**PROYECTO FINAL, INFORMÁTICA 2.**

Fabian Camilo Falla Ramírez fabian.falla@udea.edu.co

Estudiante Ingeniería de Telecomunicaciones

Universidad de Antioquia

Medellín, Antioquia

Sofia Marín Cacante sofia.marinc@udea.edu.co

Estudiante Ingeniería de Telecomunicaciones

Universidad de Antioquia

Medellín, Antioquia

**DIMENSIONAL DASH: LAS CRÓNICAS DEL MEGA ARBOL.**

1. **Contexto**

Nuestro juego está inspirado en el primer capítulo de la primera temporada de la serie. El primer capítulo de la primera temporada de "Rick y Morty" se titula "Piloto". En este episodio, conocemos a los personajes principales: Rick, un científico alcohólico y excéntrico, y su nieto Morty, un adolescente tímido. Rick convence a Morty para que lo acompañe en un viaje Inter dimensional utilizando un dispositivo que le permite viajar entre realidades paralelas. Durante el viaje, Rick y Morty se enfrentan a situaciones surrealistas y peligrosas, incluyendo la necesidad de recolectar cristales de "Mega Árbol" para curar una herida temporal. Para encontrar estas semillas se van a otra dimensión donde podemos ver diferentes criaturas, monstruos y especies demasiado extrañas, su objetivo principal es encontrar los cristales.

A lo largo del episodio, se exploran temas de ciencia ficción y comedia negra, y se sientan las bases para la serie, que combina humor absurdo con elementos de viajes Inter dimensionales y críticas a la cultura pop.

El objetivo principal de nuestro juego es encontrar las salidas a otras dimensiones. La idea es que Rick (que es el más interesado) va a ser el personaje principal de nuestro juego. Rick va a tener que abrirse paso entre esas extrañas dimensiones donde se va a encontrar con diferentes criaturas y monstruos, los cuales va a tener que esquivar para poder encontrar el portal que lo lleve al siguiente nivel hasta que en el ultimo volverá a su dimensión y ganará el juego, la idea es que el portal al siguiente nivel va a ser invisible y el jugador no sabrá con exactitud donde está, debe ser muy cuidadoso y hábil para encontrarlo rápido.

En cada nivel, el portal va a estar en diferente ubicación, tendrán diferentes escenarios y seguramente cada nivel sea más difícil que el anterior, porque cada dimensión es mas toxica, por lo que deberá tener más cuidado evitando pantanos mortales.



Figura 1. (Contexto de una dimensión)

***Análisis por niveles.***

**Nivel 1:** Rick debe encontrar una salida a una nueva realidad paralela, pero en el camino a encontrar esa “puerta” se encontrará con diversos obstáculos y enemigos que debe evitar, Rick es un científico, aunque es excéntrico conoce las debilidades de dichos enemigos y prefiere evitar estos enemigos antes que enfrentarlos. Hay dos tipos de enemigos directos en este nivel, están los fantasmas que tiene movimientos verticales y horizontales y están los monstruos que disparan con un movimiento parabólico, los fantasmas no sufren con estos disparos. Además, entre los obstáculos se encuentran objetos que son partes de esta dimensión, como troncos, que simplemente impiden que Rick avance, también hay charcos de muerte que parecen inofensivos, pero son altamente tóxicos para los seres que no son de esa realidad y si Rick los toca, muere.

**Nivel 2:** La diferencia de este nivel radica principalmente en la dificultad de este nuevo mundo ya que es una realidad muchísimo más tóxica y con más vegetación y obstáculos. Para Rick en este mundo va a ser muchísimo más complicado encontrar una salida, ya que, aunque en las diversas realidades en las que ha estado viajando no conoce exactamente la ubicación de las salidas, en esta en específico, como hay más vegetación tiene que recorrer más territorio para encontrarla y con más obstáculos y peligros, además, si logra encontrar la salida, se dará cuenta que está en frente de uno de los monstruos que disparan por lo que deberá ser muy ágil y estratégico.

**Ecuaciones de movimiento:** Estas ecuaciones las trabajamos y se nos fueron explicadas en el laboratorio y en otros cursos de la carrera.

Para simular el movimiento, los valores obtenidos en un estado se vuelven los valores iniciales del siguiente ya que la velocidad y posición varía en los instantes de muestreo.

Calcular posición:

,

,

,

,

Calcular velocidad teniendo en cuenta la gravedad:

,

.

.

Actualizar la velocidad: La magnitud de la velocidad es la raíz de la suma de sus componentes al cuadrado.

.

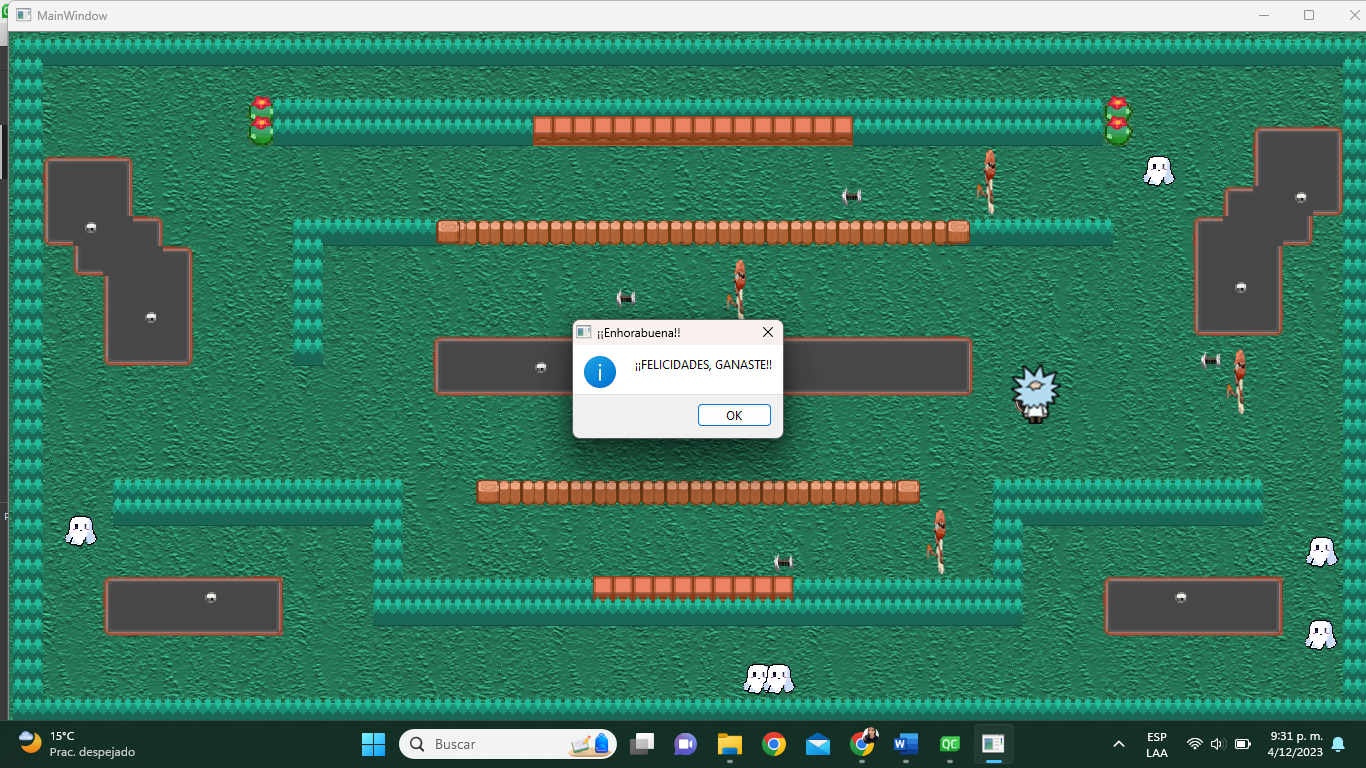
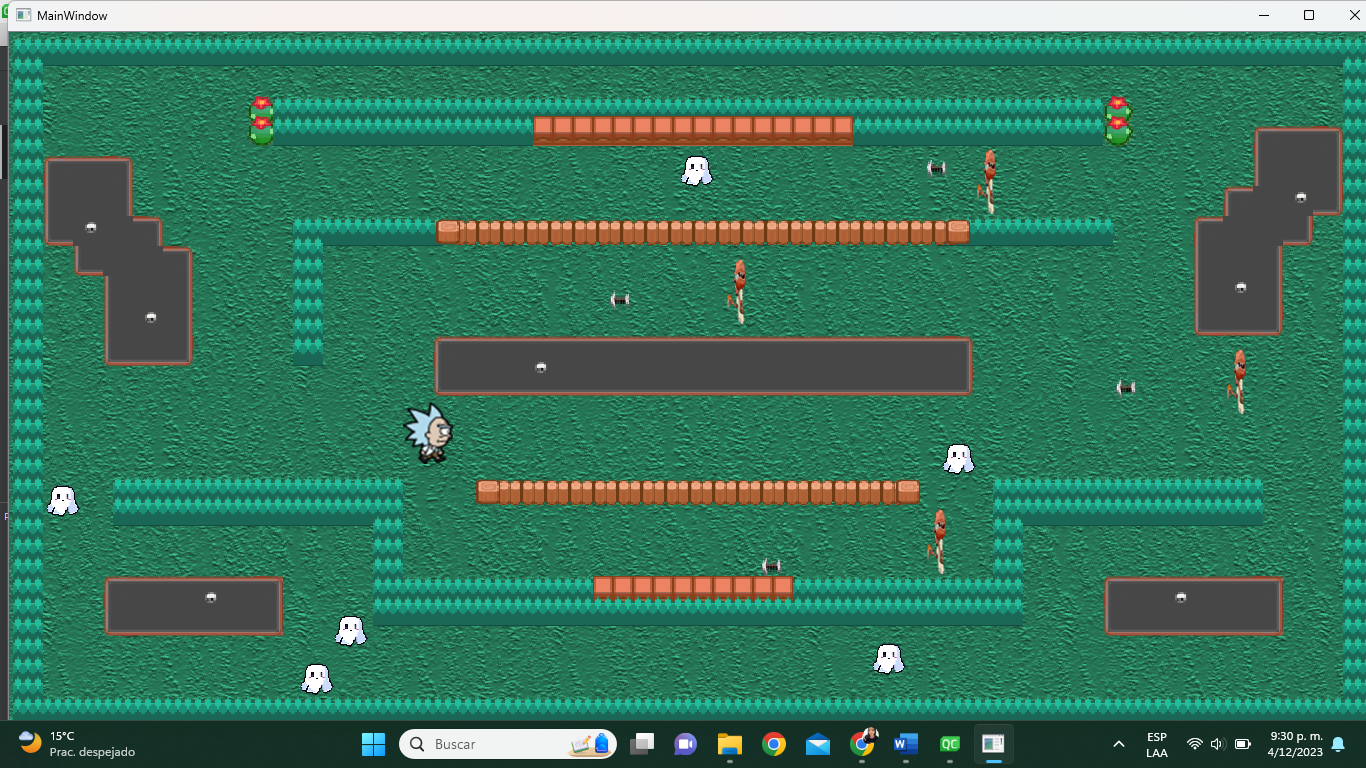
***Nivel 1***

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

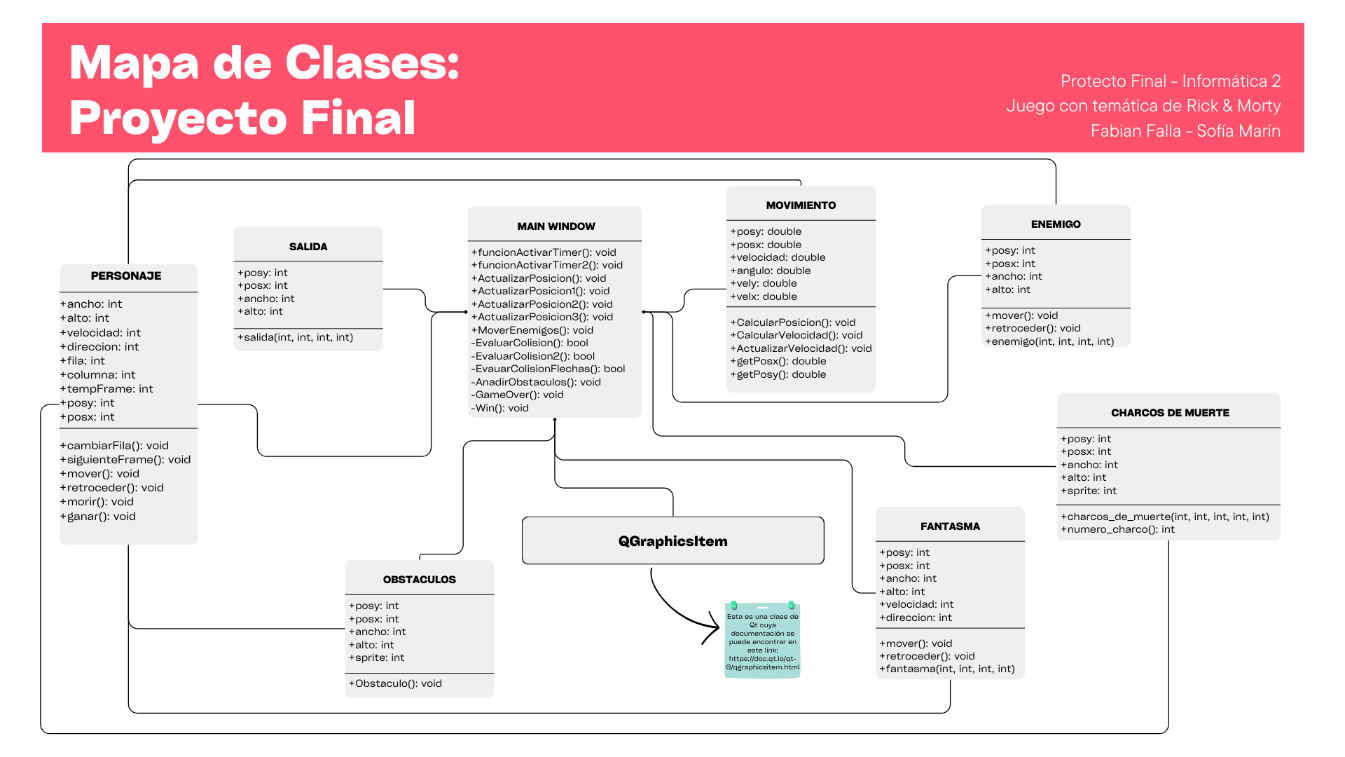
Nivel 1 después de que Rick supera los obstáculos y aparece la puerta

***Nivel 2***



Nivel 2, el jugador gana.

***Diagrama de clases final.***



***Importancia Del Control De Versiones***

Los sistemas de control de versiones son un tipo de software que ayuda a hacer un seguimiento de los cambios realizados en el código a lo largo del tiempo. A medida que un desarrollador edita el código, el sistema de control de versiones toma una instantánea de los archivos. Después, guarda esa instantánea de forma permanente para que se pueda recuperar más adelante si es necesario. Es una práctica fundamental en el desarrollo de software que implica gestionar y seguir los cambios en el código fuente a lo largo del tiempo.

Sin el control de versiones, los desarrolladores se sienten tentados a mantener varias copias del código en su equipo. Esto es peligroso, ya que es fácil cambiar o eliminar un archivo en la copia incorrecta del código, lo que podría hacer que perdieran el trabajo. Los sistemas de control de versiones solucionan este problema al administrar todas las versiones del código, pero presentan al equipo una sola versión a la vez.

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir hacia atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.  
  
¿**Por qué es importante el control de versiones?**

Hay muchas tareas que suponen una gran inversión de tiempo para los desarrolladores. La reproducción de errores, el aprendizaje de nuevas herramientas y la adición de nuevas características o contenido son solo algunos ejemplos. A medida que las demandas de los usuarios aumentan, el control de versiones ayuda a los equipos a trabajar juntos y distribuir soluciones a tiempo. El control de versiones también ayuda a los desarrolladores a moverse más rápido y permite que los equipos de software mantengan la eficiencia y la agilidad a medida que el equipo se escala para incluir más desarrolladores.

**Ventajas del control de versiones**

* ***Historial de cambios***: Permite mantener un historial detallado de todos los cambios realizados en el código. Esto facilita la identificación de quién hizo un cambio, qué cambio se hizo y cuándo se hizo. Este historial es esencial para la colaboración en equipos de desarrollo y para entender la evolución del proyecto a lo largo del tiempo.
* ***Colaboración***: Facilita la colaboración entre desarrolladores, ya que múltiples personas pueden trabajar simultáneamente en diferentes partes del código. El control de versiones gestiona de manera eficiente los conflictos que puedan surgir cuando dos o más personas intentan modificar el mismo archivo al mismo tiempo.
* ***Reversión y rollback***: Permite deshacer cambios no deseados o defectuosos. Si se introduce un error en el código, es posible volver a una versión anterior que funcione correctamente. Esto es especialmente valioso en entornos de producción, donde es crucial mantener la estabilidad del software.
* ***Branching y merging***: Facilita la creación de ramas (branches) independientes para desarrollar nuevas características o solucionar problemas sin afectar la versión principal del código (trunk o master). Luego, estas ramas pueden fusionarse (merge) de nuevo en la versión principal una vez que las modificaciones han sido probadas y verificadas.
* ***Gestión de releases***: Permite etiquetar versiones específicas del software (releases) para marcar hitos importantes en el desarrollo. Esto facilita la distribución y la implementación de versiones específicas del software.
* ***Facilita la resolución de conflictos***: Cuando varios desarrolladores trabajan en el mismo proyecto, es posible que sus cambios entren en conflicto. El control de versiones proporciona herramientas para gestionar y resolver estos conflictos de manera eficiente.
* ***Seguimiento de problemas***: Al vincular cambios específicos en el código con problemas o tareas particulares, el control de versiones facilita el seguimiento y la gestión de problemas a lo largo del ciclo de vida del desarrollo.

En resumen, el control de versiones es esencial para la eficiencia y la calidad en el desarrollo de software, facilitando la colaboración, la gestión de cambios y la mantenibilidad a lo largo del tiempo. Git es uno de los sistemas de control de versiones más populares y ampliamente utilizados en la actualidad.

**Referencias**

* Atlassian. (s. f.). Qué es el control de versiones|Atlassian Git tutorial. <https://www.atlassian.com/es/git/tutorials/what-is-version-control>
* Mijacobs. (2023, 5 octubre). ¿Qué es el control de versiones? - Azure DevOps. Microsoft Learn. <https://learn.microsoft.com/es-es/devops/develop/git/what-is-version-control>