Evaluación Paradigmas de programación 24/9

Descripción de los escenarios

Newton Lab1:

En este escenario encontramos cuatro clases “Space”, “SmoothMover” , “Body” y “vector”. La clase Space tiene cuatro métodos el primero “removeAllObjects” remueve todo lo que hay en pantalla, el segundo método se llama “sunAndPlanet” crea objetos (un sol y un planeta) tipo Body en una posición con su tamaño, fuerza, dirección, color y gravedad y el tercero “sunAndTwoPlanet” crea objetos tipo Body (un sol y dos planetas) y el cuarto “sunPlanetMoon” también crea objetos tipo Body pero en este caso un sol un planeta y una luna. Dentro de la clase Body no se implementa el método move por lo tanto los elementos permanecen en la posición inicial indicada. La clase vector al ser inicializada pide la fuerza y la dirección, obteniendo la posición en el eje cartesiano, además existen métodos para actualizar la dirección, actualizar la fuerza, revertir la dirección horizontal, revertir la dirección vertical, retornar el valor de la posición actual en el “x” e “y”, retornar el valor fuerza, actualizar la posición en “x” e “y”, sin embargo toda esta información no se utiliza porque le faltan métodos a la clase Body para poder implementarla. La clase “SmoothMover” al ser creada crea un nuevo objeto tipo vector y se le asigna una velocidad, calcula posición en “x” e “y” donde tiene que ubicarse dada una velocidad y su posición actual, el método “getExactX” retorna la posición exacta de “x” y el método “getExactY” retorna la posición exacta de “y”, también existe un método para agregar velocidad y un método para invertir la posición. Y la clase Body contiene un método para retornar la masa y tiene definida una gravedad por defecto de 5.8 y un color amarillo también por defecto.

Newton Lab2:

A diferencia del Newton Lab1 en la clase Space el método “removeAllObjets” si bien funcionan igual está expresado de diferente manera, para Newton Lab1: removeObjects (getObjects(Actor.class)); y para Newton Lab2 removeObjects (getObjects(null)); . “aplyGravity” simula la gravedad basando se en la masa y velocidad de los objetos, “aplyForce” llama al método “aplyGravity” llama al método gravedad y se lo aplica a todos los objetos tipo Body sobre Space. En este escenario en la clase Body si se aplica el método move por lo que los cuerpos tienen movimiento. Las demás clases permanecen igual.

Newton Lab3:

En este escenario se crea una clase Obstacle la cual se compone de una variable String relacionada con el sonido, una variable boolena para saber si fue tocado por un objeto body o no y los métodos son “bodyAct” pregunta si fue tocado por un cuerpo, en caso de ser

verdadero llama a un método llamado “playSong” el cual reproduce un sonido asociado a un vector, también cambia la imagen durante un instante. En la clase Body se agrega el método “bounceAtEdge” el cual consulta si el objeto tipo Body se encuentra en el extremo izquierdo o derecho de la pantalla, en este caso cambia su direccio en vertical y se acelera, y en el caso de estar en el extremo superior o inferior también cambia su dirección pero horizontalmente, también es acelerado. Al principio de la clase Space se crea un vector asociando los 14 obstáculos a crear con su respectivo sonido, se crean dos nuevos métodos reemplazando métodos anteriores, el método “createObstacle” el cual con un ciclo while crea objetos de tipo Obstacle con su correspondiente asociación a los sonidos, luego con el método “randomBodies” crea 5 objetos con tamaño, posición, dirección, velocidad y color al azar. Al principio de la clase se crea un vector asociando los 14 obstáculos a crear con su respectivo sonido

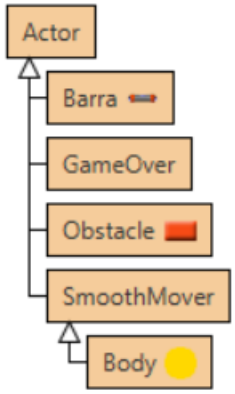
CONCLUCIÓN

Lo que trata de hacer Newton Lab es brindar la información necesaria (velocidad, dirección, masa, gravedad, etc.) para representar la fuerza gravitatoria de los cuerpos en el espacio y cómo actúan estos.

Modificación significativa

Modificamos el esenario Newton Lab3 y lo convertimos en una versión propia del juego arkanoid, que consiste en eliminar los obstáculos (se eliminan al ser tocados por la pelota) e intentar que la pelota no toque el “suelo”. Aprovechando el movimiento de la clase Body hicimos que el cuerpo (pelota) con una dirección al azar y una velocidad por defecto rebote en las paredes laterales y la superior y en la plataforma (Barra), en este caso invertimos la dirección verticalmente, mientras que si toca la pared inferior del mundo el juego termina, para esto agregamos la clase Barra la cual se puede mover con las flechas del teclado para la izquierda y derecha y la clase GameOver que nos permite mostrar un mensaje cuando termina el juego.


```
public void act()
{
    checkKeypress();
    // Add your action code here.
}
```




```
public void checkKeypress(){

    if (Greenfoot.isKeyDown("left"))
    {
        setLocation(getX()-4, getY());
    }
    if (Greenfoot.isKeyDown("right"))
    {
         setLocation(getX()+4, getY());
    }
}
```

También agregamos más objetos tipo obstáculo y modificamos su ubicación

```
public void createObstacles()
{
    int i = 0;
    while (i < 14)
    {
        addObject (new Obstacle (soundFiles + ".wav"), 80 + i*60, 110);
        addObject (new Obstacle (soundFiles + ".wav"), 80 + i*60, 180);
        i++;
    }
}
```

Creamos un objeto tipo Body con algunos datos al azar y otros fijos para lograr una mejor experiencia de juego

```
public void randomBodies(int number)
{
    while (number > 0)
    {
        int size = 30 ;
        double mass = size * 7.0;
        int direction = 180+ Greenfoot.getRandomNumber(150);
        double speed = 50; //Greenfoot.getRandomNumber(150) / 5.0
        int x = 480;
        int y = 550;
        int r =  Greenfoot.getRandomNumber(255);
        int g =  Greenfoot.getRandomNumber(255);
        int b =  Greenfoot.getRandomNumber(255);
        addObject (new Body (size, mass, new Vector(direction, speed), new Color(r, g, b)), x, y);
        number--;
    }
}
```

En el caso de que el objeto tipo Body toque algún objeto de la clase Obstacle esta reproduce un sonido y es removida. También aumenta un contador que al llegar a 28 reproduce un sonido que indica que el juego es ganado

```
public void lookForObstacle()
{
    if ( isTouching(Obstacle.class) )
    {
        removeTouching(Obstacle.class);
        Greenfoot.playSound("3b.wav");

        cont++;
        if (cont == 28)
        {
            Greenfoot.playSound("TaDa.wav");
            Greenfoot.stop();
        }
    }
}
public void lookForBarra()
{
    if ( isTouching(Barra.class) )
    {
        invertVerticalVelocity();
        accelerate(1.9);
    }
}
```

En el método bounceAtEdge chequea que estemos en un extremo derecho o izquierdo, en este caso invierte su dirección horizontalmente o que estemos en el extremos superior, en este caso invierte su dirección verticalmente

```
public void lookForObstacle()
{
    if ( isTouching(Obstacle.class) )
    {
        removeTouching(Obstacle.class);
        Greenfoot.playSound("3b.wav");

        cont++;
        if (cont == 28)
        {
            Greenfoot.playSound("TaDa.wav");
            Greenfoot.stop();
        }
    }
}
public void lookForBarra()
{
    if ( isTouching(Barra.class) )
    {
        invertVerticalVelocity();
        accelerate(1.9);
    }
}
```

En el caso que la pelota se encuentre en el extremo inferior crea una nueva instancia de la clase GameOver que indica que el jugador ha perdido y termina el juego. La imagen de Game Over es centrada en medio del mundo obteniendo la anchura de este y diviendola por 2 al igual que con la altura.

```
public GameOver(){
    setImage(new GreenfootImage("Game Over", 50, Color.GREEN,Color.BLACK));
}
```

```
private void bounceAtEdge()
{

    if ( isAtEdge()){
        if (getX() == 0 || getX() == getWorld().getWidth()-1) {
            invertHorizontalVelocity();
            accelerate(1.9);
        }
        else if (getY() == 0){
            invertVerticalVelocity();
            accelerate(1.9);
        }
        else if (getY() == getWorld().getHeight()-1){
            GameOver gamecver= new GameOver();
            int x=getWorld().getWidth()/2;
            int y=getWorld().getHeight()/2;
            getWorld().addObject(gameover,x,y);

            Greenfoot.stop();
        }
    }

}
```