

**PROYECTO FINAL**

**ArmagHEADon**

HAROLD JOSÉ URQUIJO DURÁN

CARLOS ANDRÉS MESA ROLDAN

**Docente**

**Augusto Salazar Jiménez**

Departamento de Ingeniera electrónica y Telecomunicaciones

Universidad de Antioquia

Medellín

2023

**Contexto**

En un universo alterno, existe una raza de seres, denominados los bobbleheads, son seres muy amigables y pacíficos. Su especie se caracteriza principalmente por su estructura corporal, todo su cuerpo se compone únicamente por una cabeza gigante, que se desplaza en medio del espacio, venciendo a voluntad la fuerza de gravedad del planeta en el que se encuentre. Son seres curiosos al extremo, tanto así que es de su agrado conocer nuevos lugares y seres vivos alrededor del universo.

En cierto momento caen en la cuenta de que en su planeta natal (Bobbleheadlandia) está sobre poblado, así que toman la decisión de ir a buscar más planetas en donde los bobbleheads puedan vivir tranquilamente, por tal razón los bobbleheads se convierten en una raza experta en la colonización de planetas. En cierto momento deciden invadir el planeta tierra, lo que no saben es que el planeta está preparado, dotado con helicpteros, aviones y naves para dar de baja a los bobbleheads.

Es allí donde se da inicio a una fantástica aventura, la cual invita a proteger por encima de cualquier cosa la integridad de la raza, ante las inclemencias del planeta en el que se encuentren, la protección de la raza es la prioridad. Las defensas del planeta tierra se harán mucho más agresivas, conforme se avanza al interior del planeta.

**Generalidad del juego**

Una ventana en 2D se presenta, donde a primera vista y durante el desarrollo de todo el juego. El jugador controla una cabeza voladora que se desplaza de izquierda a derecha y de arriba a abajo con el objetivo de dar de baja a los enemigos que se presentan conforme pasa el tiempo. Un primer acercamiento se da en la ***figura 1.***

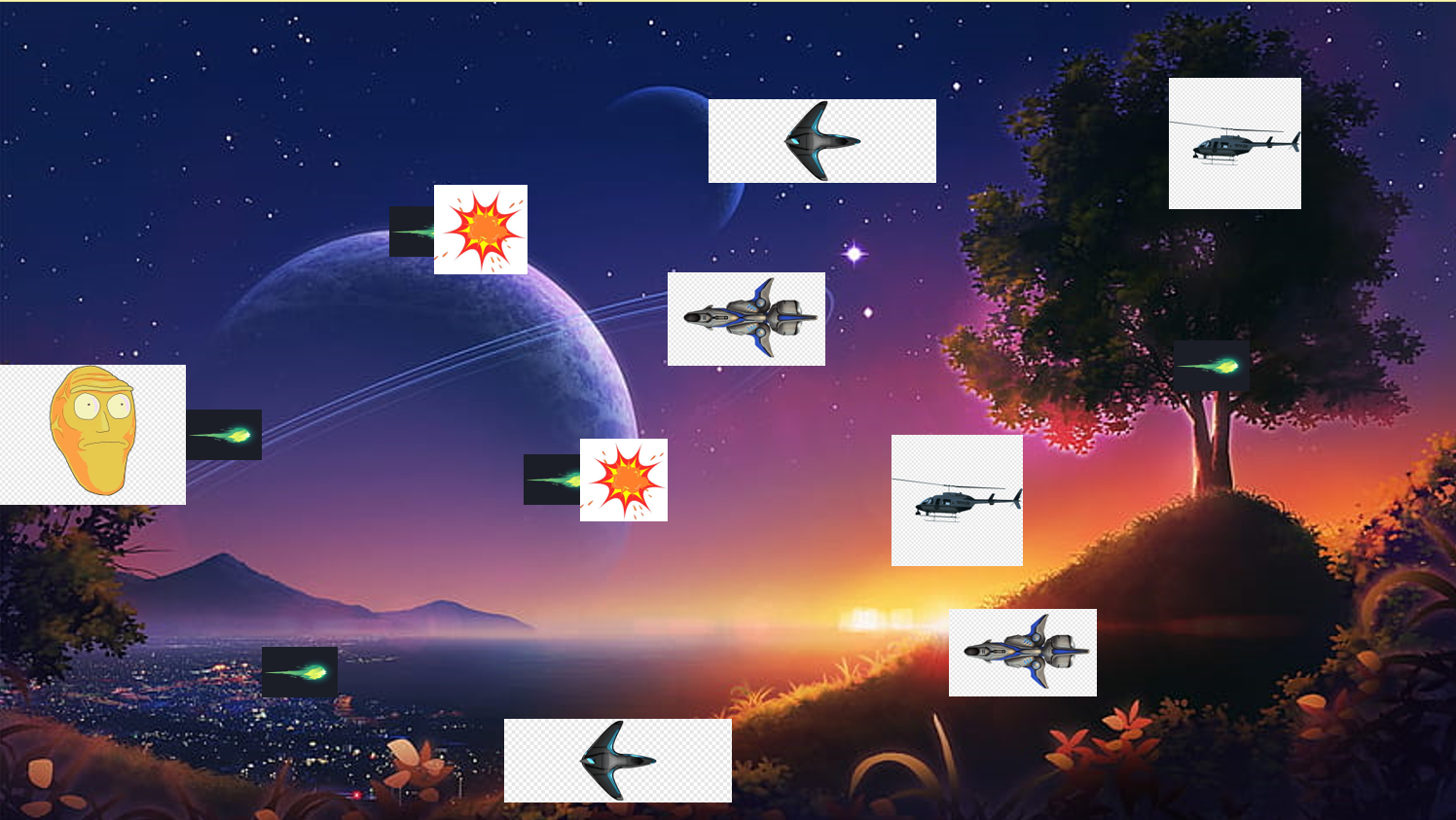


Figure 1: Primer acercamiento.

La misión consiste en destruir el máximo de enemigos posibles, para así avanzar entre las diferentes dificultades que el juego ofrecerá (un mínimo 3 niveles de dicultad). El jugador cuenta con proyectiles ilimitados y 3 vidas. Al ser tocado por uno de los aviones enemigos, el jugador pierde una vida, al ser tocado por 3 aviones el jugador perderá totalmente el juego y sus avances.

A medida que el jugador va acumulando puntos, los enemigos se moverán más rápido y será más dificultoso destruirlos o esquivarlos. El jugador tiene que tener varios elementos en cuenta, como, por ejemplo, la resistencia del aire que constantemente lo va a ir llevando hacia atrás, así que debe estar acelerando constantemente, pero no tanto, como para estar tan cerca del enemigo. La gravedad ya que esta lo va a atraer hacia abajo, por tanto, el jugador debe estar constantemente atento de no caer ni dejarse arrastrar por el viento.

El juego cuenta con 2 modos de juego, uno individual y otro multijugador, en el individual el jugador cuenta con 3 vidas, y cada vida se irá restando como se explicó anteriormente. En cambio, en el modo multijugador, se cuenta con una sola vida, primero empieza jugando uno de los dos jugadores y tiene que alcanzar el máximo puntaje, luego de que el primer jugador pierda, pasaría el 2 jugador que también cuenta solo con una vida. Al final el que tenga más puntuación será el ganador. El jugador al chocar va a tener un tiempo de inmunidad para no chocar dos veces en un tiempo muy corto, con el objetivo de que se reponga y regrese al juego.

**Físicas en el juego**

Las naves enemigas harán uso de una serie de movimientos que tienen como finalidad poner en aprietos al jugador. Dichos movimientos tendrán como finalidad realizar acciones evasivas y ataques sorpresa. Se modelarán de la siguiente manera

* **Movimientos circulares**.
* **Movimiento de caída libre**.
* **Movimiento Sinusoidal.**

El jugador por su parte hará uso de un movimiento rectilíneo uniformemente acelerado, el cual le permitirá elegir a que distancia encontrarse a la hora de atacar o esquivar al enemigo. Se contarán con las siguientes habilidades físicas:

* **Colisiones:** Colisión, cuando el jugador choca con uno de los aviones enemigos.
* **Movimiento rectilíneo uniformemente acelerado** (MRUA). A diferencia de los aviones y naves enemigas, se contará con una posibilidad de acelerar o no, cambiar de velocidad a gusto.

Se tendrán las siguientes consideraciones y modos de juego, primero se va a tener el modo 1 jugador, donde podremos escoger entre nueva partida y cargar partida, en la opción nueva partida escogemos el nombre y el tipo de personaje que queremos, que van a estar las opciones de las dos cabezas gigantes. Luego tendremos el modo multijugador solo tendremos la opción de iniciar. También tenemos la opción de ayuda donde se van a mostrar las instrucciones del juego, y también tendremos la opción de salir del juego.

**Análisis 2**

Realizando una serie de iteraciones y a partir de la experiencia dada por el laboratorio 5 del curso, se prevé la implementación de las siguientes clases, con sus algunos métodos que en un primer momento se tendrán en cuenta, pero conforme se desarrolle el juego se irán considerando.

**Clase mainwindow:** Se hace costumbre que la función de entra main sea corta y en este tipo de programas aún más, tanto así que solo tendrá una función, crear una instancia de la clase mainwindow, quizá, la clase más importante del juego. La clase mainwindow heredará una serie de métodos de QMainWindow. En otras palabras, en esta clase se gestionará el uso de la interfaz gráfica.

Se mencionarán grosso modo los métodos a utilizar de esta clase y una brave explicación acerca de lo que harán y su impacto en el juego.

Métodos públicos:

* setMenu(): De entrada, será necesario desplegar el menú y las diferentes opciones.
* Guardar\_nuevo\_jugador(): Con el objetivo de registrar y guardar nuevos jugadores.
* Cargar\_partida\_1jugador(): Cargar partidas anteriores y registros.
* nivel\_1(): Lanzar el nivel de entrada y a partir de este gestionar los siguientes niveles.
* keyPressEvent: Esté método se encargar de percibir eventos durante el juego, por ejemplo, tendrá como parámetro las teclas de desplazamiento, de una manera constante.
* Guardar\_progerso(): Encargado de guardar el proceso del juego.
* void multi\_jugador(): Una vez se despliegue el menú, tendrá la tarea de tomar las medidas necesarias en caso de que el juego se de entre más de un jugador.

Métodos privados:

Una serie de punteros eran necesarios para modelas con éxito el juego, estos serán:

* \*ui: Sera fundamental apuntar a la interfaz de usuario y a partir de allí relacionar una serie de elementos con los botones y demás posibilidades que tiene Qt.

**Clase jugador:**

En dicho objeto se tendrán una serie de atributos públicos y privados que tendrán como finalidad modelar ciertos aspectos, tales como la posición en X y en Y, las velocidades y

**Métodos:** A grandes rasgos se tendrá los siguientes métodos: - *Movimientos*, donde se añaden las diferentes imágenes del personaje, en una circunstancia dada.

- *Tiempo de inmunidad*, contador de tiempo que añade cierta inmunidad una vez se ha sido golpeado por el enemigo.

**Clase enemigo:**

En dicho objeto se tendrán una serie de atributos públicos y privados que tendrán como finalidad modelar ciertos aspectos, tales como la posición en X y en Y, las velocidades y demás. Además de que tendrá un plus en cuanto a sus movimientos, pues el enemigo hará uso de distintos movimientos ajenos para el jugador.

**Métodos:** Se tendrán en cuenta ciertos movimientos modelados físicamente, los cuales generan la dificultad al jugador. Además, se cree que se heredará de la clase jugador cierto comportamientos, pues a fin de cuentas se tienen naves espaciales que deben mostrar su posición constantemente.

- Movimientos, aquellos movimientos físicos, tales como, circular, caída libre, rectilíneo uniforme y aquello que modelará las colisiones.

**Clase Fondo:**

Un fondo en particular será el encargado de identificar el nivel. Cada nivel se abrirá paso dado un condicional, el cual evaluará

**Clase Proyectil**:

Movimiento de la bala. Por tanto, podrá heredar de la clase jugador su respectiva posición en X y Y. Esta clase estará presente tanto para el jugador como para el enemigo, por tanto, en un primer momento se va a desear que esta clase sea heredada.

* **Métodos:** A grandes rasgos se tendrá el siguiente método: *Movimientos*, para dar a conocer el movimiento de la bala en el transcurso de su recorrido y a qué velocidad viajará.

**Clase Puntaje:**

Y una serie de atributos relacionados al aumento de puntuación. Puntuación.

A grandes rasgos se tendrá los siguientes métodos: Aumentar puntos para jugador 1 y 2, vidas disponibles.

**Clase Juego:**

Menú, niveles. Escenas, disparos, fin del juego.

Se descartan una serie de elementos que son fácilmente remplazables con las diferentes posibilidades que brinda la programación orientada a objetos, por medio de la herencia de clases.

Lista de necesidades o retos:

* Indagar en las diferentes librerías de QT, con el fin de encontrar aquella que se ajuste a la integración de efectos auditivos que mejoren la interacción y experiencia del juego.
* Implementar colisionador con el fin de generar colisiones tanto para enemigos como para jugadores.