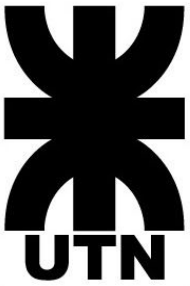


Universidad Tecnológica Nacional
Facultad Regional Villa María

Ingeniería en Sistemas de la Información

Paradigmas de Programación

Trabajo práctico

Alumnos:

- Brizzi Luka (lukabrizzi@gmail.com) (13770)
- Márquez Juan Cruz (marquezjuanchy@hotmail.com) (13359)
- Muzillo Tomás (tomimuzzillo@gmail.com) (13765)
- Guridi Ignacio Javier (nacho_g88@hotmail.com) (13506)

2- (60 pts) Sobre el escenario del Laboratorio de Newton, se deben realizar las siguientes tareas (**grupal - colaborativo**):

A: Describir las principales características de los 3 Newton Labs (funcionamiento, clases, objetos, métodos, etc)

B: A partir del Newton Lab3, incorporar una modificación **significativa** a la elección.

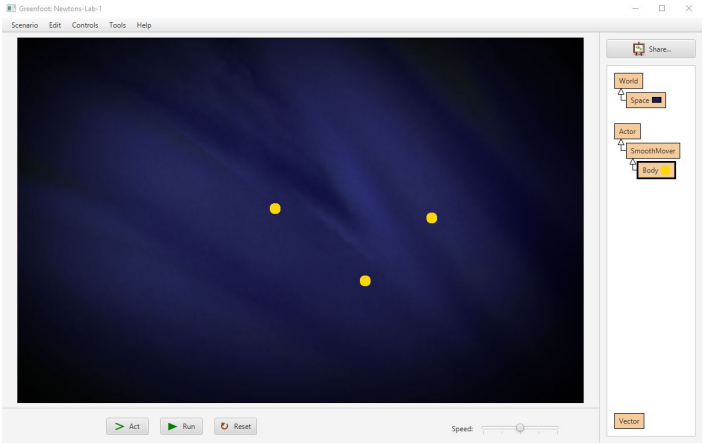
En la presentación de la solución de este punto deberá incluirse:

- una memoria descriptiva del cambio a realizar.

- imágenes, figuras, fotos, etc que documenten los cambios realizados.

- un pequeño video comentando el resultado obtenido

NEWTON LAB 1



Funcionamiento:

Cuando intente ejecutar este escenario, notará que puede colocar objetos (de tipo Body) en Space, pero estos cuerpos no se mueven, y no actúan de ninguna manera interesante.

Clases:
SmoothMover , Body*(subclase de SmoothMover*) y Vector

Objetos:
Space - Body - Vector

Métodos

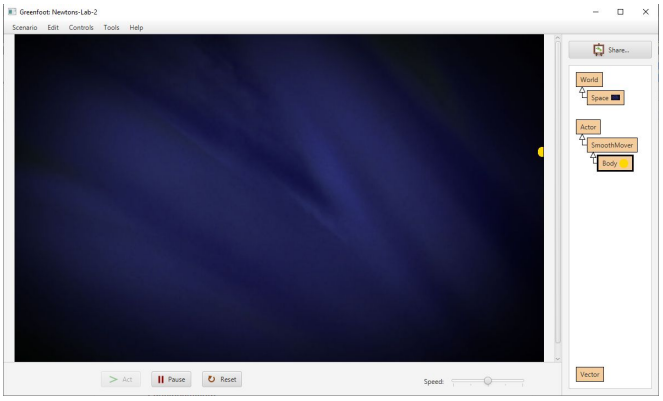
Body : Body () - act() - applyForces() - applyGravity() - getMass()

Space : Space() - sunAndPlanet() - sunAndTwoPlanets() - sunPlanetMoon() - removeAllObjects()

SmoothMover : SmoothMover() - move() - setLocation() - getExactX() - getExactY() - addToVelocity() - accelerate() - getSpeed() - invertHorizontalVelocity() - invertVerticalVelocity()

Vector: Vector() - setDirection() - add() - setLength() - scale() - setNeutral() - revertHorizontal() - revertVertical() - getX() - getY() - getDirection() - getLength() - updatePolar() - updateCartesian()

NEWTON LAB 2



Funcionamiento:
Es un mundo vacio el cual le podemos introducir unas pelotas de color amarillo, las cuales en primer lugar se atraen hasta chocarse y luego se separan en diferentes direcciones.

Superclases:
World, Actor
Clases:
SmoothMover, Vector
Subclases:
Body
Objetos:
Body - Space - Vector

Métodos

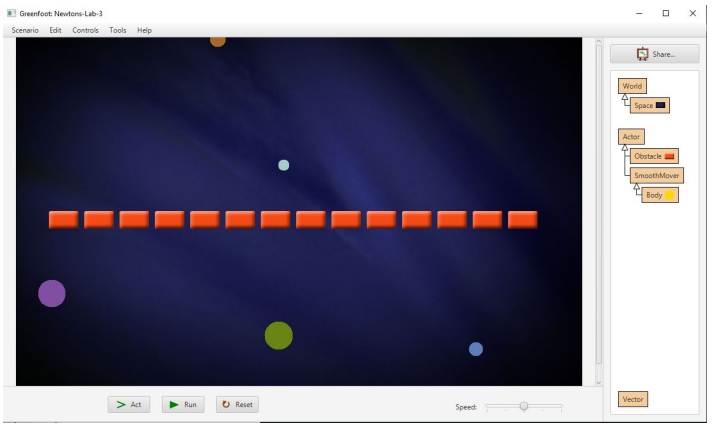
Body: void act(), double getMass(),void applyForces(), void applyGravity().

Space: void sunAndPlanet(), void sunAndTwoPlanets(), void sunPlanetMoon(), void removeAllObjects().

Smoothmover: void move(), void setLocation(), void setLocation(), double getExactX(), double getExactY(), void addToVelocity(), void accelerate(), double getSpeed(), void invertHorizontalVelocity(), void invertVerticalVelocity().

Vector: void setDirection(), void add(), void setLength(), void scale(), void setNeutral(), void revertHorizontal(), void revertVertical(), double getX(), double getY(), int getDirection(), double getLength(), void updatePolar(), void updateCartesian().

NEWTON LAB 3



Funcionamiento:

Es un mundo que posee 14 obstaculos y 5 pelotas de tamaño, color y pocisión random, le podemos introducir unas pelotas de color amarillo, todas las pelotas en primer lugar se atraen hasta chocarse y luego se separan en diferentes direcciones. Cada vez que pasa una pelota sobre un obstáculo emite sonido de piano diferente para cada uno.

Clases:
SmoothMover , Body, Vector y obstacle

Objetos:
Body - Space - Obstacle - Vector

Métodos

Space: Space(), void createObstacles(), void randomBodies().

Body: void act(), void bounceAtEdge(), void applyForces(), void applyGravity(), double getMass().

Obstacle: void act(), void playSound().

SmoothMover : SmoothMover(), void move(), void setLocation(), void setLocation(), double getExactX(), double getExactY(), void addToVelocity(), void accelerate(), double getSpeed(), void invertHorizontalVelocity(), void invertVerticalVelocity().

B: A partir del Newton Lab3, incorporar una modificación **significativa** a la elección.

En la presentación de la solución de este punto deberá incluirse:

- una memoria descriptiva del cambio a realizar.

- imágenes, figuras, fotos, etc que documenten los cambios realizados.

- un pequeño video comentando el resultado obtenido

Memoria descriptiva: Comenzamos colocando 7 filas de obstáculos (bloques) los cuales desaparecen una vez que el cuerpo los toca. Nos basamos en el clásico juego “bricks”, por lo tanto introdujimos un “player” como una clase la cual tiene como objetivo hacer rebotar al cuerpo y desplazarlo en dirección hacia los obstáculos provocando que estos desaparezcan.
El objetivo del juego es lograr romper todos los obstáculos sin que el cuerpo caiga al vacío.

Imagen Body:

```
public void moveAround(){
    setLocation(getX() + dx, getY() + dy);

}
public void bounce(){
    if(isTouching(Player.class) || isTouching(Obstacle.class) ){
        turn(Greenfoot.getRandomNumber(90)-45);
        removeTouching(Obstacle.class);
        dy = -dy;
    }
}
```

Agregamos un método llamado bounce() el cual se encarga de remover los obstáculos (bricks) si los toca y el body rebota cada vez que toca el Obstacle() o el Player().
Luego el método moveArround() que tiene como objetivo mover el body por el espacio.

Imagen Player:

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)

/**
 * Write a description of class Player here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class Player extends Actor
{
    /**
     * Act - do whatever the Player wants to do. This method is called whenever
     * the 'Act' or 'Run' button gets pressed in the environment.
     */
    public void act()
    {
        moverJugador();
    }

    public void moverJugador(){
        if (Greenfoot.isKeyDown("left"))
        {
            move(-7);
        }
        if (Greenfoot.isKeyDown("right"))
        {
            move(7);
        }
    }
}
```

El player puede desplazarse usando las flechas del teclado hacia la izquierda y derecha a una velocidad predeterminada por el método moverJugador().

Video mostrando la funcionalidad del código realizado:
https://youtu.be/-jlvW_POP2M