**Requisitos de FourElements**

**Controles:**

W: andar hacia arriba

S: andar hacia abajo

A: andar hacia izquierda

D: andar hacia derecha

ESC: pausa/unpausa y da lugar a las opciones de "Reunadar o salir a la pantalla principal"

Flecha arriba: moverte arriba en el menu de pausa

Flecha abajo: moverte abajo en el menu de pausa

Flecha izquierda: moverte a la izquierda en el menu "GameOver"

Felcha derecha: moverte a la derecha en el menu "GameOver"

La clase principal "Game" extiende de Canvas y contiene un Thread principal que es sobre el cual corre todo el juego.

El Canvas tiene una dimension equivalente a la anchura y altura escaladas a un numero de la variable Scale, en este caso es por 3. Dejando unas dimensiones de 1200x675 del JFrame que contiene el Canvas.

El juego contiene dos metodos esenciales que son update() y render(). El primero se encarga de la parte logistica y el segundo de la parte grafica.

El update() ocurre una media de 60 veces por segundo y el render() todas las veces que puede ejecutarse el run() del Thread principal. Los resultados de cada actualizacion se pueden verificar en vivo cuando se ejecuta el juego, vienen indicados en el titulo del JFrame.

La parte grafica se efectua mostrando una imagen creada a partir de una array de Pixeles. Esta array se rellena con cada ejecucion del metodo Game.render(), los nuevos datos vienen de la array Pixeles de la clase Screen. Esta clase es la que se encarga de rellenar su array con varios metodos cuales se usan por todos los elementos que tengan que ser mostrados por pantalla. Lo que significa que cada elemento que tiene que ser mostrado en pantalla contendra un metodo de la clase Screen. Estos elementos contienen, a su vez, una referencia de un Sprite que tenga toda la informacion sobre la imagen que le tenga que representar (una array de pixeles con dimensiones exactas a la imagen).

Aqui tengo que hacer un matiz sobre las array unidimensionales que uso, aun que en cierto modo deberian de ser bidimensionales, mediante el uso de identificacion de cada elemento dentro de la array de este tipo [y + x \* anchuraElemento] se comportan como tales (mas bien puedo identificarlos con coordenadas X e Y cada uno).

La clase Sprite tiene como atributo destacado, la array ya mencionada antes, esta array se obtiene de la clase SpriteSheet que es la encargada de cargar imagenes en Arrays, aun que tambien tiene otra funcionalidad que es la de cargar imagenes grandes que contengan varios elementos. Osea que de un SpriteSheet suelen salir mas de un Sprite.

Otra clase destacada de la seccion grafica es la de Font, ya que el juego usa una resolucion baja pero esta se escala, el sistema de coordenadas no se corresponde con la imagen en escala 1:1 sino de 1:3. Por lo que usar el sistema de pintado g(Graphics).drawString() sera inadecuado (aun que se podria solucionar reescalando las coordenadas tambien). Por lo que uso esta clase para leer de SpriteSheets que contengan numeros y letras, y asi usarlos para escribir lo deseado. Solo se puede escribir en mayusculas y sin simbolos "extraños", aun que siempre se puede expandir.

Los controles se encuentran en las clases Keyboard y Mouse, son extensiones de KeyListener el primero y MouseListener/MouseMotionListener, el segundo.

La clase Level, se podria decir, que es el corazon de todo el funcionamiento. Esta clase es la que se encarga de mantener una instancia de nivel y todos sus elementos actualizados y pintados. Los atributos mas destacables son anchura, altura, casillas, entidades, proyectiles, particulas, items y jugador (en el codigo estan en ingles).

Hay cuatro ArrayList (entidades, proyectiles, particulas, items) que se encargan de mantener las entidades actualizadas. La anchura y la altura son las correspondientes a las dimensiones del mapa en el que se encuentra el nivel y tiene directa relacion con el atributo casillas. Este es una array que contiene todos los tipos de casillas del nivel. Una casilla es una clase que tiene un Sprite y un metodo esencial solid() que indica si se puede atravesar o no. Existe un metodo (loadLevel()) que se encarga de recibir una direccion de una imagen cual contiene la informacion para llenar un nivel de un tipo de casillas u otras. Cada color de pixel de esta corresponde a un tipo de casilla, de esta manera la creacion de niveles se simplifica a solo pintarlos en un editor como Paint.net

Otro metodo de Level es add(Entity e), este añade una entidad al nivel y a un tipo de lista dependiendo de que tipo de entidad sea.

La clase Entity tiene como atributos x, y (sus coordenadas en el nivel), Sprite, Level y metodos como: update(), render, remove() (este pone un booleano a true y otro metodo de Level se encargara de eliminarlo de una de sus listas) e initLevel() ("inicia" el nivel a cada una de las entidades agragadas a este). Esta clase podria ser una clase abstracta ya que no se crea ninguna instancia de esta.

La clase Mob hereda de Entity, es la clase que da vida a entidades de tipo Player o Zuko, osea personajes del juego. Como destacable poseen metodos para disparar proyectiles, moverse y colisiones con proyectiles, casillas, items/buffs y otros mobs.

La clase Particle hereda de Entity, esta se usa para liberar particulas cuando ocurre un impacto. Como atributos mas destacables son vida y tiempo, se encargan de matar a la particula si sobre pasa su tiempo de vida. Metodos: collision(), asi una vez liberadas no ignoran los obstaculos como las casillas (si las entidades). Se podrian hacer fisicas de rebote en el "suelo", estan implementadas pero no se usan.

Clase Projectile, hereda de Entity. Como atributos tiene las coordenadas de origen (donde fue disparado), rango, daño y un hitBox (para la colision con las entidades).

Clase Spawner, hereda de Entity. Se usa para generar particulas (tambien para generar Mobs), particularme no me gusta y probablemente se deseche con el tiempo sustituyendose por algo mas simple.

Cabe destacar que la raiz de la estructura esta inspirada en una persona con pseudonimo "The Cherno" y se pueden encontrar sus videos en YouTube donde explica partes clave de esta arquitectura.