Universidad de Antioquia
Agudelo Zapata Andrés Felipe
Correa Morales Robinson

1 - Contextualización:

Capítulo de inspiración: Rick potion T1 E6.

Historia:

En la escuela de Morty se acerca un baile escolar y él tiene la intención de invitar a salir a una compañera. Sin embargo, sus compañeros lo avergüenzan y lo ridiculizan, dejándolo triste. Morty decide pedir ayuda a su abuelo, un genio científico, para que le proporcione una fórmula que lo haga enamorar profundamente a otra persona. Después de mucha insistencia, Morty logra obtener la fórmula y conquista el corazón de su compañera. Pero no anticipa que esto tenga un efecto inesperado, ya que provoca que toda la escuela se enamore perdidamente de él. Ante esta situación, su abuelo crea un antídoto para revertir los efectos, pero este resulta peor, convirtiendo a la gente en mantis que desean copular con Morty y luego matarlo. Estos efectos no afectan a la familia de Morty, y sus padres se ven obligados a buscar una forma de sobrevivir en este mundo apocalíptico.

Dinámicas:

Poción de vida: Estas se van a generar aleatoriamente en el mapa y permiten al jugador recuperar vida perdida.

Armas: Se van a desarrollar diferentes tipos de armas, cada una con sus habilidades especiales para permitir matar a las mantis de mejor manera.

Mantis: Las mantis religiosas son el enemigo principal del juego, existen diversos tipos de mantis, cada una con una habilidad especial que complican el avance del jugador.

Jefe final: Algunas mantis y las bombas del jefe final poseen físicas especiales.

Niveles:

Nivel 1: Separa algunos compuestos químicos para generar la fórmula capaz de enamorar a cualquier persona, hazlo antes de que se agote el tiempo y lograras pasar el nivel.

Nivel 2: Las personas se han vuelto mantis religiosas y Larry debe matarlas para salvar a su hija y detener el caos. Se implementarán distintas armas para cumplir este objetivo y un sistema de vida.

Nivel 3: Larry debe matar a la langosta jefe que tiene en su poder a su hija. La langosta jefa posee más vida, puede dejar caer bombas que explotan y hacen mucho daño.

2 - Analís de niveles:

Capítulo de inspiración: Rick potion T1 E6.

1 – Primer nivel: Vista frontal, constará de un fondo de estantes y las botellas, las botellas se pueden cambiar de posición. Se debe de aislar un compuesto que está mezclado con otro. Serán 3 botellas con 3 niveles de llenado cada una. Cada compuesto podrá estar en cada botella en una proporción de 1/3 o 2/3 o 1. Si una botella está llena, no podrá recibir contenido. El contenido del mismo tipo se junta, es decir, si la botella 1 tiene 1/3 del compuesto a y 1/3 del compuesto b y se vacía en ella el contenido de la botella 2 que es 1/3 del compuesto b, la botella 1 quedará con 1/3 del compuesto a y 2/3 del compuesto b. La interacción será por mouse, se utilizará el ratón para manipular las botellas.

Al vaciarse el contenido de una botella en otro, se ejecutará una animación representativa. El nivel cuenta con un límite de tiempo para ser completado.

Una de las tres botellas tendrá 1/3 libre, que permitirá hacer el intercambio de compuestos entre las 3 botellas.

2 – Segundo nivel: Vista frontal, movimiento horizontal y vertical (izquierda, derecha, adelante y atrás) Constará de una cantidad fija de enemigos, los cuales deben de ser eliminados. Los enemigos hacen daño al jugador.

2 – 1 Arma de bombas: Esta arma mata a un enemigo de un disparo, el disparo tiene un movimiento parabólico (caída) por lo que su distancia es limitada. Se desbloquea tras completar un evento del nivel (la eliminación de una cantidad de enemigos). Posee una cantidad de disparos baja y no se puede recargar.

Las ecuaciones paramétricas que dan el movimiento del arma de bombas están dadas por:

$X = v*\cos(\alpha)*t$

$Y = v*\sin(\alpha)*t - (1/2)*g*t2$

Con:

v = magnitud de la velocidad inicial

α = ángulo de disparo del proyectil

t = tiempo

g = gravedad

2 – 2 Pistola: posee mayor distancia, pero requiere más disparos para matar a un enemigo. Esta arma es el arma estándar del nivel.

2 – 3 Enemigos: Los enemigos son mantis religiosas que hacen daño al jugador con sus pinzas (el enemigo colisiona con el jugador para hacerle daño). Estos enemigos vendrán de puntos de generación en los extremos del mapa y se acercarán al jugador.

2 – 4 Poción de vida: Permite al jugador reestablecer sus puntos de salud, el jugador pasará por encima de ella para recuperar la vida. Si el jugador no ha perdido vida, la poción no hace efecto. Esta se generará aleatoriamente al matar un enemigo.

3 – Tercer nivel: Vista frontal, movimiento lateral (izquierda y derecha). El jugador deberá enfrentarse a la mantis jefe, la cual se encuentra volando (parte superior del mapa) Para hacerle daño se hará uso de 3 cañones apuntando hacia arriba, los cuales al ser activados disparan un proyectil hacia arriba, si el proyectil impacta en el enemigo, le restará vida.

3 – 1 Bombas: El enemigo tiene la posibilidad de disparar bombas que caen y generan daño de área, estas bombas se vuelven más frecuentes con el paso del tiempo y mientras menos vida tenga el jefe.

3 – 2: Cañones: para atacar a este enemigo debemos de hacer uso de tres cañones que se encuentran en diferentes lugares del mapa. Sin embargo, para usar estos cañones, también deben de tener en cuenta la fuerza de la gravedad cumpliendo que:

$X= v*\cos(\alpha)*t$

$Y=v*\sin(\alpha)*t-(1/2) *g*t2$

Con:

v = magnitud de la velocidad inicial

α = ángulo de disparo del proyectil

t = tiempo

g = gravedad

Los cañones se deben activar con una tecla y su ángulo de disparo se controla con el ratón.

3 – 3 Enemigo: El enemigo vuela en el mapa, se puede mover de izquierda a derecha y derecha a izquierda, este jefe tiene mucha más vida que los otros enemigos y que el

jugador haciendo que para poder derrotarlo se deban de golpearle con varias bombas de los cañones.

3 - Diseño de Clases

3.1 - Diferentes Clases implementadas

Nivel 1:

Se trata de resolver un acertijo, en donde el resultado final es la *poción de amor.*

Clase bottle: Esta clase permite modelar una botella con 3 niveles de llenado, La botella tendrá 3 objetos de la clase sustancia.

Clase liquid: Esta clase permite modelar una sustancia, habrá tres sustancias diferentes: verde, rosa, naranja.

Clase poción: Esta clase modela una poción, está compuesta por las clases anteriores.

Nivel Transición:

Este nivel conecta el nivel 1 con el nivel 2, posee características similares al nivel 2 aunque su función es servir de puente entre ambos niveles y dar contexto al segundo nivel, se conforma de un objeto tipo *player* y un *QGraphicsPixmapItem*.

Clase Player: Permite modelar un jugador, su avatar (textura) su movimiento e interacciones, Se utiliza tanto en el nivel de transición como en el nivel 2, en este último puede utilizar armas, recibir daño y acumular puntos.

Nivel 2:

Clase arma: Clase que modela un objeto que, tiene valores de daño, y rareza, interactúa con el jugador dándole la posibilidad de matar los enemigos al disparar un proyectil.

Clase enemy: Clase que modela objetos que buscan la ruta óptima hacia el jugador con el objetivo de matarlo. Estos enemigos se generan desde los bordes del mapa, al ser eliminados se puede generar un evento que lleve a la creación de un objeto tipo *item*, ya sea munición o poción de vida.

Clase item: Clase que modela un objeto que, al chocar con él, restaura una cantidad de la HP del jugador o recarga su munición.

Clase bala: Clase que modela los proyectiles lanzados por los objetos de la clase arma, interactúan con los objetos de clase *enemy* disminuyendo su HP.

Nivel 3: *no implementado*

Clase cañón: Tres objetos de esta clase se generan en el mapa, al presionar una tecla determinada se inicia un evento en el que primero se apunta el proyectil y después se dispara.

Clase bala: Este objeto tiene valores de x, y en y un ángulo inicial, además de su velocidad, su velocidad está dada por las ecuaciones paramétricas del movimiento parabólico, y cada atributo de esta clase tiene una velocidad inicial predefinida.

Clase bombas enemigas: Esta clase modela una bomba que es afectada por la gravedad y puede explotar pasado un tiempo, si el jugador se encuentra cerca, recibirá mucho daño.

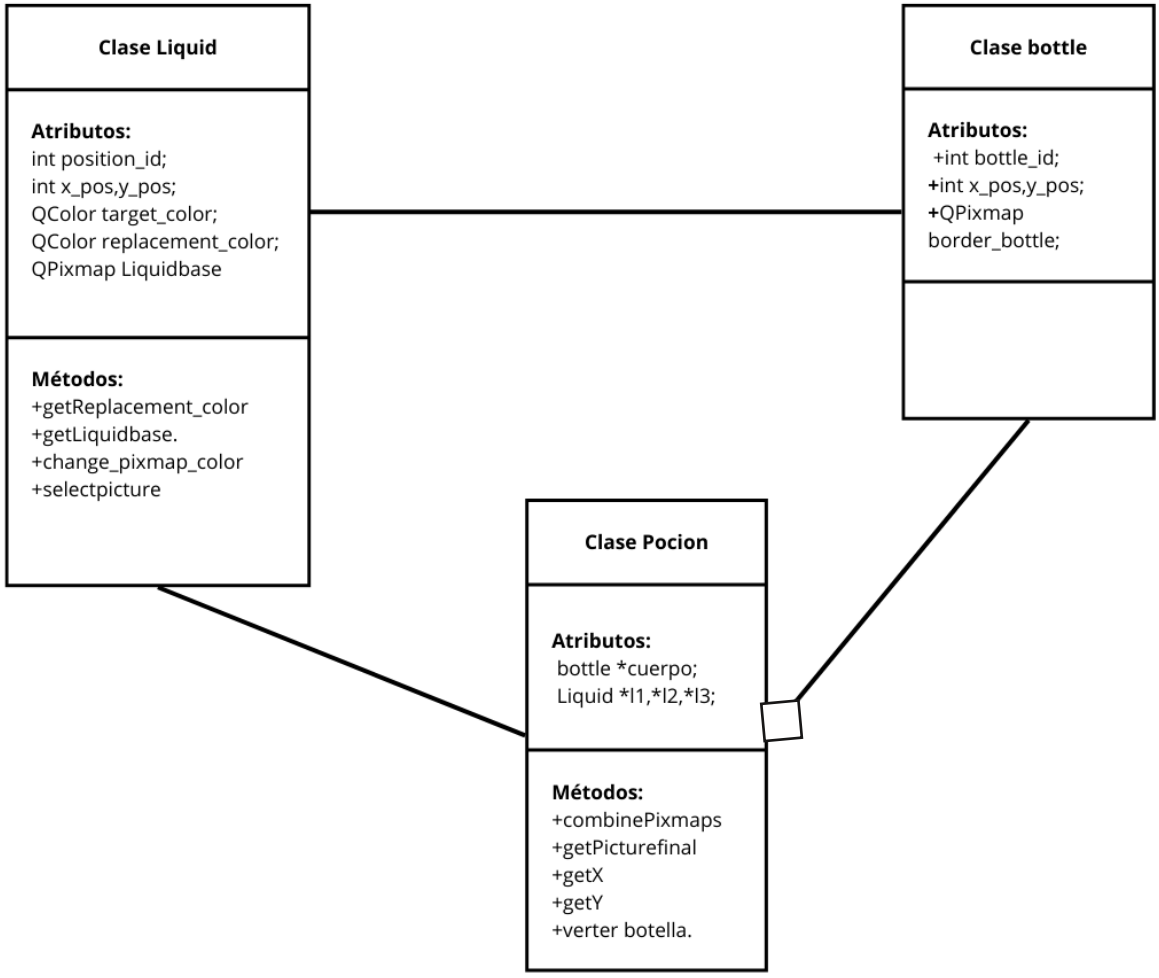
Clase jefe final: Esta clase modela el jefe final del juego, el jefe final puede generar objetos de la clase bombas enemigas cada cierto tiempo.

3.2 - Estructura de datos:

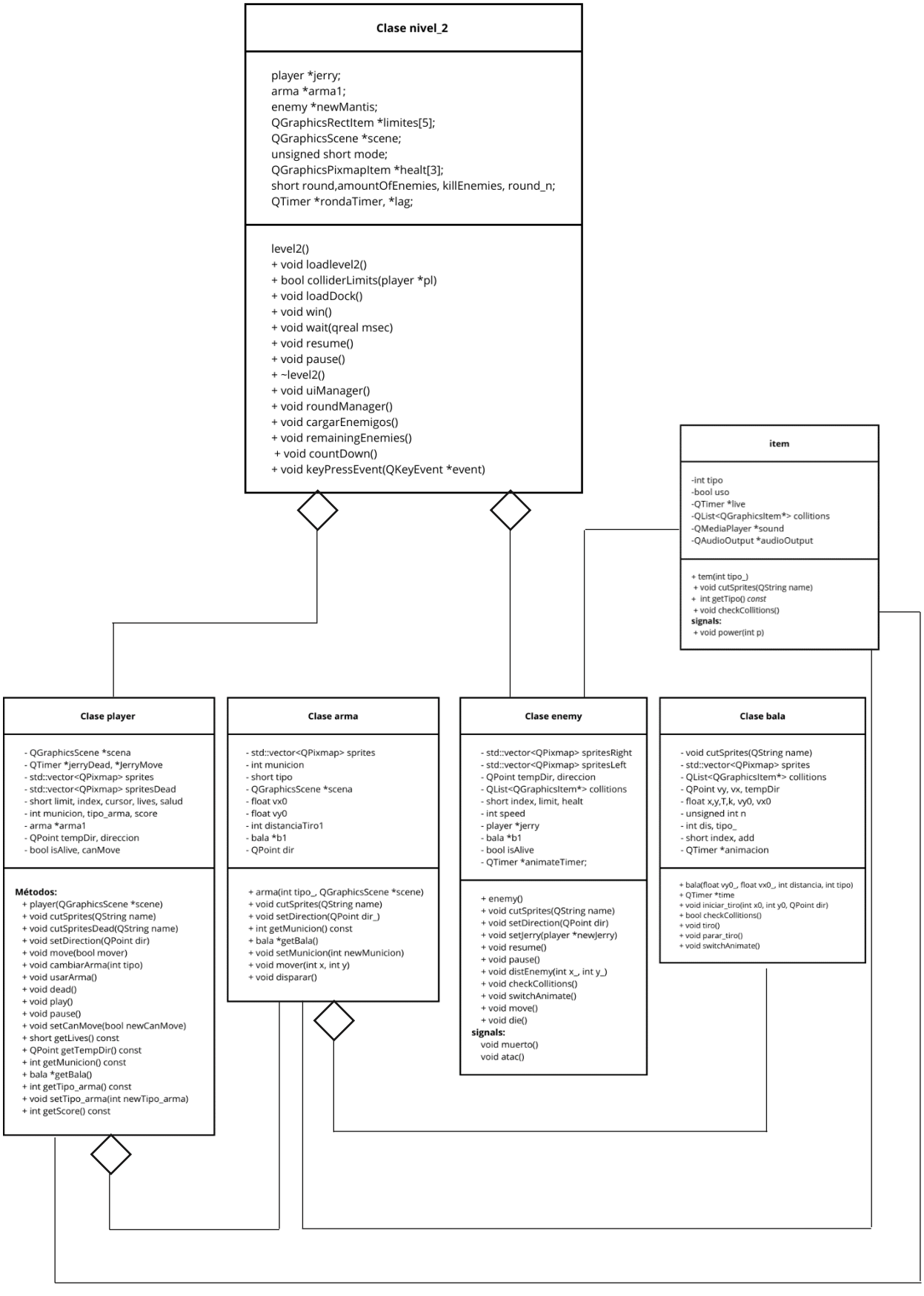
- Se utilizarán vectores para almacenar las distintas texturas de los distintos objetos.
- Habrá un arreglo de tres objetos tipo *bottle* en el primer nivel.
- En el nivel 2 se hace uso de un apuntador a objetos tipo *enemy*, los enemigos son creados y añadidos a la escena, cuando el jugador muere o sube de nivel, se envía una señal que hace que todos los enemigos creados sean eliminados de la memoria.
- Las armas contaran cada una con un puntero a objeto tipo bala, dicho objeto es creado cuando el arma es accionada y destruido cuando colisiona con un elemento cualquiera.
- Un arreglo con las bombas que deja caer el jefe final y que se encuentran activas.

3.3 - Diagramas de Clases:

Clases implementadas en los niveles uno y dos respectivamente.



Universidad de Antioquia
Agudelo Zapata Andrés Felipe
Correa Morales Robinson



3.4 – Tablas de atributos y métodos

level2 (herencia publica de QGraphicsView)		
Identificador	**Acceso**	**Descripción**
Métodos		
`level2()`	Público	Constructor.
`void loadlevel2()`	Público	Se encarga de cargar los elementos del nivel.
`bool colliderLimits(player *pl)`	Público	Permite saber si el jugador colisiona con uno de los delimitadores del mapa.
`void loadDock()`	Público	Carga los elementos gráficos de la UI.
`void win()`	Público	Indica si el nivel ha finalizado.
`void waitDead(qreal msec)`	Público	Hace una pausa cuando el jugador ha muerto.
`void wait(qreal msec)`	Público	Hace una pausa general.
`void resume()`	Público	Reanuda el nivel.
`void pause()`	Público	Detiene el nivel.
`~level2()`	Público	Destructor.
Slots públicos:		
`void uiManager()`	Público	Controla y actualiza el UI.
`void roundManager()`	Público	Controla las randas.
`void cargarEnemigos()`	Público	Genera los enemigos según la ronda.
`void remainingEnemies()`	Público	Verifica la cantidad de enemigos abatidos.
`void generateDrop(int x, int y)`	Público	Genera un item aleatorio.
`void gameFail()`	Público	Reestablece el nivel a los valores iniciales cuando el jugador muere.
`void keyPressEvent(QKeyEvent *event)`	Público	Detecta las entradas por teclado.

player (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`player(QGraphicsScene *scene, QString sprite, short borde, short d_anch, short d_alt, short scala)`	Público	Constructor, requiere de un apuntador a un QGraphicsScene, la ruta de la textura, el tamaño del borde de la textura, el tamaño de la textura (de transparencia) y la escala.
`void setTipo_arma(int newTipo_arma)`	Público	Cambia el arma del jugador (1 para pistola, 2 para lanzagranadas).
`void cutSpritesDead(QString name)`	Público	Recibe la ruta de la textura y recorta las imágenes.
`void setCanMove(bool newCanMove)`	Público	Cambia el estado de movimiento (tue o false).
`void setIsAlive(bool newIsAlive)`	Público	Cambia el estado de vida del jugador (true o false).
`void cutSprites(QString name, short borde, short ancho_transparente, short alto_transparente)`	Público	Recorta la textura del jugador, recibe la ruta de la textura, el borde de la textura, el alto y el ancho de la transparencia.
`void setDirection(QPoint dir)`	Público	Cambia la dirección del jugador.
`void cambiarArma(int tipo)`	Público	Cambia el arma en uso.
`void move(bool mover)`	Público	Mueve el jugador en la escena.
`void usarArma()`	Público	Genera un disparo del arma en uso.
`void pause()`	Público	Detiene al jugador.
`void dead()`	Público	Estado muerto del jugador, impide disparar o moverse.
`void play()`	Público	Reestablece el movimiento al jugador.
`short getLives() const`	Público	Retorna los puntos de vida del jugador.
`QPoint getTempDir() const`	Público	Retorna la dirección del jugador.
`int getMunicion() const`	Público	Retorna la cantidad de munición del arma en uso.
`bala *getBala()`	Público	Retorna un apuntador al objeto tipo bala disparado.
`int getTipo_arma() const`	Público	Retorna el tipo de arma en uso.
`int getScore() const`	Público	Retorna el puntaje del jugador.
`void setScore(int newScore)`	Público	Permite cambiar el puntaje del jugador.
`void setLives(short newLives)`	Público	Permite cambiar las vidas del jugador.
`void restartArma()`	Público	Permite Reestablecer el arma a los valores iniciales.
`void setScale(short newScale)`	Público	Cambia la escala del jugador.
`void setTieneArma(bool newTieneArma)`	Público	Permite establecer la posibilidad de usar un arma.
`~player()`	Público	Destructor.
Slots públicos		

`void switchAnimate()`	Público	Produce la animación al moverse.
`void animateDead()`	Público	Produce la animación cuando el jugador muere.
`void posicionActual()`	Público	Emite la señal movement.
`void addScore()`	Público	Añade nuevo valor al puntaje del jugador.
`void damage()`	Público	Hace que el jugador reciba daño.
`void powerUp(int a)`	Público	Permite que el jugador recoja un item.
Señales:		
`void movement(int x, int y)`	Público	Posición del jugador.
`void fail()`	Público	El jugador ha muerto.

enemy (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`enemy()`	Público	Constructor.
`void cutSprites(QString name)`	Público	Recorta la textura.
`void setDirection(QPoint dir)`	Público	Cambia la dirección del enemigo.
`void setJerry(player *newJerry)`	Público	Cambia el apuntador al jugador.
`void resume()`	Público	Reanuda a la entidad
`void pause()`	Público	Pausa a la entidad.
`void wait(qreal msec)`	Público	Genera un lapso de espera.
Slots públicos		
`void distEnemy(int x_, int y_)`	Público	Calcula la dirección a la cual debe moverse la entidad.
`void checkCollitions()`	Público	Comprueba los objetos con los que colisiona la entidad.
`void switchAnimate()`	Público	Produce la animación de movimiento.
`void move()`	Público	Mueve la entidad hacia el jugador
`void die()`	Público	Produce la muerte de la entidad.
Señales		
`void muerto()`	Público	La entidad ha muerto.
`void atac()`	Público	La entidad ha colisionado con un objeto player.
`void Drop(int x, int y)`	Público	La entidad a generado un item al morir.

arma (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`arma(int tipo_, QGraphicsScene *scene)`	Público	Constructor, recibe el tipo de arma (1: Pistola, 2: lanzagranadas) y un apuntador a un QGraphicsScene.
`void cutSprites(QString name)`	Público	Recorta y carga las texturas del objeto.
`void setDirection(QPoint dir_)`	Público	Cambia la dirección a la que apunta el arma.
`int getMunicion() const`	Público	Retorna la cantidad de balas del arma.
`bala *getBala()`	Público	Retorna un apuntador a un objeto tipo bala.
`void setMunicion(int newMunicion)`	Público	Cambia la munición del arma.
Slots públicos		
`void mover(int x, int y)`	Público	Recibe la posición del jugador y mueve el arma.
`void disparar()`	Público	Crea un objeto tipo bala y lo lanza.

item (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`item(int tipo_)`	Público	Constructor, recibe el tipo de item (1: poción de vida, 2: munición).
`void cutSprites(QString name)`	Público	Recorta y carga las texturas del item, recibe la ruta de la textura.
`int getTipo() const`	Público	Retorna el tipo de item.
Slots públicos		
`void checkCollitions()`	Público	Comprueba si el jugador ha colisionado con la entidad.
Señales		
`void power(int p)`	Público	Señal con el tipo de item.

botlle (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`bottle(int bottle_id,int x_pos,int y_pos)`	Público	Constructor, recibe un entero como id y la posición x, y del objeto.
`QPixmap getBorder_bottle() const`	Público	Retorna la textura del objeto

liquid (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`Liquid(const QColor &target_color, const QColor &replacement_color, int position_id, int x, int y)`	Público	Constructor, recibe el color del líquido, la posición del liquido en la botella y su posición x, y.
`QPixmap getLiquidbase() const`	Público	Retorna la textura de la forma del líquido.
`bool isitvoid()`	Público	Retorna true si el color del líquido es ninguno.
`QColor getReplacement_color() const`	Público	Retorna la textura del líquido con color.
`void change_pixmap_color(QPixmap &pixmap)`	Privado	Reemplaza el color del líquido.
`QPixmap selectpicture(int id)`	Privado	Retorna la forma del líquido.

pp (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`pp(int id,int x, int y)`	Público	Constructor, recibe un numero característico(ID), y las coordenadas que ocupa el objeto pp
`QPixmap getPicturefinal() const`	Público	Retorna la imagen final que aparece en el mainwindow
`int getX()`	Público	Retorna la coordenada X del pp
`void onclik1()`	Público	función que realiza las funciones que ocurren en el primer click
`void onclik2()`	Público	función que realiza las funciones que ocurren en el segundo click
`void bottledance1()`	Público	Función que simula una vibración del objeto pp de derecha a Izquierda
`void bottledance2()`	Público	función que simula una vibración del objeto pp de izquierda a derecha
`int getY()`	Público	Retorna el valor en Y de la ubicación del pp
`bool searchvoids()`	Público	Revisa si el objeto pose un objeto Liquid que sea vacío
`int getId() const`	Público	Retorna el Id del objeto
`void fillspace(Liquid* newL)`	Público	Remplaza el primer vacío de la botella por el objeto newL
`Liquid* emptyspace()`	Público	Retorna el primer objeto Liquid no vacío, y remplaza esa posición por un vacío
`void setObjective(int newObjective)`	Público	Remplaza el valor de objetive por el ingresado
`void updatepixmap()`	Público	Actualiza la imagen del Mainwindow

level1 (herencia publica de QObject y QGraphicsPixmapItem)		
Identificador	**Acceso**	**Descripción**
Métodos		
`Level1()`	Público	Constructor
`pp* searchpp(int id);`	Público	Retorna un apuntador al objeto pp que corresponde al id ingresado
`void secondanimation(int emisor,pp *receptor);`	Público	Función que invoca las funciones de la clase pp que manejan el segundo estado de movimiento
`QRectF colisiorect(pp* base);`	Público	Retorna un Qrectf con el rectángulo de los bordes de la imagen.
Slots públicos		

`void channgeLiquid();`	Público	Actualiza las botellas después de que el líquido se mueva de una a otra.
`void parar_tiro()`	Público	Detiene la simulación.
`void switchAnimate()`	Público	Produce la animación de explosión.

3.5 – Cambios realizados

- La primera versión de este entregable tenía varias falencias, algunas debido al desconocimiento del uso de Qt.
- Los diagramas y descripciones de clase se encontraban en una versión muy simplificada debido al desconocimiento de los atributos y métodos que se iban a necesitar en la versión fina, aunque lo planeado en un principio para las clases de los niveles 1 y 2 si fue implementado en la versión final.
- Se añadieron las tablas con las descripciones de los métodos de cada clase implementada.
- Se corrigió la descripción del nivel 1.

El nivel 3 no fue implementado por temas de tiempo y requisitos

No se realizaron cambios a la estructura del juego, cambios de diseño o enfoques de jugabilidad en los 2 niveles implementados.