**Clases preliminares de "Hit Drifter"**

**primero comenzaremos con las clases básicas del videojuego:**

**\*Personaje\***

La clase personaje será usada en los jugadores y enemigos de nuestro juego, en esta se gestionará tanto vida, salud, experiencia, daño y otras estadísticas de cada mob o jugador(es).

**Atributos**

int Vida

int Defensa

int Experiencia

int Nivel

Arma arma

int Velocidad (movimiento)

// qtgraphics Cuerpo.

Int x ,y;

/\*

El atributo "cuerpo" solo es una clase que heredará los métodos definidos de QGraphicsscene (QgraphicsRectItem). Definirá el cuerpo del personaje, desde esta se definirá posición del jugador o adversario en el mapa; será una escena de la misma librería Qgraphicsscene.

\*/

**Métodos**

Cada uno de estos métodos tiene como entrada un valor que se define al presionar ciertas teclas en tiempo de ejecución (Si el usuario presiona w,a,s o d. Para movimientos, o por ejemplo una tecla bien podría ser j o alguna otra para el ataque generando una interrupción y dependiendo de la tecla será la entrada a estos métodos) . Además tendrá los métodos definidos por la misma librería QGraphicsRectItem.

-void Desplazamiento ( ) Crea una trayectoria de desplazamiento para el personaje en relación a su velocidad, ya sea un salto vertical, una parabólica o un desplazamiento lineal.

-void Attack (int x int y)Genera las acciones que implican a que el jugador ataque. Como consecuencia de que este haya presionado cierta tecla.

El método Attack tomará las estadísticas de una clase llamada "Arma", calculando así todos los efectos como por ejemplo; los daños infligidos ,después de la interacción con un enemigo. //lista de enemigos

-void Area\_retroceso(int \_xj, int \_yj, int \_xe, int \_ye) Verifica si se efectuaron damages. De ser así, se realiza el cálculo del retroceso según, la velocidad del jugador y la del enemigo.

-void Estado (int \_vida) Es una función que determina la condición del jugador o enemigo en caso de que esté vivo o muerto. Si el estado de un enemigo es “Eliminado” se quitará de la pantalla y se incrementará la experiencia del jugador. Si el jugador es “Eliminado” se llama la funcion Respawn ().

-void Respawn(). en caso de que el Estado del jugador sea “Eliminado”, se regenerará al jugador y a los enemigos en el último punto de carga.

-void Progress() Al final de cada zona calcula si con los puntos de experiencia acomulados, el jugador sube de nivel. Si lo hace, podrá mejorar alguno de sus atributos.

-void AutoLevel () Dependiendo del número de zonas recorridas adapta la dificultad de los adversarios. Incrementa su vida, velocidad, defensa etc.

**\*Arma\***

Las armas en el juego vienen con propiedades preestablecidas. Sim embargo, el valor de esas características puede variar dependiendo de la experiencia del jugador.

El jugador hará uso de los atributos de esta clase para realizar sus ataques, las armas son imprescindibles para lograr sobrevivir en el juego; algunas armas son dejadas por monstruos o jefes, otras serán encontradas en cofres.

**Atributos**

string Nombre\_arma

int Daño\_ataque

int Alcance

int Probabilidad

**Métodos**

-Constructor(string \_Nombre\_arma,int \_Daño\_ataque,int \_Alcace,int \_Probabilidad)

(Definirá el arma que deseo crear, dando valores a cada atributo ej.

Espada("Excalibur", 25, 30,12) ).

-void Drop (int \_xe,int \_ye) Es un método que definirá la probabilidad de que al matar un enemigo suelte un arma o que al abrir un cofre aparezca en este.

**\*item\***

Son todos los artículos que dispone el juego, están para mejorar las propiedades del jugador.

**Atributos**

string Nombre

**Métodos**

-void Cualidad() recibe el nombre del item y según eso; devuelve el un valor que incrementará los atributos del jugador.//archivo

\***Obstáculo\***

Serán elementos que interactúan con el jugador de igual manera que los enemigos.

**Atributos**

int ancho

int largo

int posición

int Velocidad

int Daño

**Métodos**

**//**preguntar

-void Area\_Daño () Genera un ataque constante en el espacio del obstáculo.

-void Movimiento() Será una función que calcule la trayectoria de movimiento del obstáculo, si su velocidad es diferente de cero.(puede ser pendular)

**\*Escenario\* (Clase predefinida):**

//preguntar

Es una clase definida por la librería Qgraphicsscene, que almacenará todos los objetos y obstáculos del mapa para generar el inicio del juego. Contendrá los **métodos** definidos por dicha librería.

**\*Menú\***

La clase servirá para que el usuario pueda cargar alguna partida, guardar su propia partida,cerrar sesión o salir de juego.

**Atributos**

string Nombre\_partida

time fecha

**Métodos:**

-void Guardar/Cargar() Para cargar una partida, esta puede ser digitada por el usuario o buscada en el explorador de archivos. Al momento de Guardar una nueva le asignará la fecha del guardado para evitar confusiones con los nombres.

-void Cerrar\_sesion() Llama el destructor de login, jugador, enemigos, escenario, obstaculos, items, arma . Se crea una nueva instancia de login y llama el método Verificación de Login.

-void Salir\_Juego() Llama el desctructor de menu.

**\*Login\***

Es una clase que se desplegará en la primera carga del juego, pedirá al jugador su usuario y contraseña. Además, si el jugador es nuevo lo dejara registrarse.

**sin atributos**

**Métodos:**

-void verificación() Revisa que el usuario esté registrado, en ese caso pide la contraseña y evalúa si coincide con la de los datos del usuario. Por otra parte, si no se encuentra la existencia del usuario llama a Registro.

-void Registro(). verifica que el nombre del usuario no esté en uso y guarda los datos del nuevo usuario.

/\* Todas las clases tendrán una implementación grafica, creando así una facil interacción para el usuario.\*/