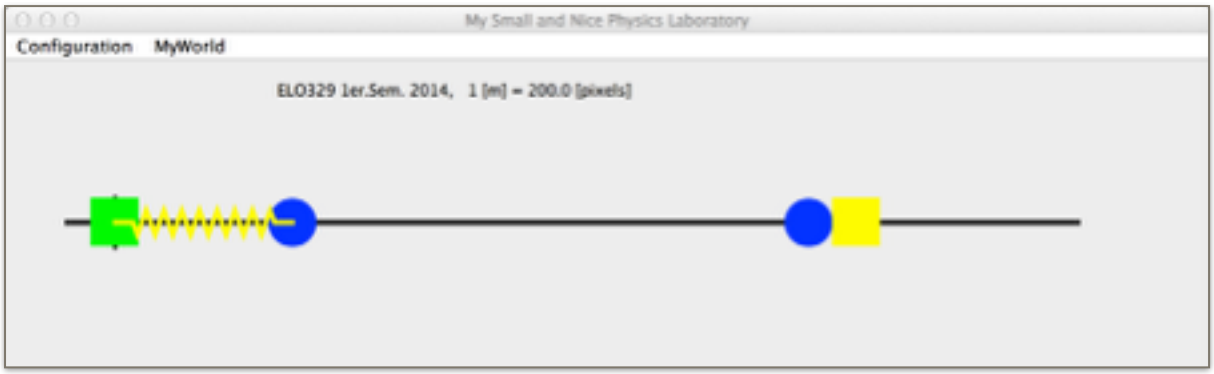
Marco Benzi, René Pozo y Andrés Ulloa
Nombre del profesor
Título del curso
28 de mayo de 2014

Tarea Nº2 de Programación Orientada a Objetos



Introducción

Segunda tarea correspondiente al ramo "Diseño y Programación Orientada a Objetos" ELO329. Este programa realizado en Java simula un laboratorio de física que comprende cuatro objetos: puntos fijos (FixedHook), bolas (Ball), bloques con roce cinético (Block) y resortes (Spring), y simula la interacción entre ellos. Los resultados de estas simulaciones se muestran a través de una interfaz gráfica en donde también se pueden ajustar los distintos parámetros de simulación.
El objetivo de esta tarea fue familiarizarse con los entornos gráficos y como implementar el uso de un mouse, es decir una extensión de la primera tarea en donde se simulaban las mismas situaciones pero sin un entorno gráfico.

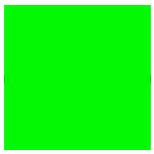
Dificultades Encontradas

- Al desarrollar la interfaz gráfica en donde al arrastrar el resorte después de haber estado conectado a objetos, este seguía conectado a ellos a pesar de ya encontrarse en otra posición. Este *bug* persistió por un tiempo, pero fue eliminado para la versión final.
- Otra dificultad fue el tener que generalizar el código para agregar un nuevo elemento capaz de colisionar, el bloque con roce, ya que el código original solo contemplaba choques entre bolas. Esto se logró generalizando los métodos de colisión para todo objeto simulable (que implemente *Simulateable*).

Objetos simulables

Bola (*Ball*)

Es colisionable, transfiere momento y no posee roce. Permite enganchar resortes



Punto Fijo (*Fixed Hook*)

No es colisionable. Permite enganchar resortes

Bloque (*Block*)

Es colisionable, transfiere momento y posee roce cinético. Permite enganchar resortes

Resorte (*Spring*)

No es colisionable, permite unir dos objetos haciéndolos oscilar.

Diseño de clases y diagrama

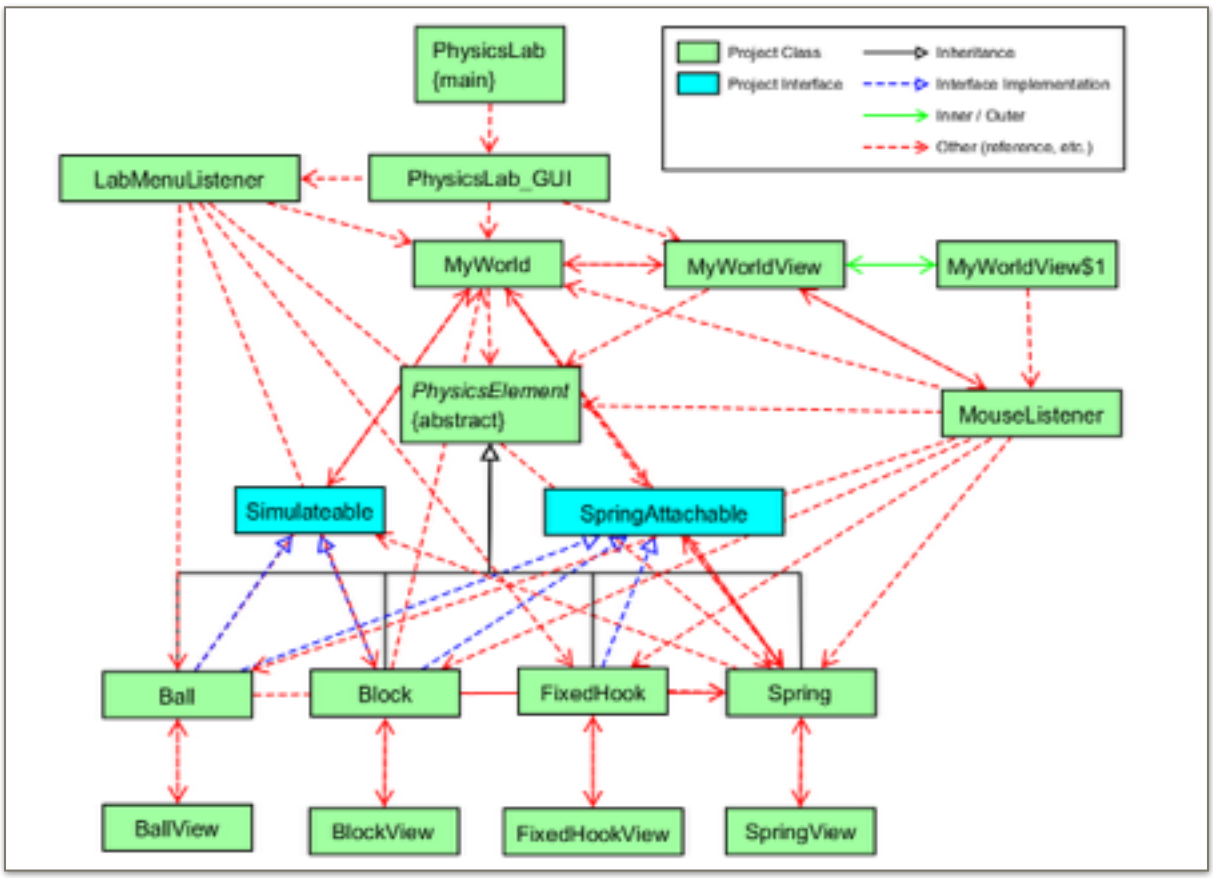


Figura 1: Diagrama de interacción de clases generado por jGrasp

En el **README** adjunto a nuestra tarea se da una explicación breve de cada clase. En esta parte intentaremos fundamentar la interacción general en la aplicación.

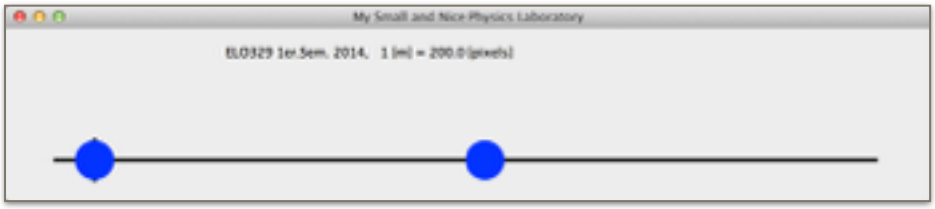
El núcleo de nuestro programa se encuentra en las clases **MyWorld** y **MyWorldView**, cuales se encargan de procesar el estado de todos los elementos y mantenerlos coordinados durante la simulación. Entre ellas, y entre todos los elementos simulables, existe una relación de forma *Modelo-Vista-Controlador*.

Todo elemento llevado al mundo virtual para ser simulado deriva de la clase abstracta **PhysicsElement**, y también posee una contraparte ***View** que dibuja la vista sobre el modelo descrito en su lógica. Además existen las interfaces **Simulateable** para definir cuales clases podrán simularse (no permanecerán estáticas) y **SpringAttachable** para cuales podrán ser enganchadas con un resorte.

Finalmente se usan tres clases de Listeners: **LabMenuListener**, asociada a eventos de menú; **MouseListener,** dedicada a eventos de mouse y una clase anónima (marcada en el diagrama como **MyWorldView$1**) para los eventos de teclado.

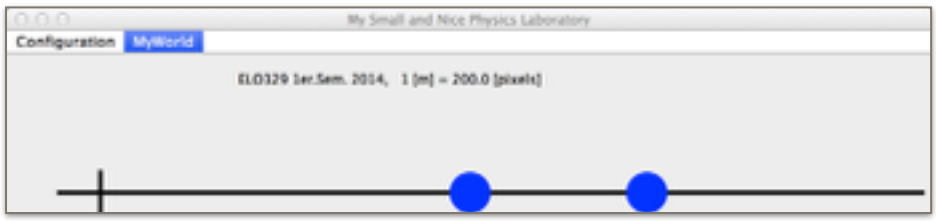
Etapas realizadas

Cada una de estas etapas cuenta con un *release* en **GitHub**.



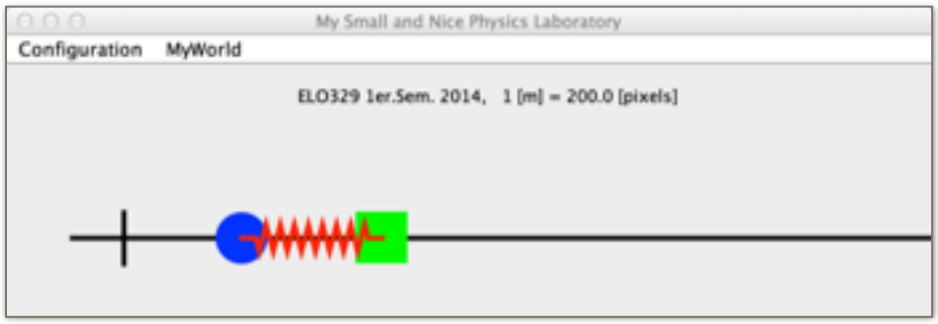
Etapa 1

Dibujando las bolas



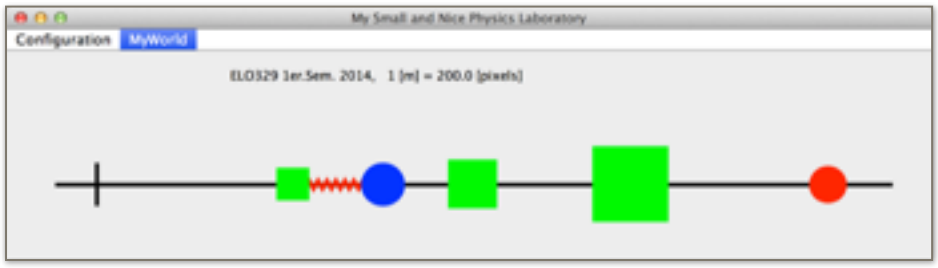
Etapa 2

Habilitamos la simulación



Etapa 3

Habilitamos resortes y puntos fijo, además de un escenario básico



Etapa 4

Habilitamos la generación de objetos con masa y tamaños distintos (aleatorios)

Etapa 4 + 1

Incorporamos bloques con roce estático