Universidad Autónoma de San Luis Potosí

Área de Computación e Informática
Ingeniería en informática

Programación Orientada a Objetos
Grupo: 222302

Contra Survive
No. Proyecto: 138

Integrantes:
Anguiano García Alfredo
Juárez Jalomo Ana Paola

Semestre 2016-2017/I

Fecha de entrega.: Lunes 5 de diciembre de 2016

Objetivo del proyecto

Es un juego de plataformas en el cual el objetivo es juntar las llaves que aparezcan en cada nivel en el menor tiempo posible ya que teniendo todas aparecerá una puerta para así pasar de nivel.

El jugador deberá de buscar hacer el menor tiempo posible para que su record aparezca en la pantalla.

Descripción del proyecto

Es un juego de plataformas en el juego aparecerán llaves en cada nivel ya que teniendo todas aparecerá una puerta para así pasar de nivel.

Existirán 3 niveles, en cada uno habrá una diferente dificultad, también un Enemigo común y a su vez tendrán un Jefe y un enemigo especial del nivel.

Personaje Principal:

El personaje se moverá con las teclas y saltará

Enemigo Común:

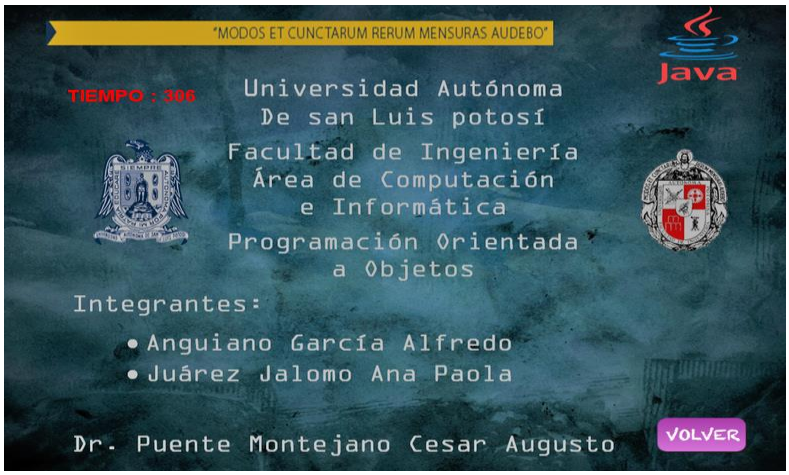
Este enemigo estará en los 3 niveles y su función será de solo moverse de forma horizontal y si el personaje principal llegará a tocarlo y así se le reducirá la vida al personaje.

Descripción e imágenes de cada nivel

El entorno principal del proyecto aparecerá como este.

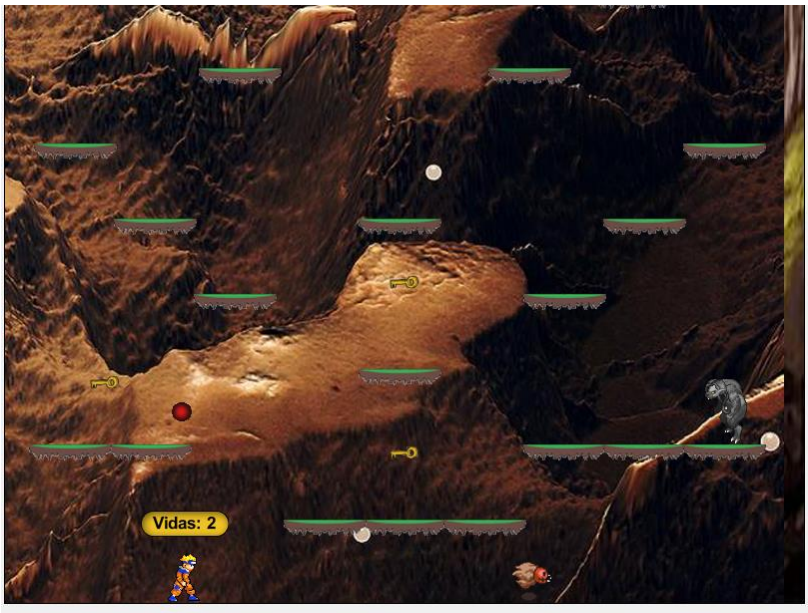
En el cual dentro de la ayuda se encontrarán los comandos para jugar, al igual que una pequeña descripción del objetivo del juego.

También está el botón de Créditos, él se mostrará una pequeña portada con los nombres de los integrantes que realizaron este proyecto, al igual que los records, en el record no se mostraran puntos ya que el objetivo del juego es solo recolectar las llaves el que menos tiempo haga aparecerá en los records.



- ## Nivel Principiante

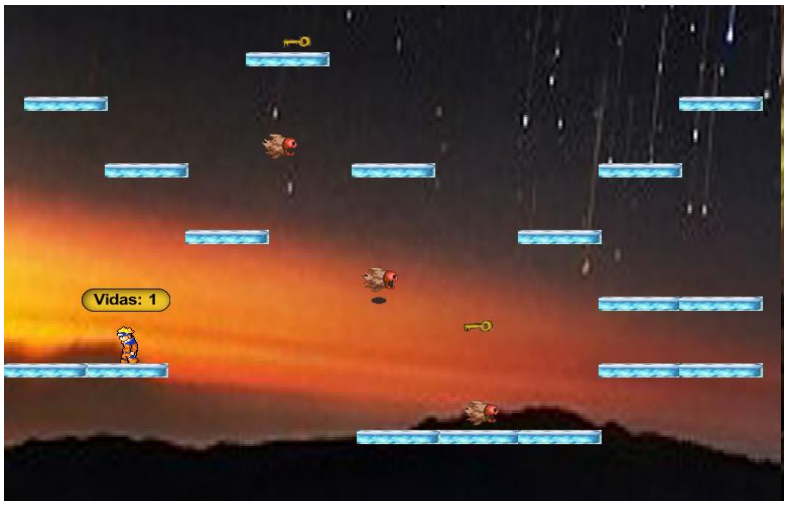
  Total, de llaves a encontrar: 2.
  Enemigo del Nivel: tendrá un enemigo el cual estará en una plataforma y disparará para que el jugador no pase.
  Si el jugador llega a subir a la parte superior de este nivel podrá ver al jefe el cual comenzará a disparar.



- ## Nivel Intermedio

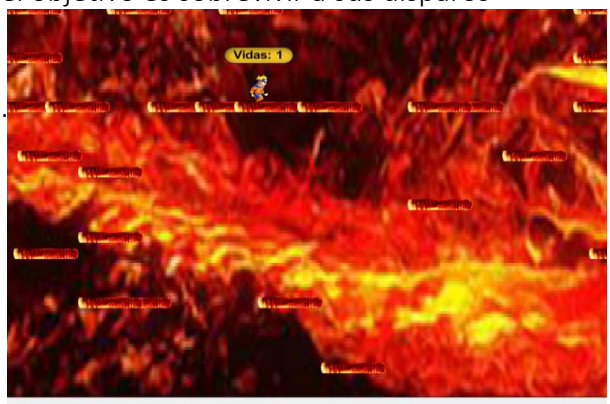
Escenario Cielo: En este escenario se encontrará un personaje que recorrerá el nivel de forma horizontal en alguna de las plataformas.

Total, de llaves a encontrar: 3.
Enemigo Especial del Nivel: La función de este enemigo es que estará volando en una posición de X fija y tendrá un movimiento vertical y a su vez deparará (se detiene y dispara cierto tiempo).



- ## Nivel Avanzado

Escenario de Fuego: En este escenario el personaje principal avanzará de forma horizontal y cuando finalice ese punto lo hará de forma vertical. Cuando se llegue con el jefe aparecerá la última llave en una de las plataformas cerca del jefe y el objetivo será recolectarla para terminar el juego.
Total, de llaves a encontrar: 4.
Jefe Fuego o dragón: El jefe es la última prueba del nivel, estará en la última plataforma y el objetivo es sobrevivir a sus disparos



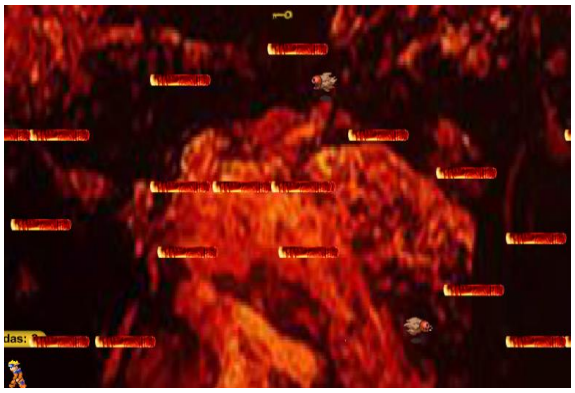
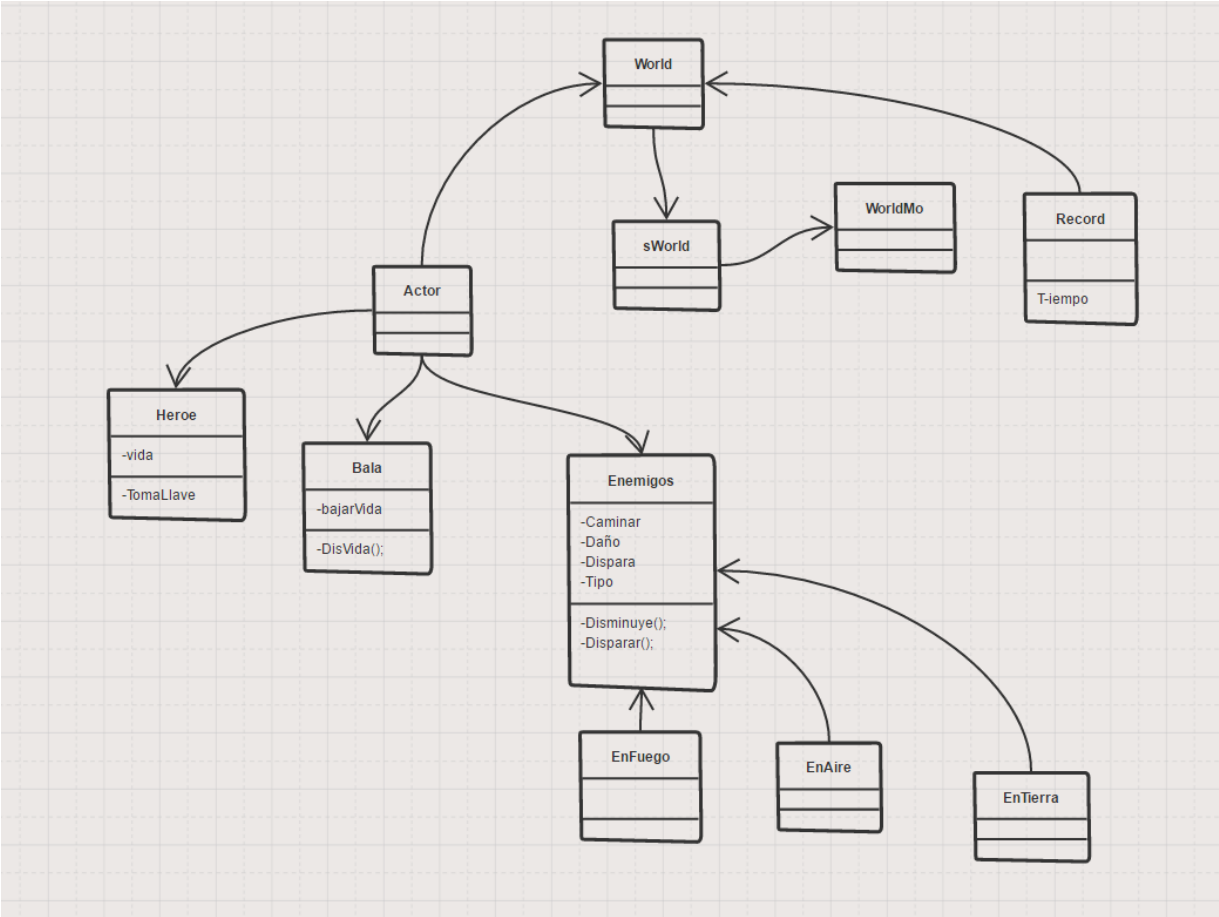


Diagrama de clases UML



Características y comportamiento de cada clase

Nombre de la clase:	Enemigo
Características:	El enemigo es malo
Comportamiento:	El enemigo evitara que el héroe avance
	Podrá disparar y moverse por el escenario
	Existen diferentes enemigos

Nombre de la clase:	Héroe
Características:	Trata de sobrevivir contra los enemigos
Comportamiento:	El héroe podrá moverse por todo el escenario
	Se moverá horizontalmente y puede saltar
	Su objetivo es conseguir todas las llaves

Nombre de la clase:	Timer
Características:	Mantiene un tiempo para el juego
Comportamiento:	Se encarga de dar un tiempo el cual los objetos van a seguir
	Mantiene el orden y secuencia de actividades

Herencia y polimorfismo

La herencia y el polimorfismo se ve reflejada dentro de las clases de enemigo y héroe para poder pasar atributos y características a sus descendientes, como el tipo de enemigos y los pies en el caso del héroe.
También se puede observar dentro de la clase plataformas para poder diferenciar los diferentes tipos de plataformas que existen dentro del juego.

Cronograma de actividades (plan de trabajo)

- Manual del usuario:

  Requisitos:

  - Instalar GreenFoot
  - Dependiendo de la versión de greenfoot que se haya

descargado, se compilará o no el proyecto por sí solo, una vez hecho esto se tendrá que inicializar el juego presionando el botón "Run".

y es así como se debe iniciar el juego.

- Código

Class SWorld

```
java.lang.Object

      greenfoot.World

          SWorld
public class SWorldextends greenfoot.World
```

Class SWorld es una super clase para hacer un mundo con scroll(Horizontal,Vertical, o ambos).

Author: danpost
Version: October 28, 2013 (v2.0)

Para implementar la Clase padre:

- (1) Crear una clase hijo de SWorld
- (2) Llamar en el constructor de la sub clase el metodo *super(....)*
- (3) Crear un Actor principal (El personaje principal que se va a ver) y llamar al metodo *setMainActor*
- (4)(Opcional) Establecer una imagen fondo scroll-able llamando al metodo *setScrollingBackground* o *fillScrollingBackground*

Nota: El order de los pasos es importante

Se encuentran dos metodos para agregar otros objetos en el mundo:

- El metodo estandar *addObject(Actor,int,int)* puede ser usado para agregar un scroll-able actor en el mundo
- Un metodo secundario *addObject(Actor,int,int,boolean)* que es equivalente al metodo estandar, pero el parametro booleano indicara el estado scroll-able del objeto. Cuando son agregados los objetos con scroll en el mundo, se usaran las

coordenadas scroll-ables, cuando se agregan objetos no scroll-ables, se usan las variables del mundo.

Nota especial:si tu decides sobre escribir el metodo "act" de esta clase con el metodo "act" de la clase hijo, vas a necesitar seguir el formato siguiente para correr el escenario scroll-able:

```
public void act()
{
    // Codigo
    super.act();
    // Codigo
}
```

SWorld

```
public SWorld(int wide,
              int high,
              int cellSize,
              int scrollWide,
              int scrollHigh)
```

The constructor for a universal scroller. Creates an unbounded world and sets the size of the scrollable area.

Parameters:

`wide` - the window width

`high` - the window height

`cellSize` - the size of each cell

`scrollWide` - the scrollable width (minimum value is window width)

`scrollHigh` - the scrollable height (minimum value is window height)

Method Detail

act

```
public void act()
```

Runs the scrolling.

Overrides:

`act` in class `greenfoot.World`

---

addMainActor

```
public void addMainActor(greenfoot.Actor main,
                         int xLoc,
                         int yLoc,
                         int xRange,
                         int yRange)
```

Adds the main actor into the world at the center of the window.

NOTE: this method must be called prior to calling *setScrollingBackground*. Sets the range in movement within the window for the actor, and determines the range of horizontal and vertical scrollable movement allowable for the actor.

Parameters:
`main` - the actor that is to always stay in view
`xLoc` - the x-coordinate of the scrolling area to place the main actor
`yLoc` - the y-coordinate of the scrolling area to place the main actor
`xRange` - the horizontal range of movement within the window
`yRange` - the vertical range of movement within the window

---

addObject

```
public void addObject(greenfoot.Actor obj,
                      int xLoc,
                      int yLoc)
```

Adds a scrollable object into the world, listing them in the Actor array.
Overrides:
`addObject` in class `greenfoot.World`
Parameters:
`obj` - the scrollable object to add to the world
`xLoc` - the x-coordinate of the scrolling area to place the object
`yLoc` - the y-coordinate of the scrolling area to place the object

---

addObject

```
public void addObject(greenfoot.Actor obj,
                      int xLoc,
                      int yLoc,
                      boolean scroller)
```

Adds an object into the world, listing it in the Actor array if it is a scrollable object; the coordinates are of the scrolling area for scrolling objects and of the world if not.
Parameters:
`obj` - the object to add to the world
`xLoc` - the x-coordinate to place the object
`yLoc` - the y-coordinate to place the object
`scroller` - a flag indicating whether this object is of scrollable type or not

---

fillScrollingBackground

```
public void fillScrollingBackground(greenfoot.GreenfootImage fillImage)
```

Fills the background of the scrolling area with the *fillImage*.

NOTE: for this method to work, the main actor must have previously been set with

setMainActor. The image will then be used to fill the background of the scrolling area and is centered in the scrollable world.

Parameters:

`fillImage` - the image to fill the background of the scrolling area with

---

getScrollingHeight

`public int` **`getScrollingHeight`**`()`

Returns the height of the scrolling area of the universe

Returns:

the height of the visible scrolling area

---

getScrollingWidth

`public int` **`getScrollingWidth`**`()`

Returns the width of the scrolling area of the universe

Returns:

the width of the visible scrolling area

---

getUnivX

`public int` **`getUnivX`**`(int worldX)`

Returns the horizonal offset from the top-left corner of the scrolling world of the 'x' value given, where 'x' is the horizontal offset from the top-left corner of the view window.

Parameters:

`worldX` - a horizontal offset from the top-left corner of the visible world

Returns:

the absolute horizontal offset from the top-left corner of the scrolling world

---

getUnivY

`public int` **`getUnivY`**`(int worldY)`

Returns the vertical offset from the top-left corner of the scrolling world of the 'y' value given, where 'y' is the vertical offset from the top-left corner of the view window.

Parameters:

`worldX` - a vertical offset from the top-left corner of the visible world

Returns:

the absolute vertical offset from the top-left corner of the scrolling world

---

removeObject

`public void` **`removeObject`**`(greenfoot.Actor obj)`

Removes an object from the world, re-defining fields as neccessary

Overrides:

`removeObject` in class `greenfoot.World`

Parameters:

`obj` - the object to be removed from the world

---

removeObjects

`public void` **`removeObjects`**`(java.util.List<greenfoot.Actor> objs)`

Removes a collection of objects from the world, calling *removeObject(Actor)* for each one in the list

Parameters:

`objs` - the collection or list of objects to be removed from the world

---

setScrollingBackground

`public void`
`setScrollingBackground``(greenfoot.GreenfootImage scrollingBackground)`

Adds a scrolling background to the world; see method description for notes on unwanted results.

NOTE: for this method to work, the main actor must have previously been set with *setMainActor*. The image will then be scaled to the appropriate size and is centered in the scrollable world.

Parameters:

`scrollingBackground` - the image to be used for the scrolling background of the world

---

Class WorldMo

```
java.lang.Object
      greenfoot.World
          SWorld
              WorldMo
```

`public class` **`WorldMo`**`extends SWorld`

Crea el los escenarios del juego de acuerdo al nivel .

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

WorldMo

`public WorldMo()`
Constructor for objects of class WorldMo.

Method Detail

act

`public void act()`
Description copied from class: `SWorld`
Runs the scrolling.
Overrides:
`act` in class `SWorld`

---

addBottom

`public void addBottom()`
Añade los botones del Menu

AddPLvl2

`public void AddPLvl2()`

AddPLvl3

`public void AddPLvl3()`

AgregaKey

`public void AgregaKey(int nKey)`
Se generan las llaves acorde a una posicion.

---

ayuda

```
public void ayuda()
```
Añade la imagen de la ayuda.

---

creditos

```
public void creditos()
```
Añade la imagen de los creditos.

---

getImagen

```
public greenfoot.GreenfootImage getImagen(int n)
```
Regresa el numero de imagen.

GetNivel

```
public int GetNivel()
```
Regresa el numero del nivel.

GetTime

```
public int GetTime()
```

IniTime

```
public void IniTime()
```

---

nivel1

```
public void nivel1()
```
Crea el nivel 1 del juego.

---

nivel2

```
public void nivel2()
```
Crea el nivel 2 del juego.

---

nivel3

```
public void nivel3()
```

Crea el nivel 3 del juego.

---

seleccionar

```
public void seleccionar()
```

Verifica cual boton se selecciona en el Menu.

Class Ball

```
java.lang.Object
    greenfoot.Actor
        Ball
```

```
public class Ballextends greenfoot.Actor
```

Es la bola que usa el enemigo de tipo Tiera.

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Ball

```
public Ball()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Ball wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

Class Rayo

```
java.lang.Object
```

```
    greenfoot.Actor
```

```
        Rayo
```

```
public class Rayoextends greenfoot.Actor
```

Es el disparo que usa el Jefe de tipo Cielo.

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Rayo

```
public Rayo()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Rayo wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

Class Ej

```
java.lang.Object
```

```
    greenfoot.Actor
```

```
        Ej
```

```
public class Ejextends greenfoot.Actor
```

Esta clase ayuda a verificar los extremos para el enmeigo Tierra

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Field Detail

bandEj

```
public static int bandEj
```

---

numX

```
public static int numX
```

Constructor Detail

Ej

```
public Ej()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Bonificacion wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

---

cambiabnd

```
public void cambiabnd()
```

GetBandEj

```
public int GetBandEj()
```

Bandera de la clase

SetRandoomPos

`public void` **`SetRandoomPos`**`()`

Hace que el enemigo fuego aparezca en posiciones aleatorias

Class Enemigo

```
java.lang.Object
```

```
    greenfoot.Actor
```

```
        Enemigo
```

`public class` **`Enemigo`**`extends greenfoot.Actor`

En esta clase es donde los enemigos adquieren elmétodo de caminar horizontal y dodne se hacen verificaciones acorde al tipo de enemigo.

Version:

(1.1 Beta)

Author:

Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Field Detail

Clado

`protected int` **`Clado`**

Tlado

`protected int` **`Tlado`**

Constructor Detail

Enemigo

`public` **`Enemigo`**`(java.lang.String file)`

Method Detail

getClado

`public int` **`getClado`**`()`

una bandera para verificar con el enemigo Cielo

---

getTlado

```
public int getTlado()
```

una bandera para verificar con el enemigo Tierra

---

move

```
public void move(java.lang.String archivo,
                 int tamX,
                 int tamY,
                 java.lang.String imaIzq,
                 java.lang.String imaDer,
                 int numSpr,
                 int limIzq,
                 int limDer)
```

Efectua el movimiento del enemigo respecto a sus limites

RevisaVista

```
public void RevisaVista(java.lang.String cadenaI,
                        java.lang.String cadenaD)
```

Verifica la vista que tiene el enemigo

---

spriteNormal

```
public void spriteNormal(java.lang.String archivo,
                         int tamX,
                         int tamY,
                         java.lang.String imaIzq,
                         java.lang.String imaDer,
                         int numSpr)
```

Hace el recorrido de los Sprites

Class Cielo

```
java.lang.Object

    greenfoot.Actor

        Enemigo

            Cielo
```

```
public class Cieloextends Enemigo
```

Crea la clase de los enemigos Tipo Tierra con sus caracteristicas

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Cielo

```
public Cielo(int Izq,
             int Der)
```

Method Detail

act

```
public void act()
```

Act - do whatever the EnemigoT wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

GetactLado

```
public int GetactLado()
```

VerifDisp

```
public void VerifDisp()
```

Class Comun

```
java.lang.Object

    greenfoot.Actor

        Enemigo
```

Comun

```
public class Comunextends Enemigo
```

Crea la clase de los enemigos Tipo Comun con sus caracteristicas

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Comun

```
public Comun(int Izq,
             int Der)
```

Method Detail

act

```
public void act()
```

Act - do whatever the booL wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

Class JefeF

```
java.lang.Object

      greenfoot.Actor

          Enemigo

              JefeF
```

```
public class JefeFextends Enemigo
```

Write a description of class JefeF here.

Constructor Detail

JefeF

```
public JefeF(int Izq,
             int Der)
```

Method Detail

act

```
public void act()
```

Act - do whatever the JefeT wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

LanzaFu

```
public void LanzaFu()
```

Hace que el Jefe lance rocas con un cierto angulo

Class Fuego

```
java.lang.Object

    greenfoot.Actor

        Enemigo

            Fuego
```

```
public class Fuegoextends Enemigo
```

Crea la clase de los enemigos Tipo Fuego con sus caracteristicas

Version:

(1.1 Beta)

Author:

Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Fuego

```
public Fuego(int limI,
             int limD)
```

Method Detail

act

```
public void act()
```

Act - do whatever the Fuego wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

Class JefeT

```
java.lang.Object

      greenfoot.Actor

          Enemigo

              JefeT
```

```
public class JefeTextends Enemigo
```

Crea la clase del Jefe del nivel Tipo Tierra con sus caracteristicas

Version:

(1.1 Beta)

Author:

Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

JefeT

```
public JefeT(int Izq,
             int Der)
```

Method Detail

act

```
public void act()
```

Act - do whatever the JefeT wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

LanzaRk

```
public void LanzaRk()
```

Hace que el Jefe lance rocas con un cierto angulo

Class Tierra

```
java.lang.Object
    greenfoot.Actor
        Enemigo
            Tierra
```

```
public class Tierraextends Enemigo
```

Crea la clase de los enemigos Tipo Cielo con sus caracteristicas

Version:

(1.1 Beta)

Author:

Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Tierra

```
public Tierra(int Izq,
              int Der)
```

Method Detail

act

```
public void act()
```

Act - do whatever the EnemigoC wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

---

addM

```
public void addM()
```

Class Texto

```
java.lang.Object

    greenfoot.Actor

        Texto
```

```
public class Textoextends greenfoot.Actor
```

Write a description of class Mensaje here.

Constructor Detail

Texto

```
public Texto(java.lang.String UnTexto,
             java.awt.Color UnColor)
```

Constructor de la clase Mensaje

Method Detail

actualizaImagen

```
public void actualizaImagen()
```

Metodo que actualiza la imagen del mensaje

---

ponTexto

```
public void ponTexto(java.lang.String texto)
```

Metodo que actualiza el texto del mesaje

Class Boton

```
java.lang.Object
    greenfoot.Actor
        Boton
```

```
public class Botonextends greenfoot.Actor
```

Inicializa las imagenes de los botones.

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Boton

```
public Boton(greenfoot.GreenfootImage ima)
```

Class barrera

```
java.lang.Object
    greenfoot.Actor
        barrera
```

```
public class barreraextends greenfoot.Actor
```

Write a description of class barrera here.

Constructor Detail

barrera

```
public barrera()
```

Method Detail

act

`public void` **`act`**`()`

Act - do whatever the barrera wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

Class barrV

```
java.lang.Object
      greenfoot.Actor
          barrera
              barrV
```

`public class` **`barrV`**`extends barrera`

Write a description of class barrT here.

Constructor Detail

barrV

`public` **`barrV`**`()`

Act - do whatever the barrT wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Method Detail

act

`public void` **`act`**`()`

Description copied from class: **`barrera`**

Act - do whatever the barrera wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `barrera`

Class barrH

```
java.lang.Object
    greenfoot.Actor
        barrera
            barrH
```

```
public class barrHextends barrera
```

Write a description of class barrH here.

Constructor Detail

barrH

```
public barrH()
```

Act - do whatever the barrH wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Method Detail

act

```
public void act()
```

Description copied from class: `barrera`

Act - do whatever the barrera wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `barrera`

Class Counter

```
java.lang.Object
    greenfoot.Actor
        Counter
```

```
public class Counterextends greenfoot.Actor
```

A Counter class that allows you to display a numerical value on screen. The Counter is an actor, so you will need to create it, and then add it to the world in Greenfoot. If you keep a reference to the Counter then you can adjust its value. Here's an example of a world class that displays a counter with the number of act cycles that have occurred:

```
class CountingWorld
{
    private Counter actCounter;

    public CountingWorld()
    {
        super(600, 400, 1);
        actCounter = new Counter("Act Cycles: ");
        addObject(actCounter, 100, 100);
    }

    public void act()
    {
        actCounter.setValue(actCounter.getValue() + 1);
    }
}
```

Version:
1.0

Author:
Neil Brown and Michael Kölling

Constructor Detail

Counter

```
public Counter()
```

Counter

```
public Counter(java.lang.String prefix)
```

Create a new counter, initialised to 0.

Method Detail

act

```
public void act()
```

Animate the display to count up (or down) to the current target value.

Overrides:

`act` in class `greenfoot.Actor`

---

add

```
public void add(int score)
```

Add a new score to the current counter value. This will animate the counter over consecutive frames until it reaches the new value.

---

getValue

```
public int getValue()
```

Return the current counter value.

---

move

```
public void move(int x,
                 int y)
```

---

setPrefix

```
public void setPrefix(java.lang.String prefix)
```

Sets a text prefix that should be displayed before the counter value (e.g. "Score: ").

---

setValue

```
public void setValue(int newValue)
```

Set a new counter value. This will not animate the counter.

Class Record

```
java.lang.Object

    greenfoot.Actor

        Record
```

```
public class Recordextends greenfoot.Actor
```

Write a description of class Archivo here.

Constructor Detail

Record

```
public Record()
```

Method Detail

comparaPuntaje

```
public void comparaPuntaje(int punt)
```

Metodo que se encarga de comparar si el puntaje que se ha hecho es mayor al highscore

---

guardaArchivo

```
public void guardaArchivo(java.lang.String arch,
                          int puntos)
```

Metodo que se encarga de guardar el archivo

---

leeArchivo

```
public int leeArchivo(java.lang.String arch)
```

Metodo que se encarga de leer el archivo

Class BFire

```
java.lang.Object

    greenfoot.Actor

        BFire
```

```
public class BFireextends greenfoot.Actor
```

Write a description of class Fire here.

Constructor Detail

BFire

```
public BFire()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Fire wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

Class Bomb

```
java.lang.Object

    greenfoot.Actor

        Bomb
```

```
public class Bombextends greenfoot.Actor
```

La bomba que usa el enemigo de tipo Cielo

Version:

(1.1 Beta)

Author:

Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Bomb

```
public Bomb()
```

Method Detail

act

`public void` **`act`**`()`

Act - do whatever the Bomb wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

Class BSky

`java.lang.Object`

`greenfoot.Actor`

`BSky`

`public class` **`BSky`**`extends greenfoot.Actor`

La bomba que usa el Jefe de tipo Cielo

Version:

(1.1 Beta)

Author:

Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

BSky

`public` **`BSky`**`()`

Method Detail

act

`public void` **`act`**`()`

Act - do whatever the BSky wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

Class Rock

`java.lang.Object`

`greenfoot.Actor`

`Rock`

```
public class Rockextends greenfoot.Actor
```

La bomba que usa el Jefe de tipo Tierra

Version:
(1.1 Beta)
Author:
Anguiano Garcia Alfredo,Juárez Jalomo Ana Paola

Constructor Detail

Rock

```
public Rock()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Rock wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

Class Plataforma

```
java.lang.Object

    greenfoot.Actor

        Plataforma
```

```
public class Plataformaextends greenfoot.Actor
```

Write a description of class Plataforma here.

Constructor Detail

Plataforma

```
public Plataforma()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Plataforma wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

---

selectImage

```
public void selectImage()
```

Class Plat2

```
java.lang.Object
    greenfoot.Actor
        Plataforma
            Plat2
```

```
public class Plat2extends Plataforma
```

Constructor Detail

Plat2

```
public Plat2()
```

Act - do whatever the Plat2 wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Method Detail

act

```
public void act()
```

Description copied from class: `Plataforma`

Act - do whatever the Plataforma wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `Plataforma`

Class Plat1

```
java.lang.Object
      greenfoot.Actor
          Plataforma
              Plat1
```

```
public class Plat1extends Plataforma
```

Constructor Detail

Plat3

```
public Plat3(int x,
             int y)
```

Act - do whatever the Plat1 wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Method Detail

act

```
public void act()
```

Description copied from class: `Plataforma`

Act - do whatever the Plataforma wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `Plataforma`

Class Plat3

```
java.lang.Object
      greenfoot.Actor
          Plataforma
              Plat3
```

```
public class Plat3extends Plataforma
```

Constructor Detail

Plat3

```
public Plat3(int x,
             int y)
```

Act - do whatever the Plat1 wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Method Detail

act

```
public void act()
```

Description copied from class: `Plataforma`

Act - do whatever the Plataforma wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `Plataforma`

Class Keyboard

```
java.lang.Object

    greenfoot.Actor

        Keyboard
```

```
public class Keyboardextends greenfoot.Actor
```

Constructor Detail

Keyboard

```
public Keyboard()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Keyboard wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:
`act` in class `greenfoot.Actor`

Class JefeC

```
java.lang.Object

    greenfoot.Actor

        JefeC
```

```
public class JefeCextends greenfoot.Actor
```

Constructor Detail

JefeC

```
public JefeC(int x,
             int y)
```

Method Detail

act

```
public void act()
```

Overrides:
`act` in class `greenfoot.Actor`

---

disparaBola

```
public void disparaBola()
```

Hace que las verificaciones de los disparos

---

mueve

```
public void mueve()
```

Hace que el Jefe se mueva de modo vertical

Class Heroe

java.lang.Object

greenfoot.Actor

Heroe

---

public class **Heroe**extends greenfoot.Actor

Field Detail

bandTE

protected static int **bandTE**

---

boom

protected static int **boom**

Hx

protected static int **Hx**

Hy

protected static int **Hy**

---

vlado

protected static int **vlado**

Constructor Detail

Heroe

public **Heroe**()

Method Detail

act

public void **act**()

Act - do whatever the Don wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.

Overrides:

`act` in class `greenfoot.Actor`

---

move

```
public void move()
```

---

regresaX

```
public int regresaX()
```

---

regresaY

```
public int regresaY()
```

---

verifExtremo

```
public void verifExtremo()
```

Aqui verificamos que el heroe se encuente dentro del area de juego y si no esta sobre algun objeto "solido", la gravedad lo aga caer bandSalto: es una bandera que sirve para que el personaje pueda saltar dependiendo donde se encuentre

---

verificaPos

```
public void verificaPos()
```

---

verificaT

```
public void verificaT()
```

Class Cuerpo

```
java.lang.Object
```

```
    greenfoot.Actor
```

Heroe

Cuerpo

---

public class **Cuerpo**extends Heroe

Constructor Detail

Cuerpo

public **Cuerpo**()

Method Detail

act

public void **act**()

Description copied from class: Heroe
Act - do whatever the Don wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
act in class Heroe

---

avanzaNivel

public void **avanzaNivel**()

---

disminuyeVidas

public void **disminuyeVidas**()

---

modificaBarrera

public void **modificaBarrera**()

---

move

public void **move**()

Overrides:
move in class Heroe

RevisaVista

```
public void RevisaVista()
```

---

spriteNormal

```
public void spriteNormal()
```

---

verifExtremo

```
public void verifExtremo()
```

Description copied from class: `Heroe`

Aqui verificamos que el heroe se encuente dentro del area de juego y si no esta sobre algun objeto "solido", la gravedad lo aga caer bandSalto: es una bandera que sirve para que el personaje pueda saltar dependiendo donde se encuentre

Overrides:

`verifExtremo` in class `Heroe`

---

verifKey

```
public int verifKey(int n)
```

Verifica la la cntidad de llaves que va recolectando el heroe

---

Class Vida

```
java.lang.Object

    greenfoot.Actor

        Vida
```

```
public class Vidaextends greenfoot.Actor
```

Constructor Detail

Vida

```
public Vida(int vidas,
            int vida)
```

Method Detail

act

```
public void act()
```

Overrides:
`act` in class `greenfoot.Actor`

Class Key

```
java.lang.Object
    greenfoot.Actor
        Key
```

```
public class Keyextends greenfoot.Actor
```

Constructor Detail

Key

```
public Key()
```

Method Detail

act

```
public void act()
```

Act - do whatever the Key wants to do. This method is called whenever the 'Act' or 'Run' button gets pressed in the environment.
Overrides:
`act` in class `greenfoot.Actor`

Class SimpleTimer

```
java.lang.Object
    SimpleTimer
```

```
public class SimpleTimerextends java.lang.Object
```

A simple timer class that allows you to keep track of how much time has passed between events. You use this class by creating a timer as a member field in your actor (or whatever):

```
private SimpleTimer timer = new SimpleTimer();
```

Then when you want to start the timer (for example, when a shot is fired), you call the mark() method:

```
timer.mark();
```

Thereafter, you can use the millisElapsed() method to find out how long it's been since mark() was called (in milliseconds, i.e. thousandths of a second). So if you want to only allow the player to fire a shot every second, you could write:

```
if (timer.millisElapsed() > 1000 && Greenfoot.isKeyDown("space"))
{
    // Code here for firing a new shot
    timer.mark(); // Reset the timer
}
```

Constructor Detail

SimpleTimer

```
public SimpleTimer()
```

Method Detail

mark

```
public void mark()
```

Marks the current time. You can then in future call millisElapsed() to find out the elapsed milliseconds since this mark() call was made. A second mark() call will reset the mark, and millisElapsed() will start increasing from zero again.

---

millisElapsed

```
public int millisElapsed()
```

Returns the amount of milliseconds that have elapsed since mark() was last called. This timer runs irrespective of Greenfoot's act() cycle, so if you call it many times during the same Greenfoot frame, you may well get different answers.

- Video https://www.youtube.com/watch?v=jKLU-i56HE8
- Pagina https://objetos16171.github.io/Contra-Survive/

Fecha de Inicio	Fecha de Término	Actividad por realizar
16/08/2016	30/08/2016	Propuesta del proyecto.
31/08/2016	08/09/2016	Diagrama de Clases.
18/10/2016	16/11/2016	Primer Avance de Proyecto.
21/11/2016	05/12/2016	Proyecto Terminado

Bitácora de actividades (historial)

Fecha (dd/mm/aa)	Descripción de la actividad realizada
16/08/2016	Dentro de esta actividad se hizo la realización de propuestas de proyecto, esto consta en dar sugerencias de lo que queremos hacer de trabajo, esto debe de cumplir con los requisitos minimos que se hayan pedido previamente, en este caso como se inicio principalmente con un equipo de 3 personas, hicimos alrededor de 4 propuestas tentativa para iniciar con la realizaion de dicho proyecto, dentro de las cuales inclimos el que les hemos presentado.
31/08/2016	Para el buen diseño del proyecto fue necesario la realización de un Diagrama de clases, para esto se nos dio la oportunidad de utilizar cualquier tipo de aplicación que nos brindara apoyo al realizar la organización de las clases, aunque en este documento no se muestra detallado todo el proyecto esn si, lo usamos como un apoyo el cual nos ayudo y guio para seguir en la elaboración de esta actividad.
18/10/2016	Al haber realizado lo anterior, contamos con un buen apoyo para comenzar a programar el código y asi poder hacer un avance para la revición, en este primer avance implementamos las clases principales, agregamos algunas imágenes para empezar con la interfaz, hicimos el movimiento del personaje principal y después implementamos los enemigos, para esto aun no verificavamos el contacto entre ello, pero agregamos los niveles que llevaría nuestro juego.
21/11/2016	Ultimo avance, al decir esto nos referimos a la culminación del proyecto en si, en esta parte implementamos las sentencias y verificaciones del contaco, hicimos varias pruebas para cuando el unuario tocara el enemigo perdiera una vida, pero en este caso tuvimos que verificar que el usuario no perdiera vida al tocar a las plataformas, para esto tivimos que hacer varias verificaciones dentro de este avance para asi poder tener terminado el proyecto dentro de la fecha indicada.