# Definición del Problema

Realice un videojuego de plataformas que permita al jugador controlar un personaje que avance enfrentándose con múltiples obstáculos y enemigos, al estilo “Súper Mario Bros.” o “Contra”. El personaje tendrá la opción de adquirir ciertos ítems que actúen como mejoras para sus factores de ataque y/o defensa. Deberán existir al menos 4 tipos de enemigos con diferentes características y comportamientos. Se le permitirá al jugador utilizar distintas técnicas/estrategias para atacar inteligentemente a sus enemigos.

Tabla de contenido

[Definición del Problema 1](#_Toc322334577)

[Análisis 1](#_Toc322334578)

[Diseño 2](#_Toc322334579)

[Código Fuente 3](#_Toc322334580)

[Condiciones de Error 4](#_Toc322334581)

[Ejemplos de Uso 5](#_Toc322334582)

[Notas al Profesor 6](#_Toc322334583)

# Análisis

El primero problema presentado en este juego es hacer que el jugador se mueva junto con la pantalla.

El jugador principal deberá tener varios ángulos de disparos para poder disparar a sus enemigos con mayor precisión, hay conseguir estos ángulos con la librería gráfica utilizada y determinar el sprite a mostrar de acuerdo con el ángulo actual del jugador. El jugador deberá saltar y volver a bajar con la gravedad. Se presenta el problema de que cuando se tira al suelo el alto del sprite es el mismo que el alto del sprite cuando el jugador está parado, por lo tanto, las balas de los enemigos colisionarían con el jugador como si estuviera parado. Para mover al jugador se utilizará el teclado.

Los disparos se deben de controlar con el tiempo ya que los caracteres deben de tener un intervalo de espera entre un disparo y otro.

Un problema grande que se presenta son los enemigos. A estos se les debe de controlar con el tiempo. También se debe de tener un límite de enemigos en juego ya que si aparecen muchos al jugador se le haría difícil ganar. Hay que determinar los movimientos de los enemigos con respecto al jugador; que tanto se deben de acercar para disparar, cuando tengan que perseguir al jugador, etc.

El juego deberá presentar obstáculos como agujeros en el piso, lava, plataformas en las que haya que saltar, etc. Para esto se debe de crear una función que detecte las colisiones para saber donde colisiona para determinar qué se debe hacer.

Para crear un dinamismo positivo y atractivo del juego se necesita agregarle la participación de más de un jugador, para que el entretenimiento pueda realizarse de una manera más social. Adaptar otro jugador al juego puede realizarse de diferentes maneras, teniendo en cuenta de que hay que utilizar diferentes sprites para cada jugador. Solamente se tendría que buscar la forma mas eficiente de manejarse los jugadores ya sea utilizando listas enlazadas o un arreglo.

Otra herramienta muy importante a utilizar en el videojuego, que también incentiva al jugador a tener un modo de juego más agradable, es el uso de joysticks o controles. Estos permitirán tener varios jugadores en la pantalla sin tener que utilizar los métodos tradicionales como los son el teclado y el mouse.

# Diseño

La función InicializarJuego inicializa todas las variables del juego y las funciones de Allegro.

La función FinalizarJuego remueve todas las funciones de la librería Allegro y también destruye todos los sprites, sonidos y fuentes cargados en el juego para no dejar espacio en memoria innecesario.

ActualizarJuego llama a todas las funciones de actualizar. También actualiza los proyectiles. Con física determina las posiciones de X y Y del proyectil. Determina si el proyectil actual es del jugador y colisiona con otro carácter que no sea el mismo. También determina si el proyectil actual es de un enemigo y colisiona solamente con el jugador. Cuando el proyectil ya no es visible en la pantalla se destruye el nodo.

Para implementar los ángulos de disparo se diseñó que el jugador tenga un total 24 ángulos de disparo, teniendo una diferencia de 15 grados entre cada ángulo. Con los sprites ya creados, se utiliza la función AnguloDeDisparo para determinar cual es el sprite que se va a mostrar. Como sólo hay sprites cada 15 grados se divide el ángulo de disparo entre una constante que indica la diferencia entre ángulos (de 15) para que se le asigne la posición del sprite a mostrar en el sprite sheet.

Para la gravedad se utiliza la función CaidaConGravedad. Aquí se determina si el caracter colisiona con una plataforma, si lo hace, la posición del jugador permanecerá encima de la plataforma, de lo contrario se utiliza una formula de física para determinar la posición a la cual el carácter deberá descender, con respecto al tiempo.

Con la función Disparar se inicializa el proyectil del carácter al que se le asignó la función. Para llamar a esta función se debe de cumplir un tiempo mínimo para cada carácter, por ejemplo, para que el jugador pueda disparar se debe de cumplir un tiempo mínimo entre un disparo y otro de 0.20 segundoss. En esta función se utiliza el seno y coseno del ángulo para determinar las velocidades en X y Y del proyectil. Se utiliza como referencia el ancho y el alto del sprite para determinar las posiciones de X y Y del proyectil.

Se utiliza la función ActualizarJugador para actualizar todo lo relacionado con el jugador principal. Luego de determinar el ángulo del mouse con respecto al jugador (con las funcione que nos ofrece Allegro de ángulos) se debe de hacer una conversión de ángulos ya que los de Allegro son diferentes a la escala de 360 de grados, la que nosotros conocemos. Con las funciones que nos ofrece Allegro del teclado se determina cuando se presiona la tecla A, W o D para mover izquierda, saltar o mover derecha, respectivamente.

Los enemigos se actualizan en la función Villanos. Para saber cuando se debe crear un enemigo se utiliza el tiempo. Cada tipo de enemigo tiene su tiempo aparte, independiente uno de otro. Para crear un enemigo se determina si el tiempo desde que se creó el último enemigo del tipo actual es mayor al requerido para crear otro y si el máximo de enemigos en juego todavía no se ha alcanzado. Luego de crear a los enemigos con un ciclo se actualizan cada uno de ellos. Se determina si la posición del enemigo está fuera del área visible y de ser así el enemigo se va a desplazar hacia el jugador. Cuando el enemigo se encuentra dentro del rango de disparo, comienza a disparar. Cuando el jugador camina hacia la derecha y el enemigo está disparando el enemigo también comienza a caminar.

La función ActualizaNave fue creada ya que la nave se comporta de manera diferente a los demás enemigos. Para hacer que la nave descienda se utiliza la física y luego de que la nave se ve completamente en la pantalla se le aplica una aceleración contraria y de igual magnitud a la aceleración de la gravedad del juego. Cuando la nave llega al limite izquierdo o derecho se la pantalla se le aplica una velocidad contraria a la que llevaba para que comience a moverse hacia a lado contrario.

La función CrearVillano se utiliza para crear cualquier tipo de enemigos. Esta se llama cuanto el tiempo de crear un enemigo específico es mayor al requerido. Cuando se cumple dicho tiempo se llama la función y con un switch determina cual enemigo es el que se va a crear.

Con la función ActualizarFondo, como su nombre lo dice, actualiza el fondo de acuerdo al jugador va avanzado. Cuando la posición de un nodo del fondo es menor que la posición actual del buffer dicho nodo se elimina.

ActualizarCarretera elimina el nodo de la carretra cuya posición del fondo es menor a la posición actual del buffer.

RestarHP es la función la cual se ejecuta cuando un proyectil colisiona con algún carácter. Este le resta al HP del carácter el factor de ataque del proyectil, pero le suma el factor defensa del carácter. Cuando el HP del carácter es menor que cero se pregunta que tipo de carácter es para determinar que se va a hacer ya que cada tipo de carácter tiene una acción diferente al llegar su HP a cero.

La función ActualizarItems se encarga de hacer aparecer un ítem cuando el tiempo requerido entre un ítem y otro es mayor al indicado. También se encarga de darle movimiento al ítem. Utiliza la física para hacer que el ítem se mueva de arriba abajo. Para determinar cuál ítem es el que va a aparecer utiliza un random que genera números de 0 a 9 y con condiciones se le da más prioridad a los ítems que son menos efectivos para que el juego quede equilibrado.

# Código Fuente

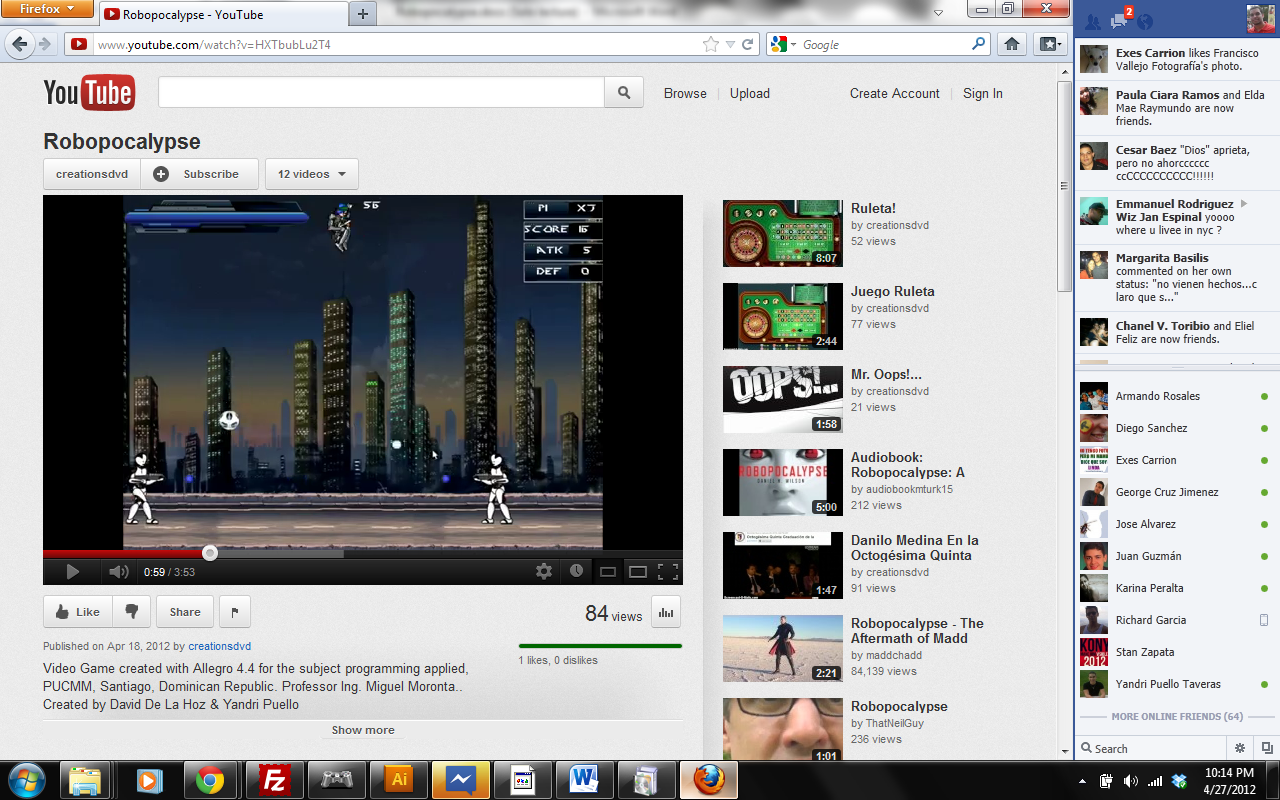
Adjunto a este documento.

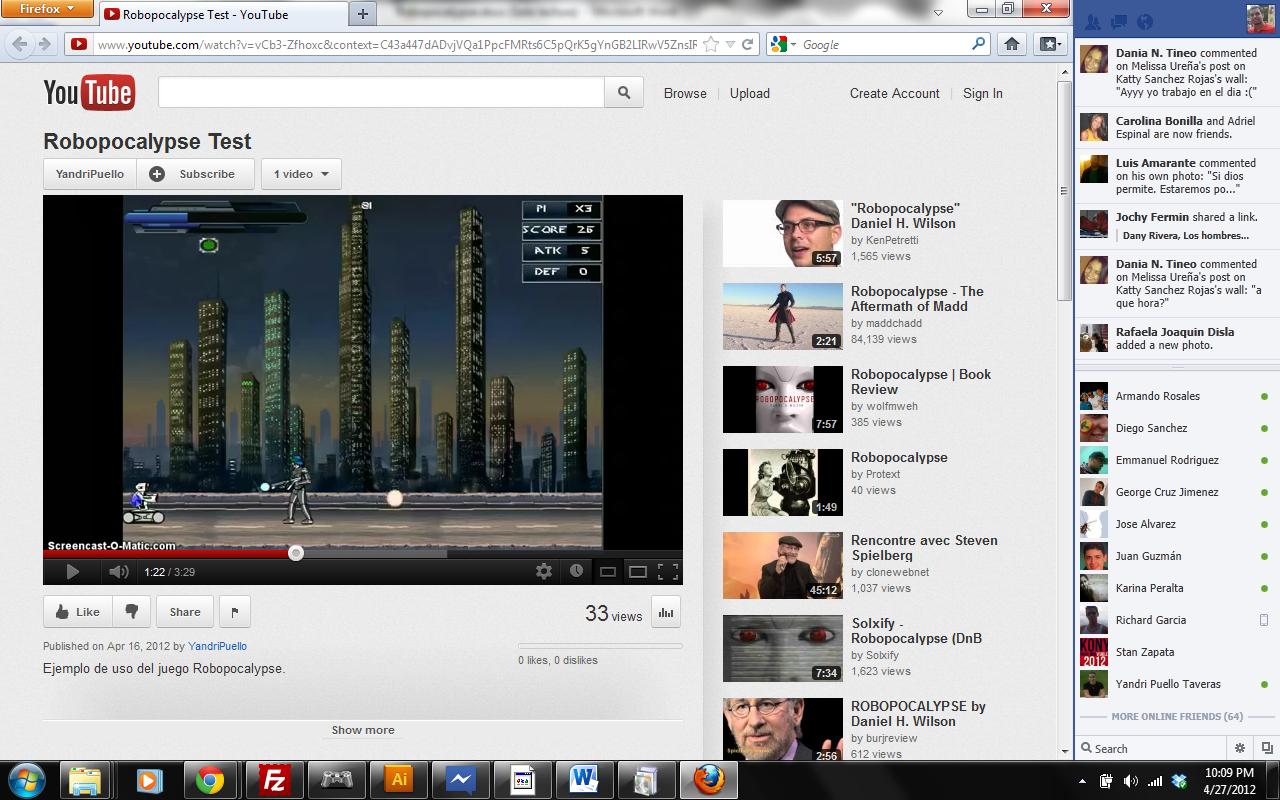
# Condiciones de Error

El programa no contiene errores.

# Ejemplos de Uso

Enlace a video a YouTube con un ejemplo de uso:

[](http://www.youtube.com/watch?v=HXTbubLu2T4)  
<http://www.youtube.com/watch?v=HXTbubLu2T4>

[](http://www.youtube.com/watch?v=vCb3-Zfhoxc)  
<http://www.youtube.com/watch?v=vCb3-Zfhoxc>

# Notas al Profesor