**Diagrama de clases:**

**Login de usuarios**

Para el logueo de usuarios se abrirá una ventana que pedirá el usuario y la contraseña, tendrá tres botones; uno para validar el usuario, otro para agregar un nuevo usuario y otro botón para leer las instrucciones del juego. Los usuarios se manejarán con archivos de texto, de la manera que se hizo en el Laboratorio 3

**Clase** Login

Privado:

String Nombre\_usuario

String Pass\_usuario

Public Slots:

Void iniciar\_sesion\_clic()

Void nuevo\_user\_clic()

Void Instrucciones\_clic()

En el botón iniciar sesión se valida que el usuario este registrado y el botón nuevo usuario abre otra ventana para llenar los datos para el registro

**Clase** Nuevo\_Usuario

Privado:

String nombre\_nuevo\_user

String pass\_nuevo\_user

Public slot:

Void Crear\_user\_clic()

Cuando se le da clic al botón crear usuario se crea un archivo de texto que guarda estas credenciales para le logueo. Esta ventana se esconde y se vuelve a abrir la del Login para que el usuario inicie sesión.

El botón de Instrucciones muestra una ventana con un texto con las instrucciones del juego, movimiento de las naves e información de los aliados.

Una vez se valida el usuario, se abre una ventana que pedirá el nombre del otro jugador, este no necesitará una contraseña.

**Clase** Oponente

Privado:

String Nombre\_oponente

Publico:

String getNombre\_oponente

Void setNombre\_oponente

Este nombre se toma para desplegarlo en el escenario junto a la barra de vida del lado de los invasores, mientras que el nombre del jugador que inicio sesión, se coloca del lado de la tierra junto a la barra de vida.

**Escena**

Para la escena tengo pensado que sea un cuadro grande, de unos 1000\*700, donde se ubicaran los jugadores y los obstáculos. También aparecerán aliados de cada bando.

En la parte superior estará el nombre de cada jugador con su vida correspondiente

**Clase** Universo: hereda de QGraphicsView

Publico:

QGraphicsScene \*scene;

Player1 \*player1

Player2 \*player2

Health \*vida1

Health \*vida2

Bala\_jugador \*bala\_jugador

Aliado \*Ally1

Aliado \*Ally2

Aliado \*Ally3

Bala\_ally1 \*b\_ally1

Bala\_ally2 \*b\_ally2

Bala\_ally2 \*b\_ally2

Wall \*wall

QLabel \*nombre\_player1

QLabel \*nombre\_player2

QTimer \*timer\_universal

Los dos jugadores se posicionan a lado y lado de la escena, solo se mueven de arriba abajo y pueden disparar balas normales. Sería la misma clase para cada jugador, pues solo aparecerían dos objetos de esta que representarían a los jugadores. Se diferenciarían por la imagen y la dirección de las balas. También se puede hacer una clase de jugador dos adicional por si se complica mucho el asunto, pero tendría la misma forma.

**Clase** Player1 hereda de QObject y QGraphicsItem

Publico:

Constructor ()

Void keyPressEvent()

Void Disparar ()

QTimer \*timer\_n

Los aliados serán objetos macro en el juego, ya que solo se programan y cada cierto tiempo aparecen ejecutando una acción y desaparecen. Las balas disparadas de este objeto no se pueden detener por los obstáculos. Serian 3 el número de aliados por cada bando; aparecería uno desde arriba, otro desde abajo y el ultimo detrás de las naves. Se necesitaría una clase para cada uno, pero cada clase luciría como muestro a continuación.

**Clase** Aliado hereda de QObject y QGraphicsItem

Q\_OBJECT

Privado:

QTimer \*timer

Publico:

Constructor ()

Public Slots:

Void Ally\_Move()

Para las balas se tendrían que hacer 4 clases, ya que son 4 tipos de balas: las disparadas por los jugadores que desaparecen en los obstáculos, las disparadas por los 3 aliados que se verán afectadas en el movimiento por algún fenómeno físico. Sin embargo, las clases de las balas tendrían el siguiente prototipo

**Clase** Bala hereda de QObject y QGraphicsItem

Q\_OBJECT

Publico:

Constructor ()

Public Slots:

Void Bullet\_move

En la función de movimiento de cada bala se definiría el fenómeno físico que las afectaría y las colisiones.

**Clase** Wall hereda de QObject y QGraphicsItem

Publico:

Constructor ();

La siguiente clase es para mostrar los nombres de los jugadores

Clase Nombre\_jugador hereda de QGraphicsTextItem

Publico:

Constructor ()

La vida de los jugadores se representará por 5 cuadros que irán desapareciendo cuando alguna bala choque con un jugador

**Clase** Health hereda de QGrapichsRectItem

Publico:

Constructor ()

Void decrease()

Void Escribir\_tablon()

Cuando la los cuadros de vida de algún jugador llegan a cero se detienen los timers y se escribe en el tablón el resultado. Este se guarda en un archivo de texto.

Cuando se acaba un encuentro (Algún jugador pierde toda la vida), aparecerá un cuadro con tres botones: Uno para iniciar una partida nueva, otro para ver el tablón de resultados y otro para salir.

**Clase** Menu\_final

Private slots:

Void Juego\_nuevo\_clic()

Void ver\_tablon\_clic()

Void salir\_clic()

**Cronograma**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Semana 1 | | | | | |
| Actividades | Lunes | Martes | Miércoles | Jueves | Viernes |
| Definición diagrama de clases |  |  |  |  |  |
| Login de Usuarios |  |  |  |  |  |
| Semana 2 | | | | | |
| Prueba de login de usuarios |  |  |  |  |  |
| Universo, Jugadores con sus balas, obstáculos y vida – Instanciamiento de clases, definición de metodos |  |  |  |  |  |
| Pruebas en el Universo |  |  |  |  |  |
| Semana 3 | | | | | |
| Aliados y sistemas físicos – Instanciamiento de clases, definición de metodos |  |  |  |  |  |
| Pruebas finales |  |  |  |  |  |