INFORME PROYECTO FINAL

JEAN CARLOS PARRA SERRANO

UNIVERSIDAD DE ANTIOQUIA

INFORMÁTICA 2

DICIEMBRE DEL 2025

MEDELLÍN, ANTIOQUIA

**Introducción**

La Programación Orientada a Objetos (POO) nos sirve para simular, programar, realizar proyectos que pueden ser usados como trabajo, entretenimiento, desarrollo, avance, implementación y muchas cosas más. La base de este documento sirve para ser la guía del programa desarrollado a lo largo de un mes de trabajo. Este proyecto es un juego que implementa conocimientos del lenguaje de programación C++. Este juego consiste en tres niveles, cada uno con su grado de dificultad, su análisis, implementación y desarrollo.

Con base a la temática planteada de algún hecho histórico relevante. El desarrollo se planteó en base a la importancia del pasado de la especie humano. Todo lo que tiene hoy la especie humana es gracias a su historia, así que sin historia nunca desarrollaríamos todas las cosas que hacemos hoy: tecnología, medicina, construcción, desarrollo y demás.

En el pasado la especie humana tuvo que pasar por muchas dificultades: escasez de comida, eventos catastróficos, pandemias, guerras y demás. Con el objetivo de desarrollar un programa que recuerde la importancia de la época prehistórica se plantea un juego llamado “The Caverman”, el cual consiste en tres niveles. El primer nivel consiste en la fase de la historia humana donde gran parte de los volcanes, estaban activos, así pues el nivel 1 llamado “La gran erupción”, un caverman sale de su cueva y se encuentra un mundo en caos, caos dado por la erupción volcánica; el objetivo es recoger frutas simulando un evento posible de nuestros antepasados ya que necesitaban sobrevivir a toda costa. El segundo nivel consiste en la caza de un mamut, ya que estos animales fueron importantes en el pasado, debido a su gran fuerza, tamaño y peso. Como todo juego es importante tener algún componente de ciencia ficción, así pues, el nivel tres cubre este hecho; este nivel tiene como personajes principales un hombre nieve malvado, que vive en un ambiente frío que a medida que se desarrolla el juego, adquiere habilidades; el objetivo es que el “caverman” lo derrote gracias a sus herramientas.

**Metodología:**

Para el desarrollo de este, se tuvo la exigencia de desarrollarlo en menos de un mes. Gracias a la ayuda de clases, diapositivas, vídeos, estudio y tiempo. La metodología fue dividir el desarrollo de este en etapas:

***Creación de la etapa lógica:***

Se plantearon las clases necesarias, sus .h y .cpp. Estas clases contendrían la lógica del juego:

-Movimientos

-Posiciones

-Actualizaciones

-Agente inteligente

-Funcionalidades del juego

-Demás lógica

Se hizo un cronograma para el desarrollo de esta. Las primeras dos semanas, se realizó el análisis previo, por ejemplo, de qué sería el juego, cómo se iba a desarrollar, las limitaciones que se debían de tener en cuenta para evitar más complicaciones. La siguiente semana se llevó el análisis del boceta de diagrama de clases. Este diagrama fue la guía para llevar a cabo la parte lógica del programa.

Para llevar organizado este desarrollo se crearon ramas en GitHub para el desarrollo de las distintas etapas. Por ejemplo, cada nivel se iba desarrollando y se montaba a la rama específica, esto para no llevar suciedad en la rama “main”, la principal.

Ramas creadas:

Interfaz de usuario gráfica, Aplicación, Tabla

El contenido generado por IA puede ser incorrecto.

* **Diseño y arquitectura del sistema**

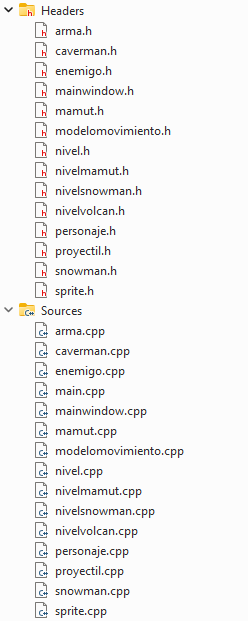
El desarrollo y la elección de cada clase, con su comportamiento da la clave para el desarrollo del programa. Gracias a distintos comportamientos que dan las clases en C++, como por ejemplo la herencia entre clase, este comportamiento simplifica el código y hace más fácil todo.

La clase nivel da herencia a los tres niveles que tienen su comportamiento propio. La clase personaje se relaciona con los niveles, y esta clase también da herencia a clases como caverman o enemigos que contienen otras clases. El manejo de los proyectiles crea cada proyectil que será usado en cada nivel específico. Los movimientos son importantes para que el juego sea dinámico.

* **Desarrollo e implementación**

La lógica interna es la base del programa, pues esta lógica controla posiciones, por ejemplo, las posiciones del caverman, del mamut, del jefe “snowman”, de los proyectiles que se usa en los niveles.

Se decidió usar vector para simplificar la cantidad de objetos que son creados, posteriormente destruidos y demás. Gracias a estos tipos de contenedores, se puede recorrerlos de forma “sencilla” y borrar objetos con métodos propios de estos.



Las clases son la base de toda la ejecución. En cada nivel, se crea una clase que controlará todo el funcionamiento de este, posiciones, habilidades y demás. Para el agente inteligente, en la clase snowman se controla el agente inteligente con cada acción que puede realizar el snowman.

* **Procedimientos de prueba**

A medida que se implementaban cosas, se iba ejecutando el programa para cuadrar: posiciones de objetos, labels, timers, revisar cómo se comporta la lógica en la interfaz gráfica de UI en QtCreator.

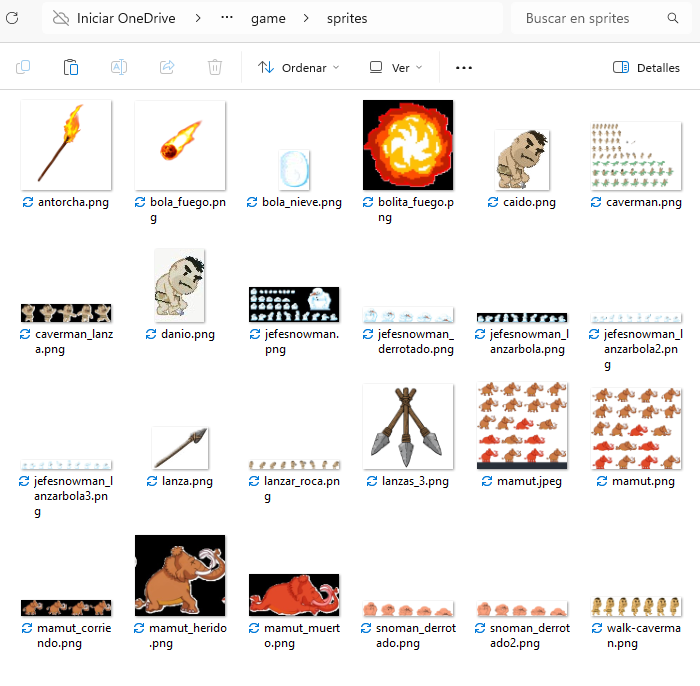
Una de las partes más complicadas del desarrollo de un juego es la implementación de la parte lógica en la parte gráfica. Esta parte lleva estudio de implementación de labels, timers, y sobre todo el manejo de sprites, manejo de la creación de objetos, vectores y demás.

Para cada nivel se realizaban pruebas como:

En los niveles se actualizarán las barras de vida, se actualizan los tiempos, la generación de los proyectiles y las eliminaciones de estos. Los movimientos de los personajes.

Se agregan los elementos que se usaron para generar estos programas:

Sprites:



Recursos juego:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

* **Resultados y discusión**

El funcionamiento logrado fue el siguiente:

Creación de un menú principal donde con darle “click” en el recuadro del nivel a jugar, se lleva a cada nivel para llevarlo a jugar:

*Nivel 1*

En el nivel 1, se puso un background de un volcán en erupción en el lado derecho, en la izquierda la cueva desde donde va el “caverman” a recoger la fruta que está en la misma línea y del volcán. La idea es evitar que las bolas de fuego que caen en todas las direcciones en el nivel y con movimiento parabólico, ya que están restan vida al caverman. El caverman debe de volver a la cueva para completar el nivel.

Se implementó el timer de 40 segundos para que el personaje llegue a la meta antes del tiempo. En caso de que el humano pierda todas las vidas o no llegue a la cueva con las frutas antes del tiempo, el nivel mostrará una imagen de “game over” y lo llevará al programa principal. El nivel también tiene un sonido de fondo para ambientarlo.



*Nivel 2*

En el nivel 2, se puso un background de un ambiente de selva, donde imita que el caverman está en el bosque. El mamut se coloca sobre la posición de la tierra para que tenga una ruta establecida y la recorra a una velocidad específica. El mamut irá de izquierda a derecha y en caso de colisionar con el caverman, le bajará vida a este. En el nivel se implementó scroll, ya que el nivel contiene sogas que se mueve con cierto movimiento pendular sin gravedad para que no se quede quieta esta. El caverman puede sujetarse y balancearse en la soga para evitar daño del mamut.

En el nivel caerán lanzar de a tres que servirán de municiones para el personaje principal y las use contra el mamut. Las lanzas que agarre tendrán un movimiento parabólico que afectarán la vida del mamut. El mamut tiene cierta vida que a medida que colisiona con las lanzas del jugador, le restan vida. El nivel también está ambientado con música de bosque.

El nivel tendrá fin cuando la vida del caverman llegue a cero o el mamut muera. El mamut muestra daño cuando la lanza lo impacta, al igual que el hombre muestra un sprite de daño cuando el mamut lo intercepta. Además en el nivel, se muestra el timer del nivel que se va actualizando, la barra de vida del caverman y la barra de vida del mamut, el caverman, el mamut, las municiones de lanzas que tiene el personaje principal.



*Nivel 3*

El nivel 3 consiste en derrotar un enemigo, un jefe snowman que adquiere habilidades a medida que la temperatura baja. En este nivel se implementa el agente inteligente que hará al personaje snowman más poderoso. La idea es que el caverman tiene ciertas herramientas que le ayudarán a derrocar este personaje. El snowman tiene tres fases, donde la primera está estático y lanza bolas de nieve a la posición del “caverman” para herirlo. Sin embargo, la temperatura se va actualizando a medida que pasa el tiempo, y a medida que esta aumenta, el snowman pasa a la siguiente fase. La segunda fase, el se puede mover en sentido horizontal, haciendo movimiento de derecha a izquierda y seguirá lanzando bolas de nieve. La tercera fase, el snowman puede seguir moviéndose igual que la segunda fase, pero comenzará a lanzar bolas en movimiento oscilatorio y esas harán más daño.

La escenografía continúa siendo similar que la del nivel 2; sin embargo, muestra el termómetro que marca la temperatura del ambiente y se va actualizando, para notar al jugador que el snowman se hace más fuerte. Además, se muestra barra de vida de los dos personajes, timer y demás. La antorcha tiene un comportamiento especial, ya que se puede mover rotando sobre su propio eje, y a cierta altura de la escena explota y lanza bolas de fuego en todas las direcciones para hacerle daño al snowman.



**Conclusiones**

Las grandes ventajas de usar este tipo de lenguaje es que nos deja realizar proyectos llamativos e implementar lógica en la parte gráfica. Se sabe que la interfaz gráfica conlleva el uso de grandes recursos que involucran implementos de software y hardware. Es claro que todo programa no funcionará si no existe una base lógica fuerte que lo soporte, así pues, se deja claro que el análisis de este problema lleva principalmente un desarrollo de diagrama de clases, soluciones propuestas, métodos y atributos de las clases que vamos a utilizar y la relación entre estas. Estas relaciones y el componente de herencia que da un polimorfismo entre clases es un detonante para desarrollar programas o proyectos más avanzados.

Sin un buen IDE, el desarrollo de un programa tendrá tanta complejidad que no será fácil manejarlo. El IDE de QTCreator, permitió implementar gran parte de lo esperado; sin embargo; es necesario saber que QT tiene decenas de clases y que la búsqueda de algunas funcionalidades es compleja. QT ofreció una amplia gama de herramientas para el uso de cualquier programador.

Aunque vemos muchos juegos que nos parecen fascinantes y pensamos que es fácil simular tantos eventos que pueden existir en estos, cada vez que nos metemos en la etapa de programar y llevar a cabo todos estos eventos, nos damos cuenta que, la complejidad que existe es gigantesca. Se entiende el por qué en una empresa de desarrollo de software o desarrollo de videojuegos, tienen personal para todo tipo de requerimientos como programadores, diseñadores y muchos campos más.

Siempre será importante no saltarse las etapas de algún problema planteado, no es solo la etapa de programación.

Agente inteligente: El snowman tendrá la posibilidad de que a medida que el termómetro baje la temperatura, él aumente sus habilidades. El percibe la temperatura y decide cómo actúa en base a esto. Además de las implicaciones que tiene que aumente la dificultad con el tiempo. El seguirá mejorando.

*En vista a un futuro:*

Se seguirá desarrollándolo para mejorar y hacer el juego más dinámico posible con la implementación de más y más acciones y la simplificación de más código que pudo ser más reducible.

Se quiso implementar más sonidos en cada acción del caverman, en el daño que recibía cada enemigo, además de mejorar la escenografía y mejorar mucho el nivel 2 y 3. Se seguirá manteniendo la idea de que el juego sea lo más llamativo y con varias funcionalidades. Sería interesante poner en cada nivel más enemigos y mejores ayudas.

A pesar de lo anterior, se llevó a cabo un buen desarrollo de este juego. Se quiso intentar que fuera dinámico y muy llamativo. Se necesita más tiempo para desarrollar mejor las ideas, las implementaciones y demás. Se logró gran parte del objetivo: aprender y aplicar lo visto en clase, o la gran parte de este. Fue un trabajo agradable y fascinante de realizar.

* **Material de apoyo**

*Librería de QTCreator.*

Enlaces de búsqueda:

<https://cppreference.com/>

Vídeos de apoyo:

<https://youtu.be/fc4hanhAgVk?si=BVfXAw0MXJgYSta1>

<https://youtu.be/BwT8dFlvwWM?si=qJS3AWLz17BJIuct>

<https://youtu.be/vRnA9hDLbV0?si=pzeakZFRhhyHRFCB>