Informe TP Programación 1

Alumnos: Tomás Montenegro, Christian Romano, Gastón Olivera

Mails:Tomymontenegro1999@hotmail.com,

Capitanmaverick1@gmail.com, gastoncarp2012@gmail.com

Universidad Nacional de General Sarmiento

Com: 1

Legajos: 2018/41874099, 2018/38616506, 40015570/2018

El juego del mago:

En este proyecto nosotros hicimos un juego en el cual un mago debía enfrentarse a los enemigos que quieren matarlo para escapar de la prisión, en el juego el mago solo puede moverse hacia la derecha e izquierda, no pude saltar y solo puede atacar a los enemigos cuando los golpea con su magia que solo puede lanzar al caer de una plataforma a la otra. Al impactar estos disparos, el enemigo queda congelado y el jugador tiene un corto periodo de tiempo para empujar al enemigo y matarlo.

Clase bala

Al crear la clase bala buscamos que el mago pudiera disparar cuando el mago choca con las plataformas entonces creamos el objeto bala, los métodos usados para esta clase son para la forma de la bala, el curso que toma y la interacción con los enemigos.

```
public double distanciaEnemigoDerecha (Enemigo  enemigo) {
    double finalEnemigo = enemigo.getX()+ enemigo.getAncho();
    double distancia = 0;
    distancia=(Math.pow(this.x-finalEnemigo,2))+Math.pow(this.y-enemigo.getY(),2);
    return distancia;
}
```

En el método distanciaEnemigoDerecha de la clase bala, usamos la fórmula de distancia para saber cuándo la bala está cerca del x del enemigo. Usamos 2 métodos uno que se fije la distancia del lado derecho del enemigo y otro del izquierdo.

Clase salud

La clase salud da al mago la cantidad de golpes que puede recibir antes de morir y aparece en la parte superior izquierda de la pantalla los métodos son para dibujarla y tiene un método para restar vida por cada ves que el mago choca con un enemigo

Clase plataforma

La clase plataforma crea las plataformas en las que el mago y los enemigos pueden moverse la clase tiene los métodos que arman las plataformas y la dibuja

Clase paredes

La clase paredes dibuja la pared que están en la derecha y en la izquierda de la pantalla de juego.

Clase mago

La clase mago crea al mago que es el objeto que controlaremos en el juego para derrotar a los enemigos en esta clase se encuentran una gran parte de los métodos del juego como por ejemplo el método chocaconplataformas que hace que el mago pueda caminar sobre las plataformas, la función caída que hace caer al mago cuando ya no está en la plataforma, mago disparo que logra que el mago dispare cuando cae y choca con la siguiente plataforma

Clase enemigos:

En la clase enemigos se crearán los enemigos que el mago tiene que vencer tiene la misma función para que choquen con las plataformas que el mago, se mueven solos con los métodos caminarderecha y caminarizquierda para moverse, la función congelar para que cuando la bala lo toque el enemigo se quede congelado, el método chocarconparedes para cambiar la dirección cuando las toca y el método rodar para que ruede si el mago lo toca cuando está congelado

Problemas a la hora de hacer el juego:

Mago y las plataformas:

Uno de los primeros problemas que nos encontramos al comenzar fueron las plataformas porque el mago no interactuaba con las plataformas y caía, también el mago caminaba por estas, pero cuando caía se quedaba en el medio de las plataformas y no terminaba de caer, esto lo fuimos solucionando creando un array con todas las plataformas para que el mago camine por todas. El otro problema lo solucionamos corrigiendo el método choca con plataforma entre esos fue que al “Y” del mago le sumamos su altura y con eso el mago no atravesaba la plataforma.

Que el mago se mueva solo:

Al principio al mago lo movíamos con las teclas izquierda y derecha, pero más adelante tuvimos que hacer que se mueva solo así que cambiamos los métodos para que al presionar las teclas que nombre antes el mago solamente cambie la dirección luego creamos la función mover para que el mago se mueva automáticamente la función lo moverá a la derecha si la variable de instancia del objeto mago dirección es true y si es false hacia la izquierda las funciones de moverderecha y moverizquierda cambian la variable dirección.

El mago y los enemigos en ciertos momentos no se tocan:

Cuando el mago caía encima de un enemigo este no moría hasta que tocaba con el ancho del enemigo para eso creamos métodos extra para que el mago choque con los enemigos tanto en los costados, como cuando un enemigo le cae encima y cuando el mago cae sobre un enemigo. Utilizamos métodos que miden la distancia entre los enemigos y el mago para resolver este problema.

También tuvimos problemas al hacer que el mago cuando cae en la plataforma por que nos disparaba antes de caer o cuando caía y para resolverlo creamos los siguientes métodos:

```
public Bala disparar(Plataformas pisos[]) {

    if(chocaConPlataforma(pisos)&&direccion==false)
    {
        return new Bala(x, y, true);
    }
    else
    {
        return new Bala(x, y, false);
    }




}

public void desabilitarDisparo() {
    disparar = false;
}

public void habilitarDisparo(Plataformas pisos[]) {
    for (int i=0;i<pisos.length;i++){
        if(this.y < pisos[i].getY()- 80 && this.y > pisos[i].getY()-160)
        {   disparar = true;}
    }
    }
```

Y tuvimos que crear la variable de instancia disparar para saber cuándo el mago podía disparar y cuando no.

Congelar a los enemigos y empujarlos:

Lograr congelar a los enemigos fue una de las cosas que más nos costó. Creamos una variable de instancia en la clase enemigo para saber cuándo el enemigo estaba congelado y cuando no, de esta manera podíamos controlar cuando queríamos que el mago muera al tocar al enemigo y cuando queríamos que solo lo empuje. Al principio lo podíamos congelar a los enemigos, pero era por tiempo infinito. Tuvimos que agregar también una variable de instancia tiempocong para saber cuánto tiempo pasaba el enemigo congelado y otra variable rueda para saber cuándo el enemigo está rodando o no (un boolean).

En la clase mago:

```
public boolean muerte(Plataformas plataformaMedio, Enemigo enemigo[], Entorno e,Plataformas pisos[]) {


    for(int i=0;i<enemigo.length;i++)
    {
        if(chocaConEnemigo(enemigo[i])&&enemigo[i].getCongelar()==0)
        {

            enemigo[i].Maxvel();
            enemigo[i].estaRodandoa();



        }


    }

    if (chocaConenemigos(enemigo) ) {
        this.x =400;
        this.y =0;
        this.desabilitarDisparo();
        for(int i=0;i<enemigo.length;i++)
            enemigo[i].Descongelar();
```

Para saber si el mago moría tuvimos que agregar condiciones para saber si el enemigo que choca estaba congelado o no y aumentarle la velocidad para hacer que se vaya hacia el agujero y muera.

Invariantes de representación.

Clase juego

```
this.saludvacia[0]=new Salud(50,20);
this.saludvacia[1]=new Salud(105,20);
this.saludvacia[2]=new Salud(160,20);
this.pared1 = new Paredes(0, entorno.alto() / 2, entorno.alto(), 50);
this.pared2 = new Paredes(entorno.ancho(), entorno.alto() / 2, entorno.alto(), 50);
this.pisos[0] = new Plataformas(entorno.ancho() / 2, 120, 500);
this.pisos[1] = new Plataformas(entorno.ancho() - 100, 250, 300);
this.pisos[2] = new Plataformas(100, 250, 300);
this.pisos[3] = new Plataformas(entorno.ancho() / 2, entorno.alto() / 2, 400);
this.pisos[4] = new Plataformas(entorno.ancho() / 2, entorno.alto() / 2 + 150, 500);
this.pisos[5] = new Plataformas(entorno.ancho() - 100, entorno.alto() - 100, 250);
this.pisos[6] = new Plataformas(100, entorno.alto() - 100, 250);
this.mago = new Mago(pisos[0].getX(), pisos[0].getY() / 3);
this.enemigos[0] = new Enemigo(pisos[6].getX(), pisos[6].getY() +7);
this.enemigos[1] = new Enemigo(pisos[1].getX(), pisos[1].getY()+25 );
this.enemigos[2] = new Enemigo(pisos[2].getX(), pisos[2].getY() / 7);
this.enemigos[3] = new Enemigo(pisos[5].getX(), pisos[5].getY() / 3);
this.enemigos2=new Enemigo(pisos[1].getX(),pisos[1].getY()+25);
this.Gameover=Herramientas.cargarImagen("gameover.jpg");
```

El X y el Y del mago al comienzo no puede ser otro porque queremos que empieza en la plataforma superior y si lo hacemos aparecer en otra plataforma podría chocarse con algún enemigo.

Las coordenadas de las paredes y los pisos tienen que valer sus “x” entre 0 y entorno.ancho y sus “y” entre 0 y entorno.alto. Lo mismo que las coordenadas de los magos, si no aparecerían fuera de los pisos y del entorno del juego.

Clase mago

```
public class Mago {
    int cantidadHP;
    double x;
    int y;
    int ancho;
    int alto;
    Bala ataque;
    private Image imagen;
    boolean direccion;
    private boolean disparar;
    int cantMuertes;
    int cantSalud;
    Salud [] salud;

    public Mago(int x, int y) {
        this.x = x;
        this.y = y;
        this.ancho = 10;
        this.alto = 50;
        this.imagen = Herramientas.cargarImagen("magoImagen.png");
        this.direccion = false;
        this.disparar = false;
        cantMuertes = 0;
        cantSalud = 3;
        salud = new Salud [3];
        salud[0]=new Salud(50,20);
```

La variable disparar tiene que ser false en principio.

La altura del mago no puede ser mayor a 50 porque podríamos tener problemas al chocar con la plataforma y no la detectaría.

La cantidad de muertes tiene que ser un número entero mayor a igual o mayor a 0 o igual o menor a 3.

El x del mago tiene que ser un numero entre 0 y el entorno.ancho.

Salud tiene que ser un array de un largo de 3.

El y del mago tiene que ser un numero entre 0 y el entorno.alto.

Dirección puede valer true o false, no importa para el lado que empieza a disparar.

Clase bala

```
this.ancho = 10;
this.alto = 5;
this.disparo = Herramientas.cargarImagen("ataque.png");
this.x = x;
this.y =y;

this.direccion=direccion;
```

El alto de la bala no puede ser mayor al alto del enemigo.

El x de la bala no puede ser mayor a entorno. Ancho.

El y de la bala no puede ser mayor a entorno. Alto.

La dirección de la bala tiene que ser la misma para donde está caminando el mago.

Clase salud

```
this.barra = Herramientas.cargarImagen("Corazonlleno.png");
this.x = x;
this.y = y;
this.ancho = 20;
this.alto = 20;
this.corazonVacio=Herramientas.cargarImagen("corazonvacio.png");
this.saludvacia = false;
```

Saludvacia no puede ser al principio del juego true por que se terminaría el juego apenas empieza.

El x de salud tiene que ser un entero mayor o igual a 0 y menor o igual a entorno.ancho.

El y de salud tiene que ser un entero mayor o igual a 0 y menor o igual a entorno.alto.

Clase Enemigo

```
private int x;
private int y;
private int ancho;
private int alto;
private double angulo;
private boolean direccion;
private int congelar=1;
private double velocidad;
private double tiempocong;
private Image imagen;
private boolean rueda;
```

El y del enemigo no puede ser mayor a entorno.alto ni menor a 0 y el x no puede ser mayor a entorno.ancho y tampoco menor a 0.

El alto del enemigo no puede ser un número mayor a 50 o menor a 0 por que tendríamos problemas al caminar en la plataforma, el enemigo flotaría.

El ancho del enemigo no puede ser un entero mayor al plataforma.getAncho() / 2.

Velocidad como máximo puede valer 15, si le ponemos un número muy alto el enemigo desaparece.

Rueda tiene que comenzar como false.

Congelar puede tener como valor 1 y 0. 1 cuando no esta congelado el enemigo y 0 cuando si esta congelado. Tiene que comenzar valiendo 1.

TiempoCongelacion tiene que ser un entero mayor a 1.

Clase Plataforma

```
private int x;
private int y;
private int ancho;
private int alto;
```

El x de la plataforma tiene que ser un entero entre 0 y entorno.ancho.

El y de la plataforma tiene que ser un entero entre 0 y entorno.alto.

El ancho tiene que ser un entero menor a entero.ancho/2 para que el mago y los enemigos puedan caer en algún momento.

El alto tiene que ser un entero mayor a 0 y menor a entorno.alto.

Conclusión

En conclusión, tuvimos muchos problemas para realizar este trabajo por los tiempos de cada integrante y nos costó mucho avanzar en el trabajo, mayormente al momento de hacer que el mago dispare cuando pisa las vigas, fue lo que más se nos dificulto. Pero pudimos solucionar los problemas que se nos presentaron y logramos incorporar todos los requerimientos obligatorios que se nos pidió para el trabajo.