17 de DICIEMBRE de 2020

laura maria zuluaga jaramillo

UniVERSIDAD DE ANTIOQUIA

Medellín - Antioquia

RESCUE KEY

idea y definición de clases

# **Rescue key**

Descripción general del juego:

Rudy, quien es este caso el personaje principal se encuentra en peligro y debe conseguir la llave que solo aparecerá después de lograr conseguir cierta cantidad de monedas justo antes que se acabe el tiempo. Conseguir la llave será la única forma de alcanzar el siguiente nivel y lograr ponerse a salvo de cada mundo; la forma en la que alcanzará el objetivo, es subiendo por medio de plataformas que se encuentran en movimiento constante y que además se encuentran llenas de obstáculos y trampas. Dentro de los obstáculos podremos encontrar diferentes tipos de mecanismos con objetos cortantes que pueden causar la muerte del jugador; así mismo, existirán algunas plataformas seguras donde Rudy podrá analizar su siguiente movimiento; en algunas de estas plataformas podrá encontrar una variedad de pociones que darán sangre al jugador para resistir durante su camino al final del mundo o le darán inmunidad por un tiempo determinado.

A medida que avanza en los diferentes niveles, el movimiento de las plataformas y la cantidad de obstáculos aumentará, mientras que el tiempo cada vez es menos.

En la opción de multijugador, se podrá elegir ser Rudy para cumplir el objetivo del juego descrito anteriormente o elegir ser un enemigo, donde se debe tratar de evitar que el personaje principal logre avanzar en los diferentes niveles.

Definición de clases:

* **Player:**

Clase que será utilizada para crear el usuario principal. Esta contendrá métodos tales como moverse hacia la izquierda, moverse a la derecha, saltar y agacharse.

Atributos:

* Float dimension[2]= {Alto, ancho} (Definición del tamaño del personaje)
* Float position[2]={PosX, PosY} (Posición del personaje en el GraphicView)
* Float speed (Velocidad con que el se mueve el personaje)

Métodos:

* Player(QString, float, float) (Se enviará el tamaño que tendrá el personaje y path de la imagen)
* Void Jump() (Utilizada para saltar)
* Void Squat() (Agacharse)
* Void Left() (Ir hacia a la izquierda)
* Void Right() (Ir hacia la derecha)

* **Villain:**

Se implementará herencia de la clase jugador, ya que se utilizarán los mismos métodos y atributos implementados en ella y se añadirán nuevos métodos como disparar.

Atributos:

* Int Shots() (Cantidad de disparos que tiene el enemigo)
* Float PosShot[2]={Posición x, Posición y} (Posiciones de la bala)
* Float VelShot[3]={Vel inicial, Vel en x, Vel en y} (Velocidades de la bala)
* Int Angle (Ángulo de disparo)

Métodos:

* Villain()
* Void Reload(int) (Modificar la cantidad de disparos, recibe un entero con la cantidad que carga)
* Void Shoot(Float \*, Float\*, int) (Método para disparar, recibe información para calcular trayectoria de la bala)
* **Poción:**

Esta clase también será utilizada para la creación de las monedas, que utilizaran algunas de los métodos implementados en ella.

Aumentar puntaje

Aumentar sangre

Generar inmunidad

* **Plataformas:**

Moverse izquierda

Moverse derecha

Realizar MCU

* **Trampas:**

MAS

Movimiento parabólico

MCU

* **Barras:**

Estas barras se utilizarán para los tipos de objetos que funcionarán como barras de progreso, tales como la cantidad de sangre de los personajes, puntaje y tiempo.

Aumentar

Disminuir

Congelarse

* **Objetos:**

Esta clase será utilizada para poner las imágenes que servirán de decoración al entorno gráfico, tales como el fondo y la llave. La llave se trabajará como Sprite para que gire todo el tiempo que este visible.

* **Datos:**

Esta clase será implementada para lo que es todo el tema de la información tales como el logueo de los usuarios y manejo de partidas.

Abrir archivos

Cargar datos

Borrar datos

Guardar datos.