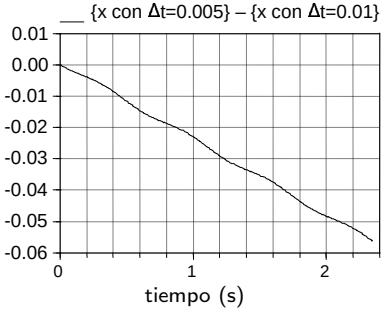
Un algoritmo para la simulación

Ejemplo (cont.)

Cabe plantearse si $\Delta t = 0.01$ es adecuado.

En la figura se muestra la diferencia entre la posición del objeto calculada usando $\Delta t = 0.005$ y $\Delta t = 0.01$.

Dependiendo cuál sea el objetivo del estudio de simulación, el error (y consecuentemente, el valor $\Delta t = 0.01$) será o no aceptable.



Causalidad computacional

Como se ha mostrado en el ejemplo anterior, 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.

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.

Causalidad computacional

Ejemplo

Se desea modelizar una resistencia eléctrica mediante la Ley de Ohm:

> *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.*

La ecuación $u = i \cdot R$ 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:

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

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.

Causalidad computacional

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.

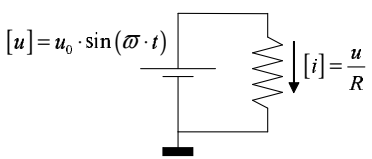
Es decir,

La causalidad computacional es una propiedad global del modelo completo.

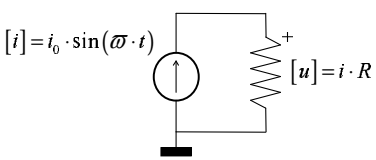
Causalidad computacional

Ejemplo

La casualidad computacional de la ecuación de la resistencia depende del resto de las ecuaciones del modelo.



a)



b)

$$[u] = u_0 \cdot \sin(\varpi \cdot t) \qquad [i] = i_0 \cdot \sin(\varpi \cdot t)$$
$$[i] = \frac{u}{R} \qquad [u] = i \cdot R$$

Causalidad computacional

A continuación, se describe un procedimiento sistemático para asignar la causalidad computacional de un modelo.

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.

Variables conocidas y desconocidas

Variables conocidas:

- La variable tiempo
- Los parámetros del modelo

  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.
- Las entradas globales al modelo

  Son 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

  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)

Variables conocidas y desconocidas

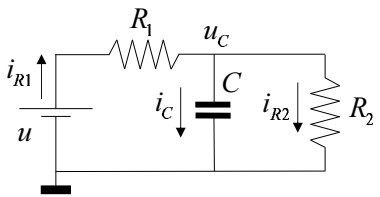
Se sustituye la derivada de cada variable de estado, allí donde aparezca, por una variable auxiliar (por ejemplo, de nombre igual al de la variable de estado, pero anteponiendo el prefijo "der"). Estas variables auxiliares se clasifican como desconocidas y se añade al modelo la condición de que cada 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.

Variables conocidas y desconocidas

Ejemplo



$$\begin{aligned} u &= u_0 \cdot \sin(\omega \cdot t) \\ i_{R1} &= i_{R2} + i_C \\ u - u_C &= R_1 \cdot i_{R1} \\ C \cdot \frac{du_C}{dt} &= i_C \\ u_C &= i_{R2} \cdot R_2 \end{aligned}$$

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

Variables conocidas y desconocidas

Ejemplo (cont.)

Con el fin de realizar la asignación de la causalidad computacional, se sustituye en el modelo $\frac{du_C}{dt}$ por la variable auxiliar $der u_C$

$$\begin{aligned} u &= u_0 \cdot \sin(\omega \cdot t) \\ i_{R1} &= i_{R2} + i_C \\ u - u_C &= R_1 \cdot i_{R1} \\ C \cdot \frac{du_C}{dt} &= i_C \\ u_C &= i_{R2} \cdot R_2 \end{aligned} \quad \Longrightarrow \quad \begin{aligned} u &= u_0 \cdot \sin(\omega \cdot t) \\ i_{R1} &= i_{R2} + i_C \\ u - u_C &= R_1 \cdot i_{R1} \\ C \cdot der u_C &= i_C \\ u_C &= i_{R2} \cdot R_2 \end{aligned}$$

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, $der u_C$, es **desconocida**.

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

Variables conocidas y desconocidas

Ejemplo (cont.)

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

11 variables: u, i_{R1}, i_{R2}, i_C, u_C, $deru_C$, u_0, ω, R_1, R_2, C

- u_C se considera **conocida** (es v.e.)
- $deru_C$ es **desconocida**.

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.

5 ecuaciones - 1 variable de estado = 4 variables algebraicas

El resto de las variables deberán ser parámetros: 5 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.
Una posible elección de los parámetros: u_0, ω, R_1, R_2 y C

Asignación de la causalidad computacional

A continuación, se describe un procedimiento sistemático para asignar la causalidad computacional de un modelo.

Consta de los dos pasos siguientes:

1. Comprobar que el **modelo no es estructuralmente singular**.
2. La asignación en sí de la causalidad computacional (llamada también **partición**).

Asignación de la causalidad computacional

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.
- 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.

Asignación de la causalidad computacional

Partición

Una vez se ha comprobado que el modelo no es *singular*, se realiza la asignación de causalidad computacional siguiendo 3 reglas:

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.

Asignación de la causalidad computacional

Partición

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

Caso 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.

Asignación de la causalidad computacional

Partición

Caso 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 sistemas de ecuaciones.

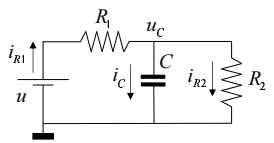
Si en este sistema las incógnitas intervienen linealmente, será posible despejarlas resolviendo el sistema simbólicamente.

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.

Asignación de la causalidad computacional

Ejemplo



u_0, ω, R_1, R_2 y C parámetros
u_C variable de estado
u, i_{R1}, i_{R2}, i_C, $deru_C$ incógnitas

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

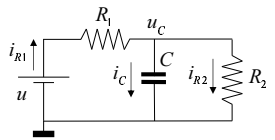
Comprobar no Singularidad Estructural

ecuaciones (5) = # incógnitas (5)

$$\begin{aligned} \boxed{u} &= u_0 \cdot \sin(\omega \cdot t) \\ i_{R1} &= i_{R2} + \boxed{i_C} \\ u - u_C &= R_1 \cdot \boxed{i_{R1}} \\ C \cdot \boxed{deru_C} &= i_C \\ u_C &= \boxed{i_{R2}} \cdot R_2 \end{aligned}$$

Asignación de la causalidad computacional

Ejemplo (cont.)



u_0, ω, R_1, R_2 y C parámetros
u_C variable de estado
u, i_{R1}, i_{R2}, i_C, $deru_C$ incógnitas

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

Partición

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

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

Fundamentos de Ejs

Motivación y contenido

En este tema se explica:

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

Contenido

1. ¿Qué es Easy Java Simulations?
2. Metodología para la creación de laboratorios virtuales
3. Definición del modelo
4. Definición de la vista
5. Ejecución y distribución del laboratorio virtual

¿Qué es Easy Java Simulations?

Easy Java Simulations (abreviado: Ejs) es un entorno de simulación gratuito que ha sido diseñado y desarrollado por el profesor Francisco Esquembre[1].

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.

http://fem.um.es/Ejs/

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

Metodología para la creación de laboratorios virtuales

Paradigma "modelo - vista - control"

La metodología para la creación de laboratorios virtuales de Ejs está basada en una simplificación del

paradigma "modelo - vista - control"

que establece que el laboratorio virtual se compone de 3 partes:

1. Modelo: describe los fenómenos bajo estudio. Está compuesto por un conjunto de variables y por las relaciones entre ellas.
2. Vista: representación gráfica de los aspectos más relevantes del fenómeno simulado.
3. Control: define las acciones que el usuario puede realizar sobre la simulación.

Metodología para la creación de laboratorios virtuales

Paradigma "modelo - vista - control"

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.

Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Fundamentos de Ejs

Metodología para la creación de laboratorios virtuales

Simplificación del paradigma "modelo - vista - control" realizada por Ejs

Ejs integra el **control** en la **vista** y en el **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 la correspondiente variable del* **modelo**.

Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Fundamentos de Ejs

Metodología para la creación de laboratorios virtuales

Páginas HTML de introducción

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.

Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Fundamentos de Ejs

Metodología para la creación de laboratorios virtuales

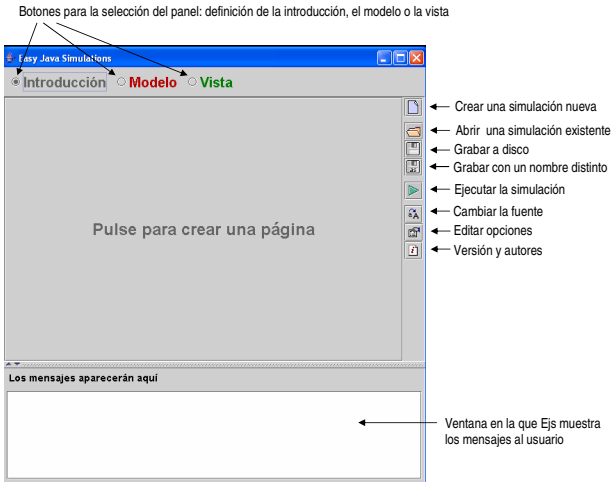
Estructura de un laboratorio en Ejs

Resumiendo lo anterior, la definición de un laboratorio virtual mediante Ejs se estructura en las siguientes 3 partes:

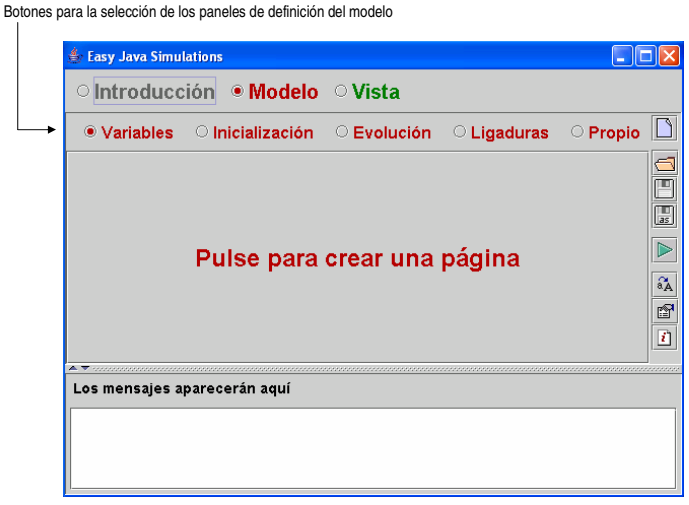
- Introducción: páginas HTML que incluyen los contenidos educativos relacionados con el laboratorio virtual.
- Modelo: modelo dinámico cuya simulación interactiva es la base del laboratorio virtual.
- Vista: interfaz entre el usuario y el modelo.
  Tiene 2 funciones:
  1. Proporciona una representación visual del comportamiento dinámico del modelo.
  2. Proporciona los mecanismos para que el usuario pueda interaccionar con el modelo durante la simulación.

Metodología para la creación de laboratorios virtuales

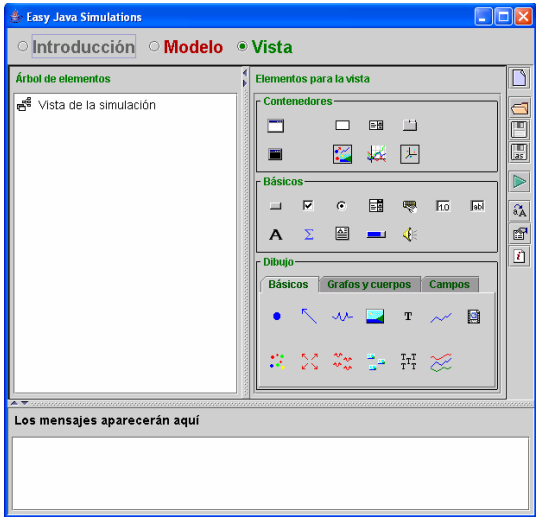
Estructura de un laboratorio en Ejs



Definición del modelo

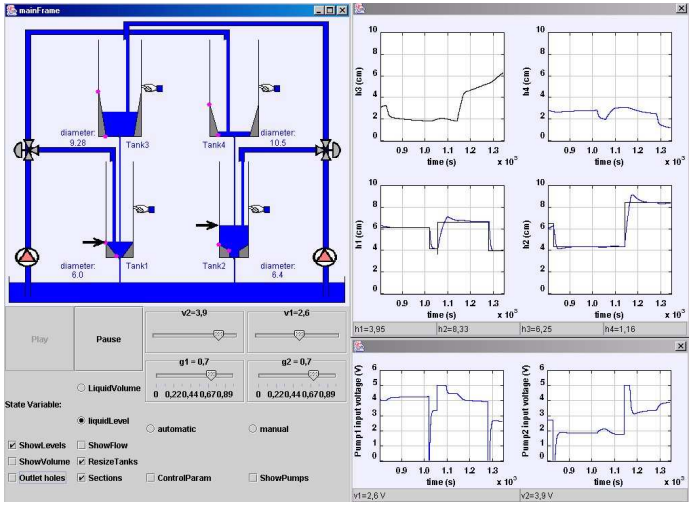


Definición de la vista



Definición de la vista

Ejemplo: laboratorio virtual de los cuatro tanques



Ejecución y distribución del laboratorio virtual

Una vez que el usuario ha definido el modelo, la vista y la introducción del laboratorio virtual, Ejs automáticamente:

- genera 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.

Ejecución y distribución del laboratorio virtual

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.
- Ejecución desde el entorno Ejs.
- Ejecución como una aplicación Java independiente.

Actividad: Ejecución de un laboratorio ya creado

Descripción del modelo en Ejs

Motivación

En este tema se explica el *algoritmo para la simulación* de Ejs.

Las explicaciones se fundamentan en los conceptos expuestos en el Tema "*Simulación de modelos de tiempo continuo*", 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 en Java.

Motivación

En este tema se explica:

- 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

Contenido

Componentes del modelo en Ejs

Para definir un modelo interactivo en Ejs, es preciso:

- Declarar las variables que intervienen en el modelo
- Describir los algoritmos para calcular 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**

Componentes del modelo en Ejs

Ejs proporciona 5 paneles para definir el modelo:

- **Inicialización de las variables**
  - Panel Variables – al declararlas
  - Panel Inicialización – algoritmos de inicialización
- **Algoritmos**
  - Panel Evolución
  - Panel Ligaduras
- **Métodos propios del usuario**

  Pueden ser invocados desde cualquier punto de la descripción del modelo o vista.

  Se emplean típicamente para describir las acciones interactivas del usuario.
  - Panel Propio

Descripción algorítmica del modelo

El modelo se describe en Ejs mediante fragmentos de código Java escritos en los paneles *Inicialización*, *Evolución*, *Ligaduras* y *Propio*.

Esto implica que:

> *el modelo matemático debe ser manipulado por el programador del laboratorio virtual con el fin de formularlo como una secuencia ordenada de asignaciones del tipo*
>
> $$variable = expresión;$$

Ejs realiza la ejecución de esta asignación 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.

El algoritmo de simulación de Ejs

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

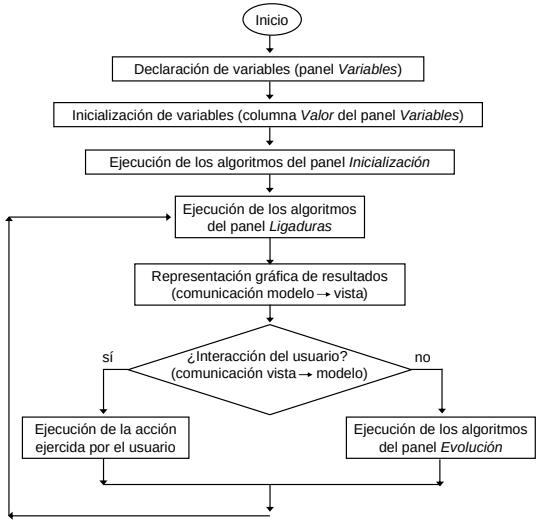
Se entiende por *algoritmo de simulación* de Ejs a

> *el orden en el que Ejs ejecuta los diferentes paneles y las diferentes ventanas dentro de cada panel*

El algoritmo de simulación de Ejs

Orden de ejecución de los paneles

Ejs ejecuta los paneles que componen la descripción del modelo en el orden mostrado en la figura



El algoritmo de simulación de Ejs

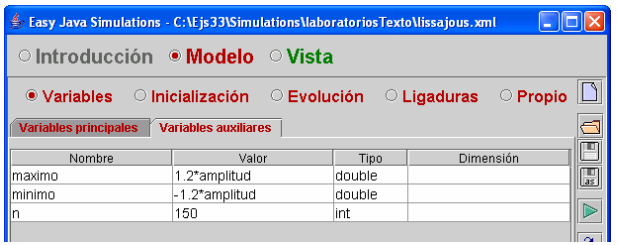
Orden de ejecución de las páginas de un panel

Si un panel consta de varias páginas, éstas se ejecutan siguiendo su orden de izquierda a la derecha.

Ejemplo

Orden de ejecución:

1. *Variables principales*
2. *Variables auxiliares*



El algoritmo de simulación de Ejs

Orden de ejecución de los algoritmos de una página

Ejs ejecuta los algoritmos de una página siguiendo el orden en que están escritos, como si se tratara de un programa en Java.

Ejemplo

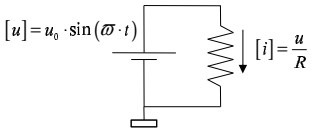
Variables | Inicialización | Evolución | Ligaduras | Propio

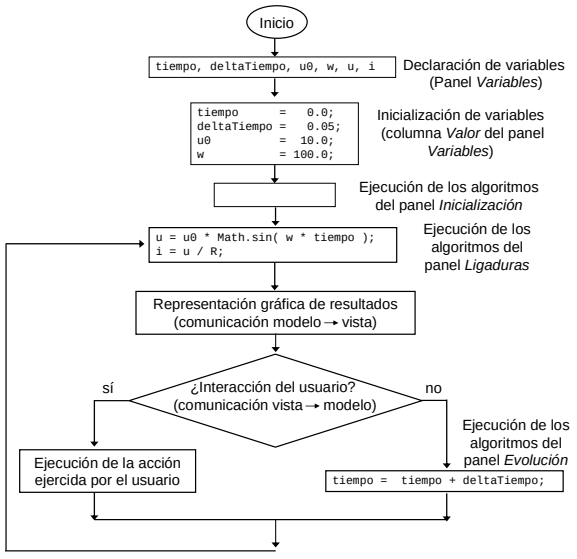
modelo matematico

```
derx1 = v1;
derx2 = v2;
Fmuelle1  = -k1 * ( x1 - x0_muelle1 );
Famortiguador  = -b * v1;
v = v1 - v2;

if (Adelante)  {   Ff = Rv * v - Rm;   }
if (Detras)    {   Ff = Rv * v + Rm;   }
if (Acoplo)    {   Ff = m2 / (m1+m2)*(Fmuelle1 + Famortiguador);   }

if ( Acoplo && Ff > R0  ) { Acoplo = false;  Detras   = true;  Ff = Rv * v + Rm; }
if ( Acoplo && Ff < -R0 ) { Acoplo = false;  Adelante = true;  Ff = Rv * v - Rm; }

Fmasa1  = Fmuelle1 + Famortiguador-Ff;
Fmasa2  = Ff;
a1   = Fmasa1 / m1;
a2   = Fmasa2 / m2;
derv1 = a1;
derv2 = a2;
```


El algoritmo de simulación de Ejs

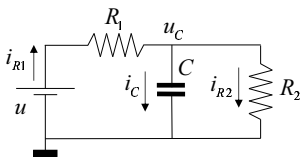
Ejemplo





El algoritmo de simulación de Ejs

Ejemplo



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

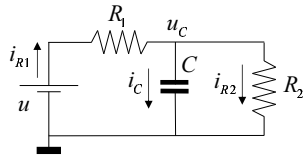
La derivadas de las variables de estado deben expresarse en función únicamente de variables de estado, parámetros y la variable tiempo.

Puesto que se ha asignado la causalidad computacional del modelo, las manipulaciones necesarias para ello se pueden realizar de manera sencilla.

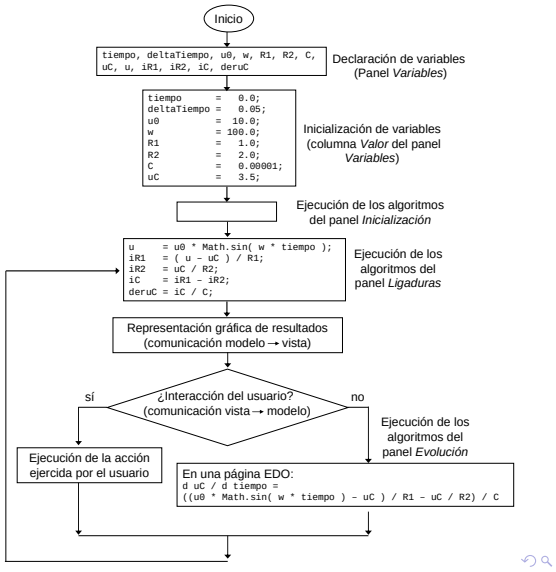
$$
\begin{aligned}
\frac{d[u_C]}{dt} &= deru_C = \frac{i_C}{C} = \frac{i_{R1} - i_{R2}}{C} \\
&= \frac{\frac{u - u_C}{R_1} - \frac{u_C}{R_2}}{C} = \frac{\frac{u_0 \cdot \sin(\omega \cdot t) - u_C}{R_1} - \frac{u_C}{R_2}}{C}
\end{aligned}
$$

El algoritmo de simulación de Ejs

Ejemplo (cont.)



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



El algoritmo de simulación de Ejs

- Panel *Ligaduras*: asignaciones para calcular las variables algebraicas y las derivadas de las variables de estado.

  La *partición* indica cómo ordenar las ecuaciones y qué variable despejar de cada ecuación.

- Panel *Evolución*: cálculo de las variables de estado, mediante integración de sus derivadas.

  Página EDO - uso de los métodos de integración de Ejs

  - La derivada de cada variable de estado debe expresarse en función únicamente de variables de estado, parámetros y la variable tiempo
  - Ejs gestiona automáticamente el incremento de la variable tiempo a lo largo de la simulación

Declaración e inicialización de las variables

Es preciso declarar y asignar valor inicial a:

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

Es preciso declarar, pero no inicializar:

- 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*.

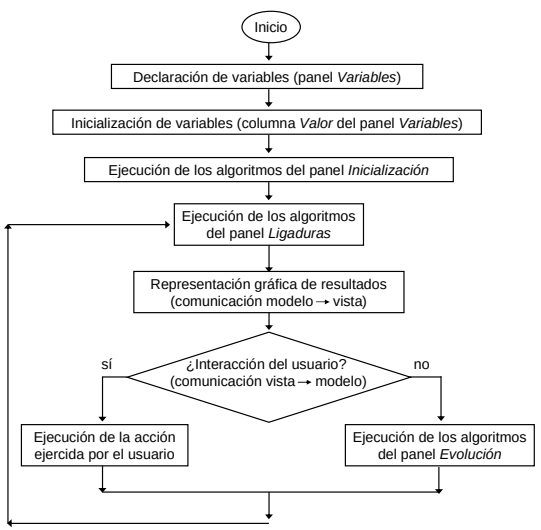
Declaración e inicialización de las variables

Declaración:

- *panel Variables*

Inicialización:

- Tiempo, constantes, parámetros, v.e.: *paneles Variables y Inicialización*
- Variables algebraicas y derivadas: *panel Ligaduras*



Declaración e inicialización de las variables

Panel Variables

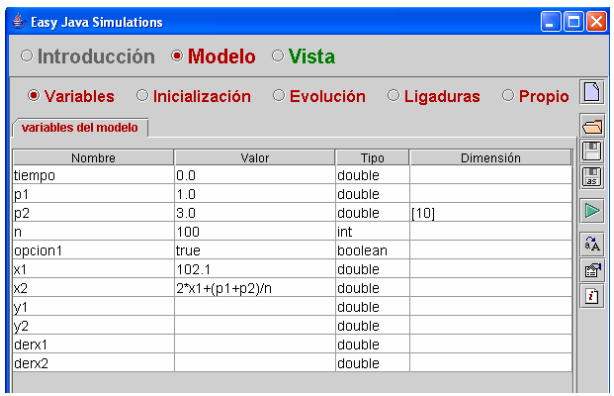
Tipos:

- *boolean*: true o false
- *int*: entero
- *double*: real
- *String*: cadena de caracteres

Dimensión:

- q dimensión [10]: $q[0], \cdots, q[9]$
- q dimensión [10] [20]: matriz de 10 filas $(0, \cdots, 9)$, 20 columnas $(0, \cdots, 19)$

Ejemplo



Declaración e inicialización de las variables

Panel Inicialización

Ejemplo

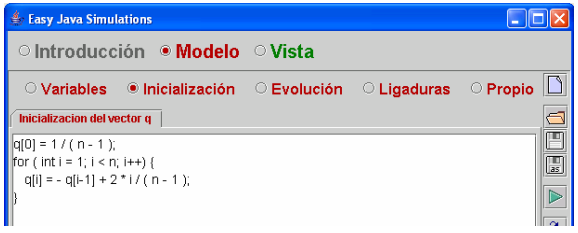
La variable q es un vector de n componentes:

$$q[0], \ldots, q[n-1]$$

que se inicializa:

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

$$q[i] = -q[i-1] + \frac{2 \cdot i}{n-1} \quad \text{para } i : 1, \ldots, n-1$$

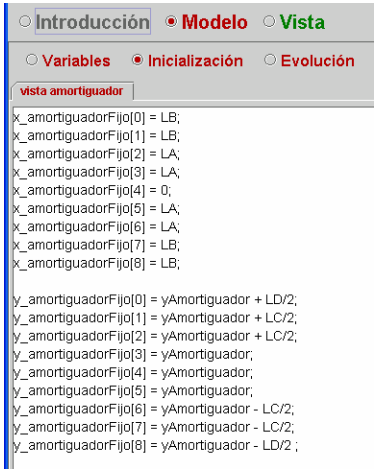


Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Descripción del modelo en Ejs

Declaración e inicialización de las variables

Panel Inicialización

Ejemplo



Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Descripción del modelo en Ejs

Descripción de la evolución

Pulse para crear una página
Permite escribir algoritmos (reinicializar variables de estado)

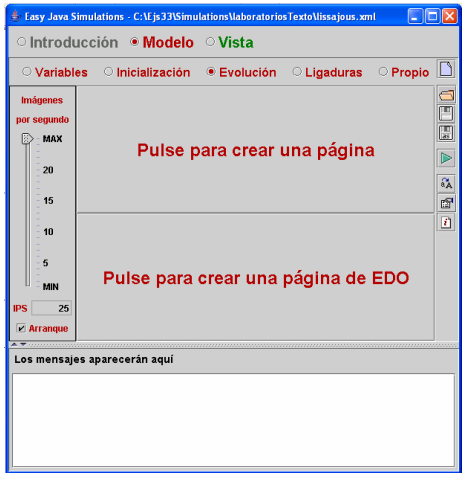
Pulse para crear una página EDO
Asistente para la definición de ecuaciones diferenciales ordinarias

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

Imágenes por segundo
Regula velocidad ejecución simulación

Arranque

- Activado: comienza la simulación al ejecutar el laboratorio virtual
- Desactivado: es preciso definir un botón de arranque, cuya acción sea una llamada al método _play()



Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
Descripción del modelo en Ejs

Descripción de la evolución

Página EDO

El código que genera Ejs para una página EDO realiza las dos tareas siguientes:

1. Calcula el valor de las variables de estado en $t + \Delta t$, aplicando el método de integración seleccionado.
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

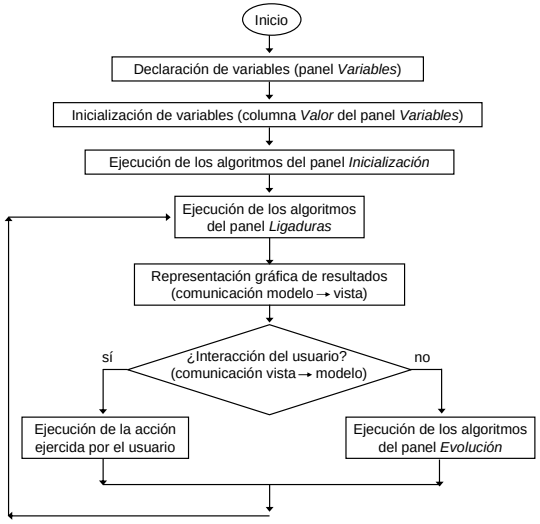
en un laboratorio virtual no puede haber más de una página de EDO

Si el laboratorio no tiene ninguna página EDO, debe programarse explícitamente el incremento de la variable tiempo

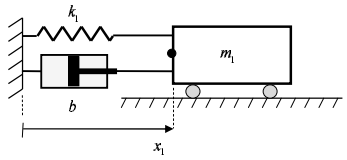
Descripción de las ligaduras

Panel Ligaduras

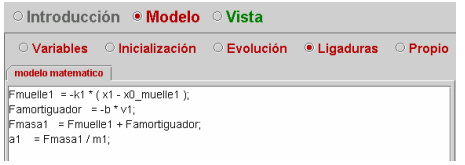
- Algoritmos para el cálculo de las variables algebraicas
- Comportamiento cuando el usuario realiza cualquier cambio interactivo.
  Esta es una diferencia conceptual entre evolución y ligadura

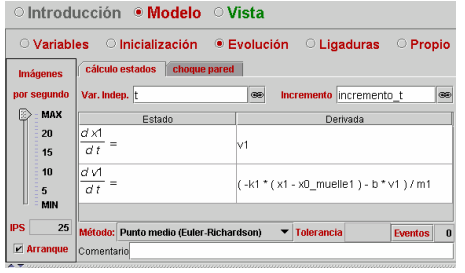


Ejemplo: paneles evolución y ligaduras

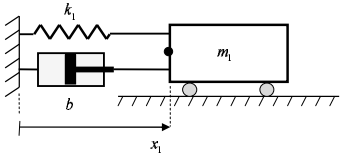


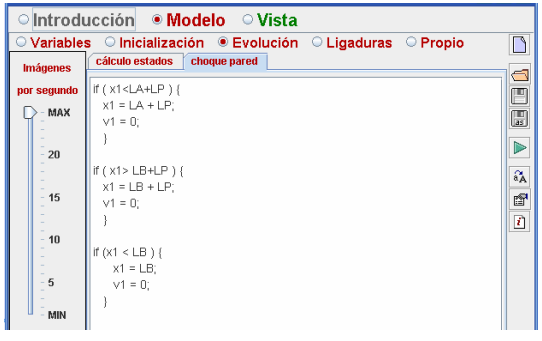
$$
\begin{aligned}
[F_{muelle1}] &= -k_1 \cdot (x_1 - x_{0,muelle1}) \\
[F_{amortiguador}] &= -b \cdot v_1 \\
[F_{masa1}] &= F_{muelle1} + F_{amortiguador} \\
[a_1] &= \frac{F_{masa1}}{m_1} \\
\frac{d\,[x_1]}{dt} &= v_1 \\
\frac{d\,[v_1]}{dt} &= \frac{-k_1 \cdot (x_1 - x_{0,muelle1}) - b \cdot v_1}{m_1}
\end{aligned}
$$





Ejemplo: paneles evolución y ligaduras





Métodos propios del usuario

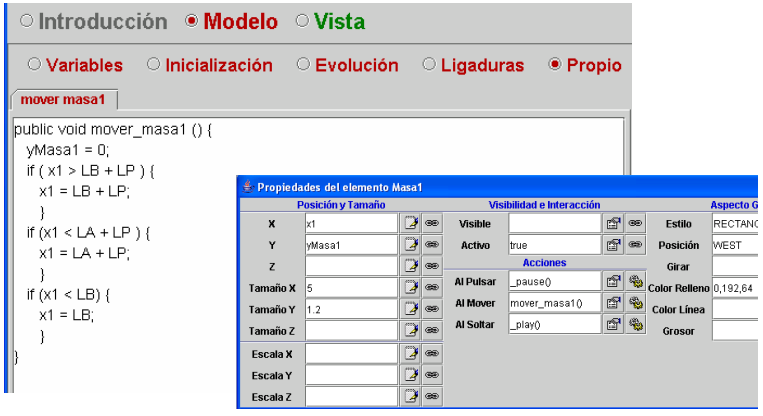
El usuario puede definir, en el panel *Propio*, todos aquellos *métodos* en lenguaje Java que precise para la definición del modelo o de la vista.

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).

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.

Métodos propios del usuario

Ejemplo



Conceptos básicos para la descripción de la vista

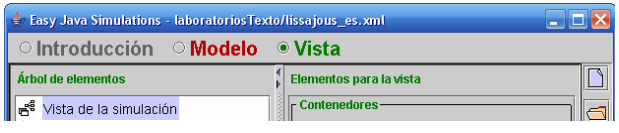
Características fundamentales de la **vista**

La **vista** del laboratorio virtual es la interfaz entre el usuario y el modelo.

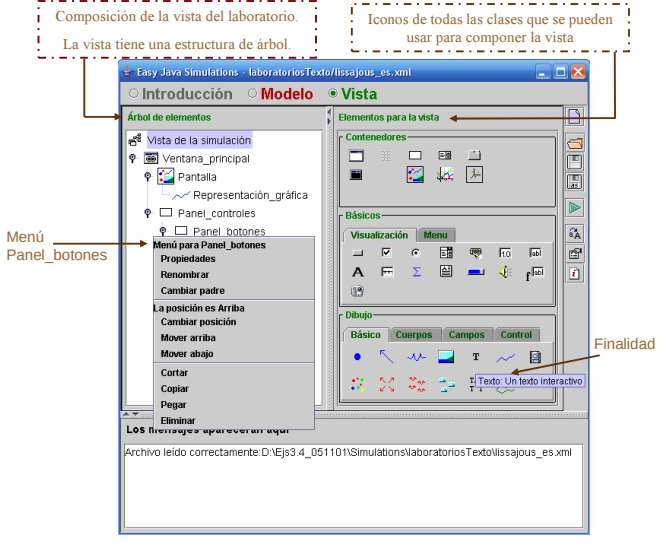
Mediante la manipulación de la **vista**, el usuario puede:

- Controlar la ejecución de la simulación
- Cambiar los parámetros, las variables de entrada o las variables de estado del modelo

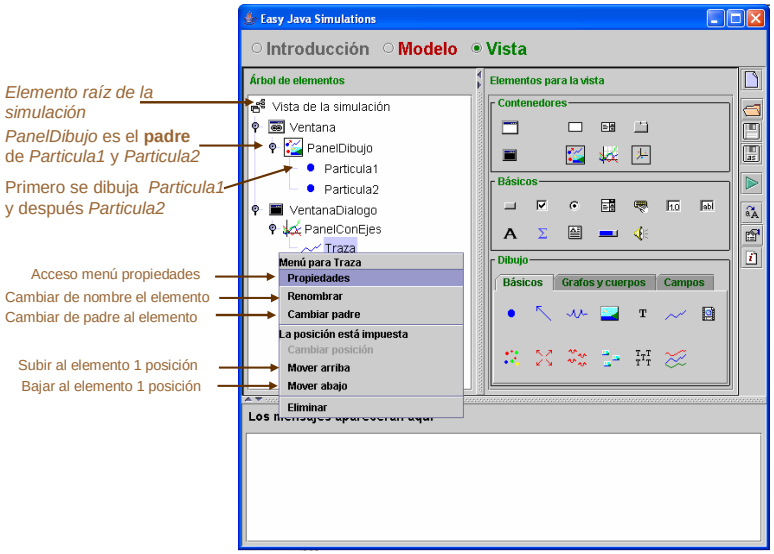
Para configurar la **vista** en Ejs debemos seleccionar el panel *Vista*. Este panel se subdivide en dos paneles: el panel *Árbol de elementos* y el panel *Elementos para la vista*.



Descripción del panel **vista**

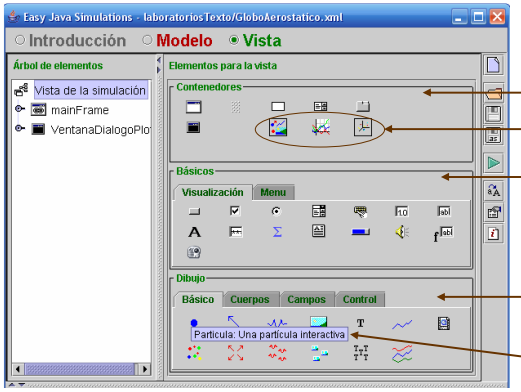


Árbol de elementos



Elementos de la vista

Referencia: http://www.um.es/fem/Ejs/LibroEjs/CD/Referencia/Referencia.html



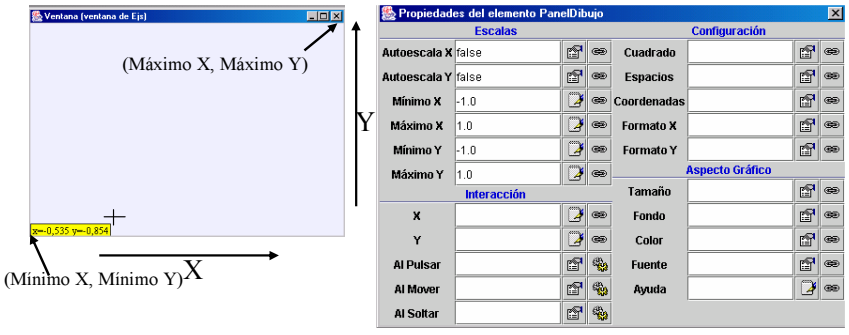
Clases de elementos de tipo Contenedor

Estas clases pueden contener otros elementos gráficos. Podemos diferenciar estos elementos según tengan o no ejes coordenados:

- Elementos que no tienen ejes coordenados: Ventana, VentanaDialogo, Panel. En todos estos elementos hay que especificar qué política de distribución se sigue para alojar a los elementos que alberga:
  - *Márgenes*: los hijos se sitúan en cinco áreas (centro, 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.
  - *Flujo izquierda/centro/derecha*: la distribución flujo hace que los hijos se alineen horizontalmente.
- Elementos con ejes coordenados: PanelDibujo, PanelDibujo3D, PanelConEjes. Para colocar un elemento dentro de un objeto de estas clases hay que especificar sus coordenadas.

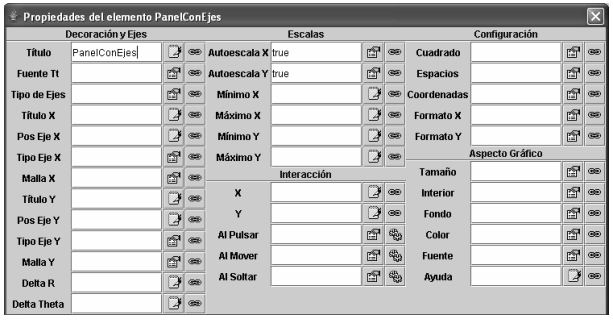
Clase PanelDibujo

Este elemento sólo contiene elementos de tipo Dibujo.

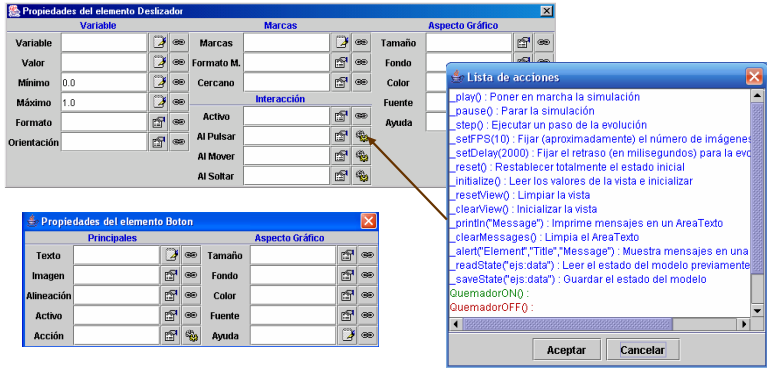


Clases PanelDibujo3D y PanelConEjes

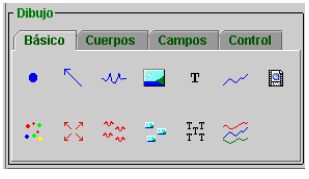
PanelConEjes: variante de la clase PanelDibujo que incluye, por defecto, un sistema de ejes coordenados.
PanelDibujo3D: contenedor especial en tres dimensiones para elementos gráficos de dibujo.

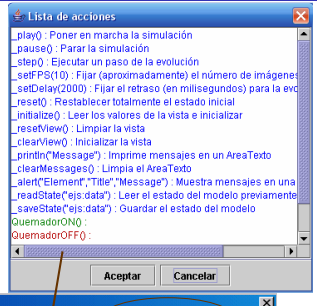


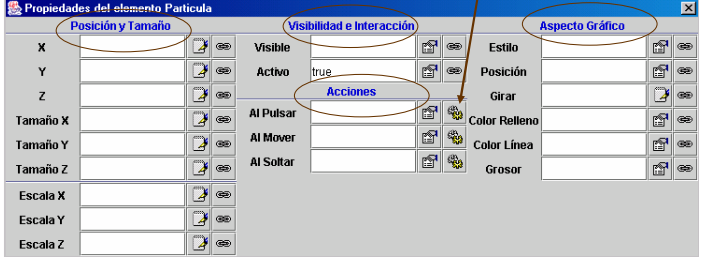
Clases de elementos de tipo Básicos



Clases de elementos de tipo Dibujo







Programación de un osciloscopio virtual con Ejs

Descripción del modelo

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

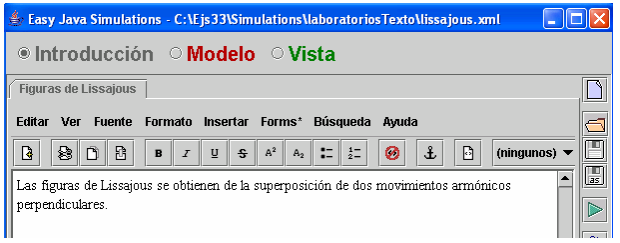
$$\begin{aligned} x &= A \cdot \cos(\omega_1 \cdot t) && \text{(movimiento horizontal)} \\ y &= A \cdot \cos(\omega_2 \cdot t + \delta) && \text{(movimiento vertical)} \end{aligned}$$

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.

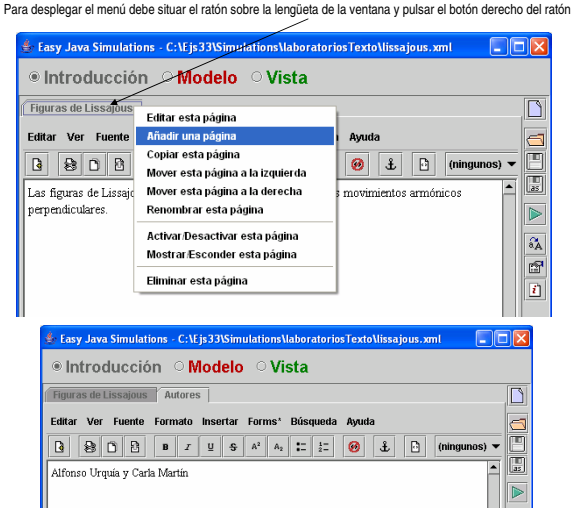
Descripción de la introducción

Pasos necesarios para describir la introducción:

1. Arrancar el entorno de simulación Ejs.
2. Crear las páginas necesarias en el panel introducción. Para ello haga clic con el ratón sobre la frase "Pulse para crear una página".
3. Escriba en la zona de texto de la página lo que desee.
4. Para ver el código HTML generado, seleccione *Ver / Ver Código Fuente*.



Descripción de la introducción



El algoritmo de la simulación

Paso previos a programar el modelo de un laboratorio virtual empleando Ejs:

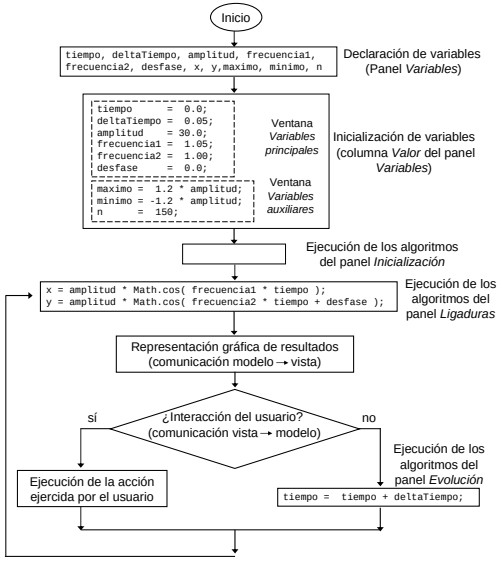
1. Clasificar las variables del modelo en conocidas y desconocidas:
   - Variables conocidas: parámetros (A, ω_1, ω_2, δ) y la variable tiempo (t).
   - Variables desconocidas: las dos variables algebraicas x e y.
2. Aplicar el algoritmo de asignación de la causalidad computacional:

$$\begin{aligned} [x] &= A \cdot \cos(\omega_1 \cdot t) \\ [y] &= A \cdot \cos(\omega_2 \cdot t + \delta) \end{aligned}$$

El algoritmo de la simulación

Variables

- del modelo
- *de los métodos numéricos*: parámetro deltaTiempo
- *de la definición de las acciones y de la vista*: parámetros `maximo`, `minimo` y `n`



Declaración e inicialización de las variables

En el caso del osciloscopio virtual, van a definirse dos páginas:

- Página "*Variables principales*"
  Se declaran las variables que intervienen en el modelo matemático y las variables necesarias para su resolución numérica.

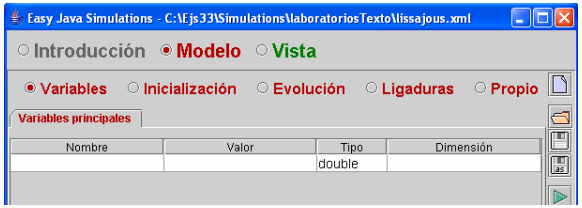
Modelo matemático	Modelo en Ejs
t	tiempo
x	x
y	y
A	amplitud
ω_1	frecuencia1
ω_2	frecuencia2
δ	desfase
deltaTiempo	deltaTiempo

- Página "*Variables auxiliares*"
  Se declaran las variables empleadas para la definición de la vista (*maximo*, *minimo* y *n*).

Declaración e inicialización de las variables

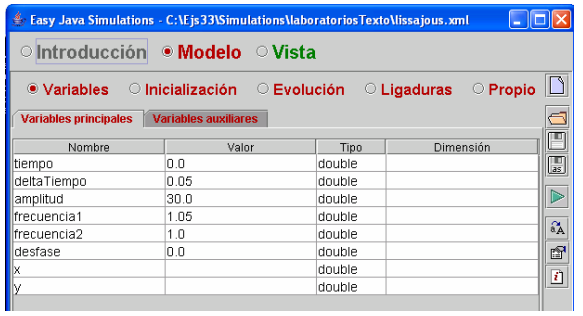
Creación de la página "*Variables principales*"

- Pulse con el ratón sobre la frase "Pulse para crear una página", que aparece en el panel *Variables*
- En la ventana que se abre, asigne a la página el nombre "*Variables principales*"



Declaración e inicialización de las variables

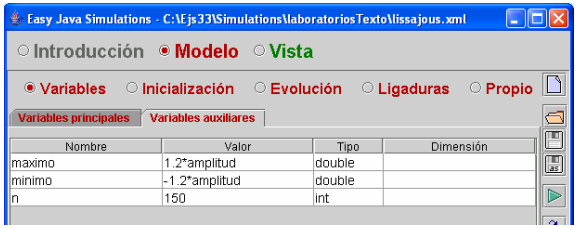
Página "*Variables principales*"



Declaración e inicialización de las variables

Creación de la página "*Variables auxiliares*"

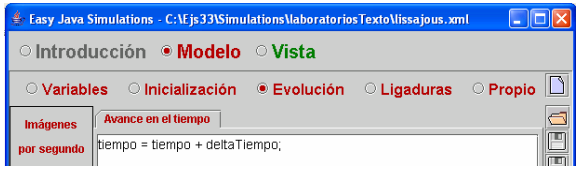
1. Pinche con el botón derecho del ratón sobre la lengüeta de la página de variables ya existente
2. En el menú desplegable que aparece, haga clic con el ratón sobre "*Añadir una página*"
3. Se abre una ventana, en la cual debe especificar el nombre de la nueva página: *Variables auxiliares*



Programación del modelo

Página de *Evolución*

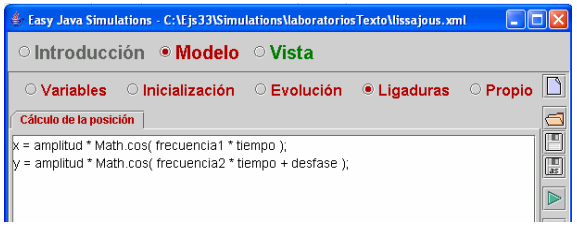
1. Para definir la ecuación de evolución, haga clic sobre el subpanel con el letrero "*Pulse para crear una página*"
2. Se abre un ventana, en la cual debe especificarse el nombre que se asigna a la nueva página que se va a crear.
   Por ejemplo, dele a la página el nombre *Avance en el tiempo*
3. Escriba la ecuación de evolución.
   Observe que la ecuación finaliza con punto y coma (;)



Programación del modelo

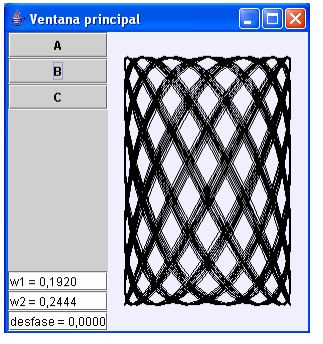
Página de *Ligaduras*

1. Haga clic sobre el botón *Ligaduras*.
2. La interfaz de Ejs muestra un panel que contiene la frase "*Pulse para crear una página*"
   Haga clic con el ratón sobre esta frase.
3. Asigne a esta página el nombre *Cálculo de la posición*.



Programación de la vista

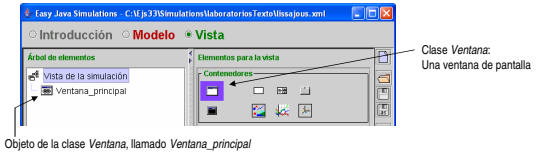
Vista que
se desea obtener



Programación de la vista

Definición de la ventana de la vista del laboratorio virtual

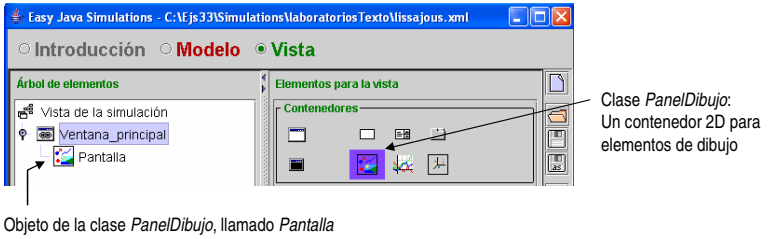
1. Haga clic sobre la clase *Ventana*
2. Haga clic sobre la frase *Vista de la simulación*
   Con ello está indicando que el objeto que está a punto de crear (de la clase *Ventana*) debe ubicarse dentro del objeto raíz de la vista
3. Se abre una ventana, en la cual debe escribir el nombre del nuevo objeto. Dele el nombre *Ventana_principal*



Programación de la vista

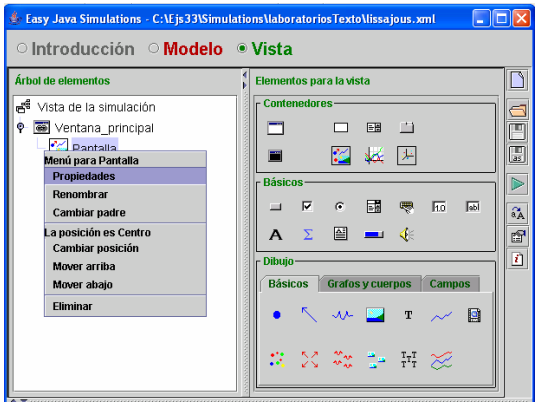
Añada un objeto de la clase *PanelDibujo*

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*.
   Asigne valor a los siguientes parámetros del objeto que está creando:
   - *Nombre del objeto*. Llámelo *Pantalla*
   - *Posición, dentro del contenedor Ventana_principal, del objeto Pantalla*.
     Acepte la posición por defecto: centro



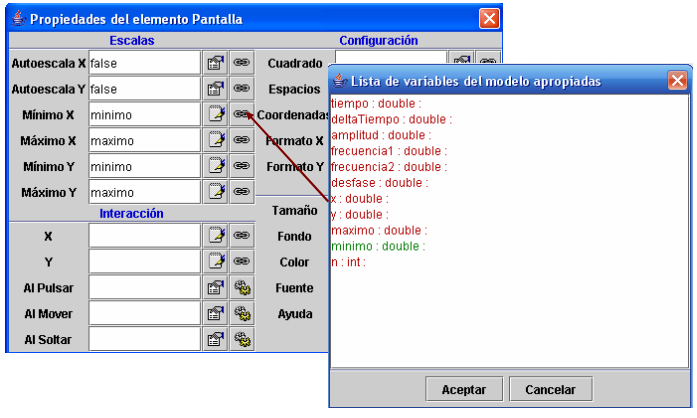
Programación de la vista

Despliegue el menú de configuración de *Pantalla* y seleccione *Propiedades*



Programación de la vista

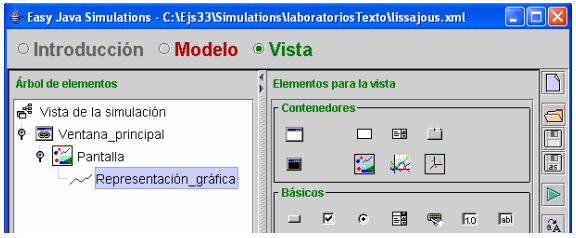
Menú de propiedades del objeto *Pantalla*



Programación de la vista

Ubique dentro del contenedor *Pantalla* un objeto de dibujo, que defina qué es lo que debe dibujarse

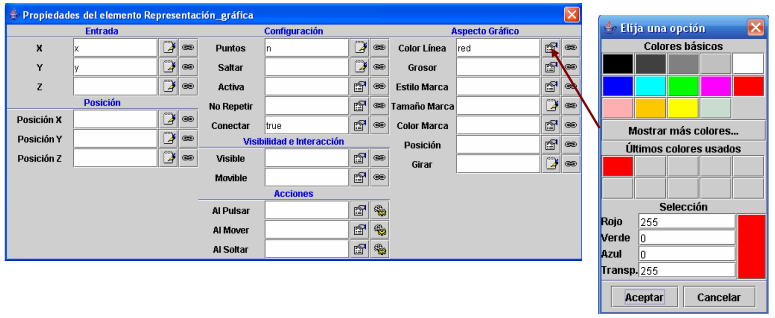
1. Seleccione la clase *Traza*
   Traza: una secuencia de puntos
2. Haga clic con el ratón sobre el objeto *Pantalla*
   Con ello se crea un objeto del tipo *Traza* y se ubica dentro del contenedor *Pantalla*
   Dele al objeto del tipo Traza que está creando el nombre *Representación_gráfica*



Programación de la vista

Asigne valor a las propiedades del objeto *Representación_gráfica*

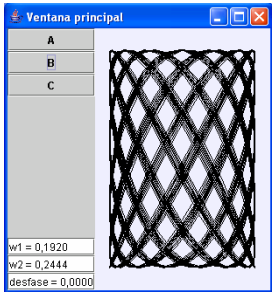
1. *Indique qué variables deben representarse*
   Enlace la propiedad *X* con la variable x, y la propiedad *Y* con la variable y
2. *Indique cuántos puntos deben dibujarse*
   Para ello, enlace la propiedad *Puntos* con la variable n
3. *Escoja el color de la línea*



Programación de las capacidades interactivas

Se va a dotar al osciloscopio virtual de dos capacidades interactivas

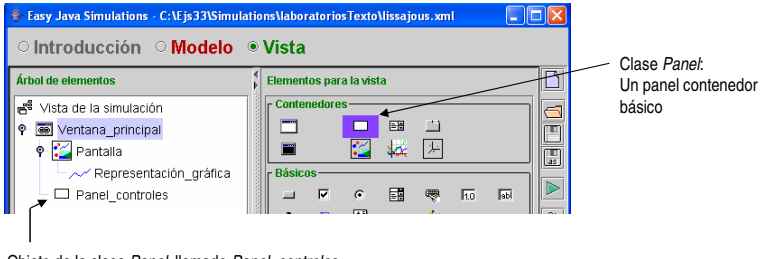
1. Van a añadirse algunos botones que permitan al usuario seleccionar determinadas frecuencias y desfases, de entre un conjunto predeterminado de ellas, las cuales dan lugar a figuras de Lissajous vistosas. Cada botón corresponderá con una determinada selección de las frecuencias y el desfase.
2. Se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



Programación de las capacidades interactivas

Cree un objeto de la clase *Panel* y ubíquelo dentro de *Ventana_principal*

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)



Programación de las capacidades interactivas

Para añadir los botones que realicen las acciones, es preciso:

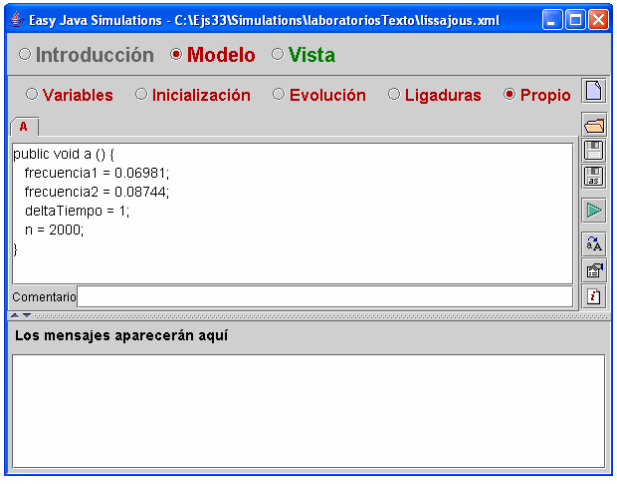
1. *Programar la acción a realizar cuando se pulse cada botón*
   Deben programarse métodos en lenguaje Java (uno por cada botón) que realicen las acciones deseadas.
   La programación de estos métodos forma parte de la definición del modelo.
2. *Incluir los tres botones en la vista y asociarle a cada 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 pretende programar tres botones:

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

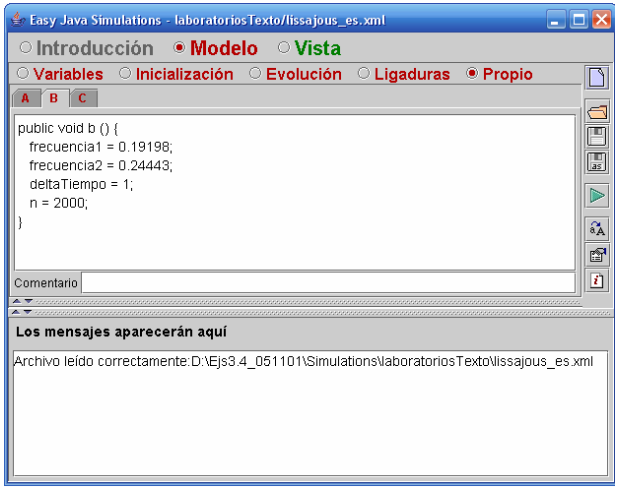
Programación de las capacidades interactivas

Método a, que se asociará con el Botón A



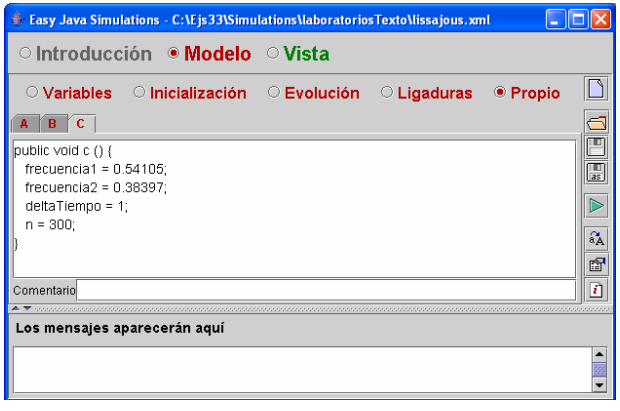
Programación de las capacidades interactivas

Método b, que se asociará con el Botón B



Programación de las capacidades interactivas

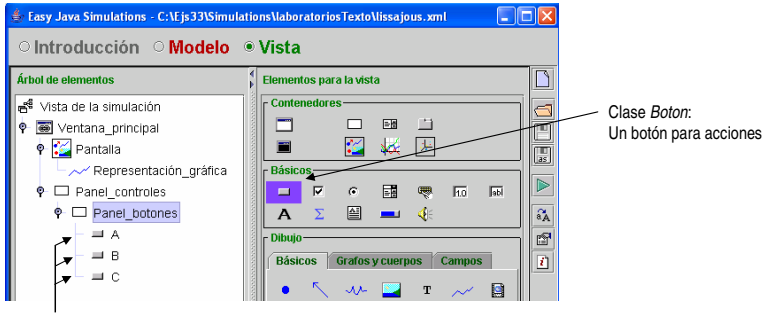
Método c, que se asociará con el Botón C



Programación de las capacidades interactivas

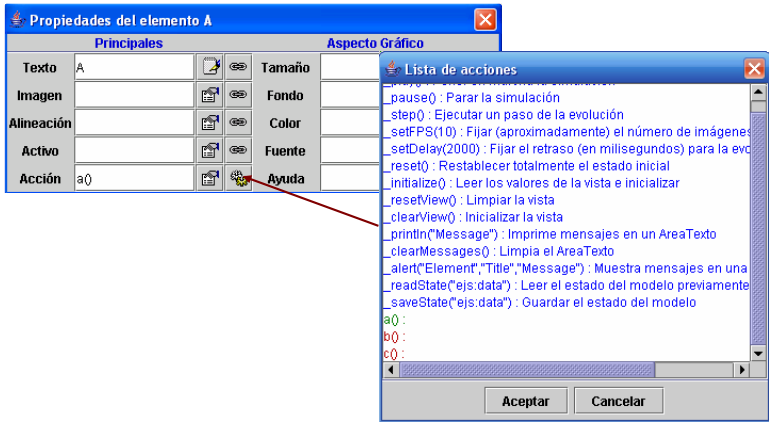
Una vez definidos los métodos, se definen los **botones** en la vista

1. Definir un objeto de la clase *Panel*, dentro del cual se ubicarán los botones. Nombre del nuevo objeto: *Panel_botones*. Posición: *Arriba*
2. Definir 3 objetos del tipo *Botón* y ubicarlos dentro de *Panel_botones* Nombres de los nuevos objetos: *A*, *B* y *C*. Posiciones: *Arriba*, *Centro* y *Abajo*



Programación de las capacidades interactivas

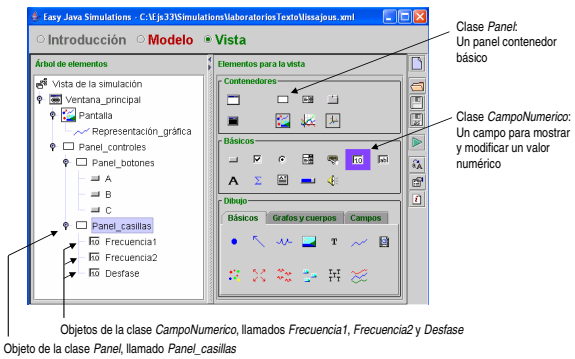
Asocie las acciones con los botones



Programación de las capacidades interactivas

Incluya las casillas numéricas

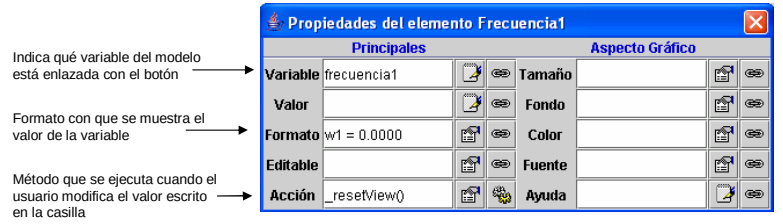
1. Cree un objeto de la clase *Panel* y ubíquelo dentro de *Panel_controles* Nombre del nuevo objeto: *Panel_casillas*. Posición: *Abajo*
2. Cree 3 objetos de la clase *CampoNumerico* y ubíquelos dentro de *Panel_casillas* Nombre de los nuevos objetos: *Frecuencia1*, *Frecuencia2* y *Desfase* Posición: *Arriba*, *Centro* y *Abajo*



Programación de las capacidades interactivas

Configure las propiedades del elemento *Frecuencia1*

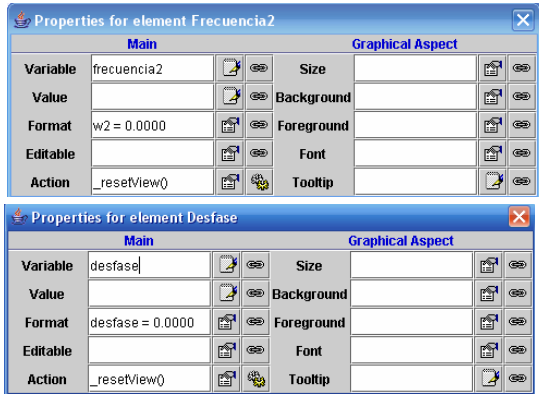
1. Sitúese sobre *Frecuencia1* y pulse el botón derecho del ratón
2. Se despliega un menú (*Menú para Frecuencia1*). Seleccione *Propiedades*
3. Se abre una ventana (*Propiedades del elemento Frecuencia1*)



El método _resetView() limpia la vista

Programación de las capacidades interactivas

Configure las propiedades de los elementos *Frecuencia2* y *Desfase*



Un laboratorio virtual para ilustrar el concepto de ciclo límite

Descripción del modelo

Un ciclo límite en el plano XY está descrito por las ecuaciones siguientes:

$$\begin{aligned}
\frac{dx}{dt} &= y + \frac{K \cdot x \cdot \left(1 - x^2 - y^2\right)}{\sqrt{x^2 + y^2}} \\
\frac{dy}{dt} &= -x + \frac{K \cdot y \cdot \left(1 - x^2 - y^2\right)}{\sqrt{x^2 + y^2}} \\
x(0) &= x_0 \qquad \text{(Condiciones iniciales)} \\
y(0) &= y_0
\end{aligned}$$

donde K es un parámetro del modelo.

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

Descripción del laboratorio virtual

El laboratorio virtual deberá permitir al alumno modificar interactivamente:

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

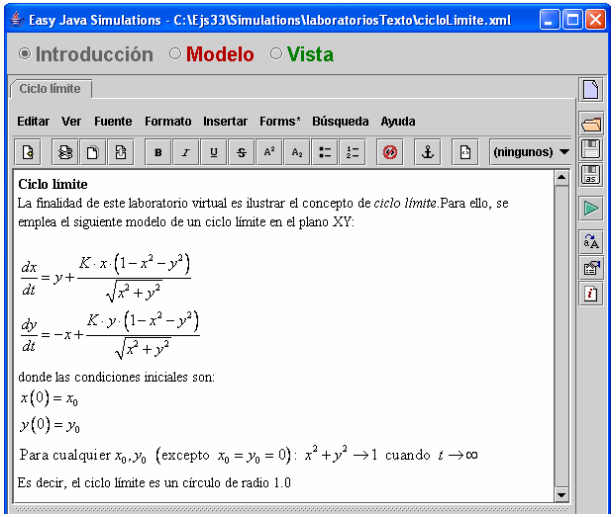
Tarea 1

Escriba la Introducción del laboratorio virtual

Para ello, use las imágenes:

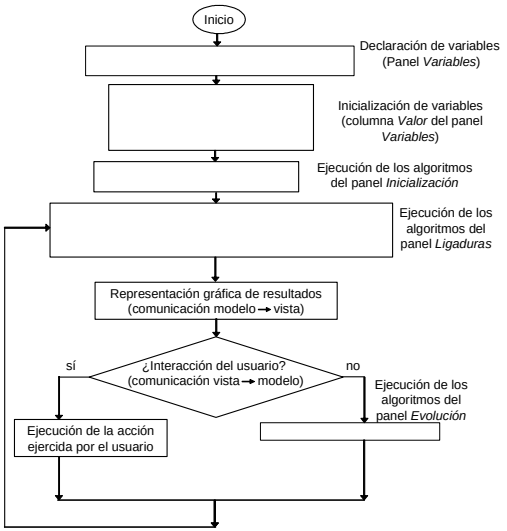
- cicloLimite1.gif
- cicloLimite2.gif
- cicloLimite4.gif

que encontrará en el directorio Imagenes



Tarea 2

Complete el algoritmo de la simulación



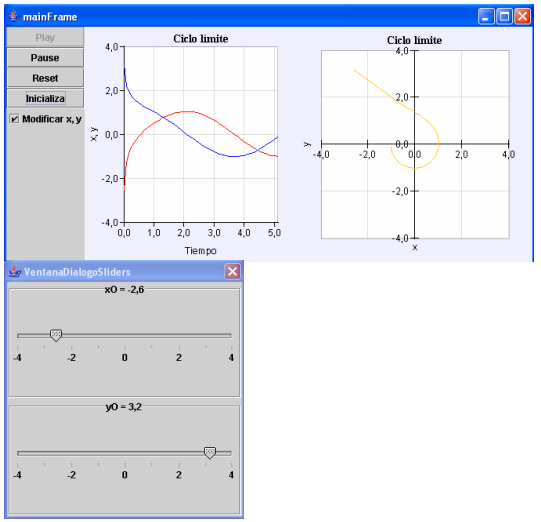
Tarea 3

Realice la definición del modelo en Ejs

1. Declaración e inicialización de las variables
2. Páginas Evolución y Ligaduras

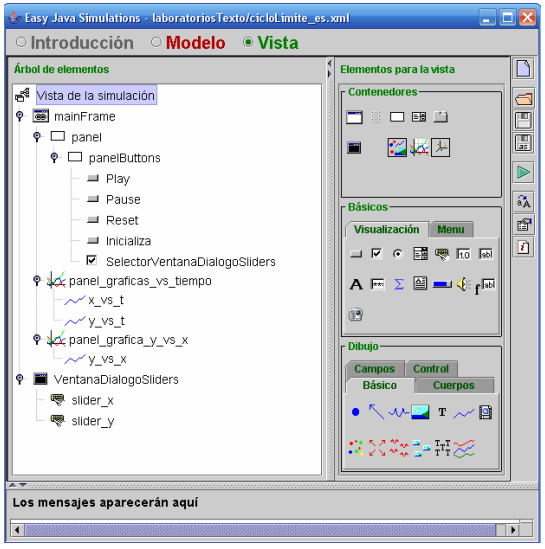
Tarea 4

Se pretende obtener la vista siguiente



Tarea 4

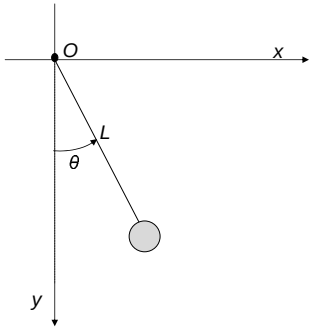
Árbol de la vista resultante



Péndulo simple

Descripción del modelo

Modelo matemático del péndulo simple linealizado entorno a la posición de equilibrio



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

Posición del péndulo en coordenadas cartesianas:

$$\begin{aligned} x &= L \cdot \cos(\theta) \\ y &= L \cdot \sin(\theta) \end{aligned}$$

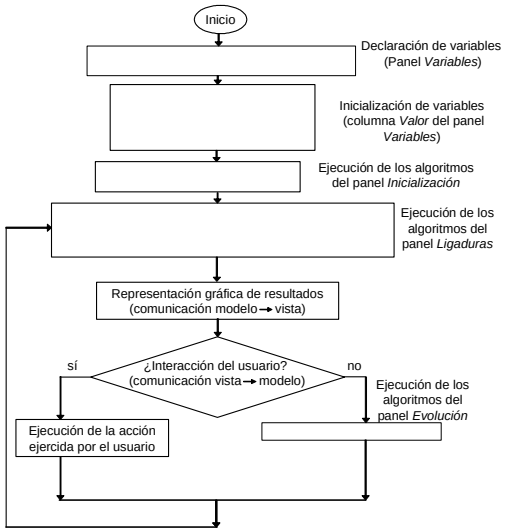
Tarea 1

Clasificar las variables del modelo en:

- Parámetros
- Variables de estado
- Variables algebraicas

Tarea 2

Complete el algoritmo de la simulación



Tarea 3

Realice la definición del modelo en Ejs

- Declaración e inicialización de las variables
- Completar la página Ligaduras
- Completar la página Propio

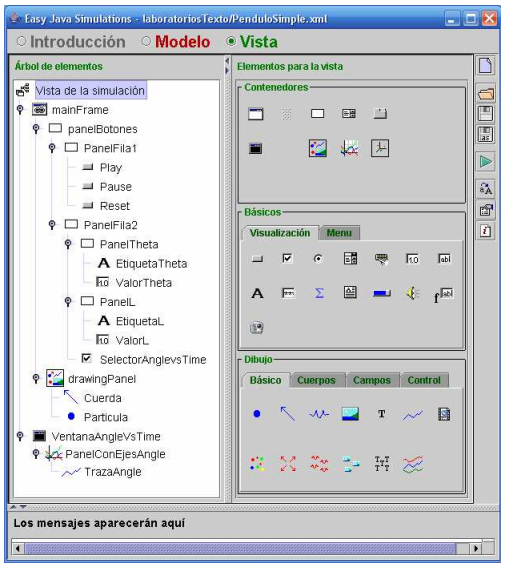
Tarea 4

Realice la **definición de la vista** en Ejs
El árbol de la vista debe contener los siguientes elementos

- Objeto de la clase Boton
- Objeto de la clase Etiqueta
- Objeto de la clase Selector
- Objeto de la clase PanelDibujo
- Objeto de la clase Flecha
- Objeto de la clase Particula
- Objeto de la clase VentanaDialogo
- Objeto de la clase PanelConEjes
- Objeto de la clase Traza

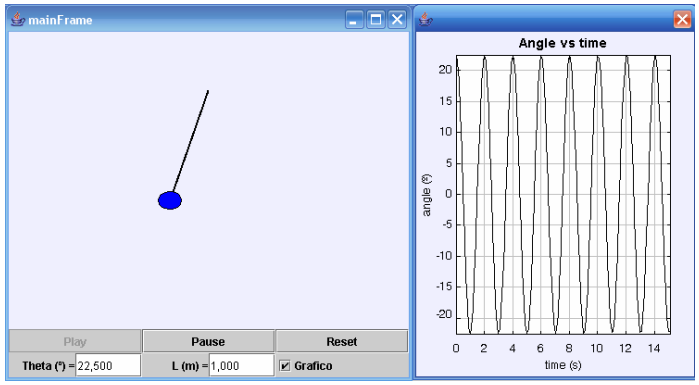
Tarea 4

Árbol de la vista resultante



Tarea 4

Vista resultante

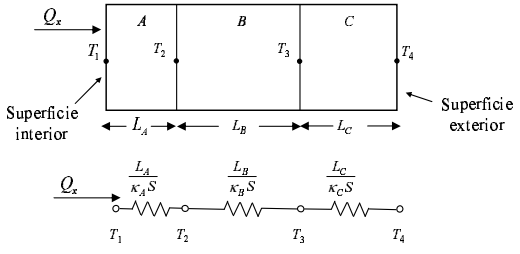


Conducción de calor en una pared múltiple

Descripción del modelo

Pared de una cámara frigorífica compuesta por 3 capas
Conducción de calor 1D en el estacionario

$$
\begin{aligned}
T_1 - T_2 &= \frac{L_A}{\kappa_A} \cdot q_x \\
T_2 - T_3 &= \frac{L_B}{\kappa_B} \cdot q_x \\
T_3 - T_4 &= \frac{L_C}{\kappa_C} \cdot q_x
\end{aligned}
$$



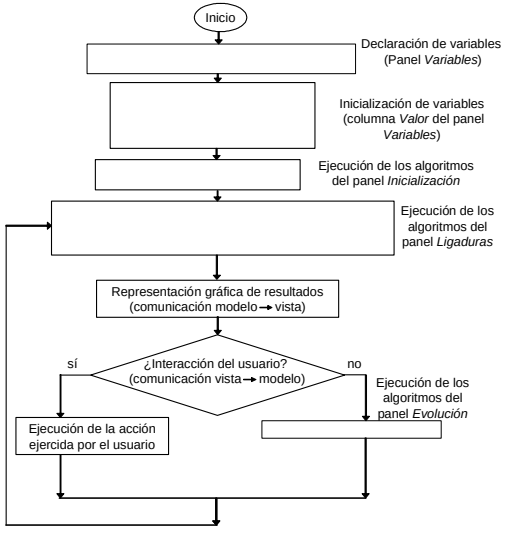
Descripción del modelo

Tras realizar la asignación de la causalidad computacional, se obtienen la ecuaciones siguientes:

$$
\begin{aligned}
[q_x] &= \frac{T_1 - T_4}{\frac{L_A}{\kappa_A} + \frac{L_B}{\kappa_B} + \frac{L_C}{\kappa_C}} \\
[T_2] &= T_1 - \frac{L_A}{\kappa_A} \cdot q_x \\
[T_3] &= T_4 + \frac{L_C}{\kappa_C} \cdot q_x
\end{aligned}
$$

Tarea 1

Complete el algoritmo de la simulación



Tarea 2

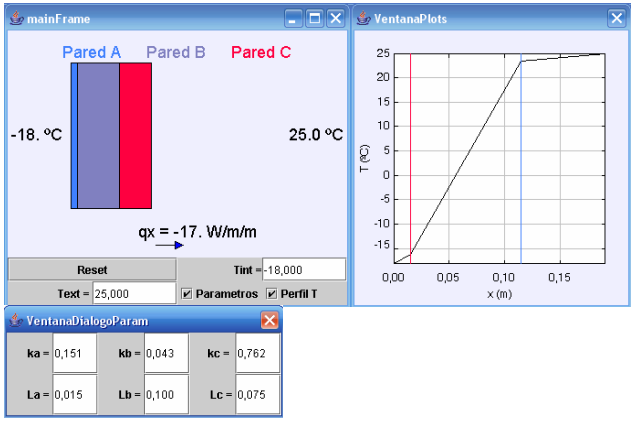
Realice la definición del modelo en Ejs

- Declaración e inicialización de las variables
- Completar la página de ligaduras

Tarea 3

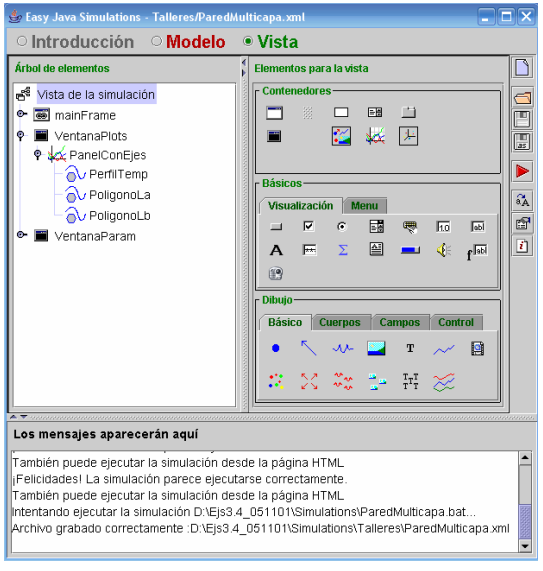
A partir del fichero ParedMulticapaTarea3.xml, completar la **definición de la vista** de Ejs

La vista ha de tener una ventana (**VentanaPlots**) donde se muestre el perfil de temperatura a lo largo de la pared (variables T y x)



Tarea 3

Árbol de la vista resultante



Cálculo de PI por el método de Monte Carlo

Descripción del modelo

Estimación del valor de π mediante el método de Monte Carlo.
El procedimiento consta de los pasos siguientes:

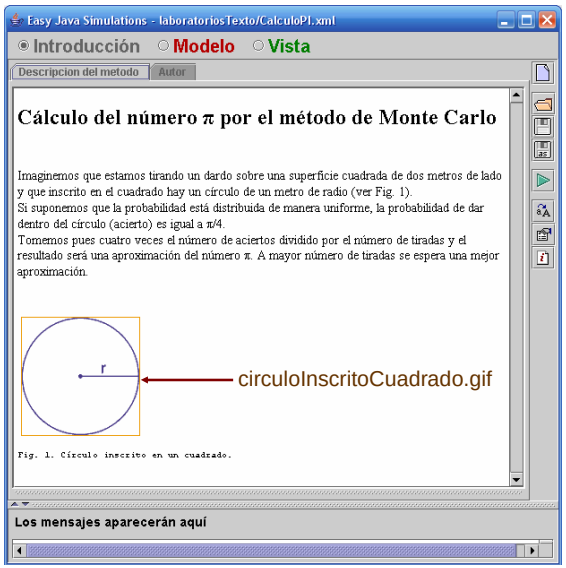
1. Se lanza el dardo de forma aleatoria n_{total} veces.
   Sea $n_{aciertos}$ el número de veces que el dardo ha quedado dentro del círculo.
2. El número π se estima de la forma siguiente:

$$\pi \approx 4 \cdot \frac{n_{aciertos}}{n_{total}}$$

A mayor número de tiradas, se espera una mejor aproximación.

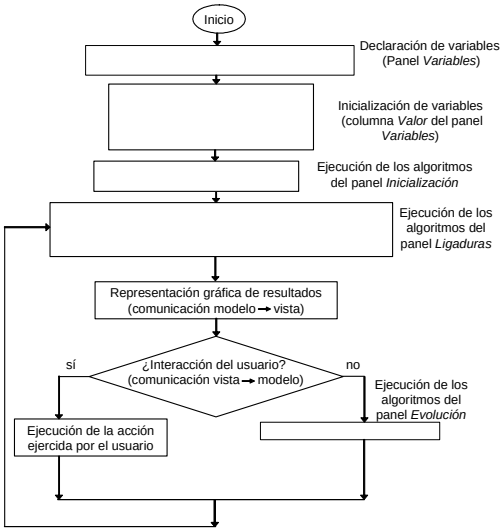
Tarea 1

Escriba la introducción del laboratorio virtual, con el contenido siguiente



Tarea 2

Complete el algoritmo de la simulación



Tarea 3

Realice la **definición del modelo** en Ejs

- Declaración e inicialización de las variables
- Completar la página de ligaduras
- Completar la página de propio

Obtener de un número aleatorio distribuido uniformemente en [-1,1]:

$$2 * Math.random() - 1$$

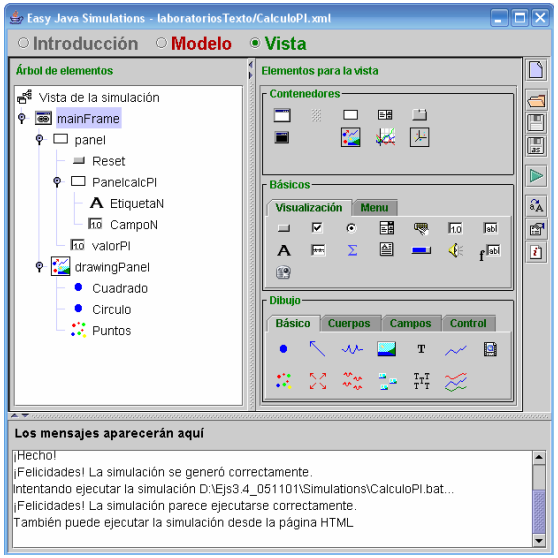
Tarea 4

Realice la definición de la vista en Ejs.
El árbol de la vista debe contener los elementos de dibujo siguientes:

- Objeto de la clase ConjuntoParticulas
- Objeto de la clase Particula

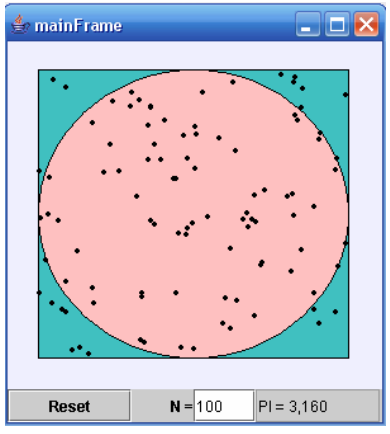
Tarea 4

Árbol de la vista resultante



Tarea 4

Vista resultante



Simulación interactiva de un globo aerostático

Descripción del modelo

"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)."

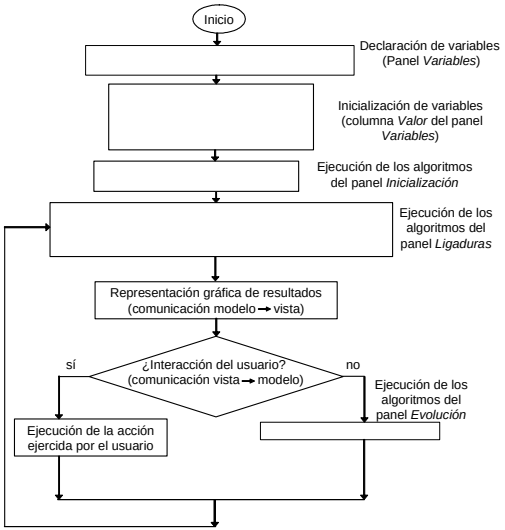
Descripción del modelo

El modelo consta de las diez ecuaciones siguientes

$$
\begin{aligned}
[T] &= T_0 - 6.5 \cdot 10^{-3} \cdot h \\
[P] &= P_0 \cdot \left(\frac{T_0}{T}\right)^{-5.256} \\
[\rho] &= \frac{P \cdot M}{R \cdot T} \\
[F_l] &= \rho \cdot V \cdot g \\
[\rho_{dg}] &= \frac{P \cdot M}{R \cdot T_{dg}} \\
[w] &= \rho_{dg} \cdot V \cdot g + w_s + w_l \\
[m] &= \frac{w}{g} \\
[derv] &= \frac{F_l - w}{m} \\
[derh] &= v \\
[derT_{dg}] &= apagado \cdot K \cdot \left(T - T_{dg}\right) + (1 - apagado) \cdot q
\end{aligned}
$$

Tarea 1

Complete el algoritmo de la simulación



Tarea 2

A partir del fichero GloboAerostaticoTarea1.xml, complete la definición del modelo en Ejs.

Para ello realice las dos tareas siguientes:

1. Rellene la página Evolución (inserte las EDO)
2. Rellene la página Ligaduras (inserte las ecuaciones restantes)

Tarea 3

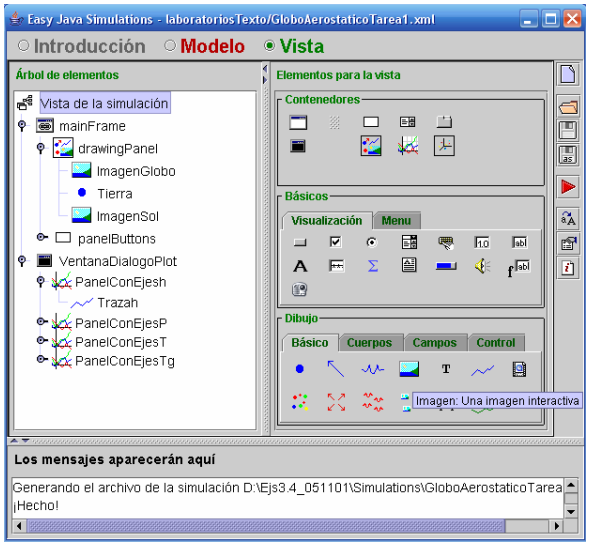
A partir del fichero GloboAerostaticoTarea2.xml, complete la definición de la vista de Ejs.

Para ello realice las dos tareas siguientes:

1. Realice una **vista animada del globo aerostático**.
   Para ello use dos objetos de la clase Imagen y un objeto de la clase Particula.
   Asocie a cada uno de los dos objetos de la clase Imagen las imágenes globoAerost.jpg y sun.gif que se encuentran ubicadas en el directorio Imagenes.
2. Cree una **ventana** donde se puedan ver la variación temporal de las variables h, P, T y Tg.
   Para ello emplee objetos de las siguientes clases: VentanaDialogo, PanelConEjes y Traza.

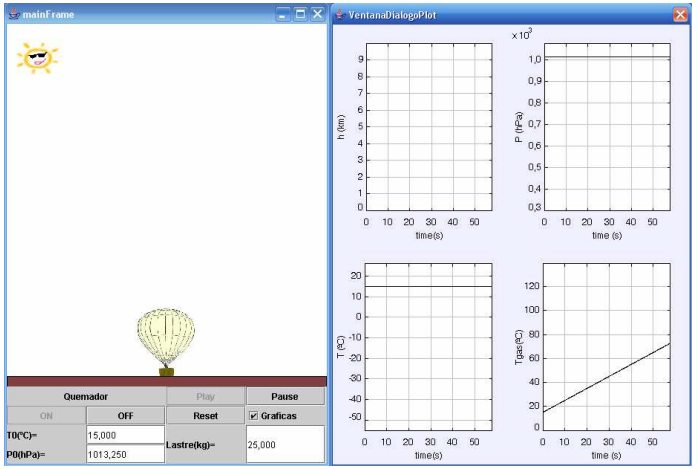
Tarea 3

Árbol de la vista resultante



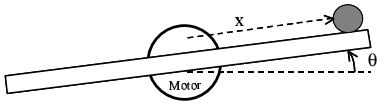
Tarea 3

Vista resultante:



Sistema bola y varilla

Descripción del modelo



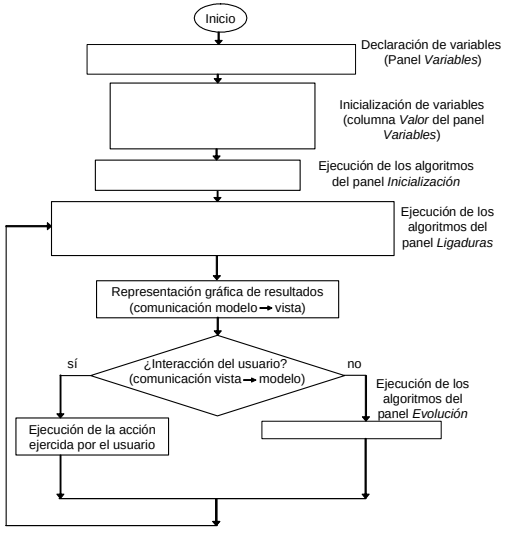
$$\frac{dx}{dt} = v$$
$$\frac{dv}{dt} = -\frac{5}{7} \cdot g \cdot \sin(\theta)$$

Se realizan las dos consideraciones siguientes:

- **Se limita la posición de la bola a la longitud de la varilla**
  Cuando la bola alcanza uno de los extremos de la varilla, no se permite que sobrepase el extremo y se iguala su velocidad a cero.
- **Modelado de la fricción**
  Cuando la velocidad de la bola y el ángulo de la varilla son pequeños, la velocidad de la bola se hace cero.

Tarea 1

Complete el algoritmo de la simulación



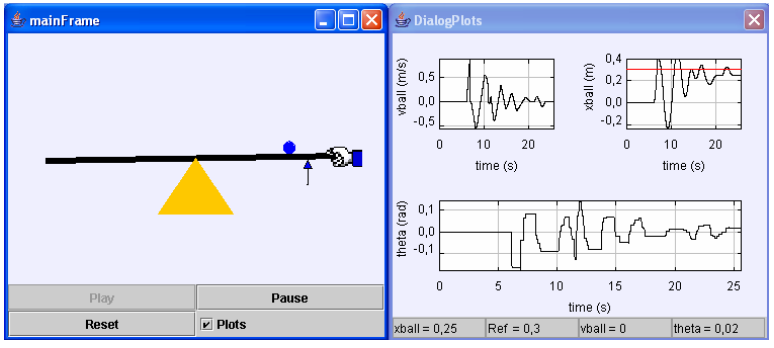
Tarea 2

Realice la definición del modelo en Ejs

1. Declaración e inicialización de las variables
2. Páginas Evolución y Ligaduras

Tarea 3

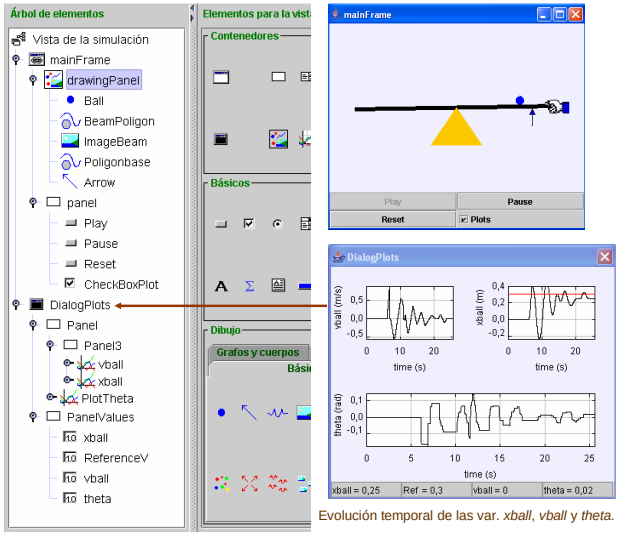
Vista resultante



Aplicación de la Simulación por Ordenador a la Enseñanza de las Ciencias
└Sistema bola y varilla

Tarea 3

A partir del fichero bolaVarillaTarea3.xml cree la ventana DialogPlots



Evolución temporal de las var. *xball*, *vball* y *theta*.