****

**La Batalla de**

**La Batalla de Ciudad Juánez**

**Índice**

* + Introducción al Trabajo 4
  + Objetivo del proyecto 4
  + Alcance 4
  + Propuesta 4
  + Análisis 5
  + Interfaz 13
  + Diagrama de Clases 18
  + Código 26
  + Administración del Proyecto 176
    - Definición de Roles 176
    - Seguimiento de Tareas 178
    - Forma de Control de Cambios 180
    - Retroalimentación del equipo 181
  + Conclusiones 184
    - Sobre el desarrollo del proyecto 184
    - Sobre la administración del proyecto 185
    - Trabajos futuros 186

**Introducción al Proyecto**

Moonligth Technologies es una empresa dedicada al desarrollo de juegos en Java. Esta empresa busca entretener a un mercado principalmente joven pero igual a todas las edades. Sigue las nuevas tendencias y lo que busca la juventud. Es por ello que en esta ocasión Moonligth Technologies desarrollara el juego la Batalla de Cd. Juánez. Un “shooter” en dos dimensiones que permitirá al usuario vivir en una realidad alterna en la que tiene que salvar a su ciudad de un ataque extraterrestre. De esta forma se sigue lo que pide el mercado juvenil el cual es el principal consumidor de videojuegos.

**Objetivo**

El objetivo de este proyecto es poder generar un juego en dos dimensiones que permita al usuario tener entretenimiento. Se busca lograr esto a través del uso de un juego de disparos en el que el objetivo principal es hacer sentir al usuario que vive en una realidad alterna y que realmente tiene que salvar a su ciudad. Esto se logrará a través del uso de diferentes retos dentro del juego.

**Alcance**

Este proyecto busca generar un juego en dos dimensiones que permita a su personaje principal realizar las funciones básicas de moverse en dos direcciones y saltar. Al mismo tiempo se busca que pueda disparar para poder atacar enemigos. Se tienen dos niveles diferentes con más enemigos en el segundo son más enemigos. También los enemigos tienen la capacidad de atacar.

El juego no tiene más de dos niveles. No puede jugarse en línea ni con varios jugadores al mismo tiempo. Solamente está en dos dimensiones y no en tres. Se permite cambiar sonidos y se llega a ganar o perder el juego. S gana cuando se destruyen a todos los enemigos.

**Propuesta**

**Nombre del proyecto:** La Batalla de Cd. Juánez

**Objetivo del proyecto (redacción centrada en el usuario):** Entretener al usuario joven y generar cierto nivel de relación con un mundo ficticio que lo haga entrar en la fantasía de intentar salvar a su ciudad.

**Descripción del proyecto:**

Un juego en el cual el personaje principal puede moverse a través de la pantalla en la cual le aparecen “enemigos” disparándole a los cuales hay que destruir utilizando balas que provienen de sus armas o de ciertos premiso que obtiene a través del juego.

**Grado de dificultado del proyecto:**

Medio

**Descripción breve del usuario al que va dirigida la aplicación:**

Es un joven de entre 13-25 años el cual este interesado en la ciencia ficción y le interese tener cierto entrenamiento mediante el uso de armas y un juego al estilo “shooter”.

**Plataforma computacional con la que cuenta el usuario:**

Se puede utilizar cualquier sistema operativo (Windows 98- en adelante, Mac OS, Linux etc.) ya que el juego será desarrollado en Java y éste es multiplataforma. Es necesario que se tenga instalado el Java Runtime Environment para que el juego pueda ejecutarse. El juego tendrá que ser descargado ya que el juego no puede jugarse en línea.

Bibliografía:

Schildt, Herbert. *Java: The complete reference J2SE.* Estados Unidos de América: Mc-

Graw Hill, 2005.

**Nombre de la tarea que va simular la computadora:**

La tarea que representa este juego de la vida real es eliminar obstáculos ficticios para poder generar en el usuario un sentido de resolución de problemas siguiendo actividades rutinarias.

**Listado de fuentes de información específicas en dónde buscarán contenidos:**

Bibliografía:

Brackeen, David. (2003). *Developing games in java*. Nueva York: New Rider Publishing.

Davison, Andrew. (2005). *Killer game programming in java*. Sebastopol: O'Really Media.

Flynt, John. (2007). *Java me game programming*. Boston: Thomson.

Schildt, Herbert. (2003). *Java the complete reference j2se*. Nueva York: McGraw Hill.

**Análisis del usuario**

El desarrollo de este juego está centrado en entretener a personas de todas las edades y con todas las capacidades pero aún así va centrada al público joven. Para poder lograr este desarrollo es necesario hace un juego con emoción que haga sentir al usuario joven que lucha por su vida en un ambiente ficticio lo cual no podría vivir en la su mundo real. Es por ello que se genera una realidad alterna que le ayuda a entretenerse.

**Usuario Primario**

El usuario primario será cualquier persona joven de 12 a 20 años. Este usuario debe estar interesado en ser retado y poder tener una realidad alterna mientras juega. Al mismo tiempo, debe ser una persona que le guste los juegos de tipo “shooter” donde se puede disparar y existe cierto grado de violencia.

**Usuario Secundario**

El usuario secundario será cualquier persona mayor de 12 años que aunque no sea joven esté interesado en ser retado y poder tener una realidad alterna mientras juega. Simplemente sería cualquier persona que quiera entretenerse por un momento.

**Usuario genérico**

El usuario genérico será toda persona que pueda entrar o no en cualquiera de los dos tipos de usuario anteriormente mencionado. En general, puede ser cualquier persona que quiera divertirse y la intención de acabar el juego con el mayor puntaje posible.

**Perfiles de Usuario**

Emmanuel

[](http://www.facebook.com/photo.php?pid=3282287&id=550296190)

Es un joven de 17 años que vive en Guadalupe N.L. y estudia en la Prepa Tec. Le interesa mucho los juegos con violencia y disparos. Siempre que tiene tiempo juega este tipo de juegos. Además le gusta vivir en una realidad alterna mientras juega se imagina que él es el personaje principal.

Daniel



Es un joven de 13 años que actualmente estudia secundaria. A él le gusta mucho jugar en su tiempo libre. En el recreo va a las salas de cómputo de su colegio para poder jugar. Le gustan mucho los juegos en línea que del tipo “first persona shooter”. Aún así le interesa cualquier tipo de shooter.

Jessica

[](http://www.facebook.com/photo.php?pid=12058782&id=527585050)

Jessica es una joven de 15 años que está a punto de acabar la secundaria. Siempre juega en la computadora de su papá y en su PSP (Play Station Portable). A ella le gustan los juegos que reten a su habilidad mental y la hagan generar estrategias. Aún así también está interesada en juegos que la hagan vivir una realidad alterna de algo que ella no haría en la vida real como matar alienígenas.

Georgina

[](http://www.facebook.com/photo.php?pid=3095152&id=665947745)

Es una joven de 19 años que le gusta jugar videojuegos en su tiempo libre sobre todo en consolas. Aún así le interesan los “shooters” en computadoras. Le gusta sobre todo que los juegos traten de ciencia ficción y poder sentirse en un mundo alterno mientras juega.

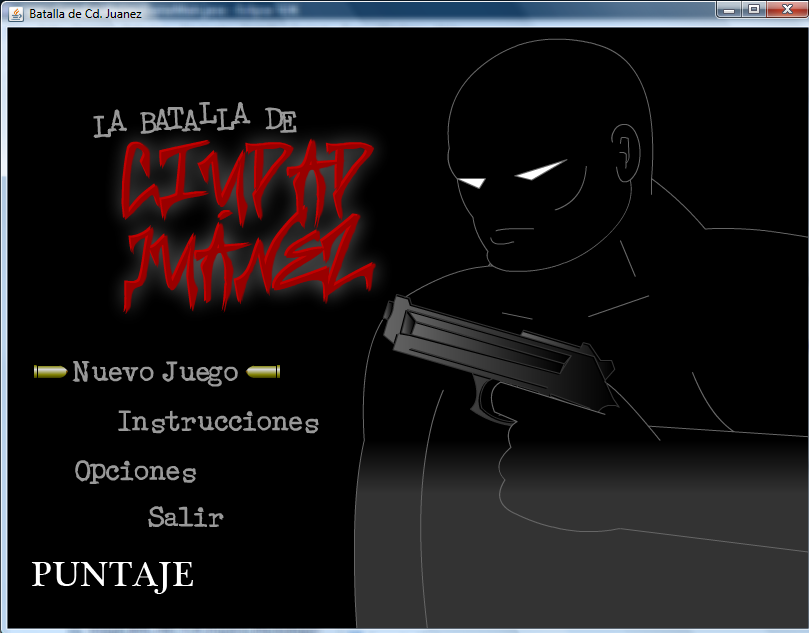
Ricardo

[](http://www.facebook.com/photo.php?pid=3340702&id=528073435)

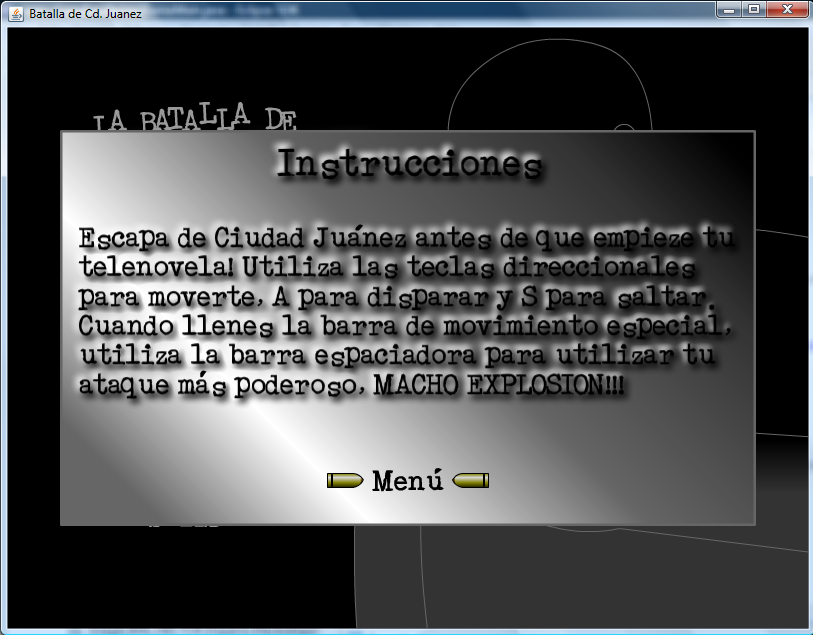
Es un joven de 20 años que no le interesan mucho los videojuegos. Su tiempo libre lo dedica a practicar deportes. Aún así siempre busca un momento para entretenerse sólo y lo hace con videojuegos de tipo “shooter”.

Según las especificaciones de los usuarios se desarrolló un prototipo beta del juego y se pidieron diferentes sugerencias tanto de un experto como de uno de los usuarios finales.

**Casos de Prueba (Prototipo Beta)**

****

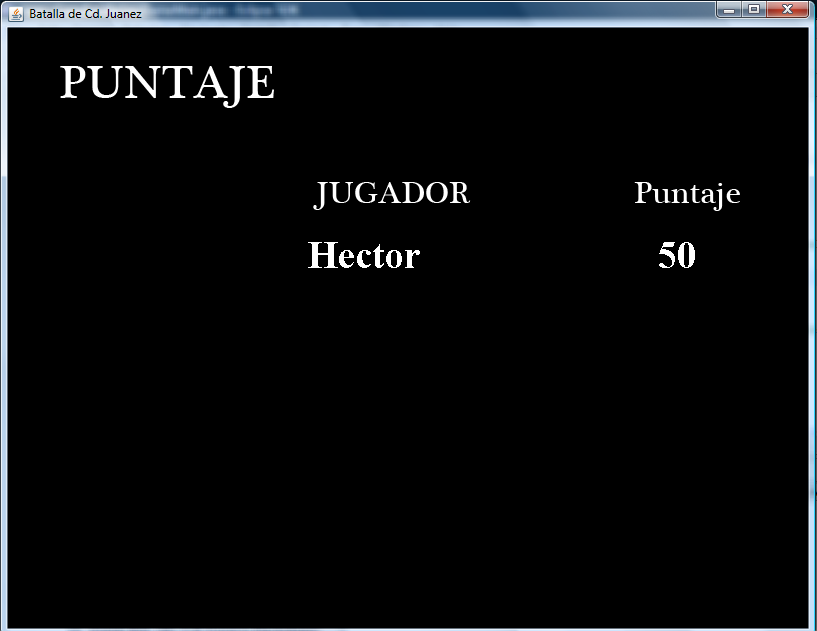
* **Si se oprime en el menú las flechas se mueve la opción.**
* **Si se oprime en el menú la flecha al final se regresa a la primera opción.**
* **Si se oprime en el menú enter en la opción Juego Nuevo aparece la primera pantalla del juego.**

****

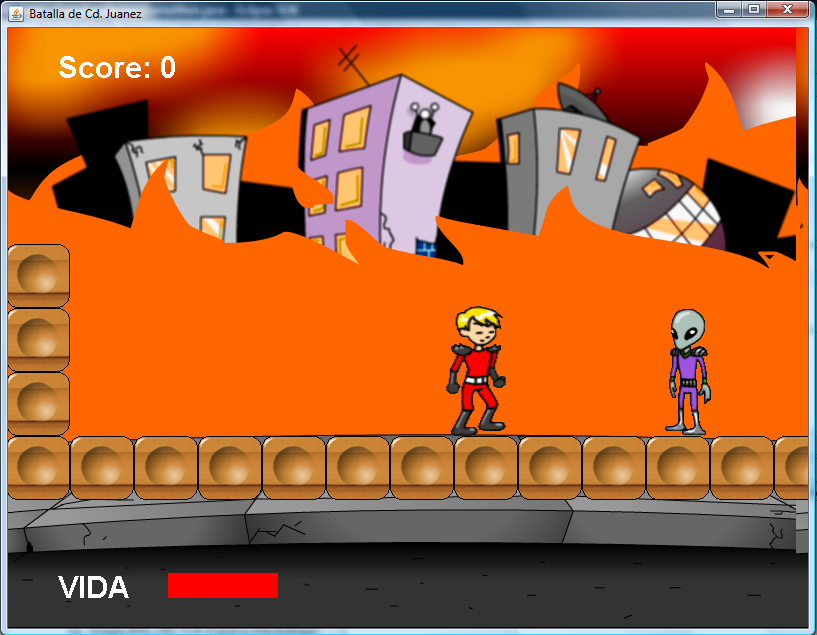
* **Si se oprime en el menú enter en la opción Instrucciones aparece la pantalla que describe cómo jugar el juego.**

****

* **Si se oprime en el menú enter en la opción Opciones aparece la pantalla de Opciones de música.** 
  + **En la pantalla de opciones al oprimir las flechas de arriba y abajo se puede escoger entre efectos de sonido y música.**
  + **En los efectos de sonido se puede apagar o prender.**
  + **En los efectos de música se puede cambiar el tipo de música.**

****

* **Si se oprime en el menú enter en la opción Puntaje aparece en pantalla los top 5 puntajes del juego.**
* **Si se oprime en el menú enter en la opción Salir se sale del juego.**

****

* **Si se presiona las flechas de derecha o izquierda el personaje principal se moverá en esa dirección.**
* **Si se presiona la letra “a” permite al personaje principal disparar.**
* **Si se presiona la letra “s” permite al personaje saltar.**
* **La bala sigue una trayectoria horizontal y desparece después de cierta distancia.**
* **Cuando las balas colisionan o desaparecen se maneja una explosión.**
* **Cuando dos balas colisionan con el enemigo, este desaparece y se incrementa el puntaje en 50 puntos.**
* **Las balas deberían de desaparecer y en su lugar aparecer una explosión cuando colisionan con los “tiles”.**
* **Cuando una bala del personaje principal colisiona con un enemigo más de una vez el enemigo desaparece y se aumenta el puntaje.**
* **El enemigo dispara hacia el personaje principal y lo sigue.**
* **Cuando la bala del enemigo o el mismo enemigo colisiona con el personaje principal, se le disminuye la barra de vida al personaje principal.**
* **Cuando se acaba la barra de vida del personaje se despliega una ventana pidiendo el nombre y se graba en un archivo el puntaje del jugador.**
* **Cuando se acaba la barra de vida del jugador aparece una pantalla de GAME OVER.**
* **Cuando el jugador se cae en un pozo aparece una pantalla de game over.**

**Nota: Todo esto se le pide a los testers que lo prueben**

**Sugerencias Experto**

****

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sugerencias**   |  |  |  |  | | --- | --- | --- | --- | | Identificador de pantalla | Descripción de la sugerencia o modificación | Justificación para hacer el cambio | Persona que lo propone | | Pantalla de Juego | El fondo parece que fue pegado hay que hacer una imagen del tamaño del juego. | El fondo no parece real por lo tanto hay que modificarlo. | Ing. Mario de la Fuente. | | Pantalla de Juego | Hay que incluir power ups. | Debe haber otra forma de ganar vida o puntos. | Ing. Mario de la Fuente. | |  |  |  |  | |  |  |  |  | |
|  |

Se le pidió al Ing. Mario de la Fuente que diera sus sugerencias acerca del juego que se desarrolló. Sus sugerencias fueron desde funcionalidad hasta interfaz. En la siguiente tabla se muestran sus sugerencias tanto como los problemas que encontró este experto.

**Sugerencias Usuario Final**

[](http://www.facebook.com/photo.php?pid=30014773&id=1456179684)

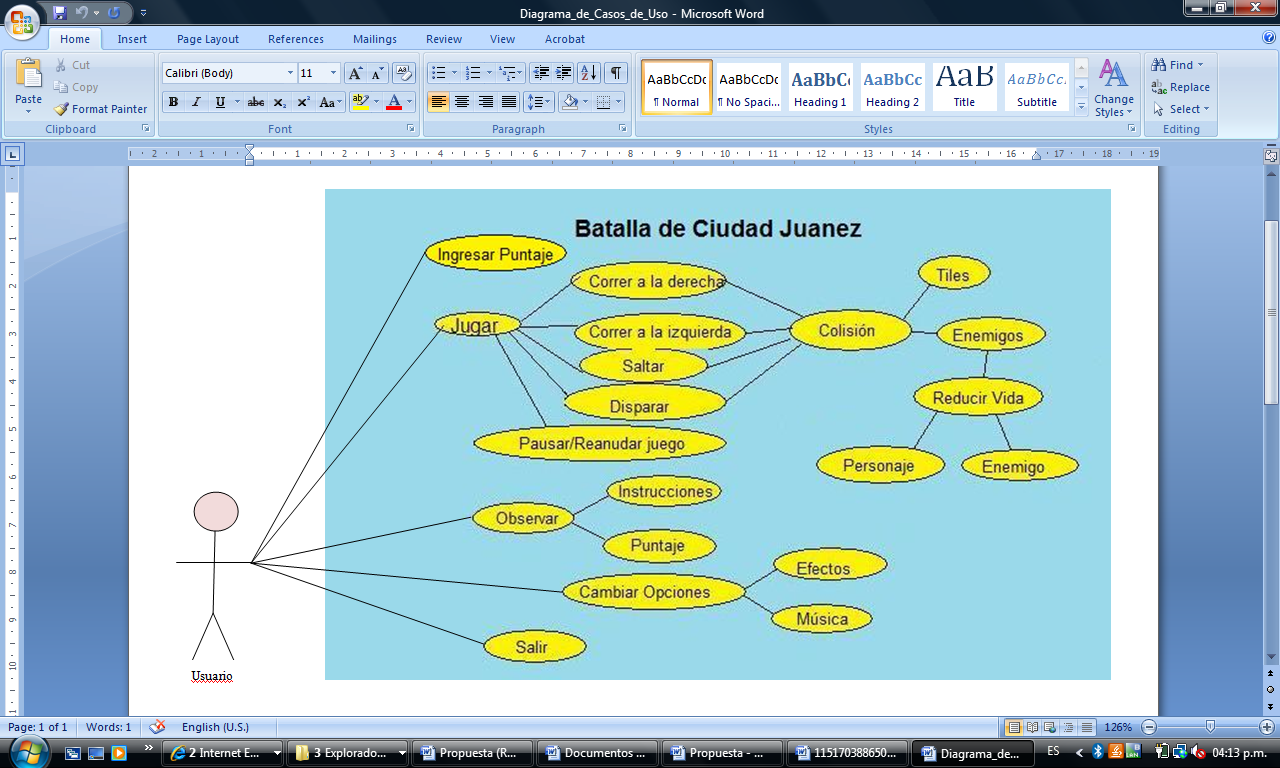
Usuario: Joaquín (Jugador de videojuegos por excelencia)

Sugerencias:

* Los controles son excelentes. Estaría interesante tener un control para poder modificar la música durante el juego y no sólo en el menú.
* Estaría interesante tener diferentes tipos de enemigos.
* También se deberían agregar “power ups” para poder tener diferentes formas de ganar vidas o score.
* En un futuro se podría buscar que el jugador pudiera elegir al personaje que va a utilizar.

**Interfaz**

**Diagrama de Casos de Uso**



**Ingresar Puntaje**

En esta parte se agrega un JOptionPane dónde se le pide al usuario que ingrese su nombre para guardar su puntaje en el archivo y así poder observar el puntaje después.

**Jugar**

En jugar comienza el juego y se le permite al jugador realizar las diversas acciones del personaje principal. Con las flechas derecha e izquierda el jugador puede moverse a través de la pantalla. Se le permite saltar con la tecla “s” y con la tecla “a” dispara. Le van apareciendo diversos obstáculos y precipicios que tiene que saltar para evitar que el juego termine y le van apareciendo los enemigos a los cuales tiene que disparar si no quiere que lo destruyan.

**Observar**

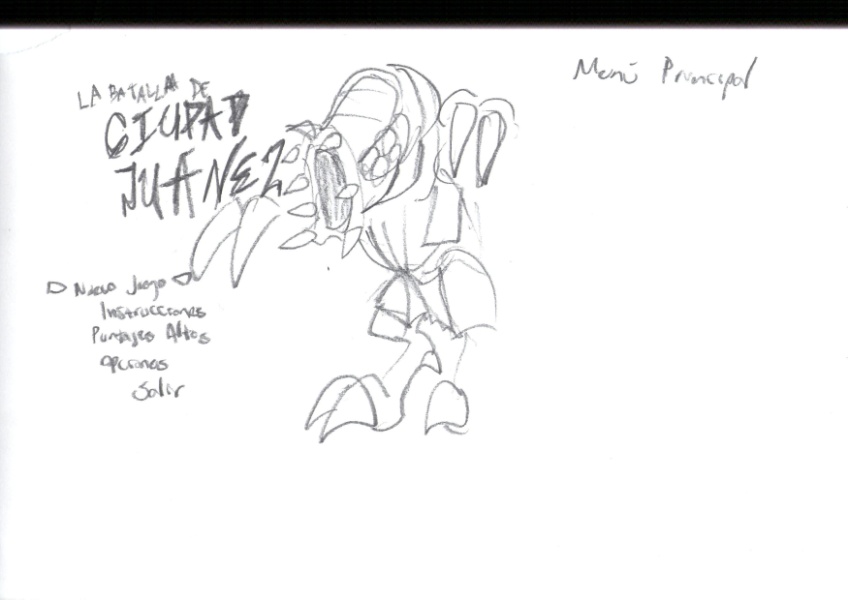
Si el usuario decide en el menú conocer las instrucciones o los más altos puntajes sólo los observa.

**Salir**

Se destruye la pantalla del juego y se acaba el mismo.

Al iniciar el juego se busca hacer sentir al usuario que comienza su batalla. Es por esto que se utiliza una imagen de un hombre con un arma para darle a conocer que va a utilizar armas para poder realizar su misión en el juego. Asimismo, al ser un juego que implica violencia se utilizaron colores oscuros para ejemplificar a través de la falta de color cierta negatividad.

**Storyboard**

****

**Texto:**

Nuevo Juego

Instrucciones

Puntajes Altos

Opciones

Salir

**Imagen:**

Logo de la Batalla de Cd. Juánez

**Sonido:**

Electrónica postmoderna

**Guión Literario**

Al iniciar el juego aparece primero la pantalla de menú. En ella se pone una imagen de una especie de extraterrestre para dar a conocer que el juego trata de una batalla contra extraterrestres que acaban de atacar a Cd. Juánez y el personaje principal tiene que salvarlo.

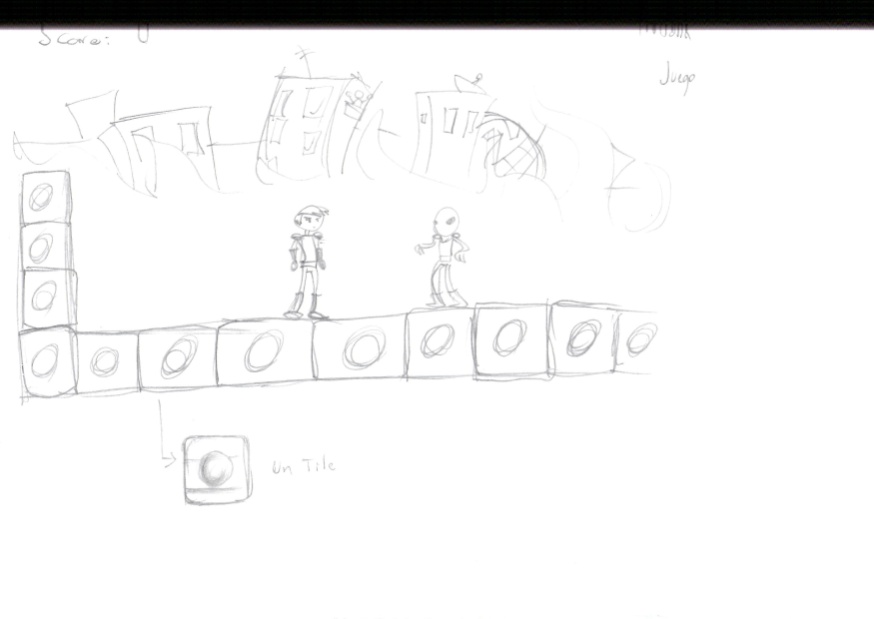
**Guión Técnico**

Con las flechas se selecciona la opción que se desea. Conforme se mueve se cambia de color la opción. Se hace un loop en el cual se regresa a la primera opción cuando se llega a la última.

**Interacción**

Cada vez que se selecciona una opción se ve resaltada y además existe un fondo de música cada vez que se cambia de opción.

**JUGANDO**

****

**Imagenes:**

* Tiles
* Personaje Principal
* Enemigo

**Sonido:**

El usuario escoge entre

Hip Hop

Electrónica

Pop

**Guión Literario**

El usuario entra a la parte del juego. La ciudad está destruida por los extraterrestres. Aún quedan algunos sobrevivientes y hay que destruir a todos los alienígenas para poder salvarlos y pasar de nivel.

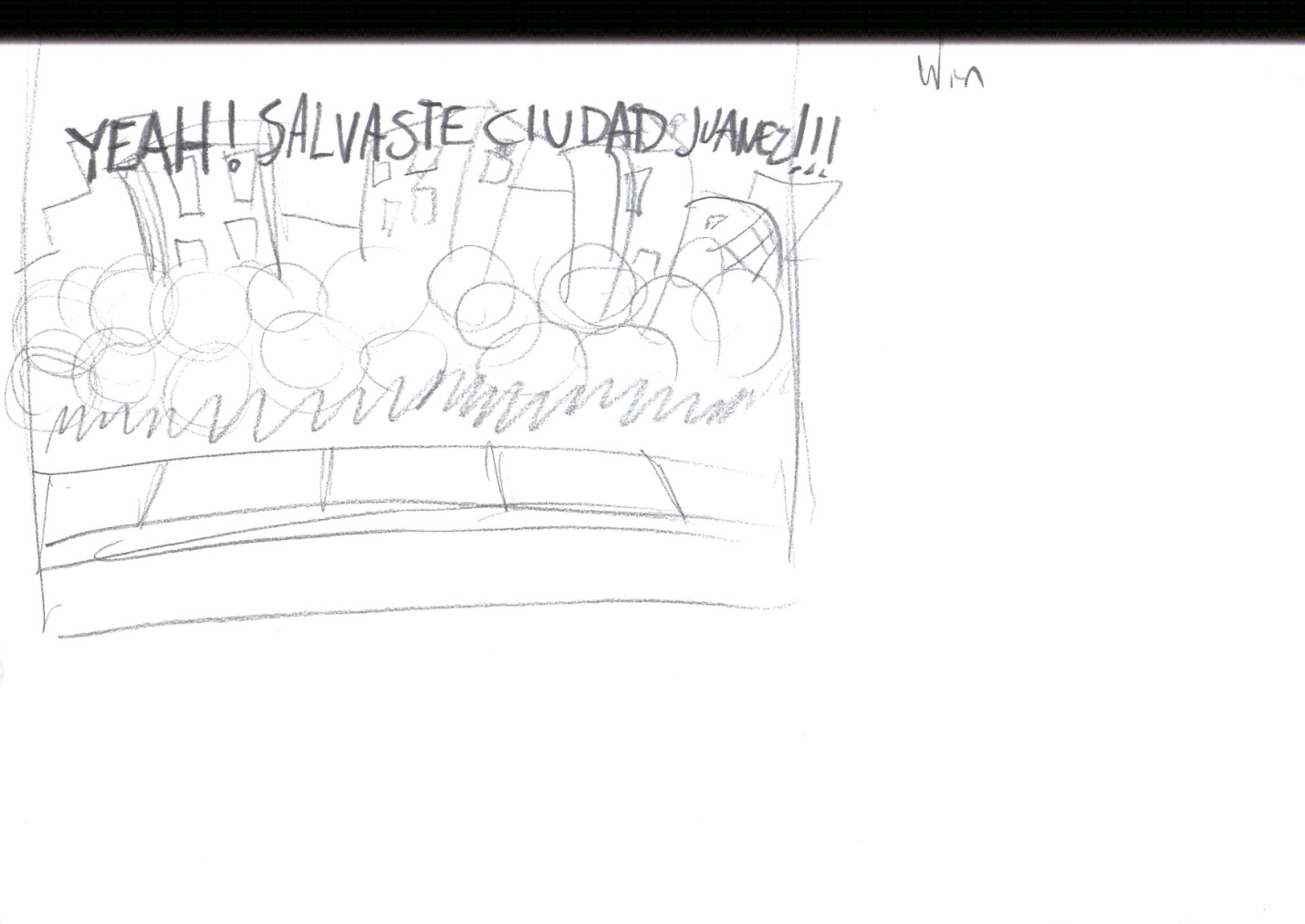
**Guión Técnico**

Con las flechas el usuario se mueve a través de la pantalla. Salta tecleando la letra “s” y con la letra “a” dispara. Las alienígenas tienen inteligencia y atacan al personaje principal con disparos.

**Interacción**

Se observa cómo se mueve el personaje principal y realiza las acciones que el usuario decide a través del teclado.

**GANAR**



**Guión Literario:**

Esta pantalla se muestra cuando el personaje principal ha pasado los tres niveles del juego y destruido a todos los extraterrestres.

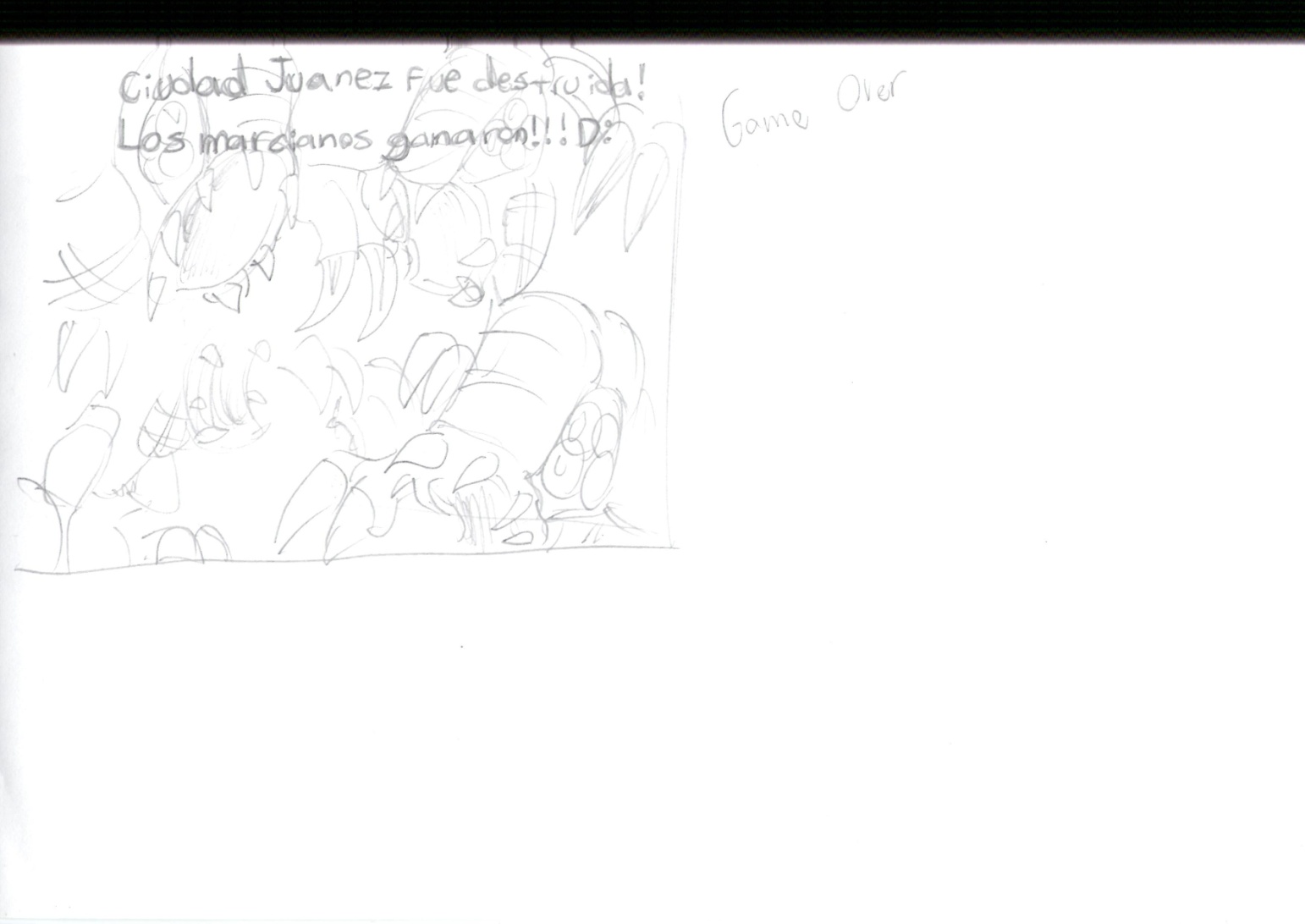
**Guión Técnico:**

Se despliega la imagen.

**Interacción:**

Se observa la imagen en la pantalla.

**PERDER**

****

**Guión Literario:**

Esta pantalla se muestra cuando se le ha acabado la vida al personaje principal.

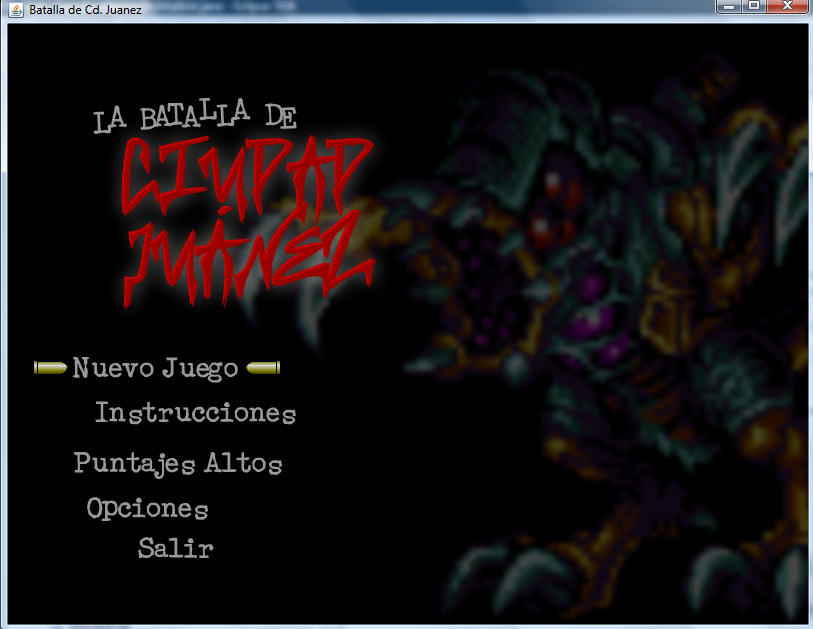
**Guión Técnico**

Se despliega en pantalla la imagen.

**Interacción**

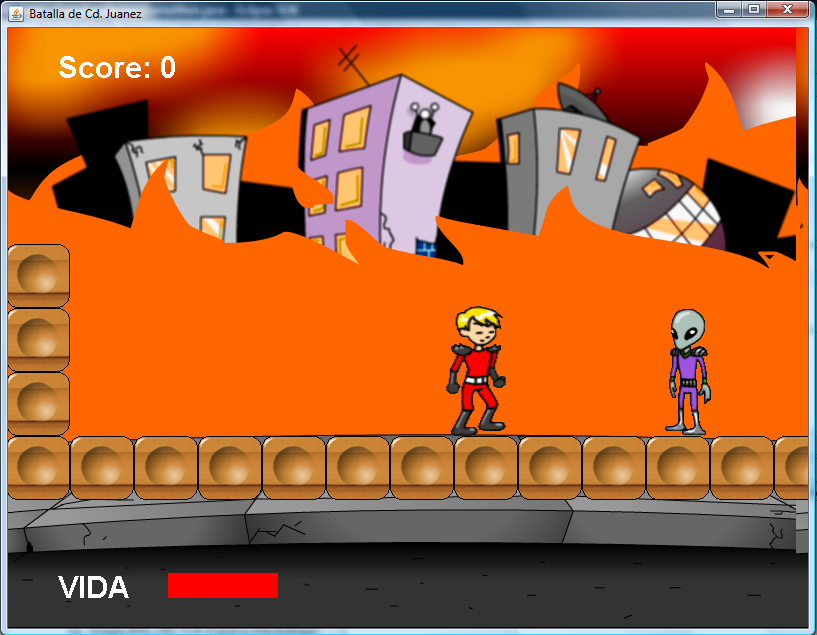
Se observa la imagen en pantalla.

**Pantalla de Menú**



El tipo de letra utilizado en el juego es el de Century Gothic, este es utilizado debido a que no es una letra difícil de leer y es amigable a la vista del usuario. También porque así se expresa el mensaje de negativismo y destrucción. Asimismo se utilizan los colores rojo y blanco en los caracteres empleados en la escritura debido a que los fondos de cada menú contienen colores oscuros y los rojos y blancos ayudan a resaltar las palabras y de esta manera se le facilita al usuario leer dichas palabras.

**Pantalla de Juego**

****

En esta pantalla se utilizó un fondo de una ciudad quemándose con sombras oscuras para dar a entender una ciudad destruida y siendo atacada. El personaje principal que se utilizó es joven para dar cierta relación con el usuario y al ser la historia sobre extraterrestres atacando se utilizó una figura de alienígenas como enemigo.

**Diagrama de Clases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| JFrame |  |  |  | |
|  |  | <<interface>> |  | |
| JFrame() |  | Runnable |  | |
|  |  | run() : void |  | |
|  |  |  |  | |
|  |  |  | <<interface>> | |
|  |  |  | KeyListener | |
|  | GameMain |  | keyPressed() : void | |
|  | serialVersionUID : long = 1L |  | keyReleased() : void | |
|  | GRAVITY: float |  | |  | | --- | | keyTyped() : void | | |
|  | HEIGHT: int |  |  | |
|  | state: state |  |  | |
|  | VELOCIDADBALA: float |  |  | |
|  | VELOCIDADCAIDA: float |  |  | |
|  | WIDTH: int |  |  | |
|  | main(String[]): void |  |  | |
|  | abajo: boolean |  |  | |
|  | alto: int |  |  | |
|  | ancho: int |  |  | |
|  | anim: animation |  |  | |
|  | arr: string |  |  | |
|  | arriba: boolean |  |  | |
|  | attack: boolean |  |  | |
|  | balas vector |  |  | |
|  | balasEnemigos: vector |  |  | |
|  | dbg: Graphics |  |  | |
|  | dbgImage: imag |  |  | |
|  | der: boolean |  |  | |
|  | elapsedTime: long |  |  | |
|  | instrucciones: image |  |  | |
|  | instruccionesz: imageIcon |  |  | |
|  | intParaPausa: int |  |  | |
|  | izq: boolean |  |  | |
|  | juego: juego |  |  | |
|  | jugador: player |  |  | |
|  | M\_INS: image |  |  | |
|  | M\_INSz: image |  |  | |
|  | M\_JUEGO: image |  |  | |
|  | M\_JUEGOz: image |  |  | |
|  | M\_OP: image |  |  | |
|  | M\_Opz: image |  |  | |
|  | M\_PUN: image |  |  | |
|  | M\_PUNz: image |  |  | |
|  | M\_SALIR: image |  |  | |
|  | M\_SALIRz: image |  |  | |
|  | mapa: tileMap |  |  | |
|  | menu: menu |  |  | |
|  | musica: soundClip |  |  | |
|  | nombreArchivo: string |  |  | |
|  | opciones: image |  |  | |
|  | opcionesz: image |  |  | |
|  | otrasImagenes: vector |  |  | |
|  | pausa: image |  |  | |
|  | pausaz: imageIcon |  |  | |
|  | pointCache: point |  |  | |
|  | prendeSonidoTemporal: int |  |  | |
|  | puntaje: image |  |  | |
|  | puntajez: imageIcon |  |  | |
|  | puntos: int |  |  | |
|  | render: tileMapRenderer |  |  | |
|  | screenSize: dimension |  |  | |
|  | seleccionMusica: int |  |  | |
|  | seleccionMusicaOn: int |  |  | |
|  | seleccionOpciones: int |  |  | |
|  | sonidoDisparo: soundClip |  |  | |
|  | sonidoDisparoMalo: soundClip |  |  | |
|  | sonidoPausa: soundClip |  |  | |
|  | space: boolean |  |  | |
|  | sueltaBotonBala: boolean |  |  | |
|  | tiempoSalto: long |  |  | |
|  | vec: vector |  |  | |
|  | vidas: int |  |  | |
|  |  |  |  | |
|  | GameMain() |  |  | |
|  | checaColision(): void |  |  | |
|  | gameDraw(Graphics2D): void |  |  | |
|  | gameInit(): void |  |  | |
|  | gameKeyPressed(int): void |  |  | |
|  | gameKeyReleased(KeyEvent): void |  |  | |
|  | gameKeyTyped(KeyEvent): void |  |  | |
|  | gameLoop(): void |  |  | |
|  | gameShutdown(): void |  |  | |
|  | gameStart(): void |  |  | |
|  | gameUpdate(): void |  |  | |
|  | getTileCollision(Sprite,float,float): Point |  |  | |
|  | grabaArchivo(): void |  |  | |
|  | keyPressed(KeyEvent): void |  |  | |
|  | keyReleased(KeyEvent): void |  |  | |
|  | keyTyped(KeyEvent): void |  |  | |
|  | leeArchivo(): void |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  |  |  |  | |
|  | | | |  | |
|  | | | |  | |
|  | | | |  | |
|  | | | | TileMap | |
| Sprite | | | | balas: vector | |
| CHOQUE\_CON\_PARED: string | | | | balasEnemigos: vector | |
| DERECHA\_ATACANDO: string | | | | explosiones: vector | |
| ESTA\_CERCA: string | | | | items: arrayList | |
| IZQUIERDA\_ATACANDO: string | | | | lista: arrayList | |
| NO\_ESTA\_CERCA: string | | | | objetos: vector | |
| PROXIMA\_CAIDA\_LIBRE: string | | | | player: player | |
| anim: animation | | | | puntos: int | |
| dx: float | | | | sprites: vector | |
| dy: float | | | | tiles: image | |
| estadoActual: string | | | |  | |
| icono: imageIcon | | | | TileMap() | |
| lado: int | | | | adBala(Bala): void | |
| matenme: boolean | | | | addBalaEnemigos(Bala): void | |
| posX: float | | | | addExplosion(Explosiones): void | |
| posY: float | | | | addSprite(Sprite): void | |
| vida: int | | | | addSprite(TileMap, Sprite, int, int): void | |
| vidaMax: int | | | | cargaTiles(): void | |
| x: float | | | | getBalas(): Vector | |
| y: float | | | | getBalasEnemigos(): Vector | |
|  | | | | getExplosiones(): Vector | |
| Sprite() | | | | getHeight(): int | |
| actualiza(long): void | | | | getPlayer(): Player | |
| atacar(): boolean | | | | getPuntos(): int | |
| cambiaVida(int): void | | | | getSprites(): Vector | |
| checaProximidad(Sprite): void | | | | getTile(int, int): Image | |
| clone(): Object | | | | getWidth(): int | |
| dibujaBala(boolean,TileMap):Bala | | | | loadImage(String): Image | |
| getAlto(): int | | | | loadMap(): TileMap | |
| getAncho(): int | | | | removeBala(Bala): void | |
| getCurrState(): String | | | | removeBalaEnemigos(Bala): void | |
| getHeight(): int | | | | removeExplosion(Explosiones): void | |
| getImageFromAnimation(): Image | | | | removeSprite(Sprite): void | |
| getImageIcon(): ImageIcon | | | | setPlayer(Player): void | |
| getImageI(): Image | | | | setPuntos(int): void | |
| getMaxVida(): int | | | | setTile(int, int, Image): void | |
| getPerimetro(): Rectangle | | | |  | |
| getPosX(): float | | | |  | |
| getPosY(): float | | | |  | |
| getVelocidadX(): float | | | |  | |
| getVelocidadY(): float | | | |  | |
| getVida(): int | | | |  | |
| getWidth(): int | | | |  | |
| indentificate(): String | | | |  | |
| iniciaVida(): void | | | |  | |
| intersecta(Sprite): boolean | | | |  | |
| intersectaAbajo(Sprite): boolean | | | |  | |
| nuevasModificaciones(): void | | | |  | |
| ontas(): void | | | |  | |
| puedeMorir(): boolean | | | |  | |
| quien(): String | | | |  | |
| revisarCambioDeEstado(String,float): void | | | |  | |
| revisarCambioDeVuelta(TileMap,float,float,float): void | | | |  | |
| setAnimation(Animation): void | | | |  | |
| setImageIcon(Image): void | | | |  | |
| setLado(int): void | | | |  | |
| setPosX(float): void | | | |  | |
| setPosY(float): void | | | |  | |
| setVidaMax(int): void | | | |  | |
| updateAnim(long): void | | | |  | |
| volteaADerecha(): boolean | | | |  | |

|  |  |  |
| --- | --- | --- |
|  |  | AnimFrame |
|  | Animacion | .image : Image |
| TileMapRenderer | .ArrayList : frames | .endTime : long |
| TILE\_SIZE: int | .currFrameIndex : int | AnimFrame() |
| TILE\_SIZE\_BITS: int | .animTime : int |  |
| pixelsToTiles(float): int | .totalDuration : int |  |
| pixelsToTiles(int): int | Animacion() |  |
| tilesToPixels(int): int | addFrame() : void |  |
| background: image | start() : void |  |
| device: GraphicDevice | update() : void |  |
| gc: GraphicsConfiguration | getImage() : Image |  |
| draw(Graphics2D,TileMap,int,int): void | getFrame() : AnimFrame |  |
| getFlippedImage(Image, Graphics): Image |  |  |
| setBackground(Image): void |  |  |

|  |  |  |
| --- | --- | --- |
| Bala |  |  |
|  |  |  |
|  | Creature |  |
|  | DIE\_TIME: int |  |
|  | STATE\_DEAD: int | Enemigo |
| animDispUno: animation | STATE\_DYING: int | ataca: boolean |
| damage: int | STATE\_NORMAL: int | atacaDerecha: animation |
| inicial: float | animationPool: animation | atacaIzquierda: animation |
| posicion: float | state: int | caminaDerecha: animation |
| rebotable: boolean | stateNames: string | caminaIzquierda: animation |
| tipoDispato: int | stateTime: long | contrincante: sprite |
|  |  | count: int |
| Bala() | Creature() | deltaEntreAtaques: long |
| actualizaPosicionI(): void | collideHorizontal(): void | estadoCaminaDerecha: string |
| cambiaInicial(float): void | collideVertical(): void | estadoCaminaIzquierda: string |
| cargaImagenes(): void | getMaxSpeed(): float | momentoDeInicioDeAtaque:long |
| getDamage(): int | getState(): int | oldX: float |
| getInicial(): float | identificate(): String | paradoDerecha:animation |
| getPosicionF(): float | isAlive(): boolean | paradoIzquierda:animation |
| getTipoDisparo(): int | isFlying(): boolean | tiempoDeAtaque: long |
| noRebota(): boolean | ontas(): void | unaVez: boolean |
|  | setState(int): void |  |
| setDamage(): int | update(long): void | Enemigo() |
| setNoRebotes(boolean): void | wakeUp(): void | atacar(): boolean |
| setTipoDisparo(int): void |  | createAnimation():void |
| updateAnim(long): void |  | dibujaBala(boolean, TileMap): Bala |
|  |  | identificate(): String |
|  |  |  |
|  |  | persigueA(Sprite): void |
|  |  | revisarCambioDeEstado(String,float): void |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  | | |
|  | | Player | | |
|  | | Ataque: enum | | |
| Explosiones | | Estado: enum | | |
| animExp:animation | | Skill: enum | | |
| tiempoAnim: int | | ataque: ataque | | |
| tiempoCreacion: long | | estado: estado | | |
| tiempoTotal: int | | JUMP\_SPEED: float | | |
|  | | anim:animation | | |
| Explosiones() | | animDisparo:animation | | |
| cargaImagenes():void | | animParado:animation | | |
| termina(): boolean | | gravity: float | | |
| updateAnim(long): void | | IzqDer: boolean | | |
|  | | MaxSpeed: float | | |
|  | | MAXVIDA: int | | |
|  | | onGround: boolean | | |
|  | | pared: int | | |
|  | | secondCharge: animation | | |
|  | | sliding: boolean | | |
|  | | tiempoDeCarga: int | | |
|  | | tiempoDisparo: long | | |
|  | | tiempoSlide: long | | |
|  | | TIMESHOT: int | | |
|  | | vida: int | | |
|  | | xPared: float | | |
|  | |  | | |
|  | | Player() | | |
|  | | cambiaVida(int): void | | |
|  | | cargaImagenes(): void | | |
|  | | collideHorizontal(): void | | |
|  | | collideVertical(): void | | |
|  | | disparo(): void | | |
|  | | estaAtacando(): boolean | | |
|  | | estaSaltado(): boolean | | |
|  | | getGravity(): float | | |
|  | | getLado(): boolean | | |
|  | | getLvlShot(): int | | |
|  | | getMaxSpeed(): float | | |
|  | | getMaxVida(): int | | |
|  | | getTiempoCarga(): int | | |
|  | | getVida(): int | | |
|  | | jump(boolean): void | | |
|  | | setGravity(float): void | | |
|  | | setPosY(float): void | | |
|  | | slide(boolean): void | | |
|  | | updateAnim(long): void | | |
|  | | wakeUp(): void | | |
|  | |  | | |
|  | |  | | |
| Puntaje |  | | | Juego | | |
| nombre : string |  | | | serialVersionUID : long = 1L | | |
| puntaje : int |  | | |  | | |
| Score() |  | | | Juego() | | |
| setNombre() : void |  | | | keyPressed(KeyEvent): void | | |
| getNombre() : string |  | | | keyReleased(KeyEvent): void | | |
| setPuntaje() : void |  | | | keyTyped(KeyEvent): void | | |
| getPuntaje() : int |  | | | paintComponent(Graphics): void | | |
| to String() : string |  | | |  | | |
|  |  | | |  | | |
|  | SoundClip | | |  | | |
|  | sample : AudioInputStream | | |  | | |
|  | clip : Clip | | |  | | |
|  | looping : boolean | | |  | | |
|  | repeat : int | | |  | | |
|  | filename : string | | |  | | |
|  | SoundClip() | | |  | | |
|  | getClip(): Clip() | | |  | | |
|  | setLooping() : void | | |  | | |
|  | getLooping() : void | | |  | | |
|  | setRepeat() : void | | |  | | |
|  | getRepeat() : void | | |  | | |
|  | setFilename() : void | | |  | | |
|  | getFilename() : string | | |  | | |
|  | isLoaded() : boolean | | |  | | |
|  | getURL() : URL() | | |  | | |
|  | load() : boolean | | |  | | |
|  | play() : void | | |  | | |
|  | stop() : void | | |  | | |
|  | | | State | | |
|  | | | GAMEOVER: state | | |
| Menu | | | INSTRUCCIONES: state | | |
| serialVersionUID : long = 1L | | | JUEGO: state | | |
|  | | | M\_INS: state | | |
| Juego() | | | M\_JUEGO: state | | |
| keyPressed(KeyEvent): void | | | M\_OP: state | | |
| keyReleased(KeyEvent): void | | | M\_PUN: state | | |
| keyTyped(KeyEvent): void | | | M\_SALIR: state | | |
| paintComponent(Graphics): void | | | OPCIONES: state | | |
|  | | | PAUSA: state | | |
|  | | | PUNTAJE: state | | |

**import** java.awt.Image;

**import** java.util.ArrayList;

/\*\*

\* Clase Animation, es la clase encargada de manejar las series

\* de imagenes y el tiempo en que esas imagemenes seran mostradas

\* para crear una animacion

\*/

**public** **class** Animation {

**private** ArrayList<AnimFrame> frames; //Estructura de datos en la que se guardan los frames

**private** **int** currFrameIndex; //Apuntador para saber la posicion en la que se encuentra actualmente el frame

**private** **long** animTime; //El tiempo de desplegado de las imagenes

**private** **long** totalDuration; //La duracion total de la animacion

/\*

\* Constructor Animacion para crear un objeto animacion vacio.

\*/

**public** Animation() {

frames = **new** ArrayList<AnimFrame>();

totalDuration = 0;

start();

}

/\*

\* Metodo addFrame se encarga de agregar un nuevo frame a la estructura

\* de datos y dandole la imagen y el tiempo en que sera desplegada

\* @param image, imagen a usar como frame

\* @param duration, duracion total del frame

\*/

**public** **synchronized** **void** addFrame(Image image,

**long** duration)

{

totalDuration += duration; //Aumenta la duracion total de la animacion

frames.add(**new** AnimFrame(image, totalDuration)); //agrega un nuevo frame con la imagen y el tiempo correspondiente

}

/\*

\* Metodo start, inicia la animacion desde el primer frame

\*/

**public** **synchronized** **void** start() {

animTime = 0;

currFrameIndex = 0;

}

/\*

\* Metodo upadate Animacion para crear un objeto animacion vacio.

\*/

**public** **synchronized** **void** update(**long** elapsedTime) {

**if** (frames.size() > 1) //Verifica que la estructura de datos tenga mas de un frame que desplegar

{

animTime += elapsedTime; //aumenta el tiempo de la animacion, esta controla que frames han sido desplegados

**if** (animTime >= totalDuration) //checha si el la animacion ha llegado a su fin

{

//Instrucciones encargadas de generar un rebobinado a la animacion

animTime = animTime % totalDuration;

currFrameIndex = 0;

}

**while** (animTime > getFrame(currFrameIndex).endTime) {

currFrameIndex++; //Aumenta el apuntador para cambiar de frame

}

}

}

/\*\*

\* Metodo getImage,regresa la imagen actual de la animacion

\* **@return** la imagen actual de la animacion

\*/

**public** **synchronized** Image getImage() {

**if** (frames.size() == 0) { //si la animacion no tiene frames regresa null

**return** **null**;

}

**else** { // Si tiene frames regrasa la imagen

**return** getFrame(currFrameIndex).image;

}

}

/\*\*

\* Metodo AnimFrame,regresa un frame especifico

\* **@param** int, funciona como el apuntador, es el

\* que permite la seleccion del frame

\* **@return** la imagen actual de la animacion

\*/

**private** AnimFrame getFrame(**int** i) {

**return** (AnimFrame)frames.get(i);

}

/\*\*

\* Constructor AnimFrame, crea un nuevo y vacio objeto AnimFrame

\*/

**private** **class** AnimFrame {

//Caracteristicas del objeto

Image image; //una imagen

**long** endTime; //el tiempo que tardara el frame en cambiar

/\*\*

\* Constructor AnimFrame, crea un nuevo objeto AnimFrame

\* **@param** image, la imagen a asignar a la AnimFrame

\* **@param** long, el tiempo de desplegado del AnimFrame

\*/

**public** AnimFrame(Image image, **long** endTime) {

**this**.image = image;

**this**.endTime = endTime;

}

}

}

**import** java.awt.Image;

**import** java.awt.Toolkit;

/\*\*

\* Clase Bala, hija de Sprite, esta clase es la encargada

\* de controlar las balas del jugador y enemigos, su impresion

\* en el mapa, movimiento y daño.

\*/

**public** **class** Bala **extends** Sprite{

**private** **int** damage=1;

**private** **float** posicion; //Una posicion X dentro del mapa

**private** **float** inicial; //La posicion inicial de la bala

**private** **int** tipoDisparo; //varible para saber si la bala es del jugador o del enemigo

**private** Animation animDispJugador; //Animacion de las balas del jugador

**private** Animation animDispEnemigo; //Animacion de las balas del Enemigo

/\*\*

\* Metodo constructor usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\*/

**public** Bala(**int** posX, **int** posY, Image image) {

**super**(posX,posY,image);

inicial=0;

cargaImagenes();

}

/\*\*

\* Metodo constructor usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\* **@param** velocidad es la <code>velocidad</code> que la bala tendra.

\*/

**public** Bala(**int** posX, **int** posY, Image image, **float** velocidad) {

**super**(posX,posY,image);

setVelocidadX(velocidad);

dx = velocidad;

inicial=0;

cargaImagenes();

}

/\*\*

\* Metodo constructor usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\* **@param** velocidadX es la <code>velocidadX</code> que la bala tendra en el eje X.

\* **@param** velocidadY es la <code>velocidadY</code> que la bala tendra en el eje Y.

\*/

**public** Bala(**int** posX, **int** posY, Image image, **float** velocidadX, **float** velocidadY) {

**super**(posX,posY,image);

setVelocidadX(velocidadX);

setVelocidadY(velocidadY);

inicial=0;

cargaImagenes();

}

/\*\*

\* Metodo <I>getDamage</I>

\* **@return** Devuelve el daño realizado por la bala

\*/

**public** **int** getDamage(){

**return** damage;

}

/\*\*

\* Metodo <I>setDamage</I>

\* Da un nuevo valor al daño de la bala

\*/

**public** **void** setDamage ( **int** num){

**this**.damage = num;

}

/\*\*

\* Metodo <I>actualizaPosicionI</I>

\* usado para actualizar la posicion en X de la Bala

\*/

**public** **void** actualizaPosicionI() {

posicion = **this**.getPosX();

}

/\*\*

\* Metodo <I>getPosicionF</I>

\* Devuelve la posicion en X de la Bala

\*/

**public** **float** getPosicionF() {

**return** posicion;

}

/\*\*

\* Metodo <I>getInicial</I>

\* Devuelve la distancia recorrida por la Bala

\*/

**public** **float** getInicial(){

**return** inicial;

}

/\*\*

\* Metodo <I>cambiaInicial</I>

\* Actualiza la distancia recorrida por la Bala

\*/

**public** **void** cambiaInicial(**float** num){

inicial=num;

}

/\*

\* Metodo <I>setTipoDisparo</I>

\* Permite cambiar el tipo de disparo

\* Enemigo o del jugador

\* Dependiendo del numero que haya sido el parametro

\*/

**public** **void** setTipoDisparo(**int** num){

tipoDisparo = num;

}

/\*

\* Metodo <I>cambiaInicial</I>

\* Regresa el tipo de disparo

\* Enemigo o del jugador

\* Dependiendo del numero regresado

\*/

**public** **int** getTipoDisparo(){

**return** tipoDisparo;

}

/\*

\* Metodo <I>updateAnim()</I> <P>

\* cambia la animacion del Player.

\* Si ha pasado cierto tiempo desde que comenzo la animacion,

\* se cambia de animacion y se cambia el Estado.

\*/

**public** **void** updateAnim(**long** elapsedTime){

**if**(tipoDisparo == 1)

{

animDispJugador.update(elapsedTime);

setImageIcon(animDispJugador.getImage());

}

**else** **if** (tipoDisparo == 4)

{

animDispEnemigo.update(elapsedTime);

setImageIcon(animDispEnemigo.getImage());

}

}

/\*\*

\* Metodo <I>cargaImagenes()</I> <P>

\* Carga todas las imagenes que se usaran por el Player

\*/

**public** **void** cargaImagenes(){

// Imagenes del disparo Jugador

Image a1 = Toolkit.*getDefaultToolkit*().getImage("images/disparocyan1.png");

Image a2 = Toolkit.*getDefaultToolkit*().getImage("images/disparocyan2.png");

Image a3 = Toolkit.*getDefaultToolkit*().getImage("images/disparocyan3.png");

//Imagenes de los disparos de enemigos

Image d1 = Toolkit.*getDefaultToolkit*().getImage("images/d1.png");

Image d2 = Toolkit.*getDefaultToolkit*().getImage("images/d2.png");

Image d3 = Toolkit.*getDefaultToolkit*().getImage("images/d3.png");

Image d4 = Toolkit.*getDefaultToolkit*().getImage("images/d4.png");

Image d5 = Toolkit.*getDefaultToolkit*().getImage("images/d5.png");

Image d6 = Toolkit.*getDefaultToolkit*().getImage("images/d6.png");

Image d7 = Toolkit.*getDefaultToolkit*().getImage("images/d7.png");

Image d8 = Toolkit.*getDefaultToolkit*().getImage("images/d8.png");

Image d9 = Toolkit.*getDefaultToolkit*().getImage("images/d9.png");

Image d10 = Toolkit.*getDefaultToolkit*().getImage("images/d10.png");

Image d11 = Toolkit.*getDefaultToolkit*().getImage("images/d11.png");

Image d12 = Toolkit.*getDefaultToolkit*().getImage("images/d12.png");

Image d13 = Toolkit.*getDefaultToolkit*().getImage("images/d13.png");

// Animacion de Disparo Jugador

animDispJugador = **new** Animation();

animDispJugador.addFrame(a1, 100);

animDispJugador.addFrame(a2, 100);

animDispJugador.addFrame(a3, 100);

// Animacion de Disparo Enemigo

animDispEnemigo = **new** Animation();

animDispEnemigo.addFrame(d1, 100);

animDispEnemigo.addFrame(d2, 100);

animDispEnemigo.addFrame(d3, 100);

animDispEnemigo.addFrame(d4, 100);

animDispEnemigo.addFrame(d5, 100);

animDispEnemigo.addFrame(d6, 100);

animDispEnemigo.addFrame(d7, 100);

animDispEnemigo.addFrame(d8, 100);

animDispEnemigo.addFrame(d9, 100);

animDispEnemigo.addFrame(d10, 100);

animDispEnemigo.addFrame(d11, 100);

animDispEnemigo.addFrame(d12, 100);

animDispEnemigo.addFrame(d13, 100);

}

}

**import** java.awt.Image;

/\*\*

\* Clase Creatura, Una creatura es un Sprite que es afectado por la gravedad

\* y puede ser destruido. Tiene 2 animaciones movimiento a la izquierda

\* y movimiento a la derecha.

\*/

**public** **abstract** **class** Creature **extends** Sprite {

//Variables usadas para saber el tiempo que pasara de el estado muriendo al estado muerto

//Estos estados sirven para determinar cuando sera eliminado la Creatura

**private** **static** **final** **int** *DIE\_TIME* = 1000;

**public** **static** **final** **int** *STATE\_NORMAL* = 0;

**public** **static** **final** **int** *STATE\_DYING* = 1;

**public** **static** **final** **int** *STATE\_DEAD* = 2;

**private** **int** state; //estados posibles de las creatras

**private** **long** stateTime; //El tiempo entre estados.

/\*\*

\* Metodo constructor de Creature usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\*/

**public** Creature(**int** posX, **int** posY, Image imagen)

{

**super**(posX,posY,imagen);

iniciaVida();

}

/\*\*

\* Metodo constructor de Creature usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\* **@param** int, seran las vidas maximas de la creatura.

\*/

**public** Creature(**int** posX, **int** posY, Image imagen, **int** vidaMax)

{

**super**(posX,posY,imagen);

setVidaMax(vidaMax);

iniciaVida();

}

/\*\*

\* Metodo constructor de Creature usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** anim es la <code>Animation</code> del objeto.

\*/

**public** Creature(**int** posX, **int** posY, Animation anim){

**super**(posX, posY, anim);

}

/\*\*

\* Metodo <I>getMaxSpeed</I> regresa la maxima velocidad posible.

\* **@return** 0, esto significa que el movimiento no tendra ningun tipo de restriccion

\* al momento de moverse

\*/

**public** **float** getMaxSpeed() {

**return** 0;

}

/\*\*

\* Metodo <I>wakeUp</I>

\* Despierta a la creatura (usado para que se despierten cuando

\* el personaje este en pantalla y que antes de eso no se muevan)

\*/

**public** **void** wakeUp() {

**if** (getState() == *STATE\_NORMAL* && getVelocidadX() == 0)

setVelocidadX(-getMaxSpeed());

}

/\*\*

\* Metodo <I>getState</I> que devuelve el Estado de la Creature.<P>

\* **@return** state, regresa el estado de la creatura.

\*/

**public** **int** getState() {

**return** state;

}

/\*\*

\* Metodo <I>setState</I> que cambia el Estado de la Creature.<P>

\*/

**public** **void** setState(**int** state) {

**if** (**this**.state != state) //Si el estado a cambiar es diferente al estado realiza el proceso

{

**this**.state = state; //asignacion del estado

stateTime = 0; //el tiempo que ha estado, como se ha cambiado se inicializa en 0

**if** (state == *STATE\_DYING*)

{

setVelocidadX(0); //detiene el movimiento de la creatura

setVelocidadY(0); //detiene el movimiento de la creatura

}

}

}

/\*\*

\* Metodo <I>isAlive</I> que devuelve un boolean para indicar

\* **@return** boolean, si la creatura no ha sido destruido regresa true,

\* si la creatura creature ha sido destruida regresa false

\*/

**public** **boolean** isAlive() {

**return** (state == *STATE\_NORMAL*);

}

/\*\*

\* Metodo <I>isFlying</I> que devuelve un boolean para indicar

\* si Creature esta volando o no.

\* **@return** false si la creatura no esta volando, true si esta volando

\*/

**public** **boolean** isFlying() {

**return** **false**;

}

/\*\*

\* Metodo <I>collideHorizontal</I> si la creatura choca con alguna pared

\* este metodo se encarga de voltearlo.

\*/

**public** **void** collideHorizontal() {

setVelocidadX(-getVelocidadX());

}

/\*\*

\* Metodo <I>collideVertical</I> que frena verticalmente a Creature.<P>

\* Si choca con un techo la velocidad en Y se vuelve cero y empieza su

\* caida libre.

\*/

**public** **void** collideVertical() {

setVelocidadY(0);

}

/\*\*

\* Metodo <I>update</I> que actualiza el estado de Creature.

\* **@return** long, el tiempo en que durar entre estados de muriendo

\* y muerto

\*/

**public** **void** update(**long** elapsedTime) {

stateTime += elapsedTime; //El tiempo que ha transcurrido en este estado mas un diferencial de tiempo

**if** (state == *STATE\_DYING* && stateTime >= *DIE\_TIME*)

{

setState(*STATE\_DEAD*);

}

}

}

**import** javax.swing.ImageIcon;

**import** java.awt.Image;

**import** java.awt.Toolkit;

/\*\*

\* Clase Enemigo, es la clase encargada de darle los aspectos

\* de inteligencia a los enemigos y sus comportamientos

\*/

**public** **class** Enemigo **extends** Creature{

//Animaciones de los enemigos: Movimientos

**private** Animation caminaDerecha = **new** Animation();

//Animaciones de los enemigos: Ataque

**private** Animation atacaDerecha = **new** Animation();

//Animacion del enemigo: StandBy

**private** Animation paradoDerecha = **new** Animation();

//El estado actual en que se encuentra el enemigo

**private** String estadoCaminaDerecha = "caminaDerecha";

**private** String estadoCaminaIzquierda = "caminaIzquierda";

Sprite contrincante; //El jugador sera considerado el contrincante, util para saber cuando atacara el enemigo

**private** **int** count;

**private** **float** oldX; //Velocidad antigua, siempre se esta cambiando cuando hay algun cambio de estado

**private** **long** tiempoDeAtaque; //la duracion del ataque

**private** **long** deltaEntreAtaques; //Tiempo entre ataques

**private** **long** momentoDeInicioDeAtaque; //cuando empieza a atacar despues de haber detectado al jugador

/\*\*

\* Constructor <I>Enemigo</I> para crear un objeto de tipo Enemigo.

\* **@param** x, posicion int del objeto en el eje X

\* **@param** y, posicion int del objeto en el eje Y

\* **@param** imagen, Imagen con la que se inicializa el Enemigo

\*/

**public** Enemigo(**int** x, **int** y, Image imagen){

**super**(x, y, imagen); //metodo heredad del padre

setVelocidadX(0.05f); //hace que la velocidad sea constante

deltaEntreAtaques = 500; //tiempo entre ataques

createAnimation(); //creacion de la animacion de ataque

count = 0; //el contador se inicializa en 0

setLado(1);// + positivo voltea a derecha, - negativo voltea a izquierda

setState(estadoCaminaDerecha); //camina hacia el lado del jugador, por default a la derecha

}

/\*\*

\* Metod void <I>revisarCambioDeEstado</I> que revisa el estado y/o cambia el estado del Enemigo.

\* **@param** identificador, String que compara cualquier estado con el estado actual del Enemigo

\* **@param** newX, float que asigna una nueva posicion para el Enemigo

\*/

**public** **void** revisarCambioDeEstado(String identificador, **float** newX)

{

**if**(identificador == Sprite.*PROXIMA\_CAIDA\_LIBRE* || identificador == Sprite.*CHOQUE\_CON\_PARED*)

{

**float** vel = dx\*(-1);

setVelocidadX(vel);

setLado((**int**)(100\*vel));

**if**(getCurrState() == estadoCaminaDerecha && newX - TileMapRenderer.*TILE\_SIZE* > 0)

setState(estadoCaminaIzquierda);

**else**{

setState(estadoCaminaDerecha); //Cambia el estado a caminar a la derecha

setAnimation(caminaDerecha); //Cambia la orientacion de las animaciones

}

}

**else** **if**(identificador == Sprite.*ESTA\_CERCA*) //Dependiendo si el jugador esta cerca:

{

momentoDeInicioDeAtaque = System.*currentTimeMillis*();

setAnimation(paradoDerecha);

setLado((**int**)(100\*dx));

oldX = dx; //oldX en este caso corresponde a la velocidad

setState(Sprite.*ESTA\_CERCA*);

dx=0;

}

//Sigue caminando si estaba en estado de atacar pero ya no esta cerca el jugador

**else** **if**(identificador == Sprite.*NO\_ESTA\_CERCA* && getCurrState() != estadoCaminaDerecha && getCurrState() != estadoCaminaIzquierda)

{

setAnimation(caminaDerecha);

**if**(volteaADerecha())

dx=Math.*abs*(oldX);

**else**

dx=Math.*abs*(oldX) \* -1;

**if**(dx>0)

{

setState(estadoCaminaDerecha);

}

**else** **if**(dx<0){

setState(estadoCaminaIzquierda);

}

}

}

/\*\*

\* Metodo <I>atacar</I> hace que el Enemigo ataque dependiendo si esta cerca el jugador

\*/

**public** **boolean** atacar(){

**if**(getCurrState() == *ESTA\_CERCA*){

**if**(System.*currentTimeMillis*() - momentoDeInicioDeAtaque > (deltaEntreAtaques + tiempoDeAtaque)){

setAnimation(paradoDerecha);

count = 0;

momentoDeInicioDeAtaque = System.*currentTimeMillis*();

}

**else** **if**(System.*currentTimeMillis*() - momentoDeInicioDeAtaque > deltaEntreAtaques){

count++;

setAnimation(atacaDerecha);

**if**(count < 2)

**return** **true**;

}

}

**return** **false**;

}

/\*\*

\* Metodo <I>persigueA</I>

\* Determina si hay un jugador cerca y le asigna al enemigo atacar al jugador

\* **@param** contrincante, Sprite al cual debe seguir el Enemigo

\*/

**public** **void** persigueA(Sprite contrincante){

**this**.contrincante = contrincante;

}

/\*\*

\* Metodo Tipo Bala <I>dibujaBala</I>

\* Encargado de obtener la imagen correspondiente y crear la bala en su posicion inicial

\* **@param** derecha, booleano que indica true si esta orientado a la derecha el Enemigo

\* **@param** mapa, TileMap al cual se agregan las balas

\*/

**public** Bala dibujaBala(**boolean** derecha, TileMap mapa){

**int** lado = (derecha)? 1:-1;

Bala bala = **new** Bala((**int**)getPosX() + 50\*lado, (**int**)(getPosY() + getAlto()/4),

Toolkit.*getDefaultToolkit*().getImage("images/IMG\_HECTOR/Enemigo/Ataca/Efectoataca/efecto08.png"),

lado\*0.5f);

bala.setTipoDisparo(4);

**return** bala;

}

/\*\*

\* Metodo <I>createAnimation</I> para cargar las imagenes de animacion del enemigo.<P>

\*/

**public** **void** createAnimation(){

//Imagenes del enemigo moviendose

Image caminaD1 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina1.png").getImage();

Image caminaD2 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina2.png").getImage();

Image caminaD3 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina3.png").getImage();

Image caminaD4 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina4.png").getImage();

Image caminaD5 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina5.png").getImage();

Image caminaD6 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina6.png").getImage();

Image caminaD7 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina7.png").getImage();

Image caminaD8 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Camina/alien1camina8.png").getImage();

//Animacion de los enemigos moviendose

caminaDerecha.addFrame(caminaD1, 120);

caminaDerecha.addFrame(caminaD2, 120);

caminaDerecha.addFrame(caminaD3, 120);

caminaDerecha.addFrame(caminaD4, 120);

caminaDerecha.addFrame(caminaD5, 120);

caminaDerecha.addFrame(caminaD6, 120);

caminaDerecha.addFrame(caminaD7, 120);

caminaDerecha.addFrame(caminaD8, 120);

//Imagenes de los enemigos atacando

Image alienA1 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Ataca/alien1ataca1.png").getImage();

Image alienA2 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Ataca/alien1ataca2.png").getImage();

Image alienA3 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Ataca/alien1ataca3.png").getImage();

Image alienA4 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Ataca/alien1ataca4.png").getImage();

Image alienA5 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Ataca/alien1ataca5.png").getImage();

Image alienA6 = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/Ataca/alien1ataca6.png").getImage();

//Animacion del enemigo atacando

atacaDerecha.addFrame(alienA1, 120);

atacaDerecha.addFrame(alienA2, 120);

atacaDerecha.addFrame(alienA3, 120);

atacaDerecha.addFrame(alienA4, 120);

atacaDerecha.addFrame(alienA5, 120);

atacaDerecha.addFrame(alienA6, 120);

//Imagen del enemigo en StandBy

Image paradoD = **new** ImageIcon("images/IMG\_HECTOR/Enemigo/alien1parado.png").getImage();

tiempoDeAtaque = 120\*6; //tiempoDeAtaque = sumatoria -> tiempo de cada frame de ataque

paradoDerecha.addFrame(paradoD, 1000);

setAnimation(caminaDerecha);

}

}

**import** java.awt.Image;

**import** java.awt.Toolkit;

**public** **class** Explosiones **extends** Sprite {

//Animacion de la explosion, tiempo de duracion en la pantalla desde que aparece

**private** Animation animExp;

**long** tiempoCreacion;

**int** tiempoAnim = 350;

**int** tiempoTotal =0;

/\*\*

\* Metodo constructor usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\*/

**public** Explosiones(**int** posX, **int** posY, Image image) {

**super**(posX,posY,image);

cargaImagenes();

tiempoCreacion = System.*currentTimeMillis*();

}

/\*\*

\* Metodo <I>updateAnim()</I> <P>

\* cambia la animacion de la explosion.

\* Si ha pasado cierto tiempo desde que comenzo la animacion,

\* se cambia de animacion y se cambia el Estado.

\*/

**public** **void** updateAnim(**long** elapsedTime){

tiempoTotal += elapsedTime;

**if**( tiempoTotal < tiempoAnim)

animExp.update(elapsedTime);

**else**

animExp.start();

setImageIcon(animExp.getImage());

}

/\*\*

\* Metodo Boleano <I>Termina()</I> <P>

\***@return** Booleano si la explosion debe de quitarse o no de la pantalla

\*/

**public** **boolean** termina(){

**if**(tiempoTotal > tiempoAnim){

**return** **true**;

}

**return** **false**;

}

/\*\*

\* Metodo <I>cargaImagenes()</I> <P>

\* Carga todas las imagenes de Explosiones que se usaran

\*/

**public** **void** cargaImagenes(){

// Animacion de Explosion

Image a1 = Toolkit.*getDefaultToolkit*().getImage("images/explosion1.png");

Image a2 = Toolkit.*getDefaultToolkit*().getImage("images/explosion2.png");

Image a3 = Toolkit.*getDefaultToolkit*().getImage("images/explosion3.png");

Image a4 = Toolkit.*getDefaultToolkit*().getImage("images/explosion4.png");

Image a5 = Toolkit.*getDefaultToolkit*().getImage("images/explosion5.png");

Image a6 = Toolkit.*getDefaultToolkit*().getImage("images/explosion6.png");

Image a7 = Toolkit.*getDefaultToolkit*().getImage("images/explosion7.png");

// Animacion de las explosiones

animExp = **new** Animation();

animExp.addFrame(a1, 50);

animExp.addFrame(a2, 50);

animExp.addFrame(a3, 50);

animExp.addFrame(a4, 50);

animExp.addFrame(a5, 50);

animExp.addFrame(a6, 50);

animExp.addFrame(a7, 50);

}

}

/\*\*

\* Clase GameMain

\*

\* JUEGO La Batalla de Ciudad Juanez

\*

\* Utiliza Game Programming

\* Es un shooter simple.

\* **@author** Samuel Heaney

\* **@author** Carlos Guillermo Elizondo Ancer

\* **@author** Jorge Saldivar

\* **@author** Hector de la Garza

\* **@version** 4.01 14/04/2010

\*/

**import** java.awt.Font;

**import** java.awt.Image;

**import** java.awt.Graphics;

**import** java.awt.Graphics2D;

**import** java.awt.Toolkit;

**import** java.awt.Dimension;

**import** java.awt.Color;

**import** java.awt.event.KeyListener;

**import** java.awt.event.KeyEvent;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Collections;

**import** java.util.Vector;

**import** java.util.Enumeration;

**import** javax.swing.ImageIcon;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.JFrame;

**import** java.awt.Point;

**import** java.awt.SplashScreen;

**public** **class** GameMain **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** *serialVersionUID* = 1L;

// Estados del juego.

**static** **enum** State {

*M\_JUEGO*, *M\_INS*, *M\_OP*, *M\_PUN*, *M\_SALIR*, *INSTRUCCIONES*, *OPCIONES*, *PUNTAJE*, *JUEGO*, *PAUSA*, *GAMEOVER*, *WIN*, *LOSE*

}

// Constantes de juego

**static** State *state*;

Dimension screenSize = Toolkit.*getDefaultToolkit*().getScreenSize();

**private** **static** **final** **int** *WIDTH*= 800;

**private** **static** **final** **int** *HEIGHT*= 600;

**public** **static** **final** **float** *GRAVITY* = 0.002f;

**public** **static** **final** **float** *VELOCIDADBALA* = .7f;

**public** **static** **final** **float** *VELOCIDADCAIDA* = 7;

**private** Point pointCache = **new** Point();

**private** Image dbImage;

**private** Graphics dbg;

//Se inicializan las imagenes del menu

**private** ImageIcon M\_INSz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0002.png"));

**private** ImageIcon M\_JUEGOz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0001.png"));

**private** ImageIcon M\_OPz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0003.png"));

**private** ImageIcon M\_SALIRz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0004.png"));

**private** ImageIcon M\_PUNz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0007.png"));

**private** ImageIcon instruccionesz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0005.png"));

**private** ImageIcon opcionesz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0006.png"));

**private** ImageIcon puntajez = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/menu0008.png"));

**private** ImageIcon pausaz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/pausa.png"));

**private** ImageIcon winz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/ganaste.png"));

**private** ImageIcon perderz = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/perdiste.png"));

//Se llaman las imagenes del menu que se inicializaron

**private** Image M\_INS = M\_INSz.getImage();

**private** Image M\_JUEGO = M\_JUEGOz.getImage();

**private** Image M\_OP = M\_OPz.getImage();

**private** Image M\_SALIR = M\_SALIRz.getImage();

**private** Image instrucciones = instruccionesz.getImage();

**private** Image opciones = opcionesz.getImage();

**private** Image puntaje = puntajez.getImage();

**private** Image M\_PUN = M\_PUNz.getImage();

**private** Image pausa = pausaz.getImage();

**private** Image win = winz.getImage();

**private** Image perder = perderz.getImage();

//elapsedTime sirve para mantener el tiempo del juego

**private** **long** elapsedTime;

**private** TileMap mapa;

**private** **int** vidas;

**private** **int** puntos;

@SuppressWarnings("unused")

**private** **boolean** izq,der,arriba,abajo,space;

**private** TileMapRenderer render;

**private** **boolean** attack;

**private** Vector<Sprite> otrasImagenes;

**private** Player jugador;

**private** Vector <Bala> balas;

@SuppressWarnings("unused")

**private** Vector <Bala> balasEnemigos;

**private** **boolean** sueltaBotonBala;

//Se llaman las clases

**private** Menu menu;

**private** Juego juego;

//variables para mantener un orden en los sonidos

**private** **int** prendeSonidoTemporal;

**private** **int** intParaPausa;

**private** **int** seleccionOpciones;

**private** **int** seleccionMusicaOn=0;

**private** **int** seleccionMusica=0;

**private** **int** nivelMapa=0;

//Variables utilizadas para el Puntaje

**private** Vector<Puntaje> vec; // Objeto vector para agregar el puntaje

**private** String nombreArchivo; //Nombre del archivo

**private** String[] arr; //Arreglo del archivo divido

//Variables para el sonido en general

**private** SoundClip []musica;

**private** SoundClip sonidoDisparo;

**private** SoundClip sonidoDisparoMalo;

**private** SoundClip sonidoPausa;

/\*\*

\* Constructor de la clase.

\*/

**public** GameMain() **throws** IOException {

sonidoDisparo = **new** SoundClip("sonidos/laser.wav");

sonidoDisparoMalo = **new** SoundClip("sonidos/lasermalo.wav");

sonidoPausa = **new** SoundClip("sonidos/Pause.wav");

//se inicalizan las canciones en un arreglo de SoundClip

musica = **new** SoundClip[3];

musica[0] = **new** SoundClip("sonidos/rock.wav"); //sonido de fondo

musica[0].setLooping(**true**);//Para que siempre se escuche el sonido de fondo.

musica[1] = **new** SoundClip("sonidos/hiphop.wav"); //sonido de fondo

musica[1].setLooping(**true**);//Para que siempre se escuche el sonido de fondo.

musica[2] = **new** SoundClip("sonidos/electronica.wav"); //sonido de fondo

musica[2].setLooping(**true**);//Para que siempre se escuche el sonido de fondo.

//Se inicalizan las variables

gameInit();

juego = **new** Juego();

menu = **new** Menu();

//options = new Options();

menu.setPreferredSize(**new** Dimension(*WIDTH*, *HEIGHT*));

juego.setPreferredSize(**new** Dimension(*WIDTH*, *HEIGHT*));

//options.setPreferredSize(new Dimension(WIDTH, HEIGHT));

**this**.setContentPane(menu);

validate();

**this**.setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);

**this**.pack();

**this**.setTitle("Batalla de Cd. Juanez");

**this**.setVisible(**true**);

gameStart();

}

/\*\*

\* Metodo <I>gameInit</I> <P>

\* En este metodo se inizializan las variables o se crean los objetos

\* a usarse en el juego.

\*/

**public** **void** gameInit() **throws** IOException {

vidas = 1; // Numero de vidas iniciales

*state* = State.*M\_JUEGO*; // Empieza en el canvas de Menu

balas = **new** Vector <Bala>(); // Vector de balas

nombreArchivo = "Puntaje.txt"; //nombre del archivo

vec = **new** Vector<Puntaje>(); //vector para puntaje

prendeSonidoTemporal=1; //sonidoTemporal prende el sonido

seleccionOpciones=0; //primera seleccion en opciones

intParaPausa=1; //con intParaPausa es 1, no hay pausa

balasEnemigos = **new** Vector <Bala>(); // Vector de balas de enemigos

sueltaBotonBala = **true**; // El boton de disparo esta suelto

puntos = 0;

der=**false**;

arriba=**false**;

abajo=**false**;

izq=**false**;

space=**false**;

mapa = **new** TileMap(10,10);

mapa = mapa.loadMap(nivelMapa);

render = **new** TileMapRenderer();

ImageIcon back = **new** ImageIcon(Toolkit.*getDefaultToolkit*().getImage("images/backgroundMejorado.png"));

Image backback = back.getImage();

render.setBackground(backback);

}

/\*\*

\* Metodo <I>gameShutdown</I> <P>

\* Metodo utilizado para cerrar la pantalla y terminar el juego.

\*/

**public** **void** gameShutdown() {

System.*exit*(0); //Cierra el juego, termina la aplicacion

}

/\*\*

\* Metodo <I>gameStart</I> <P>

\* En este metodo se crea e inicializa el hilo para la animacion.

\*/

**public** **void** gameStart() {

Thread t = **new** Thread() {

// Override run() to provide the running behavior of this thread.

**public** **void** run() {

gameLoop();

}

};

// Se inicia el hilo que a su vez llama al gameLoop

t.start();

}

/\*\*

\* Metodo <I>gameLoop</I> <P>

\* En este metodo se maneja el tiempo en el que corre y se actualiza el juego

\* hasta el momento en que termina bajo determinada condicion.

\*/

**private** **void** gameLoop() {

**long** startTime = System.*currentTimeMillis*();

**long** currTime = startTime;

//mientras no exista un GameOver sigue corriendo el juego

**while** (**true**) {

elapsedTime = System.*currentTimeMillis*() - currTime;

currTime += elapsedTime;

//llama el puntaje para poder desplegarlo cuando el estado es igual a puntaje

**if**(*state* == State.*PUNTAJE*){

**try**{

leeArchivo(); //lee el contenido del archivo

}**catch**(IOException e){

System.*out*.println("Error en " + e.toString());

}

}

//se sale del juego cuando existe esta condicion

**if** (*state* == State.*GAMEOVER*) {

**break**;

}

//para prender la musica y se pueda escuchar en el fondo

**if**(prendeSonidoTemporal==1&&seleccionMusicaOn==0){

musica[seleccionMusica].play();

prendeSonidoTemporal=2;

}

//apaga la musica

**if**(seleccionMusicaOn!=0){

musica[seleccionMusica].stop();

prendeSonidoTemporal=1;

}

//se mantiene en el juego mientras el estado esta en juego

**if**(*state* == State.*JUEGO*) {

gameUpdate();

**if**(mapa.getPuntos()>=500&&nivelMapa==0){

nivelMapa=1;

puntos=mapa.getPuntos();

**try** {

mapa = mapa.loadMap(nivelMapa);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

mapa.setPuntos(puntos);

}

**if**(mapa.getPuntos()>=1000&&nivelMapa==1){

nivelMapa=2;

puntos=mapa.getPuntos();

**try** {

mapa = mapa.loadMap(nivelMapa);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

mapa.setPuntos(puntos);

}

**if**(mapa.getPuntos()>=1550&&nivelMapa==2)

vidas=0;

**if** (vidas <= 0) { // Si las vidas son menos o igual a 0, se llega a GameOver

musica[seleccionMusica].stop();

**if**(mapa.getPuntos()>=1550&&nivelMapa==2){

*state* = State.*WIN*;

}

**else**{

*state* = State.*LOSE*;}

**this**.setContentPane(menu); // Se cambia de canvas

menu.requestFocus();

validate();

vidas=1;

String nombre = JOptionPane.*showInputDialog*("Nombre del Jugador");

**if**(nombre.length()==0){

nombre="Jugador";

}

JOptionPane.*showMessageDialog*(**null**,"El puntaje de " + nombre + " es: " + mapa.getPuntos(),"PUNTAJE", JOptionPane.*PLAIN\_MESSAGE*);

//maneja el puntaje

**try**{

vec.add(**new** Puntaje(nombre,mapa.getPuntos())); //Agrega el contenido del nuevo puntaje al vector.

grabaArchivo(); //Graba el vector en el archivo.

}**catch**(IOException e){

System.*out*.println("Error en " + e.toString());

}

**switch**(nivelMapa){

**case** 0:

nivelMapa=0;

**break**;

**case** 1:

nivelMapa=0;

**break**;

**case** 2:

nivelMapa=0;

**break**;

}

**try**{

gameInit();

}

**catch**(Exception e){

}

}

}

repaint(); // Se repinta el contenido

**try** {

// El thread se duerme.

Thread.*sleep* (20);

}

**catch** (InterruptedException ex) {

// no hace nada.

}

}

}

/\*\*

\* Metodo <I>gameUpdate</I> <P>

\* En este metodo se actualizan las posiciones de los personajes y objetos en el juego

\* en caso de que se presente algun tipo de accion.

\*/

@SuppressWarnings("unchecked")

**public** **void** gameUpdate() {

otrasImagenes = mapa.getSprites(); // Consigue el mapa

Player jugador = mapa.getPlayer(); // Consigue al jugador

**float** velocityX = 0;

**if** (der) {

velocityX += jugador.getMaxSpeed(); // Si esta presionada la derecha, la velocidad aumenta

}

**if** (izq) {

velocityX -= jugador.getMaxSpeed(); // Si esta presionada la izquierda, la velocidad disminuye

}

**if** (space) {

jugador.jump(**false**); // Obliga un salto. Envia parametro false para indicar que es salto normal (no esta rebotando)

}

**if** (attack) { // Si se presiono la tecla de ataque, se llama al metodo disparo()

jugador.disparo();

attack = **false**;

**if**(seleccionMusicaOn==0)

sonidoDisparo.play();

}

//Actualiza a los enemigos. Los obliga a atacar si el jugador esta cerca

**for** (**int** x = 0; x < otrasImagenes.size(); x++){

Sprite sprite = (Sprite)otrasImagenes.get(x);

sprite.actualiza(elapsedTime);

sprite.checaProximidad(jugador);

sprite.updateAnim(elapsedTime);

**if**(sprite.atacar()){

**if**(seleccionMusicaOn==0)

sonidoDisparoMalo.play();

mapa.addBalaEnemigos(sprite.dibujaBala(sprite.volteaADerecha(), mapa));

}

**if**(sprite.puedeMorir()){

mapa.removeSprite(sprite);

mapa.setPuntos(mapa.getPuntos()+50);

}

}

jugador.setVelocidadX(velocityX); // Cambia al jugador la velocidad horizontal dependiendo de que tecla se presiono

checaColision(); // Checa la colision del jugador con el mapa y actualiza posicion

jugador.updateAnim(elapsedTime); // Actualiza la animacion dle jugador

//Para que el jugador reaparezca despues de caerse

**if** ((jugador.getPosY() > TileMapRenderer.*tilesToPixels*(mapa.getHeight())) || (jugador.getVida() == 0)) {

jugador.setPosY(80); // Se coloca al jugador en una posicion determinada en Y

jugador.setPosX(TileMapRenderer.*tilesToPixels*(3)); // Se coloca al jugador en una posicion determinada en X

jugador.cambiaVida(15); // Se llena la vida del jugador

vidas --;

}

Vector<Explosiones> explosiones = mapa.getExplosiones();

Enumeration e = explosiones.elements();

Explosiones exp;

**while** (e.hasMoreElements()) {

exp = (Explosiones) e.nextElement();

exp.updateAnim(elapsedTime);

**if**(exp.termina()){

mapa.removeExplosion(exp);

}

}

//Seccion de BALAS

//Actualiza balas del jugador

balas = mapa.getBalas(); // Se consigue un vector de balas del jugador

e = balas.elements();

Bala b;

**while** (e.hasMoreElements()) { // Se recorre el vector

b = (Bala) e.nextElement();

b.actualiza(elapsedTime); // Se actualiza la posicion de la bala

b.updateAnim(elapsedTime);

b.cambiaInicial(b.getInicial() + 0.8f); // Se incrementa un contador de distancia recorrida por la bala

}

//Se encarga de borrar balas que ya hayan durado cierto tiempo en pantalla

e = balas.elements();

**while** (e.hasMoreElements()) {

b = (Bala) e.nextElement();

b.actualizaPosicionI();

**float** oldX = b.getPosX();

**float** newX = oldX ;

**if** (b.getInicial() > 50f || getTileCollision(b,newX,b.getPosY())!=**null**) { // Si la bala ya recorrio mas de la cantidad especificada, se quita del vector

mapa.removeBala(b);

**if** (getTileCollision(b,newX,b.getPosY())!=**null**)

mapa.addExplosion(**new** Explosiones((**int**)b.getPosX(),(**int**)b.getPosY(),Toolkit.*getDefaultToolkit*().getImage("images/explosion1.png")));// El objeto bala es removido del vector

}

**for** (**int** x = 0; x < otrasImagenes.size(); x++) { // Si una bala toca al enemigo, ambos desaparecen.

Sprite sprite = (Sprite)otrasImagenes.get(x);

**if** (b.intersecta(sprite)) {

sprite.cambiaVida(-b.getDamage());

mapa.removeBala(b);

mapa.addExplosion(**new** Explosiones((**int**)b.getPosX(),(**int**)b.getPosY(),Toolkit.*getDefaultToolkit*().getImage("images/explosion1.png")));

}

}

}

//BALAS ENEMIGOS/

balas = mapa.getBalasEnemigos(); // Se consigue el vector de balas de Enemigos

e = balas.elements();

**while** (e.hasMoreElements()) {

b = (Bala) e.nextElement();

b.actualiza(elapsedTime);

b.cambiaInicial(b.getInicial() + 0.8f);

b.updateAnim(elapsedTime);

**float** dx = b.getVelocidadX();

**float** oldX = b.getPosX();

**float** newX = oldX + dx\*elapsedTime;

**if**(b.intersecta(jugador)){

jugador.cambiaVida(b.getDamage()\*-1);

mapa.removeBalaEnemigos(b);

mapa.addExplosion(**new** Explosiones((**int**)b.getPosX(),(**int**)b.getPosY(),Toolkit.*getDefaultToolkit*().getImage("images/explosion1.png")));

}

//Despues de cierta distancia se remueve la bala de la pantalla

**if**(Math.*abs*(b.getInicial()) > 20f || getTileCollision(b, newX, b.getPosY()) != **null**){

mapa.removeBalaEnemigos(b);

}

}

}

/\*\*

\* Metodo <I>checaColision</I> <P>

\* En este metodo se manejan todo tipo de colisiones involucrando personajes.

\*/

**public** **void** checaColision() {

Player jugador = mapa.getPlayer(); // Consigue al jugador

Vector<Sprite> otrasImagenes = mapa.getSprites(); // Consigue a los enemigos

// Seccion Enemigos

**for**(**int** x = 0; x < otrasImagenes.size(); x++) { // Se recorre el vector de enemigos

Sprite sprite = (Sprite)otrasImagenes.get(x);

**float** dx = sprite.getVelocidadX(); // Se cosnigue la velcoidad actual

**float** oldX = sprite.getPosX(); // Se consigue la posicion actual

**float** newX = oldX + dx\*elapsedTime; // Se consigue la posicion a la que se va a mover

sprite.revisarCambioDeVuelta(mapa, oldX, sprite.getPosY(), newX); // Se revisa si esta en esquina, para obligar al alien a girar

Point punto = getTileCollision(sprite, newX, sprite.getPosY()); // Se revisa si hay colision en donde se va a mover

**if**(punto != **null**){

sprite.revisarCambioDeEstado(Sprite.*CHOQUE\_CON\_PARED*, newX); // Si no hay colision, se revisa si el alien necesita cambiar de estado

sprite.setPosX(TileMapRenderer.*tilesToPixels*(TileMapRenderer.*pixelsToTiles*(oldX))); //Se actualiza la posicion del alien

}

**if** (sprite.intersecta(jugador)){ // Se revisa si el enemigo toca al jugador

**boolean** izquierda = jugador.getLado();

**if**(izquierda){ // Dependiendo del lado del jugador, este es empujado

jugador.setPosX(jugador.getPosX() + sprite.getAncho());

} **else** {

jugador.setPosX(jugador.getPosX() - sprite.getAncho());

}

jugador.cambiaVida(-1); // Se reduce la vida del jugador

}

**if**(sprite.puedeMorir())

mapa.removeSprite(sprite);

}

//////////////FIN DE SECCION ENEMIGOS///////////////////

**float** dx = jugador.getVelocidadX(); //Velocidad del jugador

**float** oldX = jugador.getPosX(); //Vieja posicion

**float** newX = oldX + dx \* elapsedTime; //Nueva posicion = Vieja posicion + Velocidad\*tiempo desde ultimo checaColision

Point punto = getTileCollision(jugador, newX, jugador.getPosY()); //Checa si hubo colision o no, ver metodo getTileCollision

**if**(punto==**null**){

jugador.setPosX(newX); //Si no hubo colision, ahora si, actualiza la posicion Horizontal del jugador

}

jugador.setVelocidadY(jugador.getVelocidadY() + jugador.getGravity() \* elapsedTime); //Velocidad vertical = VelocidadY + Gravedad por el tiempo transcurrido

**float** dy = jugador.getVelocidadY(); //Velocidad en Y

**float** oldY = jugador.getPosY(); //Vieja posicion en Y

**float** newY = oldY + dy \* elapsedTime; //Posicion a la que se quiere mover= vieja posicion + velocidad\*tiempo

punto = getTileCollision(jugador, jugador.getPosX(), newY); //Checa si hay o no colision con algun Tile de piso

**if** (punto == **null**) {

jugador.setPosY(newY); //Si no hay colision (actualiza la posicion en Y)

}

**else**{

**if** (dy > 0) {

jugador.setPosY(TileMapRenderer.*tilesToPixels*(punto.y) - jugador.getAlto());

} **else** **if** (dy < 0) {

jugador.setPosY(TileMapRenderer.*tilesToPixels*(punto.y+1));

}

jugador.collideVertical(); //Si hay colision, cambia la velocidad del personaje a 0 y avisa que esta en el piso (ver Clase Player)

}

}

**public** Point getTileCollision(Sprite sprite,**float** newX, **float** newY) //Recibe como parametros el sprite y la poscion a la que se quiere mover

{

//Checa de donde a donde se quiere mover (en pixeles)

**float** fromX = Math.*min*(sprite.getPosX(), newX);

**float** fromY = Math.*min*(sprite.getPosY(), newY);

**float** toX = Math.*max*(sprite.getPosX(), newX);

**float** toY = Math.*max*(sprite.getPosY(), newY);

// Checa en que valor Entero estan ubicados los Tiles (que pueden o no existir)

// Esto es, asigna a los siguientes integers la posicion que deberian tener los Tiles de dicha area

**int** fromTileX = TileMapRenderer.*pixelsToTiles*(fromX);

**int** fromTileY = TileMapRenderer.*pixelsToTiles*(fromY);

**int** toTileX = TileMapRenderer.*pixelsToTiles*(toX + sprite.getAncho() - 1);

**int** toTileY = TileMapRenderer.*pixelsToTiles*(toY + sprite.getAlto() - 1);

// Recorre el arreglo de Tiles. Ahora si checa si hay un Tile o no en la posicion especificada

**for** (**int** x = fromTileX; x <= toTileX; x++) {

**for** (**int** y = fromTileY; y <= toTileY; y++) {

**if** (x < 0 || x >= mapa.getWidth() || mapa.getTile(x, y) != **null**)

{

//Si se encuentra un tile en la posicion, hay colision

pointCache.setLocation(x, y);

**return** pointCache;

}

}

}

//No hubo colision

**return** **null**;

}

/\*\*

\* Metodo <I>gameDraw</I> <P>

\* Metodo usado para dibujar en el frame de acuerdo

\* al estado en el que se encuentra el juego.

\* **@param** g2d, es el objeto Graphics2D en el cual se dibuja

\*/

**private** **void** gameDraw(Graphics2D g2d) {

**switch** (*state*) {

**case** *M\_JUEGO*:

g2d.drawImage(M\_JUEGO, 0,0, menu); //se dibuja la imagen de seleccion juego

**break**;

**case** *M\_INS*:

g2d.drawImage(M\_INS, 0,0, menu); //se dibuja la imagen de seleccion instrucciones

**break**;

**case** *M\_OP*:

g2d.drawImage(M\_OP, 0,0, menu); //se dibuja la imagen de seleccion opciones

**break**;

**case** *M\_SALIR*:

g2d.drawImage(M\_SALIR, 0,0, menu); //se dibuja la imagen de seleccion salir

**break**;

**case** *M\_PUN*:

g2d.drawImage(M\_PUN, 0,0, menu); //se dibuja la imagen de seleccion puntaje

**break**;

**case** *INSTRUCCIONES*:

g2d.drawImage(instrucciones, 0,0, menu); //se dibuja la imagen de instrucciones

**break**;

**case** *OPCIONES*:

g2d.drawImage(opciones, 0,0, menu); //se dibuja la imagen de opciones

**switch**(seleccionOpciones){ //se dibujan las seleccion de sonidos y musica

**case** 0:

g2d.setColor(Color.*WHITE*);

g2d.drawRect(100, 90, 100, 10);

**break**;

**case** 1:

g2d.setColor(Color.*WHITE*);

g2d.drawRect(100, 200, 100, 10);

**break**;

}

**switch**(seleccionMusica){ //se dibuja la seleccion de musica mediante un rectangulo

**case** 0:

g2d.drawRect(*WIDTH*/9, *HEIGHT*/4+*HEIGHT*/6, 160, 70);

**break**;

**case** 1:

g2d.drawRect(*WIDTH*/9+*WIDTH*/4+*WIDTH*/8-50, *HEIGHT*/4+*HEIGHT*/6, 160, 70);

**break**;

**case** 2:

g2d.drawRect(*WIDTH*/9+*WIDTH*/2+*WIDTH*/8-20, *HEIGHT*/4+*HEIGHT*/6, 160, 70);

**break**;

}

**switch**(seleccionMusicaOn){ //se dibuja la seleccion de musica prendida y apagada

**case** 0:

g2d.drawRect(*WIDTH*/2, *HEIGHT*/16, 90, 70);

**break**;

**case** 1:

g2d.drawRect(*WIDTH*/2+*WIDTH*/4, *HEIGHT*/16, 90, 70);

**break**;

}

**break**;

**case** *PUNTAJE*:

g2d.drawImage(puntaje,0,0,**this**); //se dibuja la imajen de puntaje

**int** incremento=10;

**for** (**int** i=0; i<vec.size(); i++) { //despiega el puntaje de los jugadores con los nombres de los jugadores

Puntaje x;

x = (Puntaje) vec.get(i);

g2d.setColor(Color.*BLACK*);

g2d.setFont(**new** Font("Serif", Font.*BOLD*, 38));

g2d.drawString(x.getNombre(),200,250+incremento);

g2d.drawString(""+x.getPuntaje(),450,250+incremento);

incremento+=50;

}

**break**;

**case** *PAUSA*:

g2d.drawImage(pausa, 0,0, menu); //se despiega una imagen de pausa

**break**;

**case** *WIN*:

g2d.drawImage(win,0,0,**this**); //se despliega la pantalla de ganar

**break**;

**case** *LOSE*: //se despliega la pantalla de perder

g2d.drawImage(perder,0,0,**this**);

**break**;

}

}

/\*\*

\* Metodo <I>gameKeyPressed</I> <P>

\* Metodo usado para manejar el evento que ocurre al presionar una

\* tecla en cualquier panel y dependiendo del estado.

\* **@param** keyCode es el <code>evento</code> que se genera en al presionar las teclas.

\*/

**public** **void** gameKeyPressed(**int** keyCode) {

**switch** (keyCode) {

**case** KeyEvent.*VK\_UP*:

//se cambian los estados dependiendo que se oprima la tecla de arriba

**if** (*state* == State.*M\_INS*) {

*state* = State.*M\_JUEGO*;

repaint();

} **else** **if** (*state* == State.*M\_PUN*) {

*state* = State.*M\_INS*;

repaint();

} **else** **if** (*state* == State.*M\_OP*) {

*state* = State.*M\_PUN*;

repaint();

}

**else** **if** (*state* == State.*M\_SALIR*) {

*state* = State.*M\_OP*;

repaint();

}

//se cambia la seleccion de opciones mientras este en el estado de opciones

**else** **if** (*state* == State.*OPCIONES*) {

seleccionOpciones=0;

}

**break**; // Fin del case de Arriba

**case** KeyEvent.*VK\_DOWN*:

//se cambian los estados dependiendo que se oprima la tecla de abajo

**if** (*state* == State.*M\_JUEGO*) {

*state* = State.*M\_INS*;

repaint();

} **else** **if** (*state* == State.*M\_INS*) {

*state* = State.*M\_PUN*;

repaint();

} **else** **if** (*state* == State.*M\_PUN*) {

*state* = State.*M\_OP*;

repaint();

}

**else** **if** (*state* == State.*M\_OP*) {

*state* = State.*M\_SALIR*;

repaint();

} **else** **if** (*state* == State.*JUEGO*) {

abajo=**true**;

}

//se cambia la seleccion de opciones mientras este en el estado de opciones

**else** **if** (*state* == State.*OPCIONES*) {

seleccionOpciones=1;

}

**break**; // Fin del case de tecla Abajo

**case** KeyEvent.*VK\_RIGHT*:

**if** (*state* == State.*JUEGO*) { //se mueve el jugador a la derecha

der=**true**;

}

**if** (*state* == State.*OPCIONES*&&seleccionOpciones==0) { //se apaga el sonido

seleccionMusicaOn=1;

}

**if** (*state* == State.*OPCIONES*&&seleccionOpciones==1) { //se cambia la musica

musica[seleccionMusica].stop();

prendeSonidoTemporal=1;

seleccionMusica++;

**if**(seleccionMusica>=2)

seleccionMusica=2;

}

**break**; // Fin del case de Tecla Derecha

**case** KeyEvent.*VK\_LEFT*:

**if** (*state* == State.*JUEGO*) { //se mueve el jugador a la izquierda

izq=**true**;

}

**if** (*state* == State.*OPCIONES*&&seleccionOpciones==0) { //se prende el sonido

seleccionMusicaOn=0;

}

**if** (*state* == State.*OPCIONES*&&seleccionOpciones==1) { //se cambia la musica

musica[seleccionMusica].stop();

prendeSonidoTemporal=1;

seleccionMusica--;

**if**(seleccionMusica<=0)

seleccionMusica=0;

}

**break**; // Fin del case de Tecla izquierda

**case** KeyEvent.*VK\_S*:

**if** (*state* == State.*JUEGO*) { //brinca el jugador

arriba=**true**;

space = **true**;

}

**break**;

**case** KeyEvent.*VK\_P*:

**if**(*state*==State.*JUEGO*) { //pausa

*state*=State.*PAUSA*;

repaint();

}

**else** **if**(*state*==State.*PAUSA*){ //regresa al juego en caso de pausa

*state*=State.*JUEGO*;

}

**break**;

**case** KeyEvent.*VK\_ENTER*:

**if**(*state* == State.*WIN*){

puntos=0;

*state* = State.*M\_JUEGO*;

}

**if** (*state* == State.*M\_JUEGO*) { // Pantalla de Menu Juego

*state* = State.*JUEGO*;

//musicaMenu.stop();

**this**.setContentPane(juego);

juego.requestFocus();

validate();

} **else** **if** (*state* == State.*M\_INS*) { // Pantalla de Instrucciones

*state* = State.*INSTRUCCIONES*;

repaint();

}

**else** **if** (*state* == State.*INSTRUCCIONES*) { // Pantalla de Instrucciones

*state* = State.*M\_INS*;

repaint();

}

**else** **if** (*state* == State.*M\_OP*) { // Pantalla de opciones

*state* = State.*OPCIONES*;

repaint();

}

**else** **if** (*state* == State.*OPCIONES*) { // Pantalla de opciones

*state* = State.*M\_OP*;

repaint();

}

**else** **if** (*state* == State.*M\_PUN*) { // Pantalla de puntaje

*state* = State.*PUNTAJE*;

repaint();

}

**else** **if** (*state* == State.*PUNTAJE*) { // Pantalla de puntaje

*state* = State.*M\_PUN*;

repaint();

}

**else** **if** (*state* == State.*M\_SALIR*) { //se destruye el juego

System.*exit*(0);

}

**else** **if** (*state* == State.*GAMEOVER*) { // Pantalla de Gameover

gameShutdown(); // Se cierra el juego

}

**break**;

**case** KeyEvent.*VK\_BACK\_SPACE*: // Fin del case de Tecla de Enter

**if** (*state* == State.*INSTRUCCIONES*) { // Pantalla de Instrucciones

*state* = State.*M\_INS*;

repaint();

}

**break**;

**case** KeyEvent.*VK\_ESCAPE*:

**if**(*state*==State.*JUEGO*) { //se acaba el juego

vidas=0;

}

}

}

/\*\*

\* Metodo <I>gameKeyTyped</I> <P>

\* Metodo usado para manejar el evento que ocurre al hacer click una

\* tecla en cualquier panel y dependiendo del estado.

\* **@param** e es el <code>evento</code> que se genera al hacer click en una tecla.

\*/

**public** **void** gameKeyTyped(KeyEvent e) {

**if** (*state* == State.*JUEGO*) {

**if** (e.getKeyChar() == 'a' || e.getKeyChar() == 'A' ) {

**if** (sueltaBotonBala) {

jugador = mapa.getPlayer();

sueltaBotonBala = **false**; // Se avisa que se solto el botono de bala

Bala disparo;

attack = **true**; // Se avisa que se esta atacando

**if** (jugador.getLado()) { // Se crea un nuevo objeto de bala, con velocidad dependiendo del lado del jugador

disparo = **new** Bala((**int**)jugador.getPosX(), (**int**)jugador.getPosY() + (jugador.getAlto() / 2 - 20),

Toolkit.*getDefaultToolkit*().getImage("images/disparocyan1.png"));

disparo.setVelocidadX(-*VELOCIDADBALA*);

} **else** {

disparo = **new** Bala((**int**)jugador.getPosX() + jugador.getAncho(), (**int**)jugador.getPosY() + (jugador.getAlto() / 2 - 20),

Toolkit.*getDefaultToolkit*().getImage("images/disparocyan1.png"));

disparo.setVelocidadX(*VELOCIDADBALA*);

}

disparo.setTipoDisparo(1); // Es una bala de nivel 1

mapa.addBala(disparo); // Se agrega una bala al mapa

}

}

}

}

/\*\*

\* Metodo <I>gameKeyReleased</I> <P>

\* En este metodo maneja el evento que se genera al soltar la tecla presionada en cualquier panel

\* y dependiendo del estado.

\* **@param** e es el <code>evento</code> que se genera en al soltar las teclas.

\*/

**public** **void** gameKeyReleased(KeyEvent e){

**if** (*state* == State.*JUEGO*) {

**if**(e.getKeyCode() == KeyEvent.*VK\_S*) {

space = **false**;

arriba = **false**;

} **else** **if** (e.getKeyCode() == KeyEvent.*VK\_DOWN*) {

abajo = **false**;

} **else** **if** (e.getKeyCode() == KeyEvent.*VK\_RIGHT*) {

der = **false**;

} **else** **if** (e.getKeyCode() == KeyEvent.*VK\_LEFT*) {

izq = **false**;

} **else** **if** (e.getKeyCode() == KeyEvent.*VK\_D*) {

space = **false**;

} **else** **if** (e.getKeyCode() == KeyEvent.*VK\_A*) {

jugador=mapa.getPlayer();

sueltaBotonBala = **true**; // Se avisa quie se solto el boton

}

}

}

/\*\*

\* Clase derivada de <code>JPanel</code> en donde se definen las caracteristicas

\* del panel Menu.

\*/

**class** Menu **extends** JPanel **implements** KeyListener {

**private** **static** **final** **long** *serialVersionUID* = 1L;

// Constructor

**public** Menu() {

setFocusable(**true**); // Para que sea capaz de recibir eventos

requestFocus();

addKeyListener(**this**); // Se le agrega un escuchador de eventos de teclado

}

**public** **void** paintComponent(Graphics g) {

Graphics2D g2d = (Graphics2D)g;

**super**.paintComponent(g2d);

gameDraw(g2d);

}

/\*\*

\* Metodo <I>keyPressed</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al presionar cualquier la tecla.

\* **@param** e es el <code>evento</code> generado al presionar las teclas.

\*/

**public** **void** keyPressed(KeyEvent e) {

gameKeyPressed(e.getKeyCode());

}

/\*\*

\* Metodo <I>keyReleased</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al soltar la tecla presionada.

\* **@param** e es el <code>evento</code> que se genera en al soltar las teclas.

\*/

**public** **void** keyReleased(KeyEvent e) {

gameKeyReleased(e);

}

/\*\*

\* Metodo <I>keyTyped</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al presionar una tecla que no es de accion.

\* **@param** e es el <code>evento</code> que se genera en al presionar las teclas.

\*/

**public** **void** keyTyped(KeyEvent e) {

gameKeyTyped(e);

}

}

/\*\*

\* Clase derivada de <code>JPanel</code> en donde se definen las caracteristicas

\* del panel Juego.

\*/

**class** Juego **extends** JPanel **implements** KeyListener {

**private** **static** **final** **long** *serialVersionUID* = 1L;

// Constructor

**public** Juego() {

setFocusable(**true**); //Solicita el focus

requestFocus();

addKeyListener(**this**); // Para habilitar el uso de teclado

}

// Llamado por el repaint().

**public** **void** paintComponent(Graphics g) {

Graphics2D g2d = (Graphics2D)g;

**super**.paintComponent(g2d); // paint background

//se repintara el panel mientras haya vidas

**if** (vidas > 0&&*state*!=State.*PAUSA*) {

// Inicializan el DoubleBuffer

intParaPausa=1;

**if** (dbImage == **null**){

dbImage = createImage (**this**.getSize().width, **this**.getSize().height);

dbg = dbImage.getGraphics ();

}

// Actualiza la imagen de fondo.

dbg.setColor (getBackground ());

// Actualiza el Foreground.

dbg.setColor (getForeground());

render.draw(g2d, mapa, 1000, 600);

} **else** {

**if**(intParaPausa==1){

sonidoPausa.play();

intParaPausa=2;

}

g2d.drawImage(pausa, 0, 0, **this**);

}

}

/\*\*

\* Metodo <I>keyPressed</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al presionar cualquier la tecla.

\* **@param** e es el <code>evento</code> generado al presionar las teclas.

\*/

**public** **void** keyPressed(KeyEvent e) {

gameKeyPressed(e.getKeyCode());

}

/\*\*

\* Metodo <I>keyReleased</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al soltar la tecla presionada.

\* **@param** e es el <code>evento</code> que se genera en al soltar las teclas.

\*/

**public** **void** keyReleased(KeyEvent e) {

gameKeyReleased(e);

}

/\*\*

\* Metodo <I>keyTyped</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al presionar una tecla que no es de accion.

\* **@param** e es el <code>evento</code> que se genera en al presionar las teclas.

\*/

**public** **void** keyTyped(KeyEvent e) {

gameKeyTyped(e);

}

}

/\*\*

\* Metodo <I>keyPressed</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al presionar cualquier la tecla.

\* **@param** e es el <code>evento</code> generado al presionar las teclas.

\*/

**public** **void** keyPressed(KeyEvent e) {

gameKeyPressed(e.getKeyCode());

}

/\*\*

\* Metodo <I>keyReleased</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al soltar la tecla presionada.

\* **@param** e es el <code>evento</code> que se genera en al soltar las teclas.

\*/

**public** **void** keyReleased(KeyEvent e) {

gameKeyReleased(e);

}

/\*\*

\* Metodo <I>keyTyped</I> sobrescrito de la interface <code>KeyListener</code>.<P>

\* En este metodo maneja el evento que se genera al presionar una tecla que no es de accion.

\* **@param** e es el <code>evento</code> que se genera en al presionar las teclas.

\*/

**public** **void** keyTyped(KeyEvent e) {

gameKeyTyped(e);

}

// El main del juego

**public** **static** **void** main(String[] args) **throws** IOException{

SplashScreen splash = SplashScreen.*getSplashScreen*();

**if** (splash != **null**) { // Se crea un splash Screen

Dimension splashBounds = splash.getSize();

Graphics2D g2d = splash.createGraphics();

// Simulate a progress bar

**for** (**int** i = 0; i < 100; i += 5) {

g2d.setColor(Color.*BLUE*);

// g2d.drawImage(Toolkit.getDefaultToolkit().getImage("images/corre1.png"),i,splashBounds.height -100,null);

g2d.fillRect(0, splashBounds.height -20, splashBounds.width \* i / 100, 20);

splash.update();

**try** {

Thread.*sleep*(10); // Some delays

} **catch** (Exception e) {}

}

g2d.dispose();

splash.close();

}

**new** GameMain(); // Se crea un constructor

}

/\*\*

\* Metodo <I>leeArchivo</I> que procesa eventos de KeyReleased que lee la

\* informacion de un archivo y lo agrega a un vector.

\*

\* **@throws** IOException

\*/

**public** **void** leeArchivo() **throws** IOException{

BufferedReader fileIn = **new** BufferedReader(**new** FileReader(nombreArchivo));

String dato = fileIn.readLine();

vec = **new** Vector<Puntaje>();

**while**(dato != **null**) {

arr = dato.split(",");

**int** num = (Integer.*parseInt*(arr[0]));

String nom = arr[1];

vec.add(**new** Puntaje(nom, num));

dato = fileIn.readLine();

}

fileIn.close();

}

/\*\*

\* Metodo <I>grabaArchivo</I> que agrega la informacion del vector al archivo.

\*

\* **@throws** IOException

\*/

**public** **void** grabaArchivo() **throws** IOException{

PrintWriter fileOut = **new** PrintWriter(**new** FileWriter(nombreArchivo));

Collections.*sort*(vec);

**int** limite=vec.size();

**if**(limite>5){ // Este es directamente el numero de tops score a desplegar,

limite=5;

}

**for** (**int** i=0; i<limite; i++) {

Puntaje x;

x = (Puntaje) vec.get(i);

fileOut.println(x.toString());

}

fileOut.close();

}

}

**import** java.awt.Image;

**import** java.awt.Toolkit;

/\*\*

\*La clase Jugador se encargara de manejar las animaciones del jugador,

\*validar si ha collisionado, las vidas del personaje y la velocidad con la que

\*se desplazara el jugador.

\*/

**public** **class** Player **extends** Creature {

**private** **int** vida;

//Velocidad al saltar. Negativa para que el jugador se mueva hacia arriba

**private** **static** **float** *JUMP\_SPEED* = -.95f;

**private** **int** pared=0; //Si esta collisionando con una pared.

**private** **final** **int** MAXVIDA = 15; //Las vidas del jugador, maxima para evitar que se pase de estas

**private** **float** MaxSpeed = 0.5f; //Velocidad maxima del jugador

**private** **boolean** onGround; //Si el jugador esta actualmente en el suelo

//animaciones del jugador

**private** Animation anim;

**private** Animation animParado;

**private** Animation animDisparo;

**private** **boolean** IzqDer;

**private** **float** gravity = 0.002f;

**private** **long** tiempoDisparo;

//Estados de movimiento del jugador

**static** **enum** Estado{

*CAMINANDO*,*SALTANDO*,*PARADO*, *CORRIENDO*

}

//Estados de ataque del jugador

**static** **enum** Ataque{

*NOATK*, *ATK*

}

**static** Estado *estado* = Estado.*CAMINANDO*; // Estado actual del jugador (movimiento)

**static** Ataque *ataque* = Ataque.*NOATK*; // Estado actual de ataque

/\*\*

\* Metodo constructor de Player usado para crear el objeto

\* **@param** posX es la <code>posicion en x</code> del objeto.

\* **@param** posY es la <code>posicion en y</code> del objeto.

\* **@param** image es la <code>imagen</code> del objeto.

\*/

**public** Player(**int** posX, **int** posY, Image imagen)

{

**super**(posX, posY, imagen); // Llama al super constructor

vida = MAXVIDA; // Llena la vida al maximo

cargaImagenes(); // carga las imagenes de las animaciones

setVidaMax(MAXVIDA);

iniciaVida();

}

/\*

\* Metodo <I>setGravity</I> que cambia la gravedad del Jugador.<P>

\*/

**public** **void** setGravity(**float** grav){

gravity = grav; // Le asigna al jugador cierta gravedad

}

/\*

\* Metodo <I>getGravity</I> que devuelve la gravedad del Jugador.<P>

\*/

**public** **float** getGravity(){

**return** gravity; // Regresa la gravedad

}

/\*

\* Metodo <I>estaSaltando</I> que indica si el Player salta o no.<P>

\* @return true si esta saltando, false si no esta saltando

\*/

**public** **boolean** estaSaltado(){

**if** (*estado* == Estado.*SALTANDO*) {

**return** **true**;

}

**return** **false**;

}

/\*

\* Metodo <I>estaAtacando</I> que indica si el Player ataca o no.<P>

\* @return // Devuelve true si el jugador esta atacando, False si no esta atacando

\*/

**public** **boolean** estaAtacando(){

**if** (*ataque* != Ataque.*NOATK*) //checha que el estado no sea no atacar

**return** **true**;

**return** **false**;

}

/\*

\* Metodo <I>collideVertical</I> que frena horizontalmente a Player.<P>

\*/

**public** **void** collideHorizontal() { //Se manda llamar cuando hay colision con un objeto, para que el personaje ya no avance en X

setVelocidadX(0); // Reduce la velocidad a 0

}

/\*

\* Metodo <I>collideVertical</I> que frena verticalmente a Player.<P>

\*/

**public** **void** collideVertical() { //Si hay colision con un piso (hacia abajo), se pone la velocidad en Y en 0.

**if** (getVelocidadY() > 0) {

onGround = **true**; //Al haber colision, el personaje esta en el piso.

*JUMP\_SPEED* = -.95f; // Se cambia la velocidad de salto

}

**if**(*estado* != Estado.*CORRIENDO*)//Si estaba Corriendo, sigue asi

*estado* = Estado.*CAMINANDO*;

**if** (*estado* == Estado.*CORRIENDO* && (getVelocidadX() == 0)) { //Se revisa que el jugador este quieto

**if**(onGround){

*estado* = Estado.*CAMINANDO*; // Si esta en el piso, el estado es caminando

} **else** **if**(!onGround) {

*estado* = Estado.*SALTANDO*; // Si esta inmovil y se despega del suelo, es un salto

}

}

setVelocidadY(0);

}

/\*

\* Metodo <I>setPosY</I> que establece la posicion en Y del Player.<P>

\*/

**public** **void** setPosY(**float** y) {

// Se revisa si el jugador esta cayendo

**if** (Math.*round*(y) > Math.*round*(getPosY())) {

onGround = **false**; //Si se intenta modificar la posicion en Y del jugador a algo mas bajo de lo actual, quiere decir que esta cayendo

}

**super**.setPosY(y); // Se cambia la posicion en Y del jugador

}

/\*\*

\* Metodo <I>jump()</I> <P>

\* Si el personaje esta en el piso o se le obliga a saltar, su velocidad en Y es JUMP\_SPEED.

\* Si esta en una pared, se le permite saltar de nuevo

\*/

**public** **void** jump(**boolean** forceJump) {

**if** (onGround || forceJump || (pared == 1)) {

pared = 0;

onGround = **false**; //El personaje ya no esta en el piso

setVelocidadY(*JUMP\_SPEED*);//Su velocidad es la velocidad de salto

*estado* = Estado.*SALTANDO*; // El estado cambia a Saltando

}

}

/\*\*

\* Metodo <I>updateAnim()</I> <P>

\* cambia la animacion del Player.

\* Si ha pasado cierto tiempo desde que comenzo la animacion,

\* se cambia de animacion y se cambia el Estado.

\*/

**public** **void** updateAnim(**long** elapsedTime){

setGravity(0.002f);

**if**(*estado* == Estado.*SALTANDO*){

setImageIcon((Toolkit.*getDefaultToolkit*().getImage("images/salta09.png")).getScaledInstance(64, 130, java.awt.Image.*SCALE\_SMOOTH*));

} **else** **if** ((*estado* == Estado.*CAMINANDO*) && (getVelocidadX() != 0)){ //EN movimiento

anim.update(elapsedTime);

setImageIcon((anim.getImage()).getScaledInstance(64, 130, java.awt.Image.*SCALE\_SMOOTH*)); //60,130

} **else** **if** ((*estado* == Estado.*CAMINANDO*) && (getVelocidadX() == 0)) { // Parado

**if** (*ataque* == Ataque.*ATK*) { // Cargando el arma

**long** tiempoDesdeAtk = System.*currentTimeMillis*() - tiempoDisparo;

**if** (tiempoDesdeAtk < 150) {

animDisparo.update(elapsedTime);

setImageIcon((animDisparo.getImage()).getScaledInstance(72, 130, java.awt.Image.*SCALE\_SMOOTH*));

} **else** { // Se dejo de cargar

*ataque* = Ataque.*NOATK*;

animDisparo.start();

}

} **else** {

animParado.update(elapsedTime); // Animacion de parado

setImageIcon((animParado.getImage()).getScaledInstance(64, 130, java.awt.Image.*SCALE\_SMOOTH*));

}

} **else** **if** ((*estado* == Estado.*CORRIENDO*) || (MaxSpeed == 1)) { // Corriendo

setImageIcon((Toolkit.*getDefaultToolkit*().getImage("images/Slide1.png")).getScaledInstance(64, 130, java.awt.Image.*SCALE\_SMOOTH*));

}

}

/\*\*

\* Metodo <I>getMaxSpeed()</I> <P>

\* Devuelve la Maxima velocidad posible del Player

\*/

**public** **float** getMaxSpeed() {

**return** MaxSpeed; // Devuelve la maxima velocidad posible del jugador

}

/\*\*

\* Metodo <I>disparo()</I> <P>

\* Cambia el estado de Ataque a ATK

\*/

**public** **void** disparo(){

*ataque* = Ataque.*ATK*;

tiempoDisparo = System.*currentTimeMillis*();

}

/\*\*

\* Metodo <I>cambiaVida()</I> <P>

\* Reduce o aumenta la vida del jugador.

\* **@param** int <code> cuanto</code> es la cantida de vida a modificar.

\*/

**public** **void** cambiaVida(**int** cuanto){

**if** (vida >= 0 && vida <= MAXVIDA) {

vida += cuanto;

**if** (vida > MAXVIDA){

vida = MAXVIDA; // Se asegura de no aumentar la vida mas alla del limite

}

**if** (vida < 0){

vida =0; // Se encarga de no reducir la vida mas alla de 0

}

}

}

/\*\*

\* Metodo <I>getVida()</I> <P>

\* Devuelve la vida actual del jugador

\*/

**public** **int** getVida(){

**return** vida;

}

/\*\*

\* Metodo <I>getMaxVida()</I> <P>

\* Regresa la vida Maxima posible del jugador

\*/

**public** **int** getMaxVida(){

**return** MAXVIDA;

}

/\*\*

\* Metodo <I>cargaImagenes()</I> <P>

\* Carga todas las imagenes que se usaran por el Player

\*/

**public** **void** cargaImagenes(){

// Animacion de correr

Image a1 = Toolkit.*getDefaultToolkit*().getImage("images/corre1.png");

Image a2 = Toolkit.*getDefaultToolkit*().getImage("images/corre2.png");

Image a3 = Toolkit.*getDefaultToolkit*().getImage("images/corre3.png");

Image a4 = Toolkit.*getDefaultToolkit*().getImage("images/corre4.png");

Image a5 = Toolkit.*getDefaultToolkit*().getImage("images/corre5.png");

Image a6 = Toolkit.*getDefaultToolkit*().getImage("images/corre6.png");

Image a7 = Toolkit.*getDefaultToolkit*().getImage("images/corre7.png");

Image a8 = Toolkit.*getDefaultToolkit*().getImage("images/corre8.png");

Image a9 = Toolkit.*getDefaultToolkit*().getImage("images/salta09.png");

// Animacion de parado

Image b1 = Toolkit.*getDefaultToolkit*().getImage("images/parado01.png");

Image b2 = Toolkit.*getDefaultToolkit*().getImage("images/parado02.png");

Image b3 = Toolkit.*getDefaultToolkit*().getImage("images/parado03.png");

Image b4 = Toolkit.*getDefaultToolkit*().getImage("images/parado04.png");

Image b5 = Toolkit.*getDefaultToolkit*().getImage("images/parado05.png");

Image b6 = Toolkit.*getDefaultToolkit*().getImage("images/parado06.png");

Image b7 = Toolkit.*getDefaultToolkit*().getImage("images/parado07.png");

Image b8 = Toolkit.*getDefaultToolkit*().getImage("images/parado08.png");

Image b9 = Toolkit.*getDefaultToolkit*().getImage("images/parado09.png");

Image b10 = Toolkit.*getDefaultToolkit*().getImage("images/parado10.png");

Image b11 = Toolkit.*getDefaultToolkit*().getImage("images/parado11.png");

Image b12 = Toolkit.*getDefaultToolkit*().getImage("images/parado12.png");

Image b13 = Toolkit.*getDefaultToolkit*().getImage("images/parado13.png");

Image b14 = Toolkit.*getDefaultToolkit*().getImage("images/parado14.png");

Image b15 = Toolkit.*getDefaultToolkit*().getImage("images/parado15.png");

Image b16 = Toolkit.*getDefaultToolkit*().getImage("images/parado16.png");

Image b17 = Toolkit.*getDefaultToolkit*().getImage("images/parado17.png");

// Animacion de disparo

Image c1 = Toolkit.*getDefaultToolkit*().getImage("images/dispara1.png");

Image c2 = Toolkit.*getDefaultToolkit*().getImage("images/dispara2.png");

Image c3 = Toolkit.*getDefaultToolkit*().getImage("images/dispara3.png");

// Animacion de correr

anim = **new** Animation();

anim.addFrame(a1, 100);

anim.addFrame(a2, 100);

anim.addFrame(a3, 100);

anim.addFrame(a4, 100);

anim.addFrame(a5, 100);

anim.addFrame(a6, 100);

anim.addFrame(a7, 100);

anim.addFrame(a8, 100);

anim.addFrame(a9, 100);

// Animacion de Parado

animParado = **new** Animation();

animParado.addFrame(b1, 100);

animParado.addFrame(b2, 100);

animParado.addFrame(b3, 100);

animParado.addFrame(b4, 100);

animParado.addFrame(b5, 100);

animParado.addFrame(b6, 100);

animParado.addFrame(b7, 100);

animParado.addFrame(b8, 100);

animParado.addFrame(b9, 100);

animParado.addFrame(b10, 100);

animParado.addFrame(b11, 100);

animParado.addFrame(b12, 100);

animParado.addFrame(b13, 100);

animParado.addFrame(b14, 100);

animParado.addFrame(b15, 100);

animParado.addFrame(b16, 100);

animParado.addFrame(b17, 100);

// Animacion de Disparo

animDisparo = **new** Animation();

animDisparo.addFrame(c1, 50);

animDisparo.addFrame(c2, 50);

animDisparo.addFrame(c3, 50);

}

/\*\*

\* Metodo <I>getLado()</I> <P>

\* Devuelve true si esta volteando a la izquierda, false si voltea a la derecha

\*/

**public** **boolean** getLado(){ //Devuelve True si esta volteando a la Izquierda, False si voltea a la derecha

**if** (getVelocidadX() < 0) {

IzqDer = **true**; // Si la velocidad es negativa, voltea a la izquierda

} **else** **if** (getVelocidadX() > 0) {

IzqDer = **false**; // Si la velocidad es positiva, voltea a la derecha

}

**return** IzqDer;

}

}

/\*\*

\* Clase Puntaje, es la clase encargada de contabilizar los puntos

\* logrados cada vez que se destruye un enemigo, ademas que es la

\* que crea el archivo de texto y lo maneja,

\*/

**public** **class** Puntaje **implements** Comparable <Puntaje>{

**private** String nombre; //Nombre del jugador

**private** **int** puntaje; //Puntaje que a obtenido

/\*\*

\* Constructor vacio con darle valores iniciales al momento de

\* crear el objeto Puntaje

\*/

**public** Puntaje() {

nombre = "";

puntaje = 0;

}

/\*\*

\* Metodo constructor usado para crear el objeto

\* **@param** nombre es el <code>nombre</code> del objeto.

\* **@param** puntaje es el <code>puntaje</code> del objeto.

\*/

**public** Puntaje(String nombre, **int** puntaje) {

**this**.nombre = nombre;

**this**.puntaje = puntaje;

}

/\*\*

\* Metodo modificador usado para cambiar el nombre del objeto.

\* **@param** nombre es el <code>nombre</code> del objeto.

\*/

**public** **void** setNombre(String nombre) {

**this**.nombre = nombre;

}

/\*\*

\* Metodo de acceso que regresa el nombre del objeto

\* **@return** nombre es el <code>nombre</code> del objeto.

\*/

**public** String getNombre() {

**return** nombre;

}

/\*\*

\* Metodo modificador usado para cambiar el puntaje del objeto

\* **@param** puntaje es el <code>puntaje</code> del objeto.

\*/

**public** **void** setPuntaje(**int** puntaje) {

**this**.puntaje = puntaje;

}

/\*\*

\* Metodo de acceso que regresa el puntaje del objeto

\* **@return** puntaje es el <code>puntaje</code> del objeto.

\*/

**public** **int** getPuntaje() {

**return** puntaje;

}

/\*\*

\* Metodo que regresa el objeto en formato String

\* **@return** un objeto de la clase <code>String</code>.

\*/

**public** String toString(){

**return** "" + getPuntaje() + "," + getNombre();

}

/\*\*

\* Metodo que comapra dos puntajes, para ordenar el vector de Puntajes

\* **@return** un entero.

\*/

**public** **int** compareTo(Puntaje o) {

**return** o.getPuntaje() - **this**.getPuntaje() ;

}

}

**import** javax.sound.sampled.UnsupportedAudioFileException;

**import** javax.sound.sampled.LineUnavailableException;

**import** javax.sound.sampled.AudioInputStream;

**import** javax.sound.sampled.Clip;

**import** javax.sound.sampled.AudioSystem;

**import** java.io.IOException;

**import** java.net.URL;

/\*\*

\* Clase SoundClip, es el metodo encargado de exportar los archivos tipo wav

\* y usarlos dentro del juego, esto permite que haya musica de fondo, efectos

\* de sonido

\*/

**public** **class** SoundClip {

**private** AudioInputStream sample;

**private** Clip clip;

**private** **boolean** looping = **false**;

**private** **int** repeat = 0;

**private** String filename = "";

/\*\*

\* Constructor default

\*/

**public** SoundClip() {

**try** {

//crea el Buffer de sonido

clip = AudioSystem.*getClip*();

} **catch** (LineUnavailableException e) {

}

}

/\*\*

\* Constructor con parametros, que carga manda llamar a load

\* esto carga el archivo de sonido.

\* **@param** filename es el <code>String</code> del archivo.

\*/

**public** SoundClip(String filename) {

//Llama al constructor default.

**this**();

//Carga el archivo de sonido.

load(filename);

}

/\*\*

\* Metodo de acceso que regresa un objeto de tipo Clip

\* **@return** clip es un <code>objeto Clip</code>.

\*/

**public** Clip getClip() {

**return** clip;

}

/\*\*

\* Metodo modificador usado para modificar si el sonido se repite.

\* **@param** \_looping es un valor <code>boleano</code>.

\*/

**public** **void** setLooping(**boolean** looping) {

**this**.looping = looping;

}

/\*\*

\* Metodo de acceso que regresa un booleano para ver si hay repeticion.

\* **@return** looping es un valor <code>boleano</code>.

\*/

**public** **boolean** getLooping() {

**return** looping;

}

/\*\*

\* Metodo modificador usado para definir el numero de repeticiones.

\* **@param** \_repeat es un <code>entero</code> que es el numero de repeticiones.

\*/

**public** **void** setRepeat(**int** repeat) {

**this**.repeat = repeat;

}

/\*\*

\* Metodo de acceso que regresa el numero de repeticiones.

\* **@return** repeat es un valor <code>entero</code> con el numero de repeticiones.

\*/

**public** **int** getRepeat() {

**return** repeat;

}

/\*\*

\* Metodo modificador que asigna un nombre al archivo.

\* **@param** \_filename es un <code>String</code> con el nombre del archivo.

\*/

**public** **void** setFilename(String filename) {

**this**.filename = filename;

}

/\*\*

\* Metodo de acceso que regresa el nombre del archivo.

\* **@return** filename es un <code>String</code> con el nombre del archivo.

\*/

**public** String getFilename() {

**return** filename;

}

/\*\*

\* Metodo que verifica si el archivo de audio esta cargado.

\* **@return** sample es un <code>objeto sample</code>.

\*/

**public** **boolean** isLoaded() {

**return** (sample != **null**);

}

/\*\*

\* Metodo de acceso que regresa el url del archivo

\* **@param** filename es un <code>String</code> con el nombre del archivo.

\*/

**private** URL getURL(String filename) {

URL url = **null**;

**try** {

url = **this**.getClass().getResource(filename);

}

**catch** (Exception e) {

}

**return** url;

}

/\*\*

\* Metodo que carga el archivo de sonido.

\* **@param** audiofile es un <code>String</code> con el nombre del archivo de sonido.

\*/

**public** **boolean** load(String audiofile) {

**try** {

setFilename(audiofile);

sample = AudioSystem.*getAudioInputStream*(getURL(filename));

clip.open(sample);

**return** **true**;

} **catch** (IOException e) {

**return** **false**;

} **catch** (UnsupportedAudioFileException e) {

**return** **false**;

} **catch** (LineUnavailableException e) {

**return** **false**;

}

}

/\*\*

\* Metodo que reproduce el sonido.

\*/

**public** **void** play() {

//se sale si el sonido no a sido cargado

**if** (!isLoaded())

**return**;

//vuelve a empezar el sound clip

clip.setFramePosition(0);

//Reproduce el sonido con repeticion opcional.

**if** (looping)

clip.loop(Clip.*LOOP\_CONTINUOUSLY*);

**else**

clip.loop(repeat);

}

/\*\*

\* Metodo que detiene el sonido.

\*/

**public** **void** stop() {

clip.stop();

}

}

import java.awt.Image;

import javax.swing.ImageIcon;

import java.awt.Rectangle;

/\*\*

\* Clase Sprite, es la clase encargada de darle los aspectos

\* fisico (caida libres), la posicion (movimiento) y la

\* posibilidad de atacar a los enemigos y jugador

\*/

public class Sprite {

private Animation anim; //Animacion del enemigo o jugador

private ImageIcon icono; //icono.

private float posX;// posicion en el eje X

private float posY; // posicion en el eje Y

protected float dx; //cambio de la posicion referente a una posicion inicial en X

protected float dy; //cambio de la posicion referente a una posicion inicial en Y

private int vidaMax = 3;//Le da al enemigo o jugador un numero maximo de vidas

protected int vida; //Las vidas que se necesitan quitar para destruir el personaje

private String estadoActual; //el estado en que esta el Sprite

int lado; //Lado hacia donde apunta el Sprite

//Los estados que se puede estar el Sprite

public static final String PROXIMA\_CAIDA\_LIBRE = "ProximaCaidaLibre";

public static final String CHOQUE\_CON\_PARED = "ChoqueConPared";

public static final String ESTA\_CERCA = "EstaCerca";

public static final String DERECHA\_ATACANDO = "paradoYAtacando";

public static final String IZQUIERDA\_ATACANDO="paradoYAtacandoD";

public static final String NO\_ESTA\_CERCA = "noEstaCerca";

protected boolean matenme = false;

/\*\*

\* Constructor Sprite para crear un objeto de tipo Sprite.

\* @param posX posicion en el eje X del Sprite

\* @param posY posicion en el eje Y del Sprite

\* @param image la imagen del objeto que funcionara para reconocer

\* si es el jugador o un enemigo

\*/

public Sprite(int posX, int posY ,Image image) {

this.posX=posX;

this.posY=posY;

icono = new ImageIcon(image);

}

/\*\*

\* Constructor Sprite para crear un objeto de tipo Sprite.

\* @param posX posicion en el eje X del Sprite

\* @param posY posicion en el eje Y del Sprite

\* @param Animacion el conjunto de imagenes del objeto que

\* funcionara para reconocer si es el jugador o un enemigo

\*/

public Sprite(int posX, int posY , Animation anim) {

this.anim = anim;

this.posX=posX;

this.posY=posY;

}

/\*\*

\* Metodo Sprite se encarga de asignar una imagen al objeto Sprite

\* @param image, la imagen a desplegar en el Sprite

\*/

public Sprite(ImageIcon image){

icono = image;

}

/\*\*

\* Metodo actualiza se encarga de calcular la caida libre del Sprite

\* @param tiempo, tiempo usado para calcular la caida del Sprite

\*/

public void actualiza(long tiempo) {

posX += dx \* tiempo; //movimiento tipo caida libre en eje x

posY += dy \* tiempo; //movimiento tipo caida libre en eje y

if(getVida() <= 0) //si las vidas se acaban el Sprite se elima

matenme = true;

}

/\*\*

\* Metodo atacar encargado de hacer que los enemigos ataquen.

\* Se usa en conjunto con otro metodo para saber si debe atacar

\* @return true, si el enemigo no estaba atacando lo forza a atacar

\* @return false, si el enemigo atacaba y el jugador no esta cerca

\* lo forza a detenerse

\*/

public boolean atacar(){

return false;

}

/\*\*

\* Metodo bala encargado de desplegar las balas de los enemigos

\* @param derecha, determina a que lado el Sprite de la bala se vera

\* @param mapa, si el enemigo atacaba y el jugador no esta cerca

\* @return null, al terminar de agregar la bala la saca del TileMap

\*/

public Bala dibujaBala(boolean derecha, TileMap mapa){

return null;

}

/\*\*

\* Metodo checaProximidad para determinar si hay un objeto cerca del Sprie

\* Si hay un objeto cercano, el Sprite cambia de estado.

\* @param otroSprite, sera el Sprite a comparar si esta cerca o no

\*/

public void checaProximidad(Sprite otroSprite) //Esto es usado para checar proximidad de un Enemigo con el jugador

{

//Determina si el enemigo esta a una distancia, comparando sus ejes en X y Y y mientras su estado no haya sido ESTA\_CERCA lo pone en este.

if(Math.abs(this.getPosX() - otroSprite.getPosX()) < 500 && this.getPosY() < otroSprite.getPosY() + 50

&& this.getPosY() + this.getAlto() + 50 > otroSprite.getPosY() && estadoActual != ESTA\_CERCA)

revisarCambioDeEstado(Sprite.ESTA\_CERCA, 0); //asigna el estado

//Si no esta cerca determina si el usuario esta alejado una cierta distancia para asignarle el estado de NO\_ESTA\_CERCA

else if(Math.abs(this.getPosX() - otroSprite.getPosX()) > 500 || this.getPosY() > otroSprite.getPosY() + 50

&& this.getPosY() + this.getAlto() + 50 < otroSprite.getPosY())

revisarCambioDeEstado(Sprite.NO\_ESTA\_CERCA, 0); //Asignacion del estado

//Si el Jugador esta cerca se necesita saber si el enemigo cambiara de lado

else if(getCurrState() == ESTA\_CERCA)

{

if(this.getPosX() - otroSprite.getPosX()>0) //Checa en que lado esta

{

dx = Math.abs(dx)\*-1; //checa si esta a la derecha

setLado(-1); //cambia de lado apuntando hacia la derecha

}

else{

dx = Math.abs(dx); //checa si esta a la izquierda

setLado(1); //cambia de lado apuntando hacia la izquierda

}

}

}

/\*\*

\* Metodo puedoMorir para determinar si un objeto puede ser destruido.

\* @return true, si el objeto puede ser destruido

\* @return false, si el objeto no se puede destruir

\*/

public boolean puedeMorir(){

return matenme;

}

/\*\*

\* Metodo setLado para determinar el lado hacia donde apuntara el Sprite

\* @param lado, entero que indica hacia donde apuntara el Sprite

\* 1 izquierda, -1 derecha

\*/

public void setLado(int lado){

this.lado = lado; //asignacion del lado

}

/\*\*

\* Metodo volteaADerecha para determinar el lado hacia donde debe voltear el Sprite

\* @return true, si esta orientado a la izquierda = 1,

\* @return false, si esta orientado a la derecha = -1

\*/

public boolean volteaADerecha(){

if(lado >0)

return true;

return false;

}

/\*\*

\* Metodo revisarcambioDeVueltapara revisar si el enemigo llega a una esquina.

\* Si llega a una esquina, el Sprite gira y cambia de direccion.

\* @param mapa es el TileMap.

\* @param posX es el float de la posicion en X del Sprite.

\* @param posY es el float de la posicion en Y del Sprite.

\* @param newX es el float de la futura posicion en X del Sprite.

\*/

public void revisarCambioDeVuelta(TileMap mapa, float posX, float posY, float newX){

if(dx>0) newX = newX + getWidth();

for(int x = 0; x<mapa.getWidth(); x++)

{

if(mapa.getTile(TileMapRenderer.pixelsToTiles(posX + TileMapRenderer.TILE\_SIZE\*x),

TileMapRenderer.pixelsToTiles(posY + getAlto() + TileMapRenderer.TILE\_SIZE/2)) == null)

if(Math.abs(TileMapRenderer.tilesToPixels(TileMapRenderer.pixelsToTiles(posX +

TileMapRenderer.TILE\_SIZE\*x))) - newX < TileMapRenderer.TILE\_SIZE/8)

{

setPosX(posX);

revisarCambioDeEstado(Sprite.PROXIMA\_CAIDA\_LIBRE, newX);

}

}

}

/\*\*

\* Metodo <I>getPosX</I> que devuelve la posicion en X del Sprite.<P>

\*/

public float getPosX() {

return posX;

}

/\*\*

\* Metodo <I>getPosY</I> que devuelve la posicion en Y del Sprite.<P>

\*/

public float getPosY() {

return posY;

}

/\*\*

\* Metodo <I>setPosX</I> que establece la posicion en X del Sprite.<P>

\* @param posX, la nueva posicion para el sprite

\*/

public void setPosX(float posX) {

this.posX = posX;

}

/\*\*

\* Metodo <I>setPosY</I> que establece la posicion en Y del Sprite.<P>

\* @param posY, la nueva posicion para el sprite

\*/

public void setPosY(float posY) {

this.posY = posY;

}

/\*\*

\* Metodo <I>getAncho</I> que devuelve el ancho en pixeles del Sprite.<P>

\*/

public int getAncho() {

return icono.getIconWidth();

}

/\*\*

\* Metodo <I>getAlto</I> que devuelve el alto en pixeles del Sprite.<P>

\*/

public int getAlto() {

return icono.getIconHeight();

}

/\*\*

\* Metodo <I>getVelocidadX</I> que devuelve la velocidad en X del Sprite.<P>

\*/

public float getVelocidadX() {

return dx;

}

/\*\*

\* Metodo <I>getVelocidadY</I> que devuelve la velocidad en Y del Sprite.<P>

\*/

public float getVelocidadY() {

return dy;

}

/\*\*

\* Metodo <I>setVelocidadX</I> que cambia la velocidad en X del Sprite.<P>

\* @param dx, float que cambia la velocidad de avance en el eje X

\*/

public void setVelocidadX(float dx) {

this.dx = dx;

}

/\*\*

\* Metodo <I>setVelocidadY</I> que cambia la velocidad en Y del Sprite.<P>

\* @param dy, float que cambia la velocidad de avance en el eje Y

\*/

public void setVelocidadY(float dy) {

this.dy = dy;

}

/\*\*

\* Metodo setImageIcon modificador usado para cambiar el icono del objeto

\* @param icono es el <code>icono</code> del objeto.

\*/

public void setImageIcon(Image icono) {

this.icono = new ImageIcon(icono);

}

/\*\*

\* Metodo ImageIcon de acceso que regresa el icono del objeto

\* @return ImageIcon es el <code>ImageIcon</code> del objeto.

\*/

public ImageIcon getImageIcon() {

return icono;

}

/\*\*

\* Metodo Image da acceso que regresa la imagen del icono

\* @return un objeto de la clase <code>Image</code> que es la imagen del icono.

\*/

public Image getImagenI() {

return icono.getImage();

}

/\*\*

\* Metodo Rectangle de acceso que regresa un nuevo rectangulo

\* @return un objeto de la clase <code>Rectangle</code> que es el perimetro

\* del rectangulo

\*/

public Rectangle getPerimetro(){

return new Rectangle((int)getPosX(),(int)getPosY(),getAncho(),getAlto());

}

/\*\*

\* Checa si el objeto <code>Objetos</code> intersecta a otro <code>Objetos</code>

\* @param Sprite, el objeto a comparar si hubo una colision.

\* @return un valor booleano <code>true</code> si lo intersecta <code>false</code>

\* en caso contrario

\*/

public boolean intersecta(Sprite obj){

return getPerimetro().intersects(obj.getPerimetro());

}

/\*\*

\* Checa si el objeto <code>Objetos</code> intersecta a otro <code>Objetos</code> por debajo

\* @param Sprite, el objeto a comparar si hubo una colision.

\* @return un valor booleano <code>true</code> si lo intersecta <code>false</code>

\* en caso contrario

\*/

public boolean intersectaAbajo(Sprite obj){

//Obtiene el area de la parte baja de un objeto a comparar

Rectangle parteBajaObjeto1 = new Rectangle((int)getPosX(),(int)getPosY()+getAlto()-20,getAncho(),20);

//Obtiene el area de la parte alta de un objeto a comparar

Rectangle parteAltaObjeto2 = new Rectangle((int)obj.getPosX(),(int)obj.getPosY(),obj.getAncho(),20);

//llama al metodo intersecta para saber si han colisionado y regresa el valor

return parteBajaObjeto1.intersects(parteAltaObjeto2);

}

/\*\*

\* Metodo getImageFromAnimation consigue la imagen actual de la animacion

\* @return Image regresa la imagen

\*/

public Image getImageFromAnimation(){

return anim.getImage();

}

/\*\*

\* Metodo <I>getWidth</I> que regresa el ancho de la imagen.<P>

\* @return int, que sera el valor del ancho de la imagen

\*/

public int getWidth() {

return anim.getImage().getWidth(null);

}

/\*\*

\* Metodo <I>getHeight</I> que regresa el alto de la imagen.<P>

\*@return int, que sera el valor del ancho de la imagen

\*/

public int getHeight() {

return anim.getImage().getHeight(null);

}

/\*\*

\* Metodo clone, funciona para duplicar Sprites

\* Sirve para crear multiples Sprites con la misma imagen

\* @return un nuevo Sprite con una imagen

\*/

public Object clone() {

return new Sprite(icono);

}

/\*\*

\* Metodo <I>setAnimation</I> que cambia la animacion del Sprite.<P>

\* @param la animacion nueva

\*/

protected void setAnimation(Animation anim){

this.anim = anim;

anim.getImage();

}

/\*\*

\* Metodo <I>updateAnim</I> que actualiza la animacion del Sprite.<P>

\* @param elapsedTime, servira para saber el tiempo de desplegado de la animacion

\*/

public void updateAnim(long elapsedTime){

anim.update(elapsedTime);

}

/\*\*

\* Metodo <I>revisarCambioDeEstado</I>

\*/

public void revisarCambioDeEstado(String identificador, float newX){

}

/\*\*

\* Metodo <I>getCurrState</I> que regresa el Estado del Sprite.<P>

\* @return el estado actual del objeto

\*/

public String getCurrState(){

return estadoActual;

}

/\*\*

\* Metodo <I>setState</I> que modifica el Estado del Sprite.<P>

\* @param newState, cambia el estado del Sprite

\*/

public void setState(String newState){

estadoActual = newState;

}

/\*\*

\* Metodo cambiaVida que modifica las vidas del Sprite

\* @param cant, cambia el estado del Sprite

\*/

public void cambiaVida(int cant){

if(vida>=0 && vida<=vidaMax) //Si las nuevas vidas son menores a su maxima vida le suma las vidas actuales a las nuevas

{

vida+=cant;

if(vida>vidaMax) //si las vidas son mas que la vida maxima, las vidas se ponen al maximo

vida=vidaMax;

if(vida<0) //Si las vidas son cero se queda igual

vida=0;

}

}

/\*\*

\* Metodo getVida regresa las vidas del Sprite

\* @return int, el numero de vidas del objeto

\*/

public int getVida(){

return vida;

}

/\*\*

\* Metodo setVidaMax define las vidas maximas del Sprite

\* @param vidaMax, el numero de vidas maximas.

\*/

public void setVidaMax(int vidaMax){

this.vidaMax = vidaMax;

}

/\*\*

\* Metodo iniciaVida inicializa las vidas del Sprite con la maxima posible

\*/

public void iniciaVida(){

vida = vidaMax;

}

/\*\*

\* Metodo getMaxVida regresa las vidas del Sprite

\* @return int, el numero de vidas del objeto

\*/

public int getMaxVida(){

return vidaMax;

}

}

**import** java.awt.Image;

**import** java.util.Vector;

**import** java.io.File;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.FileReader;

**import** java.util.ArrayList;

**import** java.awt.Toolkit;

**import** javax.swing.ImageIcon;

**public** **class** TileMap {

**private** Image[][] tiles;

**private** Player player; // Jugador

**private** Vector<Sprite> sprites; // Enemigos

**private** ArrayList<Image> lista; // Lista de imagenes a cargar

**private** Vector <Bala> balas; // Vector de Balas del Jugador

**private** Vector <Bala> balasEnemigos; // Vector Balas de Enemigos

**private** Vector <Explosiones> explosiones; // Vector de explosiones

**private** **int** puntos;

/\*\*

\* Constructor <I>TileMap</I> para crear un objeto de tipo TileMap.<P>

\*/

**public** TileMap(**int** width, **int** height) {

tiles = **new** Image[width][height]; // Lista en la que guarda las imagenes de los Tiles que se generen

**new** Vector <Sprite>();

balas = **new** Vector<Bala>(); // Vector de Balas

balasEnemigos = **new** Vector <Bala>(); // Vector donde iran las balas que disparen los enemigos

sprites = **new** Vector<Sprite>(); // Lista encadenada de Sprites (enemigos u objetos)

explosiones = **new** Vector<Explosiones>();// Vector de explosiones

puntos = 0;

cargaTiles(); //Carga al ArrayList lista las imagenes A,B,C,etc de la carpeta Imagenes.

}

/\*\*

\* Metodo <I>getWidth()</I> <P>

\* Devuelve el numero de Tiles a lo ancho.

\*/

**public** **int** getWidth() {

**return** tiles.length;

}

**public** **int** getPuntos(){

**return** puntos;

}

**public** **void** setPuntos(**int** P){

**this**.puntos = P;

}

/\*\*

\* Metodo <I>getHeight()</I> <P>

\* Devuelve el numero de Tiles a lo largo.

\*/

**public** **int** getHeight() {

**return** tiles[0].length;

}

/\*\*

\* Metodo <I>addExplosion()</I> <P>

\* Agrega un objeto Explosion al vector PowerUps.

\* **@param** Explosion exp es el <code>objeto Explosion</code> a agregar.

\*/

**public** **void** addExplosion(Explosiones exp){

explosiones.add(exp); // Agrega un PowerUp al vector

}

/\*\*

\* Metodo <I>removeExplosion()</I> <P>

\* Quita un objeto Explosion al vector explosions.

\* **@param** Explosiones exp es el <code>objeto Explosion</code> a quitar.

\*/

**public** **void** removeExplosion(Explosiones exp){

explosiones.removeElement(exp); // Quita un explosion del vector

}

/\*\*

\* Metodo <I>getExplosiones()</I> <P>

\* regresa el vector Explosions.

\*/

**public** Vector <Explosiones> getExplosiones(){

**return** explosiones; // Regresa el vector de Explosiones

}

/\*\*

\* Metodo <I>getTile()</I> <P>

\* Devuelve la Imagen que corresponde al Tile en cierta posicion

\* **@param** int x es el <code>entero</code> de la posicion en X del Tile.

\* **@param** int y es el <code>entero</code> de la posicion en Y del Tile.

\*/

**public** Image getTile(**int** x, **int** y) {

**if** (x < 0 || x >= getWidth() ||

y < 0 || y >= getHeight())

{

**return** **null**;

}

**else** {

**return** tiles[x][y];

}

}

/\*\*

\* Metodo <I>setTile()</I> <P>

\* Coloca un Tile en la posicion especificada.

\* **@param** int x es el <code>entero</code> de la posicion en X del Tile.

\* **@param** int y es el <code>entero</code> de la posicion en Y del Tile.

\* **@param** Image tile es el <code>Image</code> del tile.

\*/

**public** **void** setTile(**int** x, **int** y, Image tile) {

tiles[x][y] = tile; // Se coloca un tile en la posicion dada

}

/\*\*

\* Metodo <I>addBala()</I> <P>

\* Agrega un objeto Bala al vector balas.

\* **@param** Bala b es el <code>objeto Bala</code> a agregar.

\*/

**public** **void** addBala(Bala b){

balas.add(b); // Agrega una bala al vector

}

/\*\*

\* Metodo <I>removeBala()</I> <P>

\* Quita un objeto Bala al vector balas.

\* **@param** Bala b es el <code>objeto Bala</code> a agregar.

\*/

**public** **void** removeBala(Bala b){

balas.removeElement(b); // Quita una bala del vector

}

/\*\*

\* Metodo <I>getBalas()</I> <P>

\* regresa el vector balas.

\*/

**public** Vector <Bala> getBalas(){

**return** balas; // Regresa el vector de balas

}

/\*\*

\* Metodo <I>getPlayer()</I> <P>

\* Regresa el objeto Player correspondiente al jugador

\*/

**public** Player getPlayer() {

**return** player; // Regresa el objeto Jugador

}

/\*\*

\* Metodo <I>setPlayer()</I> <P>

\* Asigna al mapa el objeto Player recibido como parametro

\* **@param** Player player es el <code>Player</code> a asignar.

\*/

**public** **void** setPlayer(Player player) {

**this**.player = player;

}

//////////////////////////SECCION BALAS DE ENEMIGOS//////////////////

/\*\*

\* Metodo <I>addBalaenemigos()</I> <P>

\* Agrega un objeto Bala al vector balas de Enemigos.

\* **@param** Bala b es el <code>objeto Bala</code> a agregar.

\*/

**public** **void** addBalaEnemigos(Bala b){

balasEnemigos.add(b);

}

/\*\*

\* Metodo <I>removeBalaEnemigos()</I> <P>

\* Quita un objeto Bala al vector balas de enemigos.

\* **@param** Bala b es el <code>objeto Bala</code> a agregar.

\*/

**public** **void** removeBalaEnemigos(Bala b){

balasEnemigos.removeElement(b);

}

/\*\*

\* Metodo <I>getBalasJefs()</I> <P>

\* regresa el vector balas de Enemigos.

\*/

**public** Vector<Bala> getBalasEnemigos(){

**return** balasEnemigos;

}

//////////////////////////////FIN DE SECCION BALAS DE ENEMIGOS//////////////////////

/\*\*

\* Metodo <I>addSprite()</I> <P>

\* Coloca un Sprite en la posicion especificada.

\* **@param** int x es el <code>entero</code> de la posicion en X del Sprite.

\* **@param** int y es el <code>entero</code> de la posicion en Y del Sprite.

\* **@param** Image tile es el <code>Image</code> del Sprite.

\* **@param** TileMap map es el <code>TileMap</code> actual.

\*/

**private** **void** addSprite(TileMap map, Sprite sprite, **int** tileX, **int** tileY){

**if** (sprite != **null**) {

Image imagen = sprite.getImageFromAnimation();

// Centra al sprite

sprite.setPosX(TileMapRenderer.*tilesToPixels*(tileX) + (TileMapRenderer.*tilesToPixels*(1) - imagen.getWidth(**null**)) / 2);

// Pone al sprite en el piso.

sprite.setPosY(TileMapRenderer.*tilesToPixels*(tileY + 1) - imagen.getHeight(**null**));

//Asigna una velocidad al sprite

sprite.setVelocidadX(0.05f);

//ia se termino el bloque a quitar

map.addSprite(sprite); //Se agrega el sprite al mapa

}

}

/\*\*

\* Metodo <I>addSprite()</I> <P>

\* Agrega un objeto Sprite al vector sprites.

\* **@param** Sprite sprite es el <code> Sprite</code> a agregar.

\*/

**public** **void** addSprite(Sprite sprite) {

sprites.add(sprite);

}

/\*\*

\* Metodo <I>removeSprite()</I> <P>

\* Remueve un objeto Sprite del vector sprites.

\* **@param** Sprite sprite es el <code> Sprite</code> a quitar.

\*/

**public** **void** removeSprite(Sprite sprite) {

sprites.remove(sprite);

}

/\*\*

\* Metodo <I>getSprites()</I> <P>

\* Regresa un vector de los Sprites del mapa

\*/

**public** Vector<Sprite> getSprites() {

**return** sprites;

}

/\*\*

\* Metodo <I>loadMap()</I> <P>

\* Con base en un archivo de texto asigna tiles a una matriz

\* Tambien se encarga de asignar la posicion de inicio de los enemigos

\* y items en el mapa

\* **@param** nivel, entero que indica el numero de mapa correspondiente al nivel

\*/

**public** TileMap loadMap(**int** nivel) **throws** IOException {

ArrayList<String> lines = **new** ArrayList<String>();

**int** width = 0;

**int** height = 0;

// Se carga el archivo de texto

BufferedReader reader = **new** BufferedReader(

**new** FileReader("Mundos/Mundo"+nivel+".txt"));

**while** (**true**) {

// Se lee el archivo de texto

String line = reader.readLine();

// no quedan mas lineas

**if** (line == **null**) {

reader.close();

**break**;

}

// Cuenta cada linea que no empieze con # (comentarios)

**if** (!line.startsWith("#")) {

lines.add(line); //guarda las lineas en un ArrayList

width = Math.*max*(width, line.length()); //Saca cuanto midio la linea mas larga

}

}

height = lines.size(); //Saca cuanto fueron en total las lineas

TileMap newMap = **new** TileMap(width, height); //Genera un nuevo Mapa con el ancho y alto obtenidos

**for** (**int** y = 0; y < height; y++) {

String line = (String)lines.get(y); //Va leyendo linea por linea del ArrayList lineas

**for** (**int** x=0; x<line.length(); x++) {

**char** ch = line.charAt(x); //Lee caracter por caracter

**int** tile = ch - 'A'; //Asigna a tile el valor en entero de la Letra. Ejemplo A = 0, B=1, C=2,etc, etc.

**if** (tile >= 0 && tile < lista.size()) {

newMap.setTile(x, y, (Image)lista.get(tile)); //Agrega un Tile al mapa con el numero de columna y linea y la imagen obtenida de la lista formada por cargaTiles

}

**if**(ch == '+'){ // Agrega un enemigo al mapa

addSprite(newMap, **new** Enemigo(0, 0,

Toolkit.*getDefaultToolkit*().getImage("images/IMG\_HECTOR/Enemigo/Camina/alien1camina1.png")),x,y);

}

}

}

//Agrega al jugador al mapa

player = **new** Player(0, 0, Toolkit.*getDefaultToolkit*().getImage("images/corre1.png"));

player.setPosX(TileMapRenderer.*tilesToPixels*(3));

player.setPosY(80);

newMap.setPlayer(player);

// regresa el objeto de Mapa

**return** newMap;

}

/\*\*

\* Metodo <I>cargaTiles()</I> <P>

\* Lee las imagenes a usarse por los tiles.

\*/

**public** **void** cargaTiles() {

// Busca las imagenes de tiles, basandose en orden alfabetico

lista = **new** ArrayList<Image>();

**char** ch = 'A';

**while** (**true**) {

String name = ch + ".png";

File file = **new** File("images/" + name);

**if** (!file.exists()) {

**break**;

}

//Si encuentra un tile, lo agrega al listado de imagenes

lista.add(loadImage(name));

ch++;

}

}

/\*\*

Carga las imagenes

\*/

**public** Image loadImage(String name) {

String filename = "images/" + name;

**return** **new** ImageIcon(filename).getImage();

}

}

**import** java.awt.geom.AffineTransform;

**import** java.awt.Color;

**import** java.util.Vector;

**import** java.util.Enumeration;

**import** java.awt.\*;

**public** **class** TileMapRenderer {

**public** **static** **final** **int** *TILE\_SIZE* = 64;

**private** Image background;

/\*\*

\* Metodo int <I>pixelsToTiles()</I> <P>

\* Llama al metodo int pixelsToTiles con parametro Integer

\* **@param** pixels es el <code>float</code> del pixel a convertir.

\*/

**public** **static** **int** pixelsToTiles(**float** pixels) {

**return** *pixelsToTiles*(Math.*round*(pixels));

}

/\*\*

\* Metodo int <I>pixelsToTiles()</I> <P>

\* Devuelve el numero de Tile al que corresponde el pixel

\* **@param** pixels es el <code>float</code> del pixel a convertir.

\*/

**public** **static** **int** pixelsToTiles(**int** pixels) {

**return** (**int**)Math.*floor*((**float**)pixels / *TILE\_SIZE*);

}

/\*\*

\* Metodo int <I>tilesToPixels()</I> <P>

\* Devuelve el pixel al que corresponde el numero de Tile

\* **@param** numTiles es el <code>int</code> del Tile a convertir.

\*/

**public** **static** **int** tilesToPixels(**int** numTiles) {

**return** numTiles \* *TILE\_SIZE*;

}

/\*\*

\* Metodo <I>setBackground()</I> <P>

\* Cambia la imagen de fondo

\* **@param** background es el <code>Image</code> de la imagen de fondo.

\*/

**public** **void** setBackground(Image background) {

**this**.background = background;

}

/\*\*

\* Metodo <I>draw()</I> <P>

\* Pinta el mapa, los Sprites, el Player, las Balas y los Items

\* **@param** g es el <code>Graphics2D</code> objeto grafico.

\* **@param** map es el <code>TileMap</code>.

\* **@param** screenWidth es el <code>int</code> del ancho del mapa.

\* **@param** screenHeight es el <code>int</code> del alto del mapa.

\*/

@SuppressWarnings("unchecked")

**public** **void** draw(Graphics2D g, TileMap map, **int** screenWidth, **int** screenHeight)

{

Sprite player = map.getPlayer(); //Consigue el objeto Jugador.

Vector<Sprite> otrasImagenes = map.getSprites(); //Consigue el vector de enemigos.

**int** mapWidth = *tilesToPixels*(map.getWidth()); //Pasa el ancho del mapa de Tiles a pixeles.

**int** mapHeight = *tilesToPixels*(map.getHeight()); //Pasa la altura del mapa de Tiles a pixeles.

**int** offsetX = screenWidth / 2 - Math.*round*(player.getPosX()) - *TILE\_SIZE*; //Obtiene el desplazamiento del mapa basado en la posicion del jugador

offsetX = Math.*min*(offsetX, 0); //Decide si desplazar o no el mapa, al estar en la orilla izquierda.

offsetX = Math.*max*(offsetX, screenWidth - mapWidth); //Decide si desplazar o no el mapa, al estar en la orilla derecha.

**int** offsetY= screenHeight/2 - Math.*round*(player.getPosY()); //Igual que lo anterior, pero para un desplazamiento vertical

offsetY = Math.*min*(offsetY, 0);

offsetY = Math.*max*(offsetY, screenHeight-mapHeight);

**if** (background == **null** || screenHeight > background.getHeight(**null**)) //En caso de no tener un background, se pinta de negro el fondo

{

g.setColor(Color.*black*);

g.fillRect(0, 0, screenWidth, screenHeight);

}

// Se pinta el background, usando el desplazamiento obtenido previamente. (Brackeen)

**if** (background != **null**) {

**int** x = offsetX \* (screenWidth - background.getWidth(**null**)) / (screenWidth - mapWidth);

**int** y = screenHeight - background.getHeight(**null**);

y = offsetY \* (screenHeight - background.getHeight(**null**)) / (screenHeight - mapHeight);

g.drawImage(background, x, y, **null**);

}

// Se dibujan los Tiles visibles

**int** firstTileX = *pixelsToTiles*(-offsetX);

**int** lastTileX = firstTileX + *pixelsToTiles*(screenWidth) + 1;

**for** (**int** y = 0; y < map.getHeight(); y++) {

**for** (**int** x = firstTileX; x <= lastTileX; x++) {

Image image = map.getTile(x, y);

**if** (image != **null**) {

g.drawImage(image,

*tilesToPixels*(x) + offsetX,

*tilesToPixels*(y) + offsetY,

**null**);

}

}

}

//(Brackeen)

**for** (**int** x = 0; x < otrasImagenes.size(); x++){ // Se recorre el vector de enemigos

Sprite sprite = (Sprite)otrasImagenes.get(x);

**if** (!sprite.volteaADerecha()) { // Se ve si el enemigo voltea a la derecha

g.drawImage(getFlippedImage(sprite.getImageFromAnimation(),g), // En caso afirmativo, consigue una imagen girada

Math.*round*(sprite.getPosX()) + offsetX,

Math.*round*(sprite.getPosY()) + offsetY,**null**);

} **else** { // En caso contrario, deja la imagen como esta y la pinta

g.drawImage(sprite.getImageFromAnimation(), Math.*round*(sprite.getPosX()) + offsetX,

Math.*round*(sprite.getPosY()) + offsetY,**null**);

}

g.setColor(Color.*WHITE*);

}

//Se pinta al jugador

**if** (((Player)player).getLado()) {

g.drawImage(getFlippedImage(player.getImagenI(), g), Math.*round*(player.getPosX()) + offsetX, // Si esta volteando, gira la imagen

Math.*round*(player.getPosY()) + offsetY, **null**);

} **else** {

g.drawImage(player.getImagenI(), Math.*round*(player.getPosX()) + offsetX, // Se pinta la imagen normal, sino esta girando

Math.*round*(player.getPosY()) + offsetY, **null**);

}

Vector<Explosiones> explosiones = map.getExplosiones(); // Consigue las explosiones y las mete a un vector

//Se crea una enumeracion del vector explosiones

Enumeration e = explosiones.elements();

Explosiones exp;

**while** (e.hasMoreElements()) { // Se recorre el vector de PowerUps

exp = (Explosiones) e.nextElement();

g.drawImage(exp.getImagenI(), Math.*round*(exp.getPosX()) + offsetX - 20,

Math.*round*(exp.getPosY()) + offsetY - 40, **null**);

}

// Seccion en donde se pintan las balas

Vector <Bala> balas = map.getBalas(); // Se consigue el vector de balas

e = balas.elements();

Bala b;

**while** (e.hasMoreElements()) { // Se recorre el vector de balas

b = (Bala) e.nextElement();

**if** (b.getVelocidadX() < 0) { // Si la bala va a la izquierda, se pinta la imagen invertida de la bala

g.drawImage(getFlippedImage(b.getImagenI(), g), Math.*round*(b.getPosX()) + offsetX,

Math.*round*(b.getPosY()) + offsetY, **null**);

} **else** { // Si no, se pinta la imagen normal de la bala

g.drawImage(b.getImagenI(), Math.*round*(b.getPosX()) + offsetX,

Math.*round*(b.getPosY()) + offsetY, **null**);

}

}

// Se pintan las balas de los enemigos

balas = map.getBalasEnemigos(); //Se consigue el vector de balasde enemigos

e = balas.elements();

**while** (e.hasMoreElements()) { // Se recorre el vector de balas

b = (Bala) e.nextElement();

**if** (b.getVelocidadX()<0) { // Si la bala va a la izquierda, se consigue una imagen invertida

g.drawImage(getFlippedImage(b.getImagenI(), g), Math.*round*(b.getPosX()) + offsetX,

Math.*round*(b.getPosY()) + offsetY, **null**);

} **else** {

g.drawImage(b.getImagenI(), Math.*round*(b.getPosX()) + offsetX, // Se pinta la imagen normal

Math.*round*(b.getPosY()) + offsetY, **null**);

}

}

///PARA DESPLEGAR BARRA DE VIDA DEL PLAYER

**int** vidas = ((Player)player).getVida(); // Se Consigue la vida del personaje.

//Estetica (Colores, estilo de letra, posicion, etc.)

g.setColor(Color.*RED*);

g.fillRect(160, 545, vidas\*10, 25);

g.setFont(**new** Font("Arial", Font.*BOLD*, 30));

g.setColor(Color.*WHITE*);

g.drawString("VIDA", 50, 570);

g.drawString("Score: "+map.getPuntos(),50,50);

}

/\*\*

\* Metodo <I>getFlippedImage()</I> <P>

\* Gira la imagen horizontalmente

\* **@param** image es el <code>Image</code> de la imagen a voltear.

\* **@param** g es <code>Graphics</code> a pintar

\*/

**public** Image getFlippedImage(Image image,Graphics g) {

// Crea un objeto de Transform

AffineTransform transform = **new** AffineTransform();

transform.scale(-1, 1); //Hace que la transformacion sea horizontal.

transform.translate((-2) \* image.getWidth(**null**) / 2,(0) \* image.getHeight(**null**) / 2); //Gira la imagen, tomando como eje la mitad de la imagen

GraphicsEnvironment ge = GraphicsEnvironment.*getLocalGraphicsEnvironment*(); //Crea un objeto de Ambiente Grafico

GraphicsDevice gs = ge.getDefaultScreenDevice(); //Consigue el Dispositivo grafico

GraphicsConfiguration gc = gs.getDefaultConfiguration(); //Consigue la configuracion actual

Image newImage = gc.createCompatibleImage( //Crea una nueva imagen, manteniendo la misma altura, ancho y transparencias del original.

image.getWidth(**null**),

image.getHeight(**null**),

Transparency.*BITMASK*);

// Se dibuja la imagen transformada

Graphics2D g2 = (Graphics2D)newImage.getGraphics();

g2.drawImage(image, transform, **null**);

g2.dispose();

**return** newImage; //Se regresa la imagen invertida

}

}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | Class | Use | **Tree** | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| PREV   NEXT | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?overview-tree.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\overview-tree.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |

## Hierarchy For All Packages

## Class Hierarchy

* java.lang.Object
  + [**Animation**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html)
  + java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - java.awt.Container
      * java.awt.Window (implements javax.accessibility.Accessible)
        + java.awt.Frame (implements java.awt.MenuContainer)

javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)

[**GameMain**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html)

* + [**Puntaje**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html) (implements java.lang.Comparable<T>)
  + [**SoundClip**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html)
  + [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)
    - [**Bala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)
    - [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)
      * [**Enemigo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html)
      * [**Player**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html)
    - [**Explosiones**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)
  + [**TileMap**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html)
  + [**TileMapRenderer**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | Class | Use | **Tree** | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| PREV   NEXT | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?overview-tree.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\overview-tree.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | Class | Use | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\overview-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| PREV   NEXT | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?constant-values.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |

# Constant Field Values

**Contents**

* [<Unnamed>](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#unnamed_package)

|  |
| --- |
| **<Unnamed>.\*** |

|  |  |  |
| --- | --- | --- |
| [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html) | | |
| public static final int | [STATE\_DEAD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DEAD) | 2 |
| public static final int | [STATE\_DYING](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DYING) | 1 |
| public static final int | [STATE\_NORMAL](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_NORMAL) | 0 |

|  |  |  |
| --- | --- | --- |
| [**GameMain**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html) | | |
| public static final float | [GRAVITY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#GRAVITY) | 0.0020000000949949026f |
| public static final float | [VELOCIDADBALA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#VELOCIDADBALA) | 0.699999988079071f |
| public static final float | [VELOCIDADCAIDA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#VELOCIDADCAIDA) | 7.0f |

|  |  |  |
| --- | --- | --- |
| [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) | | |
| public static final java.lang.String | [CHOQUE\_CON\_PARED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED) | "ChoqueConPared" |
| public static final java.lang.String | [DERECHA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO) | "paradoYAtacando" |
| public static final java.lang.String | [ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA) | "EstaCerca" |
| public static final java.lang.String | [IZQUIERDA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO) | "paradoYAtacandoD" |
| public static final java.lang.String | [NO\_ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA) | "noEstaCerca" |
| public static final java.lang.String | [PROXIMA\_CAIDA\_LIBRE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) | "ProximaCaidaLibre" |

|  |  |  |
| --- | --- | --- |
| [**TileMapRenderer**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html) | | |
| public static final int | [TILE\_SIZE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#TILE_SIZE) | 64 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | Class | Use | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\overview-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| PREV   NEXT | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?constant-values.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **[Package](file:///C:\\Users\\samuel%20heaney\\workspace\\CopyBatalla\\doc\\package-summary.html)** | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Animation.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| PREV CLASS   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Animation.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#method_detail) |

## Class Animation

java.lang.Object

extended by **Animation**

public class **Animation**

extends java.lang.Object

Clase Animation, es la clase encargada de manejar las series de imagenes y el tiempo en que esas imagemenes seran mostradas para crear una animacion

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Animation**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#Animation%28%29)() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**addFrame**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#addFrame%28java.awt.Image,%20long%29)(java.awt.Image image, long duration) |
| java.awt.Image | [**getImage**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#getImage%28%29)()            Metodo getImage,regresa la imagen actual de la animacion |
| void | [**start**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#start%28%29)() |
| void | [**update**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html#update%28long%29)(long elapsedTime) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Animation

public **Animation**()

|  |
| --- |
| **Method Detail** |

### addFrame

public void **addFrame**(java.awt.Image image,

long duration)

### start

public void **start**()

### update

public void **update**(long elapsedTime)

### getImage

public java.awt.Image **getImage**()

Metodo getImage,regresa la imagen actual de la animacion

**Returns:**

la imagen actual de la animacion

## Class Bala

java.lang.Object

extended by [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

extended by **Bala**

public class **Bala**

extends [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

Clase Bala, hija de Sprite, esta clase es la encargada de controlar las balas del jugador y enemigos, su impresion en el mapa, movimiento y daño.

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [CHOQUE\_CON\_PARED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED), [DERECHA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO), [ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA), [IZQUIERDA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO), [NO\_ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA), [PROXIMA\_CAIDA\_LIBRE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Bala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#Bala%28int,%20int,%20java.awt.Image%29)(int posX, int posY, java.awt.Image image)            Metodo constructor usado para crear el objeto |  |
| [**Bala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#Bala%28int,%20int,%20java.awt.Image,%20float%29)(int posX, int posY, java.awt.Image image, float velocidad)            Metodo constructor usado para crear el objeto |  |
| [**Bala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#Bala%28int,%20int,%20java.awt.Image,%20float,%20float%29)(int posX, int posY, java.awt.Image image, float velocidadX, float velocidadY)            Metodo constructor usado para crear el objeto |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actualizaPosicionI**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#actualizaPosicionI%28%29)()            Metodo *actualizaPosicionI* usado para actualizar la posicion en X de la Bala |
| void | [**cambiaInicial**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#cambiaInicial%28float%29)(float num)            Metodo *cambiaInicial* Actualiza la distancia recorrida por la Bala |
| void | [**cargaImagenes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#cargaImagenes%28%29)()            Metodo *cargaImagenes()* |
| int | [**getDamage**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#getDamage%28%29)()            Metodo *getDamage* |
| float | [**getInicial**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#getInicial%28%29)()            Metodo *getInicial* Devuelve la distancia recorrida por la Bala |
| float | [**getPosicionF**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#getPosicionF%28%29)()            Metodo *getPosicionF* Devuelve la posicion en X de la Bala |
| int | [**getTipoDisparo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#getTipoDisparo%28%29)() |
| void | [**setDamage**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#setDamage%28int%29)(int num)            Metodo *setDamage* Da un nuevo valor al daño de la bala |
| void | [**setTipoDisparo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#setTipoDisparo%28int%29)(int num) |
| void | [**updateAnim**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html#updateAnim%28long%29)(long elapsedTime)            Metodo *updateAnim* que actualiza la animacion del Sprite. |

|  |
| --- |
| **Methods inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [actualiza](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#actualiza%28long%29), [atacar](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#atacar%28%29), [cambiaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#cambiaVida%28int%29), [checaProximidad](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#checaProximidad%28Sprite%29), [clone](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#clone%28%29), [dibujaBala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#dibujaBala%28boolean,%20TileMap%29), [getAlto](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAlto%28%29), [getAncho](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAncho%28%29), [getCurrState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getCurrState%28%29), [getHeight](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getHeight%28%29), [getImageFromAnimation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageFromAnimation%28%29), [getImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageIcon%28%29), [getImagenI](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImagenI%28%29), [getMaxVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getMaxVida%28%29), [getPerimetro](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPerimetro%28%29), [getPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosX%28%29), [getPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosY%28%29), [getVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadX%28%29), [getVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadY%28%29), [getVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVida%28%29), [getWidth](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getWidth%28%29), [iniciaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#iniciaVida%28%29), [intersecta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersecta%28Sprite%29), [intersectaAbajo](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersectaAbajo%28Sprite%29), [puedeMorir](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#puedeMorir%28%29), [revisarCambioDeEstado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeEstado%28java.lang.String,%20float%29), [revisarCambioDeVuelta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeVuelta%28TileMap,%20float,%20float,%20float%29), [setImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setImageIcon%28java.awt.Image%29), [setLado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setLado%28int%29), [setPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosX%28float%29), [setPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setState%28java.lang.String%29), [setVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadX%28float%29), [setVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadY%28float%29), [setVidaMax](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVidaMax%28int%29), [volteaADerecha](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#volteaADerecha%28%29) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Bala

public **Bala**(int posX,

int posY,

java.awt.Image image)

Metodo constructor usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

### Bala

public **Bala**(int posX,

int posY,

java.awt.Image image,

float velocidad)

Metodo constructor usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

velocidad - es la velocidad que la bala tendra.

### Bala

public **Bala**(int posX,

int posY,

java.awt.Image image,

float velocidadX,

float velocidadY)

Metodo constructor usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

velocidadX - es la velocidadX que la bala tendra en el eje X.

velocidadY - es la velocidadY que la bala tendra en el eje Y.

|  |
| --- |
| **Method Detail** |

### getDamage

public int **getDamage**()

Metodo *getDamage*

**Returns:**

Devuelve el daño realizado por la bala

### setDamage

public void **setDamage**(int num)

Metodo *setDamage* Da un nuevo valor al daño de la bala

### actualizaPosicionI

public void **actualizaPosicionI**()

Metodo *actualizaPosicionI* usado para actualizar la posicion en X de la Bala

### getPosicionF

public float **getPosicionF**()

Metodo *getPosicionF* Devuelve la posicion en X de la Bala

### getInicial

public float **getInicial**()

Metodo *getInicial* Devuelve la distancia recorrida por la Bala

### cambiaInicial

public void **cambiaInicial**(float num)

Metodo *cambiaInicial* Actualiza la distancia recorrida por la Bala

### setTipoDisparo

public void **setTipoDisparo**(int num)

### getTipoDisparo

public int **getTipoDisparo**()

### updateAnim

public void **updateAnim**(long elapsedTime)

**Description copied from class:** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29)

Metodo *updateAnim* que actualiza la animacion del Sprite.

**Overrides:**

[updateAnim](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

### cargaImagenes

public void **cargaImagenes**()

Metodo *cargaImagenes()*

Carga todas las imagenes que se usaran por el Player

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Bala.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Creature.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Creature.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#method_detail) |

## Class Creature

java.lang.Object

extended by [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

extended by **Creature**

**Direct Known Subclasses:**

[Enemigo](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html), [Player](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html)

public abstract class **Creature**

extends [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

Clase Creatura, Una creatura es un Sprite que es afectado por la gravedad y puede ser destruido. Tiene 2 animaciones movimiento a la izquierda y movimiento a la derecha.

|  |  |
| --- | --- |
| **Field Summary** | |
| static int | [**STATE\_DEAD**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DEAD) |
| static int | [**STATE\_DYING**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DYING) |
| static int | [**STATE\_NORMAL**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_NORMAL) |

|  |
| --- |
| **Fields inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [CHOQUE\_CON\_PARED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED), [DERECHA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO), [ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA), [IZQUIERDA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO), [NO\_ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA), [PROXIMA\_CAIDA\_LIBRE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#Creature%28int,%20int,%20Animation%29)(int posX, int posY, [Animation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html) anim)            Metodo constructor de Creature usado para crear el objeto |  |
| [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#Creature%28int,%20int,%20java.awt.Image%29)(int posX, int posY, java.awt.Image imagen)            Metodo constructor de Creature usado para crear el objeto |  |
| [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#Creature%28int,%20int,%20java.awt.Image,%20int%29)(int posX, int posY, java.awt.Image imagen, int vidaMax)            Metodo constructor de Creature usado para crear el objeto |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**collideHorizontal**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideHorizontal%28%29)()            Metodo *collideHorizontal* si la creatura choca con alguna pared este metodo se encarga de voltearlo. |
| void | [**collideVertical**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideVertical%28%29)()            Metodo *collideVertical* que frena verticalmente a Creature. |
| float | [**getMaxSpeed**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#getMaxSpeed%28%29)()            Metodo *getMaxSpeed* regresa la maxima velocidad posible. |
| int | [**getState**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#getState%28%29)()            Metodo *getState* que devuelve el Estado de la Creature. |
| boolean | [**isAlive**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#isAlive%28%29)()            Metodo *isAlive* que devuelve un boolean para indicar |
| boolean | [**isFlying**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#isFlying%28%29)()            Metodo *isFlying* que devuelve un boolean para indicar si Creature esta volando o no. |
| void | [**setState**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#setState%28int%29)(int state)            Metodo *setState* que cambia el Estado de la Creature. |
| void | [**update**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#update%28long%29)(long elapsedTime)            Metodo *update* que actualiza el estado de Creature. |
| void | [**wakeUp**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#wakeUp%28%29)()            Metodo *wakeUp* Despierta a la criatura (usado para que se despierten cuando el personaje este en pantalla y que antes de eso no se muevan) |

|  |
| --- |
| **Methods inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [actualiza](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#actualiza%28long%29), [atacar](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#atacar%28%29), [cambiaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#cambiaVida%28int%29), [checaProximidad](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#checaProximidad%28Sprite%29), [clone](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#clone%28%29), [dibujaBala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#dibujaBala%28boolean,%20TileMap%29), [getAlto](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAlto%28%29), [getAncho](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAncho%28%29), [getCurrState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getCurrState%28%29), [getHeight](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getHeight%28%29), [getImageFromAnimation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageFromAnimation%28%29), [getImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageIcon%28%29), [getImagenI](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImagenI%28%29), [getMaxVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getMaxVida%28%29), [getPerimetro](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPerimetro%28%29), [getPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosX%28%29), [getPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosY%28%29), [getVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadX%28%29), [getVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadY%28%29), [getVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVida%28%29), [getWidth](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getWidth%28%29), [iniciaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#iniciaVida%28%29), [intersecta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersecta%28Sprite%29), [intersectaAbajo](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersectaAbajo%28Sprite%29), [puedeMorir](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#puedeMorir%28%29), [revisarCambioDeEstado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeEstado%28java.lang.String,%20float%29), [revisarCambioDeVuelta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeVuelta%28TileMap,%20float,%20float,%20float%29), [setImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setImageIcon%28java.awt.Image%29), [setLado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setLado%28int%29), [setPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosX%28float%29), [setPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setState%28java.lang.String%29), [setVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadX%28float%29), [setVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadY%28float%29), [setVidaMax](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVidaMax%28int%29), [updateAnim](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29), [volteaADerecha](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#volteaADerecha%28%29) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Field Detail** |

### STATE\_NORMAL

public static final int **STATE\_NORMAL**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Creature.STATE_NORMAL)

### STATE\_DYING

public static final int **STATE\_DYING**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Creature.STATE_DYING)

### STATE\_DEAD

public static final int **STATE\_DEAD**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Creature.STATE_DEAD)

|  |
| --- |
| **Constructor Detail** |

### Creature

public **Creature**(int posX,

int posY,

java.awt.Image imagen)

Metodo constructor de Creature usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

### Creature

public **Creature**(int posX,

int posY,

java.awt.Image imagen,

int vidaMax)

Metodo constructor de Creature usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

int, - seran las vidas maximas de la creatura.

### Creature

public **Creature**(int posX,

int posY,

[Animation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html) anim)

Metodo constructor de Creature usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

anim - es la Animation del objeto.

|  |
| --- |
| **Method Detail** |

### getMaxSpeed

public float **getMaxSpeed**()

Metodo *getMaxSpeed* regresa la maxima velocidad posible.

**Returns:**

0, esto significa que el movimiento no tendra ningun tipo de restriccion al momento de moverse

### wakeUp

public void **wakeUp**()

Metodo *wakeUp* Despierta a la criatura (usado para que se despierten cuando el personaje este en pantalla y que antes de eso no se muevan)

### getState

public int **getState**()

Metodo *getState* que devuelve el Estado de la Creature.

**Returns:**

state, regresa el estado de la creatura.

### setState

public void **setState**(int state)

Metodo *setState* que cambia el Estado de la Creature.

### isAlive

public boolean **isAlive**()

Metodo *isAlive* que devuelve un boolean para indicar

**Returns:**

boolean, si la creatura no ha sido destruido regresa true, si la creatura creature ha sido destruida regresa false

### isFlying

public boolean **isFlying**()

Metodo *isFlying* que devuelve un boolean para indicar si Creature esta volando o no.

**Returns:**

false si la creatura no esta volando, true si esta volando

### collideHorizontal

public void **collideHorizontal**()

Metodo *collideHorizontal* si la creatura choca con alguna pared este metodo se encarga de voltearlo.

### collideVertical

public void **collideVertical**()

Metodo *collideVertical* que frena verticalmente a Creature.

Si choca con un techo la velocidad en Y se vuelve cero y empieza su caida libre.

### update

public void **update**(long elapsedTime)

Metodo *update* que actualiza el estado de Creature.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **[Package](file:///C:\\Users\\samuel%20heaney\\workspace\\CopyBatalla\\doc\\package-summary.html)** | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Creature.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Creature.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Enemigo.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Enemigo.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#fields_inherited_from_class_Creature) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#method_detail) |

## Class Enemigo

java.lang.Object

extended by [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

extended by [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

extended by **Enemigo**

public class **Enemigo**

extends [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

Clase Enemigo, es la clase encargada de darle los aspectos de inteligencia a los enemigos y sus comportamientos

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class** [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html) |
| [STATE\_DEAD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DEAD), [STATE\_DYING](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DYING), [STATE\_NORMAL](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_NORMAL) |

|  |
| --- |
| **Fields inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [CHOQUE\_CON\_PARED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED), [DERECHA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO), [ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA), [IZQUIERDA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO), [NO\_ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA), [PROXIMA\_CAIDA\_LIBRE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Enemigo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#Enemigo%28int,%20int,%20java.awt.Image%29)(int x, int y, java.awt.Image imagen)            Constructor *Enemigo* para crear un objeto de tipo Enemigo. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| boolean | [**atacar**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#atacar%28%29)()            Metodo *atacar* hace que el Enemigo ataque dependiendo si esta cerca el jugador |
| void | [**createAnimation**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#createAnimation%28%29)()            Metodo *createAnimation* para cargar las imagenes de animacion del enemigo. |
| [Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) | [**dibujaBala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#dibujaBala%28boolean,%20TileMap%29)(boolean derecha, [TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) mapa)            Metodo Tipo Bala *dibujaBala* Encargado de obtener la imagen correspondiente y crear la bala en su posicion inicial |
| void | [**persigueA**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#persigueA%28Sprite%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) contrincante)            Metodo *persigueA* Determina si hay un jugador cerca y le asigna al enemigo atacar al jugador |
| void | [**revisarCambioDeEstado**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html#revisarCambioDeEstado%28java.lang.String,%20float%29)(java.lang.String identificador, float newX)            Metod void *revisarCambioDeEstado* que revisa el estado y/o cambia el estado del Enemigo. |

|  |
| --- |
| **Methods inherited from class** [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html) |
| [collideHorizontal](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideHorizontal%28%29), [collideVertical](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideVertical%28%29), [getMaxSpeed](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#getMaxSpeed%28%29), [getState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#getState%28%29), [isAlive](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#isAlive%28%29), [isFlying](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#isFlying%28%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#setState%28int%29), [update](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#update%28long%29), [wakeUp](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#wakeUp%28%29) |

|  |
| --- |
| **Methods inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [actualiza](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#actualiza%28long%29), [cambiaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#cambiaVida%28int%29), [checaProximidad](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#checaProximidad%28Sprite%29), [clone](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#clone%28%29), [getAlto](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAlto%28%29), [getAncho](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAncho%28%29), [getCurrState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getCurrState%28%29), [getHeight](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getHeight%28%29), [getImageFromAnimation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageFromAnimation%28%29), [getImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageIcon%28%29), [getImagenI](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImagenI%28%29), [getMaxVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getMaxVida%28%29), [getPerimetro](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPerimetro%28%29), [getPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosX%28%29), [getPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosY%28%29), [getVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadX%28%29), [getVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadY%28%29), [getVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVida%28%29), [getWidth](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getWidth%28%29), [iniciaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#iniciaVida%28%29), [intersecta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersecta%28Sprite%29), [intersectaAbajo](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersectaAbajo%28Sprite%29), [puedeMorir](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#puedeMorir%28%29), [revisarCambioDeVuelta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeVuelta%28TileMap,%20float,%20float,%20float%29), [setImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setImageIcon%28java.awt.Image%29), [setLado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setLado%28int%29), [setPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosX%28float%29), [setPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setState%28java.lang.String%29), [setVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadX%28float%29), [setVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadY%28float%29), [setVidaMax](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVidaMax%28int%29), [updateAnim](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29), [volteaADerecha](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#volteaADerecha%28%29) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Enemigo

public **Enemigo**(int x,

int y,

java.awt.Image imagen)

Constructor *Enemigo* para crear un objeto de tipo Enemigo.

**Parameters:**

x, - posicion int del objeto en el eje X

y, - posicion int del objeto en el eje Y

imagen, - Imagen con la que se inicializa el Enemigo

|  |
| --- |
| **Method Detail** |

### revisarCambioDeEstado

public void **revisarCambioDeEstado**(java.lang.String identificador,

float newX)

Metod void *revisarCambioDeEstado* que revisa el estado y/o cambia el estado del Enemigo.

**Overrides:**

[revisarCambioDeEstado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeEstado%28java.lang.String,%20float%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

**Parameters:**

identificador, - String que compara cualquier estado con el estado actual del Enemigo

newX, - float que asigna una nueva posicion para el Enemigo

### atacar

public boolean **atacar**()

Metodo *atacar* hace que el Enemigo ataque dependiendo si esta cerca el jugador

**Overrides:**

[atacar](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#atacar%28%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

**Returns:**

true, si el enemigo no estaba atacando lo forza a atacar

### persigueA

public void **persigueA**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) contrincante)

Metodo *persigueA* Determina si hay un jugador cerca y le asigna al enemigo atacar al jugador

**Parameters:**

contrincante, - Sprite al cual debe seguir el Enemigo

### dibujaBala

public [Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) **dibujaBala**(boolean derecha,

[TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) mapa)

Metodo Tipo Bala *dibujaBala* Encargado de obtener la imagen correspondiente y crear la bala en su posicion inicial

**Overrides:**

[dibujaBala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#dibujaBala%28boolean,%20TileMap%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

**Parameters:**

derecha, - booleano que indica true si esta orientado a la derecha el Enemigo

mapa, - TileMap al cual se agregan las balas

**Returns:**

null, al terminar de agregar la bala la saca del TileMap

### createAnimation

public void **createAnimation**()

Metodo *createAnimation* para cargar las imagenes de animacion del enemigo.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Explosiones.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Explosiones.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#fields_inherited_from_class_Sprite) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#method_detail) |

## Class Explosiones

java.lang.Object

extended by [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

extended by **Explosiones**

public class **Explosiones**

extends [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [CHOQUE\_CON\_PARED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED), [DERECHA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO), [ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA), [IZQUIERDA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO), [NO\_ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA), [PROXIMA\_CAIDA\_LIBRE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Explosiones**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#Explosiones%28int,%20int,%20java.awt.Image%29)(int posX, int posY, java.awt.Image image)            Metodo constructor usado para crear el objeto |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**cargaImagenes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#cargaImagenes%28%29)()            Metodo *cargaImagenes()* |
| boolean | [**termina**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#termina%28%29)()            Metodo Boleano *Termina()* |
| void | [**updateAnim**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#updateAnim%28long%29)(long elapsedTime)            Metodo *updateAnim()* |

|  |
| --- |
| **Methods inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [actualiza](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#actualiza%28long%29), [atacar](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#atacar%28%29), [cambiaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#cambiaVida%28int%29), [checaProximidad](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#checaProximidad%28Sprite%29), [clone](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#clone%28%29), [dibujaBala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#dibujaBala%28boolean,%20TileMap%29), [getAlto](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAlto%28%29), [getAncho](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAncho%28%29), [getCurrState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getCurrState%28%29), [getHeight](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getHeight%28%29), [getImageFromAnimation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageFromAnimation%28%29), [getImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageIcon%28%29), [getImagenI](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImagenI%28%29), [getMaxVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getMaxVida%28%29), [getPerimetro](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPerimetro%28%29), [getPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosX%28%29), [getPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosY%28%29), [getVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadX%28%29), [getVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadY%28%29), [getVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVida%28%29), [getWidth](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getWidth%28%29), [iniciaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#iniciaVida%28%29), [intersecta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersecta%28Sprite%29), [intersectaAbajo](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersectaAbajo%28Sprite%29), [puedeMorir](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#puedeMorir%28%29), [revisarCambioDeEstado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeEstado%28java.lang.String,%20float%29), [revisarCambioDeVuelta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeVuelta%28TileMap,%20float,%20float,%20float%29), [setImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setImageIcon%28java.awt.Image%29), [setLado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setLado%28int%29), [setPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosX%28float%29), [setPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setState%28java.lang.String%29), [setVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadX%28float%29), [setVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadY%28float%29), [setVidaMax](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVidaMax%28int%29), [volteaADerecha](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#volteaADerecha%28%29) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Explosiones

public **Explosiones**(int posX,

int posY,

java.awt.Image image)

Metodo constructor usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

|  |
| --- |
| **Method Detail** |

### updateAnim

public void **updateAnim**(long elapsedTime)

Metodo *updateAnim()*

cambia la animacion de la explosion. Si ha pasado cierto tiempo desde que comenzo la animacion, se cambia de animacion y se cambia el Estado.

**Overrides:**

[updateAnim](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

### termina

public boolean **termina**()

Metodo Boleano *Termina()*

**Returns:**

Booleano si la explosion debe de quitarse o no de la pantalla

### cargaImagenes

public void **cargaImagenes**()

Metodo *cargaImagenes()*

Carga todas las imagenes de Explosiones que se usaran

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Explosiones.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Enemigo.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Explosiones.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#fields_inherited_from_class_Sprite) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\GameMain.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?GameMain.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: [NESTED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#nested_classes_inherited_from_class_java.awt.Component) | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#method_detail) |

## Class GameMain

java.lang.Object

extended by java.awt.Component

extended by java.awt.Container

extended by java.awt.Window

extended by java.awt.Frame

extended by javax.swing.JFrame

extended by **GameMain**

**All Implemented Interfaces:**

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

public class **GameMain**

extends javax.swing.JFrame

**See Also:**

[Serialized Form](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\serialized-form.html#GameMain)

|  |
| --- |
| **Nested Class Summary** |

|  |
| --- |
| **Nested classes/interfaces inherited from class java.awt.Component** |
| java.awt.Component.BaselineResizeBehavior |

|  |  |
| --- | --- |
| **Field Summary** | |
| static float | [**GRAVITY**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#GRAVITY) |
| static float | [**VELOCIDADBALA**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#VELOCIDADBALA) |
| static float | [**VELOCIDADCAIDA**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#VELOCIDADCAIDA) |

|  |
| --- |
| **Fields inherited from class javax.swing.JFrame** |
| EXIT\_ON\_CLOSE |

|  |
| --- |
| **Fields inherited from class java.awt.Frame** |
| CROSSHAIR\_CURSOR, DEFAULT\_CURSOR, E\_RESIZE\_CURSOR, HAND\_CURSOR, ICONIFIED, MAXIMIZED\_BOTH, MAXIMIZED\_HORIZ, MAXIMIZED\_VERT, MOVE\_CURSOR, N\_RESIZE\_CURSOR, NE\_RESIZE\_CURSOR, NORMAL, NW\_RESIZE\_CURSOR, S\_RESIZE\_CURSOR, SE\_RESIZE\_CURSOR, SW\_RESIZE\_CURSOR, TEXT\_CURSOR, W\_RESIZE\_CURSOR, WAIT\_CURSOR |

|  |
| --- |
| **Fields inherited from class java.awt.Component** |
| BOTTOM\_ALIGNMENT, CENTER\_ALIGNMENT, LEFT\_ALIGNMENT, RIGHT\_ALIGNMENT, TOP\_ALIGNMENT |

|  |
| --- |
| **Fields inherited from interface javax.swing.WindowConstants** |
| DISPOSE\_ON\_CLOSE, DO\_NOTHING\_ON\_CLOSE, HIDE\_ON\_CLOSE |

|  |
| --- |
| **Fields inherited from interface java.awt.image.ImageObserver** |
| ABORT, ALLBITS, ERROR, FRAMEBITS, PROPERTIES, SOMEBITS |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**GameMain**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#GameMain%28%29)()            Constructor de la clase. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**checaColision**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#checaColision%28%29)()            Metodo *checaColision* |
| void | [**gameInit**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameInit%28%29)()            Metodo *gameInit* |
| void | [**gameKeyPressed**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameKeyPressed%28int%29)(int keyCode)            Metodo *gameKeyPressed* |
| void | [**gameKeyReleased**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameKeyReleased%28java.awt.event.KeyEvent%29)(java.awt.event.KeyEvent e)            Metodo *gameKeyReleased* |
| void | [**gameKeyTyped**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameKeyTyped%28java.awt.event.KeyEvent%29)(java.awt.event.KeyEvent e)            Metodo *gameKeyTyped* |
| void | [**gameShutdown**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameShutdown%28%29)()            Metodo *gameShutdown* |
| void | [**gameStart**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameStart%28%29)()            Metodo *gameStart* |
| void | [**gameUpdate**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#gameUpdate%28%29)()            Metodo *gameUpdate* |
| java.awt.Point | [**getTileCollision**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#getTileCollision%28Sprite,%20float,%20float%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) sprite, float newX, float newY) |
| void | [**grabaArchivo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#grabaArchivo%28%29)()            Metodo *grabaArchivo* que agrega la informacion del vector al archivo. |
| void | [**keyPressed**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#keyPressed%28java.awt.event.KeyEvent%29)(java.awt.event.KeyEvent e)            Metodo *keyPressed* sobrescrito de la interface KeyListener. |
| void | [**keyReleased**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#keyReleased%28java.awt.event.KeyEvent%29)(java.awt.event.KeyEvent e)            Metodo *keyReleased* sobrescrito de la interface KeyListener. |
| void | [**keyTyped**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#keyTyped%28java.awt.event.KeyEvent%29)(java.awt.event.KeyEvent e)            Metodo *keyTyped* sobrescrito de la interface KeyListener. |
| void | [**leeArchivo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#leeArchivo%28%29)()            Metodo *leeArchivo* que procesa eventos de KeyReleased que lee la informacion de un archivo y lo agrega a un vector. |
| static void | [**main**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#main%28java.lang.String[]%29)(java.lang.String[] args) |

|  |
| --- |
| **Methods inherited from class javax.swing.JFrame** |
| getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar, setLayeredPane, setLayout, setTransferHandler, update |

|  |
| --- |
| **Methods inherited from class java.awt.Frame** |
| addNotify, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated |

|  |
| --- |
| **Methods inherited from class java.awt.Window** |
| addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getIconImages, getInputContext, getListeners, getLocale, getModalExclusionType, getMostRecentFocusOwner, getOwnedWindows, getOwner, getOwnerlessWindows, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindows, getWindowStateListeners, hide, isActive, isAlwaysOnTop, isAlwaysOnTopSupported, isFocusableWindow, isFocusCycleRoot, isFocused, isLocationByPlatform, isShowing, pack, postEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, reshape, setAlwaysOnTop, setBounds, setBounds, setCursor, setFocusableWindowState, setFocusCycleRoot, setIconImages, setLocationByPlatform, setLocationRelativeTo, setMinimumSize, setModalExclusionType, setSize, setSize, setVisible, show, toBack, toFront |

|  |
| --- |
| **Methods inherited from class java.awt.Container** |
| add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, transferFocusBackward, transferFocusDownCycle, validate |

|  |
| --- |
| **Methods inherited from class java.awt.Component** |
| action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, dispatchEvent, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, requestFocus, requestFocusInWindow, resize, resize, setBackground, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setName, setPreferredSize, show, size, toString, transferFocus, transferFocusUpCycle |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, wait, wait, wait |

|  |
| --- |
| **Methods inherited from interface java.awt.MenuContainer** |
| getFont, postEvent |

|  |
| --- |
| **Field Detail** |

### GRAVITY

public static final float **GRAVITY**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#GameMain.GRAVITY)

### VELOCIDADBALA

public static final float **VELOCIDADBALA**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#GameMain.VELOCIDADBALA)

### VELOCIDADCAIDA

public static final float **VELOCIDADCAIDA**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#GameMain.VELOCIDADCAIDA)

|  |
| --- |
| **Constructor Detail** |

### GameMain

public **GameMain**()

throws java.io.IOException

Constructor de la clase.

**Throws:**

java.io.IOException

|  |
| --- |
| **Method Detail** |

### gameInit

public void **gameInit**()

throws java.io.IOException

Metodo *gameInit*

En este metodo se inizializan las variables o se crean los objetos a usarse en el juego.

**Throws:**

java.io.IOException

### gameShutdown

public void **gameShutdown**()

Metodo *gameShutdown*

Metodo utilizado para cerrar la pantalla y terminar el juego.

### gameStart

public void **gameStart**()

Metodo *gameStart*

En este metodo se crea e inicializa el hilo para la animacion.

### gameUpdate

public void **gameUpdate**()

Metodo *gameUpdate*

En este metodo se actualizan las posiciones de los personajes y objetos en el juego en caso de que se presente algun tipo de accion.

### checaColision

public void **checaColision**()

Metodo *checaColision*

En este metodo se manejan todo tipo de colisiones involucrando personajes.

### getTileCollision

public java.awt.Point **getTileCollision**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) sprite,

float newX,

float newY)

### gameKeyPressed

public void **gameKeyPressed**(int keyCode)

Metodo *gameKeyPressed*

Metodo usado para manejar el evento que ocurre al presionar una tecla en cualquier panel y dependiendo del estado.

**Parameters:**

keyCode - es el evento que se genera en al presionar las teclas.

### gameKeyTyped

public void **gameKeyTyped**(java.awt.event.KeyEvent e)

Metodo *gameKeyTyped*

Metodo usado para manejar el evento que ocurre al hacer click una tecla en cualquier panel y dependiendo del estado.

**Parameters:**

e - es el evento que se genera al hacer click en una tecla.

### gameKeyReleased

public void **gameKeyReleased**(java.awt.event.KeyEvent e)

Metodo *gameKeyReleased*

En este metodo maneja el evento que se genera al soltar la tecla presionada en cualquier panel y dependiendo del estado.

**Parameters:**

e - es el evento que se genera en al soltar las teclas.

### keyPressed

public void **keyPressed**(java.awt.event.KeyEvent e)

Metodo *keyPressed* sobrescrito de la interface KeyListener.

En este metodo maneja el evento que se genera al presionar cualquier la tecla.

**Parameters:**

e - es el evento generado al presionar las teclas.

### keyReleased

public void **keyReleased**(java.awt.event.KeyEvent e)

Metodo *keyReleased* sobrescrito de la interface KeyListener.

En este metodo maneja el evento que se genera al soltar la tecla presionada.

**Parameters:**

e - es el evento que se genera en al soltar las teclas.

### keyTyped

public void **keyTyped**(java.awt.event.KeyEvent e)

Metodo *keyTyped* sobrescrito de la interface KeyListener.

En este metodo maneja el evento que se genera al presionar una tecla que no es de accion.

**Parameters:**

e - es el evento que se genera en al presionar las teclas.

### main

public static void **main**(java.lang.String[] args)

throws java.io.IOException

**Throws:**

java.io.IOException

### leeArchivo

public void **leeArchivo**()

throws java.io.IOException

Metodo *leeArchivo* que procesa eventos de KeyReleased que lee la informacion de un archivo y lo agrega a un vector.

**Throws:**

java.io.IOException

### grabaArchivo

public void **grabaArchivo**()

throws java.io.IOException

Metodo *grabaArchivo* que agrega la informacion del vector al archivo.

**Throws:**

java.io.IOException

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\GameMain.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?GameMain.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: [NESTED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#nested_classes_inherited_from_class_java.awt.Component) | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Player.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Player.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#fields_inherited_from_class_Creature) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#method_detail) |

## Class Player

java.lang.Object

extended by [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

extended by [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

extended by **Player**

public class **Player**

extends [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

La clase Jugador se encargara de manejar las animaciones del jugador, validar si ha collisionado, las vidas del personaje y la velocidad con la que se desplazara el jugador.

|  |
| --- |
| **Field Summary** |

|  |
| --- |
| **Fields inherited from class** [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html) |
| [STATE\_DEAD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DEAD), [STATE\_DYING](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_DYING), [STATE\_NORMAL](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#STATE_NORMAL) |

|  |
| --- |
| **Fields inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [CHOQUE\_CON\_PARED](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED), [DERECHA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO), [ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA), [IZQUIERDA\_ATACANDO](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO), [NO\_ESTA\_CERCA](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA), [PROXIMA\_CAIDA\_LIBRE](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Player**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#Player%28int,%20int,%20java.awt.Image%29)(int posX, int posY, java.awt.Image imagen)            Metodo constructor de Player usado para crear el objeto |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**cambiaVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#cambiaVida%28int%29)(int cuanto)            Metodo *cambiaVida()* |
| void | [**cargaImagenes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#cargaImagenes%28%29)()            Metodo *cargaImagenes()* |
| void | [**collideHorizontal**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#collideHorizontal%28%29)()            Metodo *collideHorizontal* si la creatura choca con alguna pared este metodo se encarga de voltearlo. |
| void | [**collideVertical**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#collideVertical%28%29)()            Metodo *collideVertical* que frena verticalmente a Creature. |
| void | [**disparo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#disparo%28%29)()            Metodo *disparo()* |
| boolean | [**estaAtacando**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#estaAtacando%28%29)() |
| boolean | [**estaSaltado**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#estaSaltado%28%29)() |
| float | [**getGravity**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#getGravity%28%29)() |
| boolean | [**getLado**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#getLado%28%29)()            Metodo *getLado()* |
| float | [**getMaxSpeed**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#getMaxSpeed%28%29)()            Metodo *getMaxSpeed()* |
| int | [**getMaxVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#getMaxVida%28%29)()            Metodo *getMaxVida()* |
| int | [**getVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#getVida%28%29)()            Metodo *getVida()* |
| void | [**jump**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#jump%28boolean%29)(boolean forceJump)            Metodo *jump()* |
| void | [**setGravity**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#setGravity%28float%29)(float grav) |
| void | [**setPosY**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#setPosY%28float%29)(float y)            Metodo *setPosY* que establece la posicion en Y del Sprite. |
| void | [**updateAnim**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#updateAnim%28long%29)(long elapsedTime)            Metodo *updateAnim()* |

|  |
| --- |
| **Methods inherited from class** [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html) |
| [getState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#getState%28%29), [isAlive](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#isAlive%28%29), [isFlying](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#isFlying%28%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#setState%28int%29), [update](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#update%28long%29), [wakeUp](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#wakeUp%28%29) |

|  |
| --- |
| **Methods inherited from class** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) |
| [actualiza](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#actualiza%28long%29), [atacar](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#atacar%28%29), [checaProximidad](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#checaProximidad%28Sprite%29), [clone](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#clone%28%29), [dibujaBala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#dibujaBala%28boolean,%20TileMap%29), [getAlto](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAlto%28%29), [getAncho](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAncho%28%29), [getCurrState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getCurrState%28%29), [getHeight](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getHeight%28%29), [getImageFromAnimation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageFromAnimation%28%29), [getImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageIcon%28%29), [getImagenI](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImagenI%28%29), [getPerimetro](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPerimetro%28%29), [getPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosX%28%29), [getPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosY%28%29), [getVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadX%28%29), [getVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadY%28%29), [getWidth](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getWidth%28%29), [iniciaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#iniciaVida%28%29), [intersecta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersecta%28Sprite%29), [intersectaAbajo](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersectaAbajo%28Sprite%29), [puedeMorir](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#puedeMorir%28%29), [revisarCambioDeEstado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeEstado%28java.lang.String,%20float%29), [revisarCambioDeVuelta](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeVuelta%28TileMap,%20float,%20float,%20float%29), [setImageIcon](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setImageIcon%28java.awt.Image%29), [setLado](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setLado%28int%29), [setPosX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosX%28float%29), [setState](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setState%28java.lang.String%29), [setVelocidadX](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadX%28float%29), [setVelocidadY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadY%28float%29), [setVidaMax](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVidaMax%28int%29), [volteaADerecha](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#volteaADerecha%28%29) |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Player

public **Player**(int posX,

int posY,

java.awt.Image imagen)

Metodo constructor de Player usado para crear el objeto

**Parameters:**

posX - es la posicion en x del objeto.

posY - es la posicion en y del objeto.

image - es la imagen del objeto.

|  |
| --- |
| **Method Detail** |

### setGravity

public void **setGravity**(float grav)

### getGravity

public float **getGravity**()

### estaSaltado

public boolean **estaSaltado**()

### estaAtacando

public boolean **estaAtacando**()

### collideHorizontal

public void **collideHorizontal**()

**Description copied from class:** [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideHorizontal%28%29)

Metodo *collideHorizontal* si la creatura choca con alguna pared este metodo se encarga de voltearlo.

**Overrides:**

[collideHorizontal](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideHorizontal%28%29) in class [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

### collideVertical

public void **collideVertical**()

**Description copied from class:** [**Creature**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideVertical%28%29)

Metodo *collideVertical* que frena verticalmente a Creature.

Si choca con un techo la velocidad en Y se vuelve cero y empieza su caida libre.

**Overrides:**

[collideVertical](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#collideVertical%28%29) in class [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

### setPosY

public void **setPosY**(float y)

**Description copied from class:** [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29)

Metodo *setPosY* que establece la posicion en Y del Sprite.

**Overrides:**

[setPosY](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

### jump

public void **jump**(boolean forceJump)

Metodo *jump()*

Si el personaje esta en el piso o se le obliga a saltar, su velocidad en Y es JUMP\_SPEED. Si esta en una pared, se le permite saltar de nuevo

### updateAnim

public void **updateAnim**(long elapsedTime)

Metodo *updateAnim()*

cambia la animacion del Player. Si ha pasado cierto tiempo desde que comenzo la animacion, se cambia de animacion y se cambia el Estado.

**Overrides:**

[updateAnim](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

### getMaxSpeed

public float **getMaxSpeed**()

Metodo *getMaxSpeed()*

Devuelve la Maxima velocidad posible del Player

**Overrides:**

[getMaxSpeed](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html#getMaxSpeed%28%29) in class [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html)

**Returns:**

0, esto significa que el movimiento no tendra ningun tipo de restriccion al momento de moverse

### disparo

public void **disparo**()

Metodo *disparo()*

Cambia el estado de Ataque a ATK

### cambiaVida

public void **cambiaVida**(int cuanto)

Metodo *cambiaVida()*

Reduce o aumenta la vida del jugador.

**Overrides:**

[cambiaVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#cambiaVida%28int%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

**Parameters:**

int - cuanto es la cantida de vida a modificar.

### getVida

public int **getVida**()

Metodo *getVida()*

Devuelve la vida actual del jugador

**Overrides:**

[getVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVida%28%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

**Returns:**

int, el numero de vidas del objeto

### getMaxVida

public int **getMaxVida**()

Metodo *getMaxVida()*

Regresa la vida Maxima posible del jugador

**Overrides:**

[getMaxVida](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getMaxVida%28%29) in class [Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)

**Returns:**

int, el numero de vidas del objeto

### cargaImagenes

public void **cargaImagenes**()

Metodo *cargaImagenes()*

Carga todas las imagenes que se usaran por el Player

### getLado

public boolean **getLado**()

Metodo *getLado()*

Devuelve true si esta volteando a la izquierda, false si voltea a la derecha

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Player.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\GameMain.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Player.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#fields_inherited_from_class_Creature) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Puntaje.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Puntaje.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#method_detail) |

## Class Puntaje

java.lang.Object

extended by **Puntaje**

**All Implemented Interfaces:**

java.lang.Comparable<[Puntaje](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)>

public class **Puntaje**

extends java.lang.Object

implements java.lang.Comparable<[Puntaje](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)>

Clase Puntaje, es la clase encargada de contabilizar los puntos logrados cada vez que se destruye un enemigo, ademas que es la que crea el archivo de texto y lo maneja,

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Puntaje**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#Puntaje%28%29)()            Constructor vacio con darle valores iniciales al momento de crear el objeto Puntaje |  |
| [**Puntaje**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#Puntaje%28java.lang.String,%20int%29)(java.lang.String nombre, int puntaje)            Metodo constructor usado para crear el objeto |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| int | [**compareTo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#compareTo%28Puntaje%29)([Puntaje](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html) o)            Metodo que comapra dos puntajes, para ordenar el vector de Puntajes |
| java.lang.String | [**getNombre**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#getNombre%28%29)()            Metodo de acceso que regresa el nombre del objeto |
| int | [**getPuntaje**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#getPuntaje%28%29)()            Metodo de acceso que regresa el puntaje del objeto |
| void | [**setNombre**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#setNombre%28java.lang.String%29)(java.lang.String nombre)            Metodo modificador usado para cambiar el nombre del objeto. |
| void | [**setPuntaje**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#setPuntaje%28int%29)(int puntaje)            Metodo modificador usado para cambiar el puntaje del objeto |
| java.lang.String | [**toString**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#toString%28%29)()            Metodo que regresa el objeto en formato String |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### Puntaje

public **Puntaje**()

Constructor vacio con darle valores iniciales al momento de crear el objeto Puntaje

### Puntaje

public **Puntaje**(java.lang.String nombre,

int puntaje)

Metodo constructor usado para crear el objeto

**Parameters:**

nombre - es el nombre del objeto.

puntaje - es el puntaje del objeto.

|  |
| --- |
| **Method Detail** |

### setNombre

public void **setNombre**(java.lang.String nombre)

Metodo modificador usado para cambiar el nombre del objeto.

**Parameters:**

nombre - es el nombre del objeto.

### getNombre

public java.lang.String **getNombre**()

Metodo de acceso que regresa el nombre del objeto

**Returns:**

nombre es el nombre del objeto.

### setPuntaje

public void **setPuntaje**(int puntaje)

Metodo modificador usado para cambiar el puntaje del objeto

**Parameters:**

puntaje - es el puntaje del objeto.

### getPuntaje

public int **getPuntaje**()

Metodo de acceso que regresa el puntaje del objeto

**Returns:**

puntaje es el puntaje del objeto.

### toString

public java.lang.String **toString**()

Metodo que regresa el objeto en formato String

**Overrides:**

toString in class java.lang.Object

**Returns:**

un objeto de la clase String.

### compareTo

public int **compareTo**([Puntaje](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html) o)

Metodo que comapra dos puntajes, para ordenar el vector de Puntajes

**Specified by:**

compareTo in interface java.lang.Comparable<[Puntaje](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)>

**Returns:**

un entero.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Puntaje.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Puntaje.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\SoundClip.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?SoundClip.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#method_detail) |

## Class SoundClip

java.lang.Object

extended by **SoundClip**

public class **SoundClip**

extends java.lang.Object

Clase SoundClip, es el metodo encargado de exportar los archivos tipo wav y usarlos dentro del juego, esto permite que haya musica de fondo, efectos de sonido

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**SoundClip**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#SoundClip%28%29)()            Constructor default |  |
| [**SoundClip**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#SoundClip%28java.lang.String%29)(java.lang.String filename)            Constructor con parametros, que carga manda llamar a load esto carga el archivo de sonido. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| javax.sound.sampled.Clip | [**getClip**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#getClip%28%29)()            Metodo de acceso que regresa un objeto de tipo Clip |
| java.lang.String | [**getFilename**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#getFilename%28%29)()            Metodo de acceso que regresa el nombre del archivo. |
| boolean | [**getLooping**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#getLooping%28%29)()            Metodo de acceso que regresa un booleano para ver si hay repeticion. |
| int | [**getRepeat**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#getRepeat%28%29)()            Metodo de acceso que regresa el numero de repeticiones. |
| boolean | [**isLoaded**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#isLoaded%28%29)()            Metodo que verifica si el archivo de audio esta cargado. |
| boolean | [**load**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#load%28java.lang.String%29)(java.lang.String audiofile)            Metodo que carga el archivo de sonido. |
| void | [**play**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#play%28%29)()            Metodo que reproduce el sonido. |
| void | [**setFilename**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#setFilename%28java.lang.String%29)(java.lang.String filename)            Metodo modificador que asigna un nombre al archivo. |
| void | [**setLooping**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#setLooping%28boolean%29)(boolean looping)            Metodo modificador usado para modificar si el sonido se repite. |
| void | [**setRepeat**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#setRepeat%28int%29)(int repeat)            Metodo modificador usado para definir el numero de repeticiones. |
| void | [**stop**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#stop%28%29)()            Metodo que detiene el sonido. |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### SoundClip

public **SoundClip**()

Constructor default

### SoundClip

public **SoundClip**(java.lang.String filename)

Constructor con parametros, que carga manda llamar a load esto carga el archivo de sonido.

**Parameters:**

filename - es el String del archivo.

|  |
| --- |
| **Method Detail** |

### getClip

public javax.sound.sampled.Clip **getClip**()

Metodo de acceso que regresa un objeto de tipo Clip

**Returns:**

clip es un objeto Clip.

### setLooping

public void **setLooping**(boolean looping)

Metodo modificador usado para modificar si el sonido se repite.

**Parameters:**

\_looping - es un valor boleano.

### getLooping

public boolean **getLooping**()

Metodo de acceso que regresa un booleano para ver si hay repeticion.

**Returns:**

looping es un valor boleano.

### setRepeat

public void **setRepeat**(int repeat)

Metodo modificador usado para definir el numero de repeticiones.

**Parameters:**

\_repeat - es un entero que es el numero de repeticiones.

### getRepeat

public int **getRepeat**()

Metodo de acceso que regresa el numero de repeticiones.

**Returns:**

repeat es un valor entero con el numero de repeticiones.

### setFilename

public void **setFilename**(java.lang.String filename)

Metodo modificador que asigna un nombre al archivo.

**Parameters:**

\_filename - es un String con el nombre del archivo.

### getFilename

public java.lang.String **getFilename**()

Metodo de acceso que regresa el nombre del archivo.

**Returns:**

filename es un String con el nombre del archivo.

### isLoaded

public boolean **isLoaded**()

Metodo que verifica si el archivo de audio esta cargado.

**Returns:**

sample es un objeto sample.

### load

public boolean **load**(java.lang.String audiofile)

Metodo que carga el archivo de sonido.

**Parameters:**

audiofile - es un String con el nombre del archivo de sonido.

### play

public void **play**()

Metodo que reproduce el sonido.

### stop

public void **stop**()

Metodo que detiene el sonido.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\SoundClip.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Puntaje.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?SoundClip.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Sprite.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\SoundClip.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?Sprite.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#method_detail) |

## Class Sprite

java.lang.Object

extended by **Sprite**

**Direct Known Subclasses:**

[Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html), [Creature](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Creature.html), [Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)

public class **Sprite**

extends java.lang.Object

Clase Sprite, es la clase encargada de darle los aspectos fisico (caida libres), la posicion (movimiento) y la posibilidad de atacar a los enemigos y jugador

|  |  |
| --- | --- |
| **Field Summary** | |
| static java.lang.String | [**CHOQUE\_CON\_PARED**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#CHOQUE_CON_PARED) |
| static java.lang.String | [**DERECHA\_ATACANDO**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#DERECHA_ATACANDO) |
| static java.lang.String | [**ESTA\_CERCA**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#ESTA_CERCA) |
| static java.lang.String | [**IZQUIERDA\_ATACANDO**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#IZQUIERDA_ATACANDO) |
| static java.lang.String | [**NO\_ESTA\_CERCA**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#NO_ESTA_CERCA) |
| static java.lang.String | [**PROXIMA\_CAIDA\_LIBRE**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#PROXIMA_CAIDA_LIBRE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#Sprite%28javax.swing.ImageIcon%29)(javax.swing.ImageIcon image)            Metodo Sprite se encarga de asignar una imagen al objeto Sprite |  |
| [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#Sprite%28int,%20int,%20Animation%29)(int posX, int posY, [Animation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html) anim)            Constructor Sprite para crear un objeto de tipo Sprite. |  |
| [**Sprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#Sprite%28int,%20int,%20java.awt.Image%29)(int posX, int posY, java.awt.Image image)            Constructor Sprite para crear un objeto de tipo Sprite. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actualiza**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#actualiza%28long%29)(long tiempo)            Metodo actualiza se encarga de calcular la caida libre del Sprite |
| boolean | [**atacar**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#atacar%28%29)()            Metodo atacar encargado de hacer que los enemigos ataquen. |
| void | [**cambiaVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#cambiaVida%28int%29)(int cant)            Metodo cambiaVida que modifica las vidas del Sprite |
| void | [**checaProximidad**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#checaProximidad%28Sprite%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) otroSprite)            Metodo checaProximidad para determinar si hay un objeto cerca del Sprie Si hay un objeto cercano, el Sprite cambia de estado. |
| java.lang.Object | [**clone**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#clone%28%29)()            Metodo clone, funciona para duplicar Sprites Sirve para crear multiples Sprites con la misma imagen |
| [Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) | [**dibujaBala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#dibujaBala%28boolean,%20TileMap%29)(boolean derecha, [TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) mapa)            Metodo bala encargado de desplegar las balas de los enemigos |
| int | [**getAlto**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAlto%28%29)()            Metodo *getAlto* que devuelve el alto en pixeles del Sprite. |
| int | [**getAncho**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getAncho%28%29)()            Metodo *getAncho* que devuelve el ancho en pixeles del Sprite. |
| java.lang.String | [**getCurrState**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getCurrState%28%29)()            Metodo *getCurrState* que regresa el Estado del Sprite. |
| int | [**getHeight**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getHeight%28%29)()            Metodo *getHeight* que regresa el alto de la imagen. |
| java.awt.Image | [**getImageFromAnimation**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageFromAnimation%28%29)()            Metodo getImageFromAnimation consigue la imagen actual de la animacion |
| javax.swing.ImageIcon | [**getImageIcon**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImageIcon%28%29)()            Metodo ImageIcon de acceso que regresa el icono del objeto |
| java.awt.Image | [**getImagenI**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getImagenI%28%29)()            Metodo Image da acceso que regresa la imagen del icono |
| int | [**getMaxVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getMaxVida%28%29)()            Metodo getMaxVida regresa las vidas del Sprite |
| java.awt.Rectangle | [**getPerimetro**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPerimetro%28%29)()            Metodo Rectangle de acceso que regresa un nuevo rectangulo |
| float | [**getPosX**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosX%28%29)()            Metodo *getPosX* que devuelve la posicion en X del Sprite. |
| float | [**getPosY**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getPosY%28%29)()            Metodo *getPosY* que devuelve la posicion en Y del Sprite. |
| float | [**getVelocidadX**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadX%28%29)()            Metodo *getVelocidadX* que devuelve la velocidad en X del Sprite. |
| float | [**getVelocidadY**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVelocidadY%28%29)()            Metodo *getVelocidadY* que devuelve la velocidad en Y del Sprite. |
| int | [**getVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getVida%28%29)()            Metodo getVida regresa las vidas del Sprite |
| int | [**getWidth**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#getWidth%28%29)()            Metodo *getWidth* que regresa el ancho de la imagen. |
| void | [**iniciaVida**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#iniciaVida%28%29)()            Metodo iniciaVida inicializa las vidas del Sprite con la maxima posible |
| boolean | [**intersecta**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersecta%28Sprite%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) obj)            Checa si el objeto Objetos intersecta a otro Objetos |
| boolean | [**intersectaAbajo**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#intersectaAbajo%28Sprite%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) obj)            Checa si el objeto Objetos intersecta a otro Objetos por debajo |
| boolean | [**puedeMorir**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#puedeMorir%28%29)()            Metodo puedoMorir para determinar si un objeto puede ser destruido. |
| void | [**revisarCambioDeEstado**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeEstado%28java.lang.String,%20float%29)(java.lang.String identificador, float newX)            Metodo *revisarCambioDeEstado* |
| void | [**revisarCambioDeVuelta**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#revisarCambioDeVuelta%28TileMap,%20float,%20float,%20float%29)([TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) mapa, float posX, float posY, float newX)            Metodo revisarcambioDeVueltapara revisar si el enemigo llega a una esquina. |
| void | [**setImageIcon**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setImageIcon%28java.awt.Image%29)(java.awt.Image icono)            Metodo setImageIcon modificador usado para cambiar el icono del objeto |
| void | [**setLado**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setLado%28int%29)(int lado)            Metodo setLado para determinar el lado hacia donde apuntara el Sprite |
| void | [**setPosX**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosX%28float%29)(float posX)            Metodo *setPosX* que establece la posicion en X del Sprite. |
| void | [**setPosY**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setPosY%28float%29)(float posY)            Metodo *setPosY* que establece la posicion en Y del Sprite. |
| void | [**setState**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setState%28java.lang.String%29)(java.lang.String newState)            Metodo *setState* que modifica el Estado del Sprite. |
| void | [**setVelocidadX**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadX%28float%29)(float dx)            Metodo *setVelocidadX* que cambia la velocidad en X del Sprite. |
| void | [**setVelocidadY**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVelocidadY%28float%29)(float dy)            Metodo *setVelocidadY* que cambia la velocidad en Y del Sprite. |
| void | [**setVidaMax**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#setVidaMax%28int%29)(int vidaMax)            Metodo setVidaMax define las vidas maximas del Sprite |
| void | [**updateAnim**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#updateAnim%28long%29)(long elapsedTime)            Metodo *updateAnim* que actualiza la animacion del Sprite. |
| boolean | [**volteaADerecha**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html#volteaADerecha%28%29)()            Metodo volteaADerecha para determinar el lado hacia donde debe voltear el Sprite |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Field Detail** |

### PROXIMA\_CAIDA\_LIBRE

public static final java.lang.String **PROXIMA\_CAIDA\_LIBRE**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Sprite.PROXIMA_CAIDA_LIBRE)

### CHOQUE\_CON\_PARED

public static final java.lang.String **CHOQUE\_CON\_PARED**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Sprite.CHOQUE_CON_PARED)

### ESTA\_CERCA

public static final java.lang.String **ESTA\_CERCA**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Sprite.ESTA_CERCA)

### DERECHA\_ATACANDO

public static final java.lang.String **DERECHA\_ATACANDO**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Sprite.DERECHA_ATACANDO)

### IZQUIERDA\_ATACANDO

public static final java.lang.String **IZQUIERDA\_ATACANDO**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Sprite.IZQUIERDA_ATACANDO)

### NO\_ESTA\_CERCA

public static final java.lang.String **NO\_ESTA\_CERCA**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#Sprite.NO_ESTA_CERCA)

|  |
| --- |
| **Constructor Detail** |

### Sprite

public **Sprite**(int posX,

int posY,

java.awt.Image image)

Constructor Sprite para crear un objeto de tipo Sprite.

**Parameters:**

posX - posicion en el eje X del Sprite

posY - posicion en el eje Y del Sprite

image - la imagen del objeto que funcionara para reconocer si es el jugador o un enemigo

### Sprite

public **Sprite**(int posX,

int posY,

[Animation](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Animation.html) anim)

Constructor Sprite para crear un objeto de tipo Sprite.

**Parameters:**

posX - posicion en el eje X del Sprite

posY - posicion en el eje Y del Sprite

Animacion - el conjunto de imagenes del objeto que funcionara para reconocer si es el jugador o un enemigo

### Sprite

public **Sprite**(javax.swing.ImageIcon image)

Metodo Sprite se encarga de asignar una imagen al objeto Sprite

**Parameters:**

image, - la imagen a desplegar en el Sprite

|  |
| --- |
| **Method Detail** |

### actualiza

public void **actualiza**(long tiempo)

Metodo actualiza se encarga de calcular la caida libre del Sprite

**Parameters:**

tiempo, - tiempo usado para calcular la caida del Sprite

### atacar

public boolean **atacar**()

Metodo atacar encargado de hacer que los enemigos ataquen. Se usa en conjunto con otro metodo para saber si debe atacar

**Returns:**

true, si el enemigo no estaba atacando lo forza a atacar

### dibujaBala

public [Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) **dibujaBala**(boolean derecha,

[TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) mapa)

Metodo bala encargado de desplegar las balas de los enemigos

**Parameters:**

derecha, - determina a que lado el Sprite de la bala se vera

mapa, - si el enemigo atacaba y el jugador no esta cerca

**Returns:**

null, al terminar de agregar la bala la saca del TileMap

### checaProximidad

public void **checaProximidad**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) otroSprite)

Metodo checaProximidad para determinar si hay un objeto cerca del Sprie Si hay un objeto cercano, el Sprite cambia de estado.

**Parameters:**

otroSprite, - sera el Sprite a comparar si esta cerca o no

### puedeMorir

public boolean **puedeMorir**()

Metodo puedoMorir para determinar si un objeto puede ser destruido.

**Returns:**

true, si el objeto puede ser destruido

### setLado

public void **setLado**(int lado)

Metodo setLado para determinar el lado hacia donde apuntara el Sprite

**Parameters:**

lado, - entero que indica hacia donde apuntara el Sprite 1 izquierda, -1 derecha

### volteaADerecha

public boolean **volteaADerecha**()

Metodo volteaADerecha para determinar el lado hacia donde debe voltear el Sprite

**Returns:**

true, si esta orientado a la izquierda = 1,

### revisarCambioDeVuelta

public void **revisarCambioDeVuelta**([TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) mapa,

float posX,

float posY,

float newX)

Metodo revisarcambioDeVueltapara revisar si el enemigo llega a una esquina. Si llega a una esquina, el Sprite gira y cambia de direccion.

**Parameters:**

mapa - es el TileMap.

posX - es el float de la posicion en X del Sprite.

posY - es el float de la posicion en Y del Sprite.

newX - es el float de la futura posicion en X del Sprite.

### getPosX

public float **getPosX**()

Metodo *getPosX* que devuelve la posicion en X del Sprite.

### getPosY

public float **getPosY**()

Metodo *getPosY* que devuelve la posicion en Y del Sprite.

### setPosX

public void **setPosX**(float posX)

Metodo *setPosX* que establece la posicion en X del Sprite.

**Parameters:**

posX, - la nueva posicion para el sprite

### setPosY

public void **setPosY**(float posY)

Metodo *setPosY* que establece la posicion en Y del Sprite.

**Parameters:**

posY, - la nueva posicion para el sprite

### getAncho

public int **getAncho**()

Metodo *getAncho* que devuelve el ancho en pixeles del Sprite.

### getAlto

public int **getAlto**()

Metodo *getAlto* que devuelve el alto en pixeles del Sprite.

### getVelocidadX

public float **getVelocidadX**()

Metodo *getVelocidadX* que devuelve la velocidad en X del Sprite.

### getVelocidadY

public float **getVelocidadY**()

Metodo *getVelocidadY* que devuelve la velocidad en Y del Sprite.

### setVelocidadX

public void **setVelocidadX**(float dx)

Metodo *setVelocidadX* que cambia la velocidad en X del Sprite.

**Parameters:**

dx, - float que cambia la velocidad de avance en el eje X

### setVelocidadY

public void **setVelocidadY**(float dy)

Metodo *setVelocidadY* que cambia la velocidad en Y del Sprite.

**Parameters:**

dy, - float que cambia la velocidad de avance en el eje Y

### setImageIcon

public void **setImageIcon**(java.awt.Image icono)

Metodo setImageIcon modificador usado para cambiar el icono del objeto

**Parameters:**

icono - es el icono del objeto.

### getImageIcon

public javax.swing.ImageIcon **getImageIcon**()

Metodo ImageIcon de acceso que regresa el icono del objeto

**Returns:**

ImageIcon es el ImageIcon del objeto.

### getImagenI

public java.awt.Image **getImagenI**()

Metodo Image da acceso que regresa la imagen del icono

**Returns:**

un objeto de la clase Image que es la imagen del icono.

### getPerimetro

public java.awt.Rectangle **getPerimetro**()

Metodo Rectangle de acceso que regresa un nuevo rectangulo

**Returns:**

un objeto de la clase Rectangle que es el perimetro del rectangulo

### intersecta

public boolean **intersecta**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) obj)

Checa si el objeto Objetos intersecta a otro Objetos

**Parameters:**

Sprite, - el objeto a comparar si hubo una colision.

**Returns:**

un valor booleano true si lo intersecta false en caso contrario

### intersectaAbajo

public boolean **intersectaAbajo**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) obj)

Checa si el objeto Objetos intersecta a otro Objetos por debajo

**Parameters:**

Sprite, - el objeto a comparar si hubo una colision.

**Returns:**

un valor booleano true si lo intersecta false en caso contrario

### getImageFromAnimation

public java.awt.Image **getImageFromAnimation**()

Metodo getImageFromAnimation consigue la imagen actual de la animacion

**Returns:**

Image regresa la imagen

### getWidth

public int **getWidth**()

Metodo *getWidth* que regresa el ancho de la imagen.

**Returns:**

int, que sera el valor del ancho de la imagen

### getHeight

public int **getHeight**()

Metodo *getHeight* que regresa el alto de la imagen.

**Returns:**

int, que sera el valor del ancho de la imagen

### clone

public java.lang.Object **clone**()

Metodo clone, funciona para duplicar Sprites Sirve para crear multiples Sprites con la misma imagen

**Overrides:**

clone in class java.lang.Object

**Returns:**

un nuevo Sprite con una imagen

### updateAnim

public void **updateAnim**(long elapsedTime)

Metodo *updateAnim* que actualiza la animacion del Sprite.

**Parameters:**

elapsedTime, - servira para saber el tiempo de desplegado de la animacion

### revisarCambioDeEstado

public void **revisarCambioDeEstado**(java.lang.String identificador,

float newX)

Metodo *revisarCambioDeEstado*

### getCurrState

public java.lang.String **getCurrState**()

Metodo *getCurrState* que regresa el Estado del Sprite.

**Returns:**

el estado actual del objeto

### setState

public void **setState**(java.lang.String newState)

Metodo *setState* que modifica el Estado del Sprite.

**Parameters:**

newState, - cambia el estado del Sprite

### cambiaVida

public void **cambiaVida**(int cant)

Metodo cambiaVida que modifica las vidas del Sprite

**Parameters:**

cant, - cambia el estado del Sprite

### getVida

public int **getVida**()

Metodo getVida regresa las vidas del Sprite

**Returns:**

int, el numero de vidas del objeto

### setVidaMax

public void **setVidaMax**(int vidaMax)

Metodo setVidaMax define las vidas maximas del Sprite

**Parameters:**

vidaMax, - el numero de vidas maximas.

### iniciaVida

public void **iniciaVida**()

Metodo iniciaVida inicializa las vidas del Sprite con la maxima posible

### getMaxVida

public int **getMaxVida**()

Metodo getMaxVida regresa las vidas del Sprite

**Returns:**

int, el numero de vidas del objeto

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\Sprite.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\TileMap.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?TileMap.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#method_detail) |

## Class TileMap

java.lang.Object

extended by **TileMap**

public class **TileMap**

extends java.lang.Object

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**TileMap**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#TileMap%28int,%20int%29)(int width, int height)            Constructor *TileMap* para crear un objeto de tipo TileMap. |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**addBala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#addBala%28Bala%29)([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)            Metodo *addBala()* |
| void | [**addBalaEnemigos**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#addBalaEnemigos%28Bala%29)([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)            Metodo *addBalaenemigos()* |
| void | [**addExplosion**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#addExplosion%28Explosiones%29)([Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html) exp)            Metodo *addExplosion()* |
| void | [**addSprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#addSprite%28Sprite%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) sprite)            Metodo *addSprite()* |
| void | [**cargaTiles**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#cargaTiles%28%29)()            Metodo *cargaTiles()* |
| java.util.Vector<[Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)> | [**getBalas**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getBalas%28%29)()            Metodo *getBalas()* |
| java.util.Vector<[Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)> | [**getBalasEnemigos**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getBalasEnemigos%28%29)()            Metodo *getBalasJefs()* |
| java.util.Vector<[Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)> | [**getExplosiones**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getExplosiones%28%29)()            Metodo *getExplosiones()* |
| int | [**getHeight**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getHeight%28%29)()            Metodo *getHeight()* |
| [Player](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html) | [**getPlayer**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getPlayer%28%29)()            Metodo *getPlayer()* |
| int | [**getPuntos**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getPuntos%28%29)() |
| java.util.Vector<[Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)> | [**getSprites**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getSprites%28%29)()            Metodo *getSprites()* |
| java.awt.Image | [**getTile**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getTile%28int,%20int%29)(int x, int y)            Metodo *getTile()* |
| int | [**getWidth**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#getWidth%28%29)()            Metodo *getWidth()* |
| java.awt.Image | [**loadImage**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#loadImage%28java.lang.String%29)(java.lang.String name)            Carga las imagenes |
| [TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) | [**loadMap**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#loadMap%28int%29)(int nivel)            Metodo *loadMap()* |
| void | [**removeBala**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#removeBala%28Bala%29)([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)            Metodo *removeBala()* |
| void | [**removeBalaEnemigos**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#removeBalaEnemigos%28Bala%29)([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)            Metodo *removeBalaEnemigos()* |
| void | [**removeExplosion**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#removeExplosion%28Explosiones%29)([Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html) exp)            Metodo *removeExplosion()* |
| void | [**removeSprite**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#removeSprite%28Sprite%29)([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) sprite)            Metodo *removeSprite()* |
| void | [**setPlayer**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#setPlayer%28Player%29)([Player](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html) player)            Metodo *setPlayer()* |
| void | [**setPuntos**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#setPuntos%28int%29)(int P) |
| void | [**setTile**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#setTile%28int,%20int,%20java.awt.Image%29)(int x, int y, java.awt.Image tile)            Metodo *setTile()* |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Constructor Detail** |

### TileMap

public **TileMap**(int width,

int height)

Constructor *TileMap* para crear un objeto de tipo TileMap.

|  |
| --- |
| **Method Detail** |

### getWidth

public int **getWidth**()

Metodo *getWidth()*

Devuelve el numero de Tiles a lo ancho.

### getPuntos

public int **getPuntos**()

### setPuntos

public void **setPuntos**(int P)

### getHeight

public int **getHeight**()

Metodo *getHeight()*

Devuelve el numero de Tiles a lo largo.

### addExplosion

public void **addExplosion**([Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html) exp)

Metodo *addExplosion()*

Agrega un objeto Explosion al vector PowerUps.

**Parameters:**

Explosion - exp es el objeto Explosion a agregar.

### removeExplosion

public void **removeExplosion**([Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html) exp)

Metodo *removeExplosion()*

Quita un objeto Explosion al vector explosions.

**Parameters:**

Explosiones - exp es el objeto Explosion a quitar.

### getExplosiones

public java.util.Vector<[Explosiones](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Explosiones.html)> **getExplosiones**()

Metodo *getExplosiones()*

regresa el vector Explosions.

### getTile

public java.awt.Image **getTile**(int x,

int y)

Metodo *getTile()*

Devuelve la Imagen que corresponde al Tile en cierta posicion

**Parameters:**

int - x es el entero de la posicion en X del Tile.

int - y es el entero de la posicion en Y del Tile.

### setTile

public void **setTile**(int x,

int y,

java.awt.Image tile)

Metodo *setTile()*

Coloca un Tile en la posicion especificada.

**Parameters:**

int - x es el entero de la posicion en X del Tile.

int - y es el entero de la posicion en Y del Tile.

Image - tile es el Image del tile.

### addBala

public void **addBala**([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)

Metodo *addBala()*

Agrega un objeto Bala al vector balas.

**Parameters:**

Bala - b es el objeto Bala a agregar.

### removeBala

public void **removeBala**([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)

Metodo *removeBala()*

Quita un objeto Bala al vector balas.

**Parameters:**

Bala - b es el objeto Bala a agregar.

### getBalas

public java.util.Vector<[Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)> **getBalas**()

Metodo *getBalas()*

regresa el vector balas.

### getPlayer

public [Player](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html) **getPlayer**()

Metodo *getPlayer()*

Regresa el objeto Player correspondiente al jugador

### setPlayer

public void **setPlayer**([Player](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Player.html) player)

Metodo *setPlayer()*

Asigna al mapa el objeto Player recibido como parametro

**Parameters:**

Player - player es el Player a asignar.

### addBalaEnemigos

public void **addBalaEnemigos**([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)

Metodo *addBalaenemigos()*

Agrega un objeto Bala al vector balas de Enemigos.

**Parameters:**

Bala - b es el objeto Bala a agregar.

### removeBalaEnemigos

public void **removeBalaEnemigos**([Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html) b)

Metodo *removeBalaEnemigos()*

Quita un objeto Bala al vector balas de enemigos.

**Parameters:**

Bala - b es el objeto Bala a agregar.

### getBalasEnemigos

public java.util.Vector<[Bala](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Bala.html)> **getBalasEnemigos**()

Metodo *getBalasJefs()*

regresa el vector balas de Enemigos.

### addSprite

public void **addSprite**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) sprite)

Metodo *addSprite()*

Agrega un objeto Sprite al vector sprites.

**Parameters:**

Sprite - sprite es el Sprite a agregar.

### removeSprite

public void **removeSprite**([Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html) sprite)

Metodo *removeSprite()*

Remueve un objeto Sprite del vector sprites.

**Parameters:**

Sprite - sprite es el Sprite a quitar.

### getSprites

public java.util.Vector<[Sprite](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)> **getSprites**()

Metodo *getSprites()*

Regresa un vector de los Sprites del mapa

### loadMap

public [TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) **loadMap**(int nivel)

throws java.io.IOException

Metodo *loadMap()*

Con base en un archivo de texto asigna tiles a una matriz Tambien se encarga de asignar la posicion de inicio de los enemigos y items en el mapa

**Parameters:**

nivel, - entero que indica el numero de mapa correspondiente al nivel

**Throws:**

java.io.IOException

### cargaTiles

public void **cargaTiles**()

Metodo *cargaTiles()*

Lee las imagenes a usarse por los tiles.

### loadImage

public java.awt.Image **loadImage**(java.lang.String name)

Carga las imagenes

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\TileMap.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\Sprite.html)   [**NEXT CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html) | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?TileMap.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#method_summary) | DETAIL: FIELD | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html#method_detail) |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\TileMapRenderer.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html)   NEXT CLASS | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?TileMapRenderer.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#method_detail) |

## Class TileMapRenderer

java.lang.Object

extended by **TileMapRenderer**

public class **TileMapRenderer**

extends java.lang.Object

|  |  |
| --- | --- |
| **Field Summary** | |
| static int | [**TILE\_SIZE**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#TILE_SIZE) |

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**TileMapRenderer**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#TileMapRenderer%28%29)() |  |

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**draw**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#draw%28java.awt.Graphics2D,%20TileMap,%20int,%20int%29)(java.awt.Graphics2D g, [TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) map, int screenWidth, int screenHeight)            Metodo *draw()* |
| java.awt.Image | [**getFlippedImage**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#getFlippedImage%28java.awt.Image,%20java.awt.Graphics%29)(java.awt.Image image, java.awt.Graphics g)            Metodo *getFlippedImage()* |
| static int | [**pixelsToTiles**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#pixelsToTiles%28float%29)(float pixels)            Metodo int *pixelsToTiles()* |
| static int | [**pixelsToTiles**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#pixelsToTiles%28int%29)(int pixels)            Metodo int *pixelsToTiles()* |
| void | [**setBackground**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#setBackground%28java.awt.Image%29)(java.awt.Image background)            Metodo *setBackground()* |
| static int | [**tilesToPixels**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#tilesToPixels%28int%29)(int numTiles)            Metodo int *tilesToPixels()* |

|  |
| --- |
| **Methods inherited from class java.lang.Object** |
| equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait |

|  |
| --- |
| **Field Detail** |

### TILE\_SIZE

public static final int **TILE\_SIZE**

**See Also:**

[Constant Field Values](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\constant-values.html#TileMapRenderer.TILE_SIZE)

|  |
| --- |
| **Constructor Detail** |

### TileMapRenderer

public **TileMapRenderer**()

|  |
| --- |
| **Method Detail** |

### pixelsToTiles

public static int **pixelsToTiles**(float pixels)

Metodo int *pixelsToTiles()*

Llama al metodo int pixelsToTiles con parametro Integer

**Parameters:**

pixels - es el float del pixel a convertir.

### pixelsToTiles

public static int **pixelsToTiles**(int pixels)

Metodo int *pixelsToTiles()*

Devuelve el numero de Tile al que corresponde el pixel

**Parameters:**

pixels - es el float del pixel a convertir.

### tilesToPixels

public static int **tilesToPixels**(int numTiles)

Metodo int *tilesToPixels()*

Devuelve el pixel al que corresponde el numero de Tile

**Parameters:**

numTiles - es el int del Tile a convertir.

### setBackground

public void **setBackground**(java.awt.Image background)

Metodo *setBackground()*

Cambia la imagen de fondo

**Parameters:**

background - es el Image de la imagen de fondo.

### draw

public void **draw**(java.awt.Graphics2D g,

[TileMap](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html) map,

int screenWidth,

int screenHeight)

Metodo *draw()*

Pinta el mapa, los Sprites, el Player, las Balas y los Items

**Parameters:**

g - es el Graphics2D objeto grafico.

map - es el TileMap.

screenWidth - es el int del ancho del mapa.

screenHeight - es el int del alto del mapa.

### getFlippedImage

public java.awt.Image **getFlippedImage**(java.awt.Image image,

java.awt.Graphics g)

Metodo *getFlippedImage()*

Gira la imagen horizontalmente

**Parameters:**

image - es el Image de la imagen a voltear.

g - es Graphics a pintar

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | [**Package**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-summary.html) | **Class** | [**Use**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\class-use\TileMapRenderer.html) | [**Tree**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\package-tree.html) | [**Deprecated**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\deprecated-list.html) | [**Index**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index-files\index-1.html) | [**Help**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\help-doc.html) | | |  |
| [**PREV CLASS**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMap.html)   NEXT CLASS | [**FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\index.html?TileMapRenderer.html)    [**NO FRAMES**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html)    [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) [**All Classes**](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#field_summary) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#constructor_summary) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#method_summary) | DETAIL: [FIELD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#field_detail) | [CONSTR](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#constructor_detail) | [METHOD](file:///C:\Users\samuel%20heaney\workspace\CopyBatalla\doc\TileMapRenderer.html#method_detail) |

**ADMINISTRACION DE PROYECTOS**

**Definición de Roles**

Los roles que fueron desempeñados por los miembros del equipo fueron los siguientes:

**Administración de sonido:** Jorge Saldívar

**Diseño grafico:** Jorge Saldívar

**Programador de objetos:** Samuel Heaney

**Programador de colisiones:** Carlos Elizondo

**Física que maneja el juego:** Carlos Elizondo

**Inteligencia artificial del juego:** Héctor Garza

**Programador líder:** Samuel Heaney

**Documentación del programa:** Héctor Garza

**Administración de sonido**

La función de este rol es encargarse de conseguir los sonidos apropiados para los efectos especiales que se pueden observar en el juego. Así como conseguirlos, su rol también es implementarlos en el juego para que puedan ser utilizados y se puedan escuchar en los momentos apropiados. Estos sonidos son la música de fondo que se escucha y los efectos de sonidos cuando se presentan acciones en el juego.

**Diseño gráfico**

La función de este rol es encargarse del manejo de las imágenes que contiene el juego. Para esto el diseñador grafico ocupa saber las funciones básicas para poder crear y modificar imágenes que se manejan. Estas imágenes son prácticamente todo lo que se puede observar en el menú y en el juego. Estas intentan mostrar un mensaje visual para poder ser capaz de transmitir la idea principal con la cual se construyo el juego.

**Programador de objetos**

La función de este rol es encargarse de la administración de los objetos y de los atributos que contienen estos. Además de esto se encarga de que puedan ser desplegados en el juego así como la función de estos pueda ser de la manera esperada.

**Programador de colisiones**

La función de este rol es encargarse específicamente en las colisiones que se observan en el juego. Estas colisiones incluyen interacciones entre las balas y los enemigos del juego así como las colisiones entre las balas y el mapa. Otro tipo de interacciones que se pueden observar son las interacciones del personaje con el mapa para que no caiga en caída libre en caso de encontrarse arriba de un objeto. También las interacciones entre el personaje principal con los enemigos que se observan.

**Física que maneja el juego**

La función de este rol es encargarse que el juego tenga elementos de física para que pueda que pueda tener ciertas semejanzas a la vida real. Un ejemplo de la física en el juego puede ser la gravedad que maneja el juego y como interactúa esta con los objetos que se pueden observar.

**Inteligencia artificial del juego**

La función de este rol es encargarse de la inteligencia que contienen los enemigos del juego para que el usuario que se encuentra jugando pueda tener una mejor experiencia. En caso de no contener algún tipo de inteligencia artificial, la experiencia que se llevaría el usuario no sería óptima y probablemente no retozaría nuevamente nuestro juego.

**Programador líder**

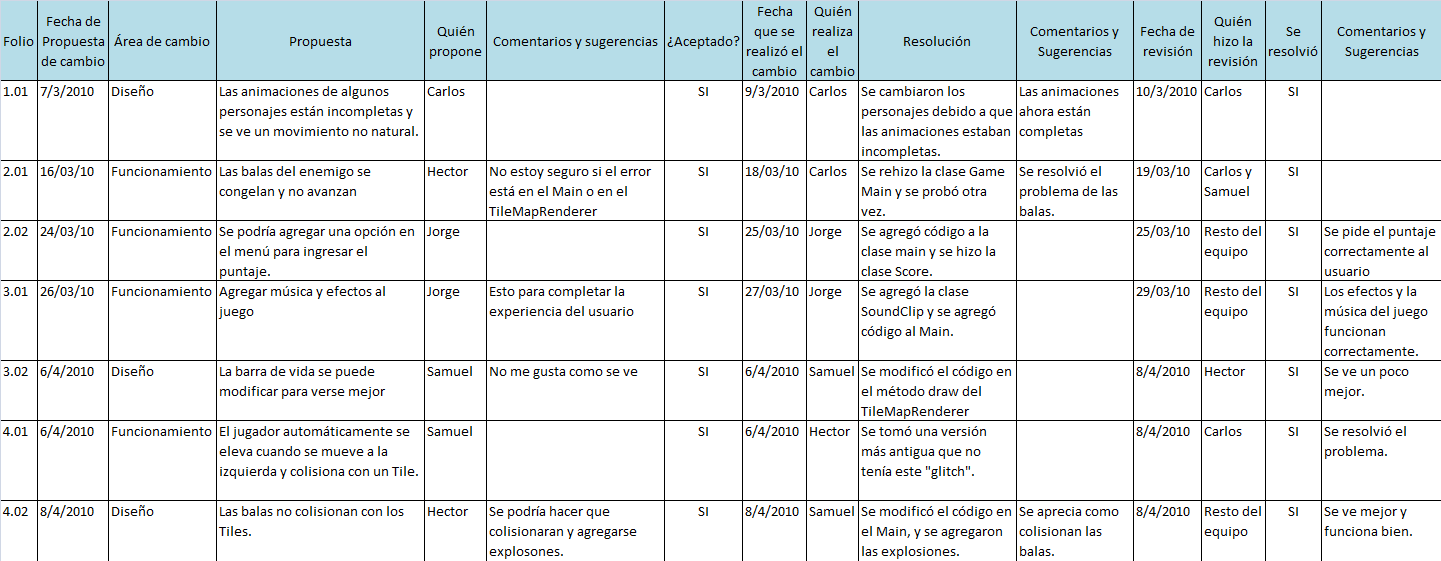
La función de este rol es encargarse de la programación en general y poder analizar y solucionar cualquier dificultad que se encuentre en el programa. Cuando un integrante del equipo tiene dificultades para programar algo, recurre al programador líder para ver si él puede ser capaz de solucionar su problema.

**Documentación del programa**

La función de este rol es encargarse de que el código pueda ser leído por otras personas y pueda ser entendido fácilmente como para que otras personas ajenas puedan ser capaces de poder modificar el programa. Este rol es crucial a la hora de que se quiera empezar a modificar el programa en dado caso que surjan nuevas ideas. Con una buena documentación, a la hora de modificar el programa, el tiempo en entenderlo es significantemente reducido a que el programa no se encuentre documentado.

**Seguimiento de tareas**

Desde la primera reunión que tuvimos en grupo, lo primero que hicimos fue idear un plan para poder empezar a trabajar en el programa de la manera más optima. Para esto, nos dividimos los roles en los cuales cada miembro del equipo se iba a encargar principalmente. Cada miembro del quipo tenía la libertad de trabajar en otras áreas para poder ayudarnos mutuamente y poder ser capaces de superar cualquier dificultad que se nos presentara. Los roles que tuvimos fue más que todo como una base para poder enfocarnos, pero a la vez todos nos encontrábamos programando el juego. Tuvimos varias reuniones afuera del aula de clase para poder darle seguimiento al programa y poder finalizarlo para el tiempo requerido. Todos los integrantes del equipo participaron de manera que todos mostraban interés en el programa. En la mayoría de las reuniones que teníamos en equipo todos asistían puntualmente y se trabajaba en el seguimiento del juego.

**Forma de Control de Cambios**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del Equipo: Moonlight Tecnologies** | | **Alumno Evaluado: Carlos Elizondo** | | |
| **Responsabilidad** en las actividades individuales | Siempre responsable de elaborar la actividad individual previa | La mayoría del tiempo responsable de elaborar la actividad individual previa | Algunas veces responsable de elaborar la actividad individual previa | La responsabilidad en el trabajo fue muy escasa o nula |
| **Puntualidad** en el trabajo encargado a desarrollar | Siempre puntual en el trabajo encargado a desarrollar | La mayoría del tiempo puntual en el trabajo encargado a desarrollar | Algunas veces puntual en el trabajo encargado a desarrollar | La puntualidad en el trabajo fue muy escasa o nula |
| **Cantidad de trabajo** desarrollado | La cantidad de trabajo desarrollado siempre fue satisfactoria | La cantidad de trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la cantidad de trabajo desarrollado fue satisfactoria | La cantidad en el trabajo fue muy escasa o nula |
| **Calidad del trabajo** desarrollado | La calidad del trabajo desarrollado siempre fue satisfactoria | La calidad del trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la calidad del trabajo desarrollado fue satisfactoria | La calidad en el trabajo fue muy escasa o nula |
| **Interés** en el trabajo en equipo | Siempre estuvo interesado en el trabajo en equipo | La mayoría del tiempo estuvo interesado en el trabajo en equipo | Algunas veces estuvo interesado en el trabajo en equipo. | El interés en el trabajo en equipo fue escaso o nulo. |
| **La colaboración** en el trabajo de equipo. | Siempre colaboro intensamente en el trabajo de equipo aportando ideas e investigando. | La mayoría de las veces colaboró en el trabajo en equipo aportando ideas e investigando. | Algunas veces colaboró en el trabajo en equipo aportando ideas e investigando. | La colaboración del trabajo en equipo aportando ideas e investigando fue escasa o nula. |
| **Cumplimiento de la tarea.** | Siempre cumplió con la tarea asignada. | La mayoría de las veces cumplió con la tarea asignada. | Algunas veces cumplió con la tarea asignada | El cumplimiento con la tarea asignada fue escaso o nulo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del Equipo: Moonlight Tecnologies** | | **Alumno Evaluado: Héctor Garza** | | |
| **Responsabilidad** en las actividades individuales | Siempre responsable de elaborar la actividad individual previa | La mayoría del tiempo responsable de elaborar la actividad individual previa | Algunas veces responsable de elaborar la actividad individual previa | La responsabilidad en el trabajo fue muy escasa o nula |
| **Puntualidad** en el trabajo encargado a desarrollar | Siempre puntual en el trabajo encargado a desarrollar | La mayoría del tiempo puntual en el trabajo encargado a desarrollar | Algunas veces puntual en el trabajo encargado a desarrollar | La puntualidad en el trabajo fue muy escasa o nula |
| **Cantidad de trabajo** desarrollado | La cantidad de trabajo desarrollado siempre fue satisfactoria | La cantidad de trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la cantidad de trabajo desarrollado fue satisfactoria | La cantidad en el trabajo fue muy escasa o nula |
| **Calidad del trabajo** desarrollado | La calidad del trabajo desarrollado siempre fue satisfactoria | La calidad del trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la calidad del trabajo desarrollado fue satisfactoria | La calidad en el trabajo fue muy escasa o nula |
| **Interés** en el trabajo en equipo | Siempre estuvo interesado en el trabajo en equipo | La mayoría del tiempo estuvo interesado en el trabajo en equipo | Algunas veces estuvo interesado en el trabajo en equipo. | El interés en el trabajo en equipo fue escaso o nulo. |
| **La colaboración** en el trabajo de equipo. | Siempre colaboro intensamente en el trabajo de equipo aportando ideas e investigando. | La mayoría de las veces colaboró en el trabajo en equipo aportando ideas e investigando. | Algunas veces colaboró en el trabajo en equipo aportando ideas e investigando. | La colaboración del trabajo en equipo aportando ideas e investigando fue escasa o nula. |
| **Cumplimiento de la tarea.** | Siempre cumplió con la tarea asignada. | La mayoría de las veces cumplió con la tarea asignada. | Algunas veces cumplió con la tarea asignada | El cumplimiento con la tarea asignada fue escaso o nulo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del Equipo: Moonlight Tecnologies** | | **Alumno Evaluado: Jorge Saldívar** | | |
| **Responsabilidad** en las actividades individuales | Siempre responsable de elaborar la actividad individual previa | La mayoría del tiempo responsable de elaborar la actividad individual previa | Algunas veces responsable de elaborar la actividad individual previa | La responsabilidad en el trabajo fue muy escasa o nula |
| **Puntualidad** en el trabajo encargado a desarrollar | Siempre puntual en el trabajo encargado a desarrollar | La mayoría del tiempo puntual en el trabajo encargado a desarrollar | Algunas veces puntual en el trabajo encargado a desarrollar | La puntualidad en el trabajo fue muy escasa o nula |
| **Cantidad de trabajo** desarrollado | La cantidad de trabajo desarrollado siempre fue satisfactoria | La cantidad de trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la cantidad de trabajo desarrollado fue satisfactoria | La cantidad en el trabajo fue muy escasa o nula |
| **Calidad del trabajo** desarrollado | La calidad del trabajo desarrollado siempre fue satisfactoria | La calidad del trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la calidad del trabajo desarrollado fue satisfactoria | La calidad en el trabajo fue muy escasa o nula |
| **Interés** en el trabajo en equipo | Siempre estuvo interesado en el trabajo en equipo | La mayoría del tiempo estuvo interesado en el trabajo en equipo | Algunas veces estuvo interesado en el trabajo en equipo. | El interés en el trabajo en equipo fue escaso o nulo. |
| **La colaboración** en el trabajo de equipo. | Siempre colaboro intensamente en el trabajo de equipo aportando ideas e investigando. | La mayoría de las veces colaboró en el trabajo en equipo aportando ideas e investigando. | Algunas veces colaboró en el trabajo en equipo aportando ideas e investigando. | La colaboración del trabajo en equipo aportando ideas e investigando fue escasa o nula. |
| **Cumplimiento de la tarea.** | Siempre cumplió con la tarea asignada. | La mayoría de las veces cumplió con la tarea asignada. | Algunas veces cumplió con la tarea asignada | El cumplimiento con la tarea asignada fue escaso o nulo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del Equipo: Moonlight Tecnologies** | | **Alumno Evaluado: Samuel Heaney** | | |
| **Responsabilidad** en las actividades individuales | Siempre responsable de elaborar la actividad individual previa | La mayoría del tiempo responsable de elaborar la actividad individual previa | Algunas veces responsable de elaborar la actividad individual previa | La responsabilidad en el trabajo fue muy escasa o nula |
| **Puntualidad** en el trabajo encargado a desarrollar | Siempre puntual en el trabajo encargado a desarrollar | La mayoría del tiempo puntual en el trabajo encargado a desarrollar | Algunas veces puntual en el trabajo encargado a desarrollar | La puntualidad en el trabajo fue muy escasa o nula |
| **Cantidad de trabajo** desarrollado | La cantidad de trabajo desarrollado siempre fue satisfactoria | La cantidad de trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la cantidad de trabajo desarrollado fue satisfactoria | La cantidad en el trabajo fue muy escasa o nula |
| **Calidad del trabajo** desarrollado | La calidad del trabajo desarrollado siempre fue satisfactoria | La calidad del trabajo desarrollado la mayoría del tiempo fue satisfactoria | Algunas veces la calidad del trabajo desarrollado fue satisfactoria | La calidad en el trabajo fue muy escasa o nula |
| **Interés** en el trabajo en equipo | Siempre estuvo interesado en el trabajo en equipo | La mayoría del tiempo estuvo interesado en el trabajo en equipo | Algunas veces estuvo interesado en el trabajo en equipo. | El interés en el trabajo en equipo fue escaso o nulo. |
| **La colaboración** en el trabajo de equipo. | Siempre colaboro intensamente en el trabajo de equipo aportando ideas e investigando. | La mayoría de las veces colaboró en el trabajo en equipo aportando ideas e investigando. | Algunas veces colaboró en el trabajo en equipo aportando ideas e investigando. | La colaboración del trabajo en equipo aportando ideas e investigando fue escasa o nula. |
| **Cumplimiento de la tarea.** | Siempre cumplió con la tarea asignada. | La mayoría de las veces cumplió con la tarea asignada. | Algunas veces cumplió con la tarea asignada | El cumplimiento con la tarea asignada fue escaso o nulo. |

**CONCLUSIONES**

**Sobre el desarrollo del proyecto**

**Carlos Elizondo:**

En la etapa de desarrollo del proyecto existieron algunos problemas sobre todo en la programación ya que no entendíamos por completo la manera de generar un juego con tiles e ir moviendo el fondo en vez del objeto para poder hacer sentir el movimiento al usuario. Esto lo logramos superar con apoyo mutuo en el que algunos estudiaban algo y otros seguían intentando para luego llegar al funcionamiento óptimo del juego. El código de Brackeen no lo entendíamos por completo pero lo que nos ayudó fue combinar tanto código de Brackeen como de nuestras propias ideas para poder adecuar el juego. De esta forma llegó a la conclusión que si bien es necesario adecuar la programación a plantillas ya existentes. También es necesario poder modificar con ideas propias para un mejor funcionamiento.

**Héctor Garza:**

En el desarrollo del proyecto obtuvimos una gran cantidad de aprendizaje en organización de clases. El desarrollo permite generar diagramas UML para que todos puedan seguir cada parte del proyecto. En un principio teníamos que entender el código de Brackeen para poder trabajar. Sin embargo, éste era un poco complejo por lo que logramos superar el problema utilizando código propio y adecuando el de Brackeen. Por ello es interesante utilizar lo ya existente y combinarlo con lo propio.

**Jorge Saldívar:**

Durante la etapa del proyecto tuvimos varias complicaciones que dificultaban el desarrollo y el seguimiento del programa. Estas complicaciones las pudimos superar gracias al trabajo mutuo que tuvimos y a la preparación que hemos tenido durante el curso. Para poder crear el juego que teníamos pensado desde un principio, nos basamos en el código de Brackeen y lo reutilizamos para poder crear nuestro juego en base a nuestras ideas. Cuando recién comenzamos a programar el juego, nos enfocamos prácticamente en pocas cosas debido a que para que funcionara correctamente, primero teníamos que enfocarnos en pocas cosas. La primera de las cosas en la cual nos enfocamos, fue básicamente en la utilización del mapa como lo utiliza el código de Brackeen. Una vez ya que nos iban saliendo las cosas como queríamos íbamos enfocando en cosas diferentes para ir acabando el juego.

**Samuel Heaney:**

Se aprendió como trabajar como equipo en el desarrollo del proyecto de un juego de programación. Fue importante la colaboración y esfuerzo de todos los integrantes del equipo. Se logró sacar el proyecto a tiempo y con la dedicación debida. Hubo muchos errores pero pudimos resolverlos en base a las fortalezas de cada integrante y se logró avanzar el proyecto.

**Sobre la administración del proyecto**

**Carlos Elizondo:**

En esta área no tuvimos una gran cantidad de obstáculos ya que todos nos poníamos de acuerdo para lograr las reuniones. Desde un principio generábamos metas y tiempos de corte para llegar a alguna meta. De esta manera logramos cumplir con los tiempos de entrega y al mismo tiempo entregar un proyecto de calidad.

**Héctor Garza:**

En esta área se logró administrar el tiempo para cumplir con lo esperado por nuestro cliente (el profesor). Se logró generar diferentes formas para poder trabajar como el formato de cambios y así podíamos saber y llevar un control sobre lo que hacíamos. Esto, en conclusión, permite trabajar de una manera eficiente y así cumplir con los tiempos de entrega.

**Jorge Saldívar:**

En la parte administrativa del proyecto no tuvimos muchas complicaciones. Definimos los roles de que es lo que cada quien iba hacer antes que nada para así poder enfocarnos en más cosas a la vez. Tuvimos varias reuniones para trabajar en equipo y así poder ayudarnos entre todos en caso de tener complicaciones. Fuimos capaces de administrar bien el tiempo debido a que siempre cumplíamos con el tiempo de entrega.

**Samuel Heaney:**

La administración del proyecto fue constante en donde cada integrante se encargaba de tener copias de respaldo en caso de problemas que pudieran suscitar y de tal forma cada quien logró trabajar en su parte de forma ordenada. Tuvimos varias juntas en donde nos enfocamos a compartir ideas y a avanzar el proyecto en equipo de igual forma a como lo estábamos avanzando individualmente.

**Trabajos futuros**

**Carlos Elizondo:**

Con los conocimientos obtenidos en este proyecto. La empresa se puede dedicar a generar diferentes aplicaciones basadas en juegos sobre todo para publicidad y así llegar a un concepto de advergaming. Además aprendimos a administrar el tiempo de manera eficiente. En un futuro esperamos poder tener contratos de empresas para generarle s juegos.

**Héctor Garza:**

En este momento he aprendido a realizar un desarrollo con la organización del tiempo y a administrar el software. De esta manera puedo buscar clientes que busquen un desarrollo de juegos y hacerlo de una manera eficiente. Espero poder desarrollar proyectos de juegos futuros.

**Jorge Saldívar:**

Nuestro equipo está dispuesto a trabajar en más proyectos en caso necesario debido a que conseguimos los conocimientos para poder crear juegos y pudimos organizarnos bien. Yo opino que este proyecto nos va ayudar en un futuro cuando ocupemos crear alguna aplicación o simplemente cuando trabajemos en equipo. Nuestro equipo fue muy responsable en cuanto a trabajar y esto nos puede ayudar en un futuro.

**Samuel Heaney:**

Aunque no nos dediquemos a desarrollar juegos en un futuro, este trabajo nos ha servido para organizar un proyecto de software y nos da las herramientas para trabajar en nuevos proyectos a corto, mediano y largo plazo. Tuvimos que aprender a asignar roles de manera que coincidiera con las fortalezas de cada integrante del equipo. Aprendimos a recolectar ideas y plasmarlas eficaz y eficientemente en el proyecto, habilidad que nos ayudará en futuros proyectos.