# Documentación Tarea nº3

Por:
- *Luis Fuentes*
- *Benjamín Ginouvès*

# Introducción

En esta tarea se pretende dar una versión para la implementación de un laboratorio virtual, pudiendo ser incluido dentro de una página WEB o ser capaz de ejecutarse como una aplicación separada. Este laboratorio virtual, da 3 vistas del movimiento de uno de los bloques a elección, movimiento en el eje x versus el tiempo, eje y versus el tiempo, y desplazamiento del bloque en el plano utilizando una ventana de tiempo para mostrar su desplazamiento.

Este laboratorio virtual puede observarse como un sistema que interactúa "casi" en tiempo real y con elementos físicos representados por objetos de software.

La aplicación pretende dar una visión gráfica en dos dimensiones, con el movimiento de la interacción entre bloques, resortes y elásticos, y mostrando éste movimiento de varias formas. La aplicación brinda la libertad de posicionar tanto los bloques como elásticos y resortes de cualquier forma, con la restricción de no poder interconectar los extremos de resortes con elásticos, restringiéndose a la conexión solamente a los bloques.

# Descripción y Diseño

Se modela la interacción de objetos reales, elásticos, resortes y bloques, de la naturaleza a partir de modelos que son representados en clases y objetos de software, que a su vez se presentan de forma gráfica para su fácil manipulación. La ejecución de la aplicación se puede realizar mediante alguno de los dos comandos:

```
java –jar PhysicsLab.jar

make run
```

El ultimo además de ejecutar el programa, compila y crea el archivo .jar en caso de ser requerido. Otra forma es a través de una página WEB, incluyendo dentro de su código lo siguiente:

```
<applet code="PhysicsLabApplet.class" archive="PhysicsLab.jar" width="800" height="600">
       <param name="RefreshTime" value="0.01"/>
       <param name="DeltaTime" value="0.001"/>
       <param name="timeWindowSize" value="2"/>
       <param name="Gravity" value="100"/>
</applet>
```

## Aplicación

Para la versión ejecutada con alguna de las primeras opciones de línea de comandos, el manifiesto del archivo PhysicsLab.jar define como clase main a la clase *PhysicsLab.class.* Al iniciar la ejecución, el programa busca un archivo llamado PhysicsLab.ini que contiene los valores de inicio para la simulación en la forma "Propiedad" = "valor", al no encontrarse este archivo, se muestra un mensaje de error y se utilizan los valores por defecto. Si el archivo está presente pero falta una o más propiedades, el programa sigue su ejecución normalmente, tomando también los valores por defecto para las propiedades ausentes.

El programa genera un panel *JFrame* dividiéndolo a través del uso de *JSplitPane* indicando los paneles que se encuentran a cada lado y la forma de división, para luego configurar el tamaño proporcional de cada uno de los paneles. A la izquierda de centrará el panel *labPanel*, el cual no presenta mayores cambios desde la tarea anterior, y al lado derecho el graphPanel, donde se colocan 3 gráficos para mostrar el movimiento del bloque seleccionado. En la figura nº1 se muestra el panel del laboratorio completo.

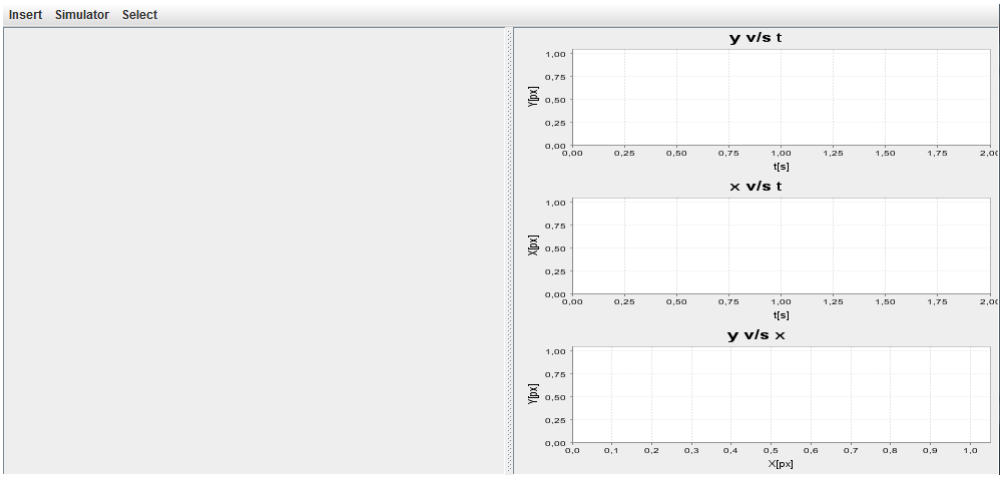



**Figura nº1.- Panel principal del laboratorio virtual.**

## Applet

En el caso de la ejecución por página web, la inicialización se realiza por la clase *PhysicsLabApplet.class*, en la cual se redefinen las funciones de interés que llamará el navegador. Dentro de estas funciones, *init()* representa el método que será ejecutado una vez el navegador cargue la página con la llamada al applet. Las llamadas *start(), stop() y destroy()*, serán llamadas cuando se vuelva a la página o navegador, ida a otra aplicación o página, y se cierre el navegador respectivamente, pero como en nuestro programa no deseamos realizar alguna acción especial con estas funciones, estos no se redefinen y quedan de la misma forma que en la clase JApplet.

En ambos casos, el tamaño de los gráficos corresponde a 1/3 del área total del panel derecho, cambiando de tamaño si el panel se agranda o achica. Para realizar estos gráficos se ha utilizado la librería *JFreeChart*, a través de la utilización de ésta clase, la implementación de los gráfico se aliviana considerablemente, siendo requerido sólo la configuración de 3 *ChartPanel* en el objeto *graphPane* (clase *GraphPanel*) junto a sus respectivos datasets y series de datos.

Al utilizar una librería externa a java, se requiere tener esta para la compilación y ejecución del programa, por lo que se incluye tanto en el código fuente como en el PhysicsLab.jar dentro de la carpeta org.

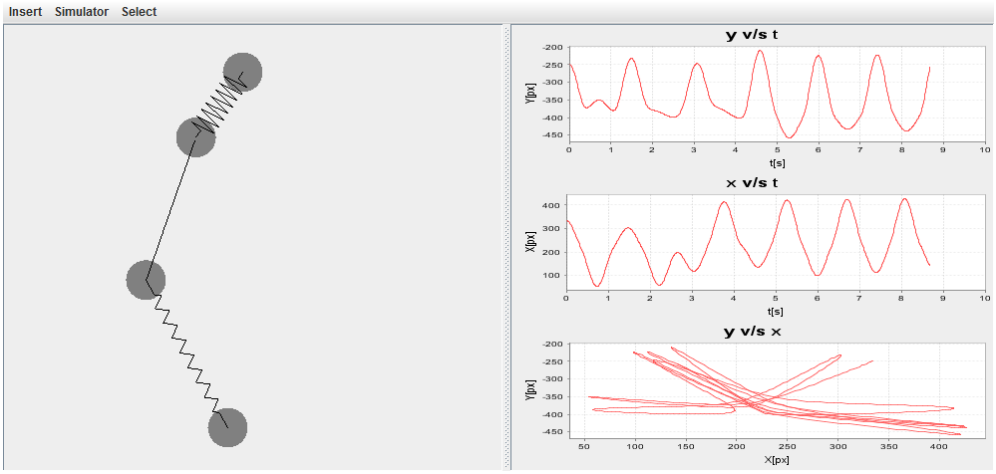
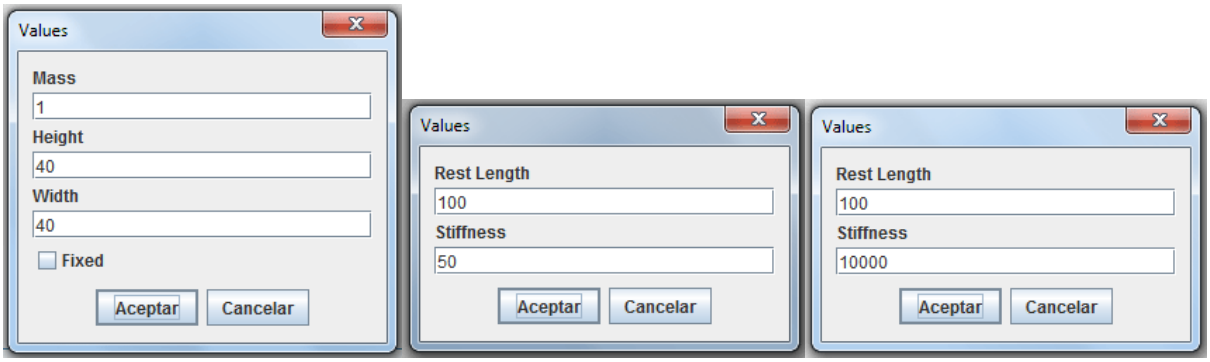



**Figura nº2.- Panel principal del laboratorio virtual realizando una simulación.**

La barra de menú para las opciones de los objetos a insertar y las opciones de la simulación no han sufrido grandes modificaciones, debido a que los requerimientos indicados son cumplidos tanto por la aplicación como por el applet.

Las masas se representan como elipses con posibilidades de cambiar las dimensiones, valor de su masa y si deben quedar fijos dentro del panel de simulación. Los resortes se representan como líneas poligonales pudiendo desplazar sus extremos para vincular con cualquier bloque, pudiendo ingresar los valores de su largo y coeficiente al momento de crear el objeto físico, de igual forma pero representados como una línea, se insertan los elásticos en la simulación. Para mayor simplicidad, tanto los resortes como los elásticos se han considerado sin masa y cumpliendo con la ley de Hooke a lo largo de toda su extensión. Para facilitar la inserción de elementos, los menús de atributos vienen con valores predeterminados para cada elemento.

El sistema inicialmente se encuentra afectado por la gravedad, pero debido al cambio de elementos (cuerdas por elásticos), la amortiguación para la cuerda introducida en la tarea 2 fue retirada como opción por defecto, pero esta se puede volver a activar dentro de Setting -> String damping, para que el elástico pueda adoptar la forma de cuerda modelada en la tarea anterior.




**Figuranº3: Menús de selección de atributos de los elementos físicos a insertar a)Masa, b)Resorte y c)Elástico.**

A su vez, inicialmente el grafico esta en blanco, pero el programa permite la selección de la masa a graficar de 2 formas, directamente desde el panel y mediante el identificador de la masa, cuyo ítem se agrega al menú al insertar la masa. Haciendo click en "From panel", la simulación se detiene en caso de estar corriendo y al hacer click en una masa esta comienza a graficarse y, en caso de que la simulación se halla detenido, esta se reinicia automáticamente. La segunda opción permite seleccionar la masa sin detener la simulación ni saber la posición de la masa, simplemente con su identificador, que corresponde al orden en el que se ingreso la masa.

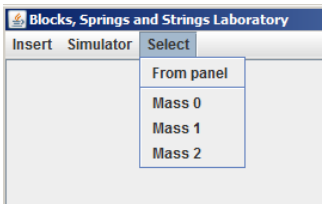



**Figura nº4.- Selección de masa a graficar.**

Dado que el código del programa está basado en el código entregado para la tarea anterior, no se entrará en detalles del código que se ha dejado sin modificar. Los cambios hechos son:

- Se cambia el nombre a la clase *MyString* a *RubberBand* sin cambios mayores. Del mismo modo se ha cambiado el nombre a la clase GString a GRubberBand para la representación de los elásticos.

- Se agregan 2 nuevas clases, *GraphPanel* y *SelectListener*, la primera correspondiente al panel que contiene a los gráficos junto a sus métodos para actualizarlos, y la segunda implementa a action listener para la selección de la masa a graficar.

- Se modifica la clase *LabFrame* para implementar el splitpane, inicializar el panel de graficos y el listener encargado de manejar las acciones del menú Select.

- Se modifica la clase *LabMenuBar* para agregar el menú Select, y agregarle los menú ítems a este a medida que se van insertando masas.

- Se crea la clase *PhysicsLabApplet*, la que contiene la inicialización del programa para ser ejecutado como applet, esta combina las clases *PhysicsLab* y *LabFrame*, siendo esta ultima eliminada dado a la no necesidad de un frame dentro del applet.

- Se modifican las clases *Simulator*, *BlockSpringConfiguration* y *MouseListener*, modificando y agregando métodos que permitan la identificación y selección del bloque que se desea graficar.

- Se modifican las clases *Simulator* y *Block* para añadir las mejoras en la simulación planteadas por el profesor, pero debido a que nuestro grafico utiliza el parámetro t de *Simulator* para el eje del tiempo, no fue posible implementar el cambio planteado por el, pero este se cambio de float a double, para que al menos el programa pueda correr por un tiempo prolongado sin fallar.

La documentación del código se generará a partir del código fuente de la aplicación, en la carpeta Doc, utilizando:

```
make doc
```

# Casos de Uso

En la figura nº 5, se presenta un diagrama de Casos de Uso para las acciones del ingreso de los elementos al panel, y las acciones de mostrar las simulaciones que dependen de que se haya ingresado o no los elementos al panel, de otra forma no puede hace nada la aplicación.

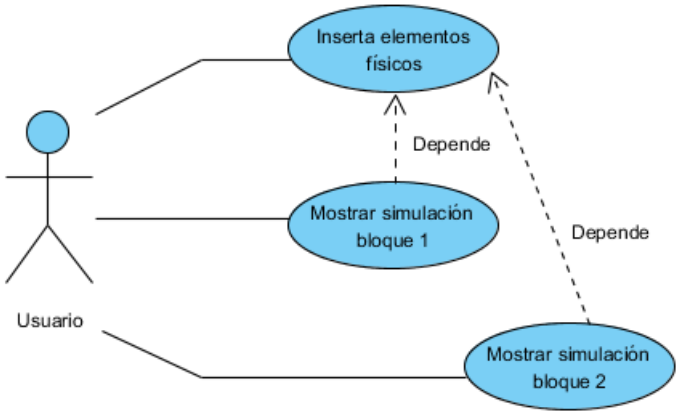



**Figura 5.- Diagrama de Casos de Uso para 3 casos.**

| Nombre | **Inserta Elementos Físicos** |
|---|---|
| Propósito | Insertar elementos en el panel de simulación, presentando una vista inicial al usuario de lo que desea simular. |
| Actores | Usuario |
| Pre-Condiciones | No haber iniciado la simulación, o haber detenido la simulación. |
| Evento | Inicio de la aplicación. |
| Pos-Condiciones | Disponibilidad de inicial la simulación |
| Tipo | Manual |

| Nombre | **Mostrar Simulación del bloque 1** |
|---|---|
| Propósito | Inicia la simulación en el panel de la aplicación, mostrando los gráficos del movimiento del bloque 1 al usuario. |
| Actores | Usuario |
| Pre-Condiciones | haber insertado elementos en el panel. |
| Evento | Acción del usuario |
| Pos-Condiciones | Disponibilidad de detener la simulación y/o cambiar los ajustes de las variables que condicionan la simulación. |
| Tipo | Manual |

| Nombre | **Mostrar Simulación del bloque 2** |
|---|---|
| Propósito | Inicia la simulación en el panel de la aplicación, mostrando los gráficos del movimiento del bloque 2 al usuario. |
| Actores | Usuario |
| Pre-Condiciones | haber insertado elementos en el panel. |
| Evento | Acción del usuario |
| Pos-Condiciones | Disponibilidad de detener la simulación y/o cambiar los ajustes de las variables que condicionan la simulación. |
| Tipo | Manual |

# Tarjetas CRC

A continuación se presentan las Tarjetas CRC básicas de dos de las clases implementadas en la aplicación. A la izquierda se indican las responsabilidades de la clase y a la derecha los colaboradores de la clase.

Clase Simulador, clase que controla la inicialización de las variables por defecto (antes de leer de otras fuentes) y la que inicia y detiene la simulación.

| **Simulator** | |
|---|---|
| *Inicializar las variables.*<br>*Iniciar la simulación.*<br>*Detener la simulación.* | BlockSpringConfiguration<br>GraphPanel<br>Block |

Clase que controla los elementos gráficos del panel del laboratorio donde se realiza la simulación, se encarga de volver a pintar los elementos al ser cambiados por la simulación o por acción del usuario.

| **LabPanel** | |
|---|---|
| *Repintar los elementos gráficos del panel.* | BlockSpringConfiguration |

# Conclusiones

El trabajo de los gráficos del movimiento del bloque seleccionado se llevó a cabo con la implementación de una clase importada, la *JFreeChart* que representa en variadas formas los gráficos, su inclusión en el proyecto se basó plenamente en la documentación disponible en la clase y ejemplos de su uso. Esto ayudó en el aprendizaje de la mecánica de importación y uso de clases diseñadas por terceros para fines propios.

La implementación del *Applet* se basó según los pasos indicados, liberando a éste de crear el panel contenedor (*JFrame*) e inicializando las clases, objetos e interfaces respectivas de la aplicación, ahora como *Applet*. Esta implementación no tuvo mayores problemas, dado que la mayoría del código fue reutilizado de la tarea anterior, donde ya se habían realizado las modificaciones de la entrada de datos solicitada en esta oportunidad.

Dentro de las dificultades, la más difícil de solucionar fue posicionar e indicar correctamente la ubicación de las librerías de JFreeChart durante la creación del archivo .jar y la compilación en aragorn, ya que en NetBeans es simple, pero útil solamente durante la creación del programa.

El cambio de nombre de la clase, bajo el NetBeans, no tuvo mayores problemas dado que lo realiza de forma automática, cambiando todas las referencias asociadas en las clases y objetos.

En algunos Browsers o Navegadores se puede demoren mucho en limpiar la memoria al cerrar un applet, lo que puede derivar en un mal funcionamiento.

El diseño del los gráficos de Casos de Uso, se realizó de acuerdo a las acciones concretas que se requieren y son visibles al actuador, en este caso el usuario.

Las tarjetas CRC se han diseñado del visto en clases, ya que según programas más elaborados y orientados al maneja de estos formatos, requieren de la entrada de las clases superiores, atributos y de una descripción de las clases.