17 DE ABRIL DE 2018

JUEGO EN GREENFOOT

CARLOS GARRIDO JUNCO
IES BARAJAS

ÍNDICE:

ANÁLISIS:

INTRODUCCIÓN:

El juego consiste en el típico planet shouter que cada vez que se matan enemigos saldrán más y más rápido y la única defensa que tendremos serán los asteroides.

PLANTEAMIENTO DEL PROBLEMA:

Los principales problemas son: las colisiones con otros objetos o actores, el manejo de disparo, cómo recoger la puntación del héroe y su vida .Además del diseño de niveles, actores y cómo tratar la música.

OBJETIVOS:

El objetivo es aprender a utilizar un motor gráfico como Greenfoot y manejar las colisiones y eventos.

TECNOLOGÍAS:

Paint Tool Sai:

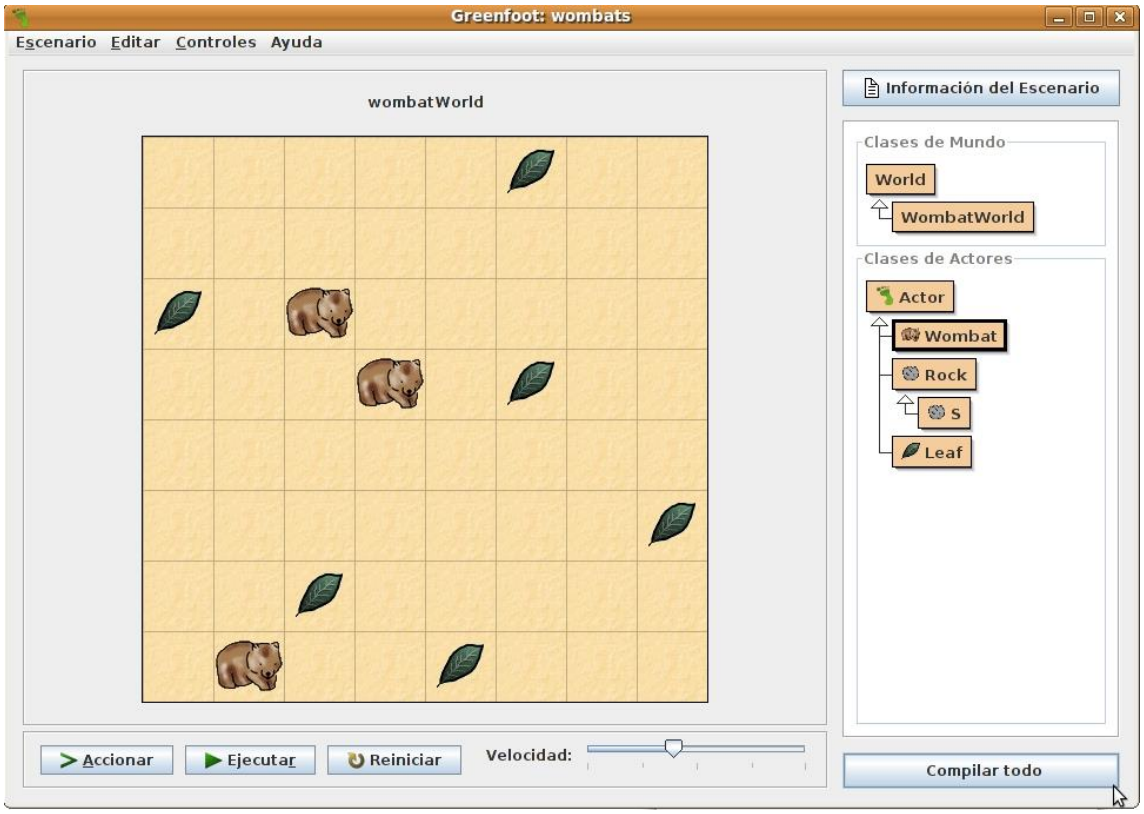
Paint Tool SAI (nombre oficial: PaintTool SAI) es la herramienta de dibujo para PC, que nos permite pintar mediante capas. La cual junto a una tableta gráfica he usado para pintar los escenarios y los sprites.

Greenfoot:

Es un motor gráfico para desarrollar aplicaciones gráficas, en dos dimensiones, en java con propósitos educacionales.

A diferencia de Alice que también es otro motor gráfico para desarrollar juegos en java, este no tiene un entorno sencillo de arrastrar y soltar sino que hay utilizar el editor de texto para colocar nuestros objetos y darles una funcionalidad.

Además, el programa divide los objetos en escenario y en los actores que interactúan en el escenario.

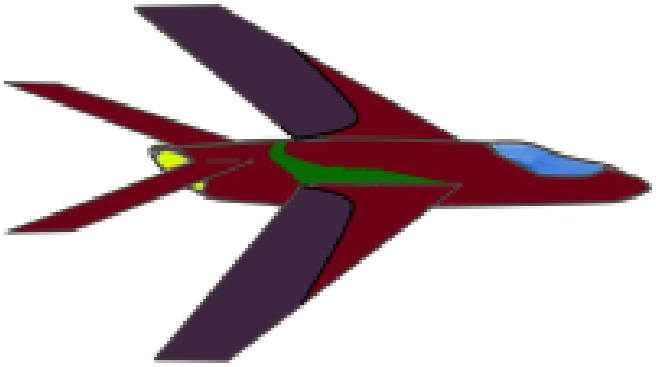


DISEÑO:

IMÁGENES:

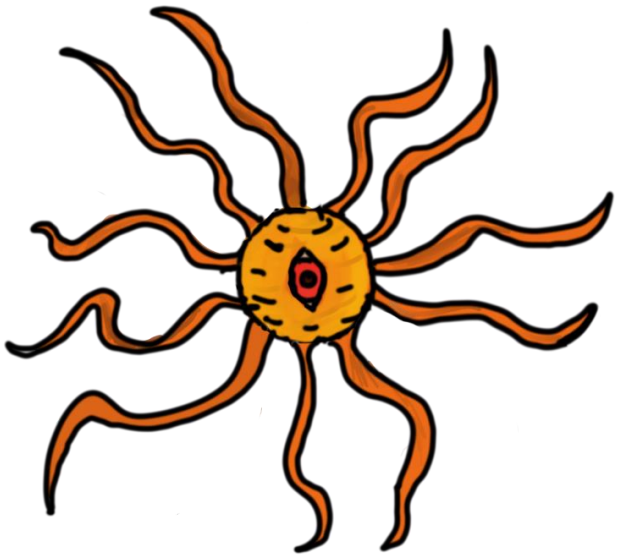
Todas las imágenes han sido dibujados han sido dibujadas a mano utilizando Paint tool sai:

NAVE:

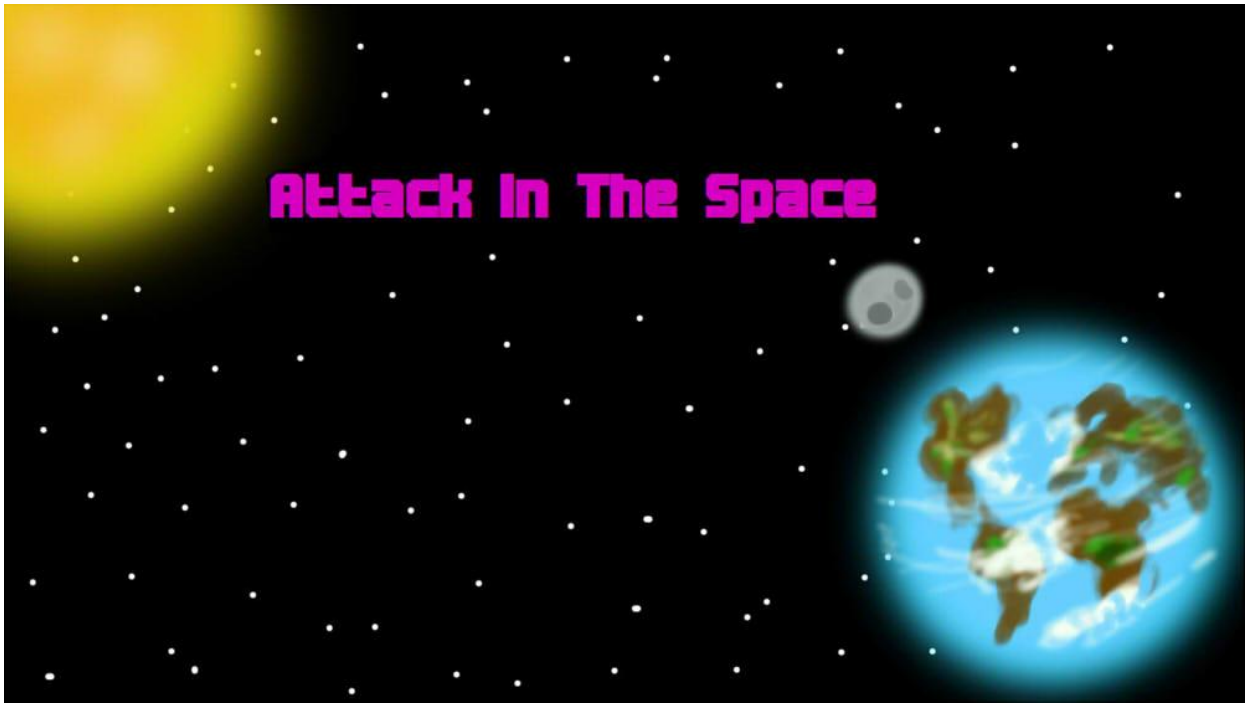


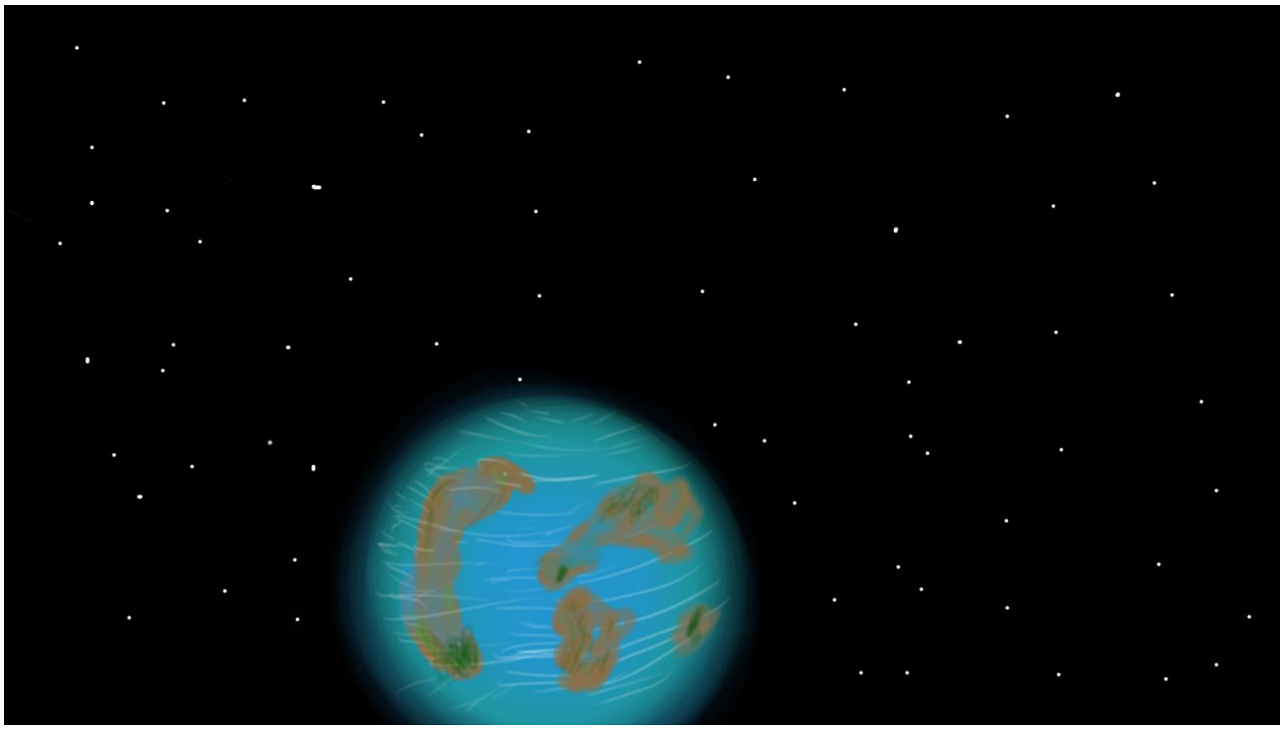
ASTEROIDES:

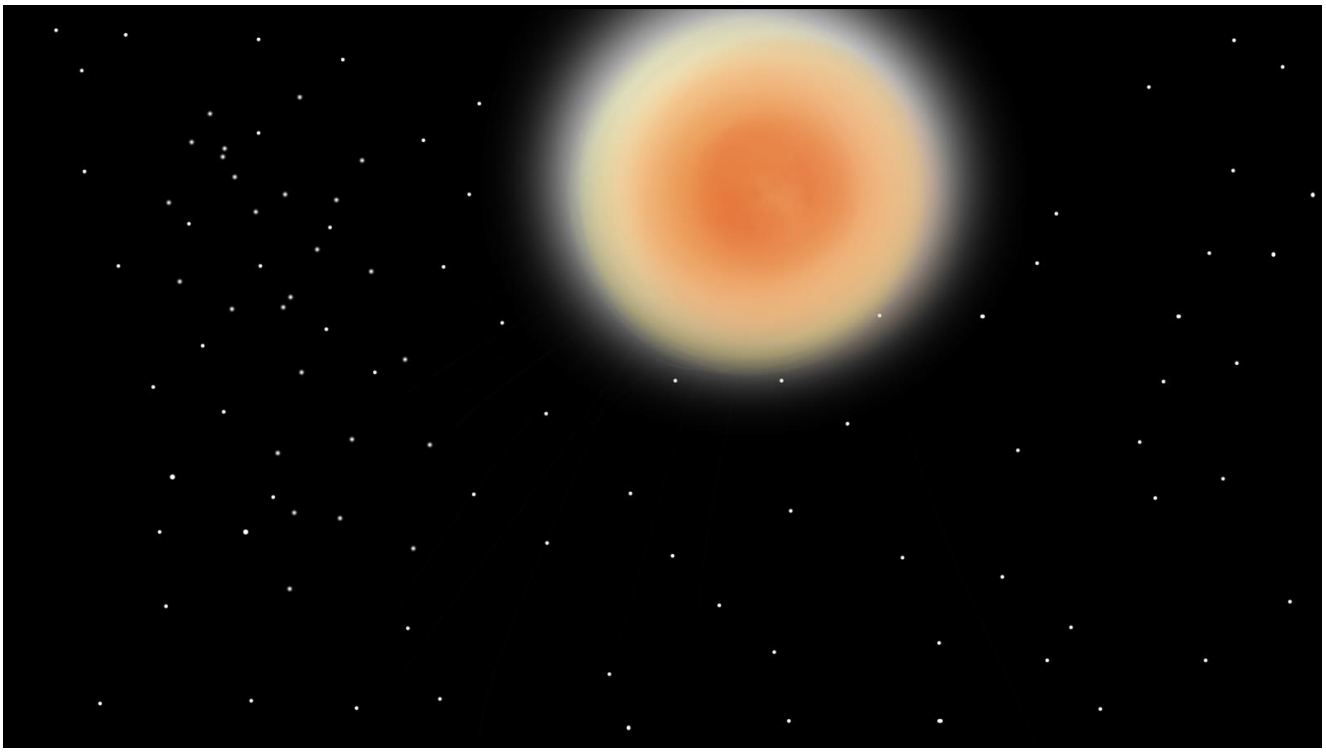
ENEMIGOS:

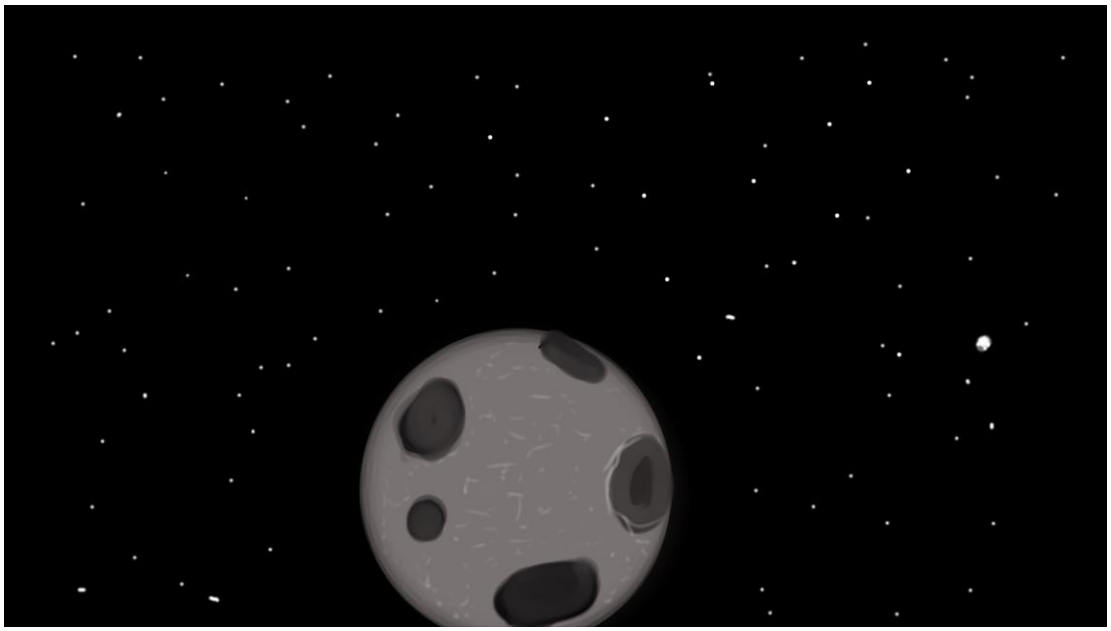


ESCENARIOS:









IMPLEMENTACIÓN:

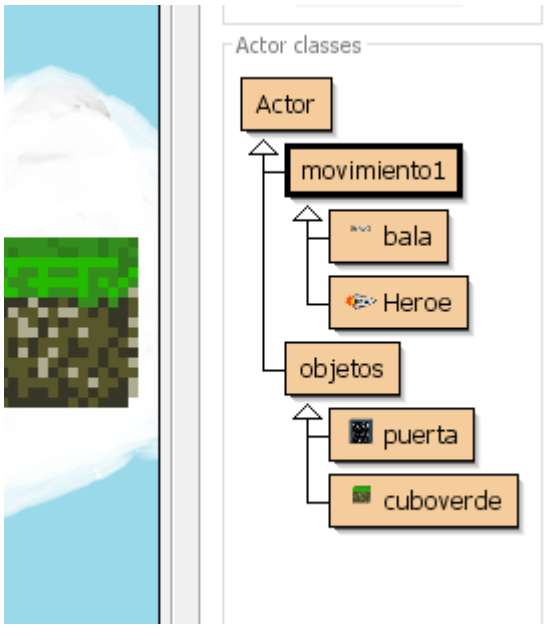
MOVIMIENTO DEL HEROE:

Capturamos los eventos de las teclas con el método isKeyDown() y utilizamos el setLocation para movernos.

```
public void controles()
{
    if(vida<0|| vida == 0) {
        getWorld().removeObject(this);
        mimumdo.inimundo(false);
    }else{
        if(Greenfoot.isKeyDown("D")) {
            move(4);
        }
        if(Greenfoot.isKeyDown("A")) {
            move(-4);
        }
        if(Greenfoot.isKeyDown("W")) {
            setLocation(getX(), getY()-2);
        }

        if(Greenfoot.isKeyDown("S")) {
            setLocation(getX(), getY()+2);
        }
        if(Greenfoot.isKeyDown("F")){
            if(carga<5)
                carga++;
            gravedad(gravedad);
            if(carga==5){
                carga=0;
                Greenfoot.playSound("disparo.mp3");

                getWorld().addObject(mibala, getX(), getY());

            }
        }
```

EFECTO HACIA ATRÁS:

Para hacer el efecto hacia atrás, he creado una subclase primero para que todos los actores con movimiento hereden de ella y así vayan por defecto hacia la izquierda:




```
public class movimiento1 extends Actor
{

    boolean suelo;
    public void act()
    {

    }
       public void derecIzq(int movimiento)
    {
        if(movimiento==-1){
            setLocation(getX()-1, getY());
        }
        if(movimiento==2){
            setLocation(getX()+4, getY());
        }
        if(movimiento==-2){
            setLocation(getX()-4, getY());
        }
    }

}
```

El parámetro hace referencia si el héroe va hacia la izquierda o hacia la derecha.

DISPARO:

En esta clase tengo que comprobar que el objeto no está en la esquina y no ha colisionado con otro para que se mueva porque si es al revés, es decir se mueve y luego comprueba si ha colisionado o ha llegado al borde del mapa y se borra, da error y la aplicación se para porque al volver a recorrer act() el método moviendo() necesita una instancia del objeto la cual no estará porque se habrá borrado y dará el error.

```
if(isAtEdge()){
    getWorld().removeObject(this);

}else{

    if(miamalgama!=null){
        miamalgama.setvida(10);
        getWorld().removeObject(this);
    }else{
        if(actor!=null){
            getWorld().removeObject(this);
        }
        else{
            moviendo();
        }
    }
}
```

Luego en la clase donde se ejecuta el dispara utilizo un entero que va servir para que haya una pausa entre bala y bala sin tener que parar el hilo principal.

```
else if(Greenfoot.isKeyDown("F")){
    if(carga<5)
        carga++;
    if(carga==5){
        carga=0;
        Greenfoot.playSound("disparo.mp3");
        getWorld().addObject(new bala(20), getX(), getY());
    }
}
```

COLISIÓN CON LOS BLOQUES:

Sobrescribir el método setLocation por el cual si encuentra un objeto cuboverde se queda en la posción en la que está llamando al setlocation padre y si no lo encuentra será como un setLocation normal.

```
public void setLocation(int x, int y)
{
    int oldX = getX();
    int oldY = getY();
    super.setLocation(x, y);
    if(!getIntersectingObjects(cuboverde.class).isEmpty())
    {
        super.setLocation(oldX, oldY);
        suelo=true;
    }
}
```

ENEMIGOS:

Los enemigos se mueven mientras su vida sea mayor que cero y no están al principio del eje x y si los enemigos chocan con nuestro héroe este es borrado, para ello he utilizado polimorfismo.

```
public void movimiento(){
    Actor heroe=(Actor)getOneIntersectingObject(Heroe.class);

    if(getX()==0||vida==0||vida<0){
        getWorld().removeObject(this);
    }else{
        if(heroe!=null){
            getWorld().removeObject(heroe);

        }
        else{
            if(isAtEdge()){
                mover=mover*-1;
                setLocation(getX(), getY()+mover);
            }

            setLocation(getX(), getY()+mover);
            gravedad(-2);

        }

    }
}
```

He utilizado if anidados dentro de los else porque si no, ocurre un exception de Nullpoint exception por intentar utilizar un objeto que ha sido borrado.

He actualizado la clase bala para que cuando toque a un enemigo este le disminuye la vida y desaparezca la bala al hacer contacto con este:

```
public class bala extends movimientol
{
    int dir=1;
    // boolean disparo
    public bala(int dir){
        this.dir=dir;
    }

    public void act()
    {
        Actor actor = getOneIntersectingObject(cuboverde.class);
        Actor actor1=getOneIntersectingObject(amalgama.class);
        amalgama miamalgama=(amalgama)actor1;

        if(isAtEdge()){
            getWorld().removeObject(this);
        }else{
            if(miamalgama!=null){
                miamalgama.setvida(10);
                getWorld().removeObject(this);
            }else{
                if(actor!=null){
                    getWorld().removeObject(this);
                }
                else{
                    moviendo();
                }
            }
        }
```

GENERACIÓN DE ENEMIGOS Y BLOQUES EN EL MUNDO DE FORMA DINÁMICA:

Hay dos formas de generar nuestros objetos, la primera es estática es decir antes de dar a run en el constructor de la clase World se carga los objetos mediante el método:

```
addObject(new cuboverde(),getWidth()-Greenfoot.getRandomNumber(100), Greenfoot.getRandomNumber(getHeight()));
```

Que es de la propia clase World.

Para hacerlo de una forma dinámica, es decir que se vaya instanciando con la ejecución del juego sobrescribimos el método act() de la misma clase que es el que se va ejecutar de forma periódica y metemos:

```
public void act(){

    if(empezar){
        music.playLoop();
    }else
        music.stop();
    if(getObjects(amalgama.class).isEmpty()){
        ponerenemigos();
    }
    if(getObjects(cuboverde.class).isEmpty()){
      ponerbloques();
    }

}
```

En este caso para probarlo, por cada vuelta de bucle comprueba e sí hay enemigos o cubos en el caso en que no haya llama un

```
public void ponerbloques(){
    for(int i=0;i<5;i++){

        addObject(new cuboverde(),getWidth()-Greenfoot.getRandomNumber(100), Greenfoot.getRandomNumber(getHeight()));
    }
}
```

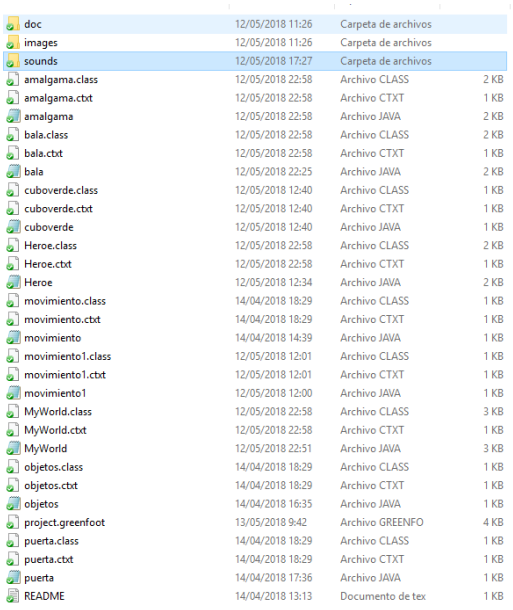
```
public void ponerenemigos(){
    for(int i =0 ;i<10 ;i++){
        if(i%2==0){
            modulo=-1;
            addObject(new amalgama(modulo),getWidth()-Greenfoot.getRandomNumber(100), Greenfoot.getRandomNumber(getHeight()));
        }
        else{
            modulo=1;
            addObject(new amalgama(modulo),getWidth()-Greenfoot.getRandomNumber(100), Greenfoot.getRandomNumber(getHeight()));
        }
    }
}
```

método que los genera:

El módulo del método de poneremigos() es para pasarlo a la clase amalgama en su contructor para que salga en distintas direcciones para ello he utilizado la función módulo.

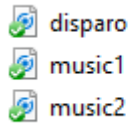
MÚSICA O EFECTOS DE SONIDO:

Para introducir música o efectos de sonido lo introducimos en la carpeta del proyecto dentro de sounds:



doc	12/05/2018 11:26	Carpeta de archivos	
images	12/05/2018 11:26	Carpeta de archivos	
sounds	12/05/2018 17:27	Carpeta de archivos	
amalgama.class	12/05/2018 22:58	Archivo CLASS	2 KB
amalgama.ctxt	12/05/2018 22:58	Archivo CTXT	1 KB
amalgama	12/05/2018 22:58	Archivo JAVA	2 KB
bala.class	12/05/2018 22:58	Archivo CLASS	2 KB
bala.ctxt	12/05/2018 22:58	Archivo CTXT	1 KB
bala	12/05/2018 22:25	Archivo JAVA	2 KB
cuboverde.class	12/05/2018 12:40	Archivo CLASS	1 KB
cuboverde.ctxt	12/05/2018 12:40	Archivo CTXT	1 KB
cuboverde	12/05/2018 12:40	Archivo JAVA	1 KB
Heroe.class	12/05/2018 22:58	Archivo CLASS	2 KB
Heroe.ctxt	12/05/2018 22:58	Archivo CTXT	1 KB
Heroe	12/05/2018 12:34	Archivo JAVA	2 KB
movimiento.class	14/04/2018 18:29	Archivo CLASS	1 KB
movimiento.ctxt	14/04/2018 18:29	Archivo CTXT	1 KB
movimiento	14/04/2018 14:39	Archivo JAVA	1 KB
movimiento1.class	12/05/2018 12:01	Archivo CLASS	1 KB
movimiento1.ctxt	12/05/2018 12:01	Archivo CTXT	1 KB
movimiento1	12/05/2018 12:00	Archivo JAVA	1 KB
MyWorld.class	12/05/2018 22:58	Archivo CLASS	3 KB
MyWorld.ctxt	12/05/2018 22:58	Archivo CTXT	1 KB
MyWorld	12/05/2018 22:51	Archivo JAVA	3 KB
objetos.class	14/04/2018 18:29	Archivo CLASS	1 KB
objetos.ctxt	14/04/2018 18:29	Archivo CTXT	1 KB
objetos	14/04/2018 16:35	Archivo JAVA	1 KB
project.greenfoot	13/05/2018 9:42	Archivo GREENFO	4 KB
puerta.class	14/04/2018 18:29	Archivo CLASS	1 KB
puerta.ctxt	14/04/2018 18:29	Archivo CTXT	1 KB
puerta	14/04/2018 17:36	Archivo JAVA	1 KB
README	14/04/2018 13:13	Documento de tex	1 KB

Y metemos la música o los efectos de sonido en formato mp3.



Luego en la clase World, sobrescribimos el método stopped() y started() para cuando se ejecute el juego se inicie la música de fondo.

```
public void started(){
    empezar=true;

}

public void stopped()
{
    empezar = false;
    music.stop();
}
```

Utilizo una variable boleana para iniciar la ejecución de la música en el act()

```
public void act(){

    if(empezar){
        music.playLoop();
    }else
        music.stop();
    if(getObjects(amalgama.class).isEmpty()){
        ponerenemigos();
    }
    if(getObjects(cuboverde.class).isEmpty()){
      ponerbloques();
    }

}
```

Para utilizar los métodos playLoop() y stop() tengo que primero crear un objeto de la clase GreenfootSound

```
int modulo=5;
GreenfootSound music = new GreenfootSound("music2.mp3");
public MyWorld()
```

PUNTUACIÓN:

Para hacer la puntuación, he tenido que crear una clase que contenga un array con las imágenes, una variable int que tendrá su

setter y su getter y con esa variable pondré la imagen que este en el array.

```
public class Puntos extends Valores
{
    int puntos=0;
    String[] imagenes={"0.png","1.png","2.png","3.png","4.png","5.png","6.png","7.png","8.png","9.png"};
    public void act()
    {
            setImage(imagenes[puntos]);

    }
    public int getpuntos(){
        return puntos;
    }
    public void sumapuntos(int nuevospuntos){
        puntos= nuevospuntos;
    }
}
```

Como se puede ver, ahora solamente podemos tener una puntuación del 0-9 y en el momento en que tengamos 10 o más no va a dar una excepción, para solucionar esto, en nuestra clase principal creamos cuatro objetos de la clase punto que van a representar las unidades, decenas, centenas, millares:

```
Puntos unidades= new Puntos();
Puntos decenas= new Puntos();
Puntos centenas = new Puntos();
Puntos millares= new Puntos();
```

Y las añadimos en el mundo:

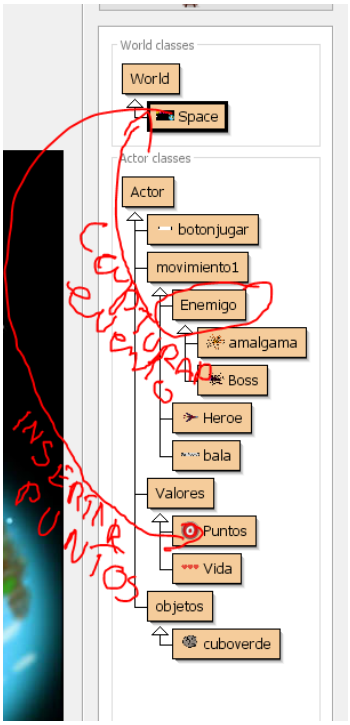
```
addObject(unidades, getWidth()-40,50);
addObject(decenas, getWidth()-80,50);
addObject(centenas, getWidth()-120,50);
addObject(millares, getWidth()-160,50);
```

Y también una variable punto de tipo int que inicializamos a cero:

```
int puntos =0;
```

Esta variable la utilizaremos como contador cuando matemos a un enemigo y luego la descompondremos en unidades de millar, centenas, decenas y unidades y se lo introduciremos a los setter del objeto punto para que cambien las imágenes de los números

```
public void insertarnum( int sumpuntos){
    puntos = puntos+sumpuntos;

    int punmillares=puntos/1000;
    millares.sumapuntos(punmillares);

    int puncentenas=(puntos-(punmillares*1000))/100;
    centenas.sumapuntos(puncentenas);

    int pundecenas=(puntos- (punmillares*1000 + puncentenas*100))/10;
    decenas.sumapuntos(pundecenas);

    int pununidades=puntos-(punmillares*1000 + puncentenas*100 + pundecenas*10 );
    unidades.sumapuntos(pununidades);
}
```



Ahora el problema es el siguiente: cuando muere un enemigo , la clase space (es la clase principal donde se desarrolla el bucle principal y donde se inserta los actores) no sabe que ha muerto por lo cual no se puede capturar el evento y suma al contador por lo cual tenemos que capturarlo en la misma clase enemigo, pasarle la puntuación y eliminar el enemigo, y luego este valor utilizando el método insertnum() que está en la clase principal, se envía los puntos a las instancias de la clase punto para refrescar la imagen:

Ahora para llamar insertarnum() desde la clase Enemigo, tenemos el problema de que no podemos instanciarla en la clase enemigo con el operador new y llamar al método porque al hacerlo estaríamos creando una copia, de la clase principal, por lo cual la puntuación sería enviada a la copia y no a la principal, y también al no ser insertarnum() y la variable puntos static (se comparten en todas las instancias) porque nos interesa utilizar esta variable en métodos como act() que no son static. Para ello en la clase enemigo declaramos una variable de tipo Space llamada mimundo:

```
private Space mimumdo;
```

Y sobrescribimos un método propio de Greenfoot con lo cual conseguimos llamar a la clase principal sin tener que crear una nueva instancia:

```
public void addedToWorld(World world) {
    mimumdo = (Space) world;
}
```

Y ya con esto podemos utilizar insertarnum() desde la clase enemigo cuando muera:

```
if(vida==0||vida<0){
    mimumdo.insertarnum(1);
    getWorld().removeObject(this);
```

VIDA DEL HEROE:

Para la vida del héroe tenemos que hacer algo parecido que, con la puntuación, primero tenemos que crear una clase vida que va a tener lo siguiente:

```
public class Vida extends Valores
{
    int vida=3;
    public void act()
    {
        if(vida==0||vida<0){
            getWorld().removeObject(this);
        }
        else{
            if(vida==3){
                setImage("corazon3.png");
            }
            if(vida==2){
                setImage("corazon2.png");
            }
            if(vida==1){
                setImage("corazon1.png");
            }
        }
    }

    public void restarvida(){
        vida=vida-1;
    }

    public void restablecer(){
        vida=3;
    }
}
```

Para cuando vaya cambiado el contador la imagen vaya cambiando y vaya disminuyendo los corazones.

Ahora creamos nuestro objeto Vida en la clase Space:

```
Vida vidaheroe = new Vida();
```

Y un método que devuelva la vida, para que podamos obtener el objeto desde la clase Heroe para así tener accesos a sus métodos:

```
public Vida muestravida()
{
    return vidaheroe;
}
```

Luego en la clase Heroe creamos un objeto Vida donde almacenará lo que devuelve el método de muestravida() y también creamos una variable de la clase Space.

```
Vida  vida2;
```

```
private Space mimundo;
```

Sobrescribiremos el siguiente método para cargar la variable mimundo:

```
public void addedToWorld(World world) {
    mimundo = (Space) world;
}
```

Ahora ya podemos cargar el objeto vida en la variable vida2:

```
vida2=mimumdo.muestravida();
```

Y actualizar el objeto vida

```
public void  dano(int resta){

    this.vida= vida-resta;
    vida2.restarvida();

}
```

BOTÓN INICIO:

Ahora creamos el botón para iniciar el nivel y el autogenerado del mundo y los objetos, además tendremos que programarlo para que cuando nuestro héroe muera que se vuelva añadir en el mundo el botón.

Para ello vamos a utilizar una variable booleana que activará un if en el cual se iniciará el nivel, además de llamar a otro método que instanciará los objetos y cambiará el fondo de pantalla.

Se inicializa la variable con false.

```
boolean mundo1=false;
```

El método inimundo() el cual se llamará desde nuestra clase jugar:

```
public  void inimundo(boolean valor) {
    mundol=valor;
    if(valor){
        empezar=true;
        miheroe= new Heroe();
        vidaheroe.restablecer();
        addObject(unidades, getWidth()-40,50);
        addObject(decenas, getWidth()-80,50);
        addObject(centenas, getWidth()-120,50);
        addObject(millares, getWidth()-160,50);
        addObject(vidaheroe, 150,50);
        addObject(miheroe, 50,100);
    }
    else{
        empezar=false;
        music.stop();
        ronda=1;
        puntos =0;
        removeObject(unidades);
        removeObject(decenas);
        removeObject(centenas);
        removeObject(millares);
        removeObject(vidaheroe);
        removeObjects(getObjects(cuboverde.class));
        removeObjects(getObjects(Enemigo.class));
        addObject(jugar, getWidth()/2-20 ,getHeight()/2 +100
        background= new GreenfootImage("fondocarga.jpg");
        setBackground(background);

    }
}
```

Nuestra variable booleana afectará en el act:

```
public void act(){
    if(mundol){
        buclemundol();
    }
}
```

Y ahora en la clase de nuestro botón

```
import greenfoot.*;  // (World, Actor, GreenfootImage, Greenfo

/**
 * Write a description of class botonjugar here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class botonjugar extends Actor
{
    private Space mimundo;
    public void act()
    {
        if(Greenfoot.mouseClicked(this)){
            mimundo.inimundo(true);
            getWorld().removeObject(this);

        }
    }

    public void addedToWorld(World world) {
        mimundo = (Space) world;
    }
}
```

Hacemos como en los anteriores pasos llamamos a la clase principal y usamos el método inimundo() en el cual se cambiará también nuestra variable booleana.

Ahora para hacer el efecto contrario, añadir la pantalla de inicio y el botón, tenemos que ir a nuestra clase héroe, en el método controles() y llamar a nuestro método inimundo pero con el parámetro false para cambiar la variable booleana mundo1.

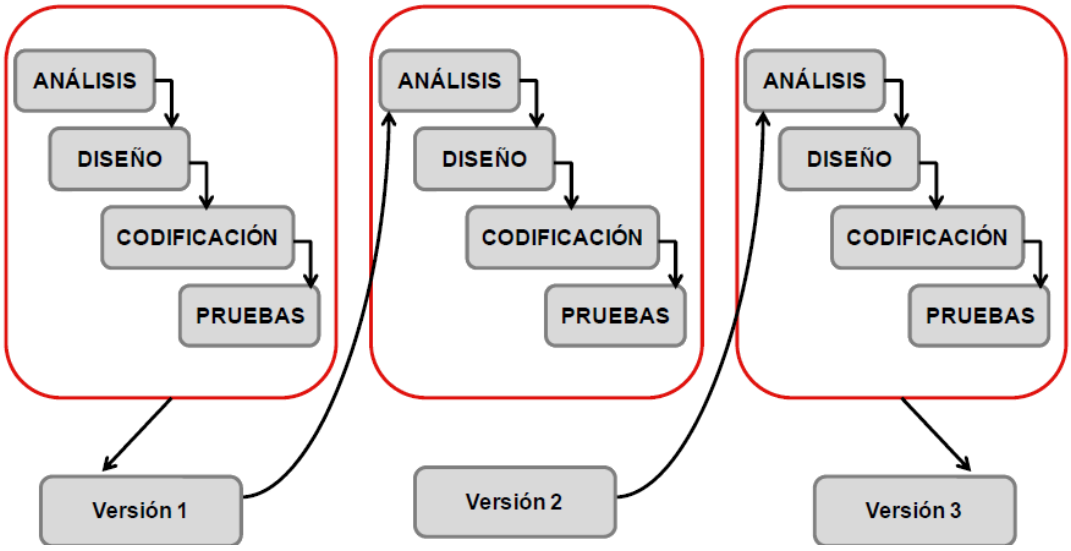
CAMBIO DE ESCENARIO:

Para cambiar de escenario he utilizado números aleatorios generados con la clase Random:

```
public  void inimundo(boolean valor) {
    mundo1=valor;
    if(valor){
        empezar=true;
        miheroe= new Heroe();
        vidaheroe.restablecer();
        addObject(unidades, getWidth()-40,50);
        addObject(decenas, getWidth()-80,50);
        addObject(centenas, getWidth()-120,50);
        addObject(millares, getWidth()-160,50);
        addObject(vidaheroe, 150,50);
        addObject(miheroe, 50,100);
        //insertar fondo aleatorio cada vez que se pulsa el botón
        int i = numal.nextInt(3);
        background= new GreenfootImage("escenario"+i+".png");
        setBackground(background);

    }
    else{
        empezar=false;
        music.stop();
        ronda=1;
        puntos =0;
        insertarnum(0);
        removeObject(unidades);
        removeObject(decenas);
        removeObject(centenas);
        removeObject(millares);
        removeObject(vidaheroe);
        removeObjects(getObjects(cuboverde.class));
        removeObjects(getObjects(Enemigo.class));
        removeObjects(getObjects(bala.class));
        addObject(jugar, getWidth()/2-20 ,getHeight()/2 +100);
        background= new GreenfootImage("fondocarga.jpg");
        setBackground(background);

    }
}
```

PRUEBAS:

Las pruebas se han ido realizando cada vez que hemos implementado código de forma de prueba y error ya que hemos utilizado un ciclo de vida incremental e iterativa.



CONCLUSIÓN:

Este proyecto me ha servido para aprender a cómo desarrollar un juego utilizando Greenfoot que a pesar de ser un entorno con propósitos educacionales, los conocimientos adquiridos los puedo extrapolar a entornos más profesionales como Unity o Unreal. Además de aprender a utilizar Paint tool sai para la creación de sprites y escenario utilizando capas.