**Proyecto final - Informática 2**

**Integrantes:**

[DAVID ALEJANDRO MORON ACACIO](mailto:david.moron@udea.edu.co) PPT 5065210

[LUISA MARIA BOHORQUEZ ARDILA](mailto:luisam.bohorqueza@udea.edu.co) TI 1082875815

**Anatomy park** - Capítulo 3 | Temporada 1

**Resumen:** Es navidad y Rick busca salvarle la vida a un indigente llamado Rubén, para hacerlo le pide ayuda a Morty, cuando Morty accede Rick lo encoge y lo introduce en el cuerpo de Rubén por medio de una inyección. La primera misión de Morty es encontrar al doctor Bloom, co-creador del parque anatómico que se encuentra dentro del indigente. El parque Anatómico posee las enfermedades más mortales del mundo exhibidas como atracciones, pero algo ha salido mal y estas han escapado. El cuerpo de Rubén comienza a colapsar y poco tiempo después muere. Morty debe salir lo más rápido posible del parque pero las enfermedades se pondrán en su camino. Mientras tanto el resto de la familia de Morty pasa tiempo con los padres de Jerry y a Rick se le ocurre una forma de sacar a Morty y a los sobrevivientes del cadáver.

**Guión/Síntesis del juego:** El jugador va a estar controlando a Morty por medio de la consola, dirigiendo sus movimientos y ataques para vencer a sus enemigos. El jugador deberá disparar, arrojar granadas, saltar y correr para evitar ser asesinado por Hepatitis B, las múltiples Tuberculosis y la temible Gonorrea, hasta que Rick saque al vagabundo al espacio y lo haga explotar.

**Desarrollo del juego:**

**Introducción:**

* Aparece el logo del Rick and Morty mientras suena la intro

**Apertura (Viñetas):**

* Rick ingresando al vagabundo a la casa
* Imagen de Rick pidiéndole a Morty que lo acompañe (Vista desde Morty)
* Ingresa al cuerpo
* Presentación del parque Anatómico
* Conoce a los personajes y le presentan el problema

**Inicia el juego:**

Se presenta a Morty en el escenario de batalla (una imagen estática de fondo, todo en dos dimensiones), donde hay plataformas sobre las que puede moverse Morty, y a lo largo del escenario van apareciendo los enemigos (los cuales persiguen a Morty) y los items que le proporcionan salud (sustancia roja), energía para el rayo láser (sustancia verde) y munición (cargas explosivas como las granadas).

El jugador

* Nivel 1: Luchar contra hepatitis B
* Viñetas llegan al aparato respiratorio
* Nivel 2: Luchar contra la tuberculosis
* Viñetas donde muere Rubén
* Viñetas del viaje por el intestino delgado y sospecha de un traidor

\*Si tenemos suficiente tiempo:

* Nivel 3: Luchar contra la gonorrea
* Viñetas que muestran la traición de poncho y su muerte
* Viñetas dónde se le ocurrió a Rick explotar el cuerpo de Rubén en el espacio
* Viñetas de la muerte de Roger
* Sacrificio del doctor Bloom
* Nivel 4: Luchar contra E. coli en el tren

**Cierre:**

* Annie y Morty son perseguidos por Hepatitis B
* Los salva Hepatitis C
* Se suben a la nava de Rick
* Explotan al vagabundo
* **Extra:** Implementación de Rick en la nave

Ideas:

* Personajes en el nivel 1: Bloom, Poncho, Morty, Annie y Roger
* Personajes en el nivel 2: Bloom, Poncho, Morty, Annie y Roger (+ alexander ¿? si lo incluímos también mostraremos su muerte)
* Personajes nivel 3: Bloom, Poncho, Morty, Annie y Roger
* Personajes nivel 4: Morty y Annie
* Crear el resto de los personajes( Roger, Annie, Poncho, Dr. Xenon Bloom)
* Cuando Morty venza a sus enemigos aparecerán los que siguen vivos y se moverán con él hacia el siguiente nivel

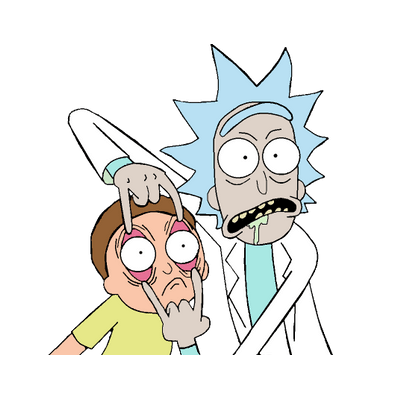
Ideas del logo del juego:



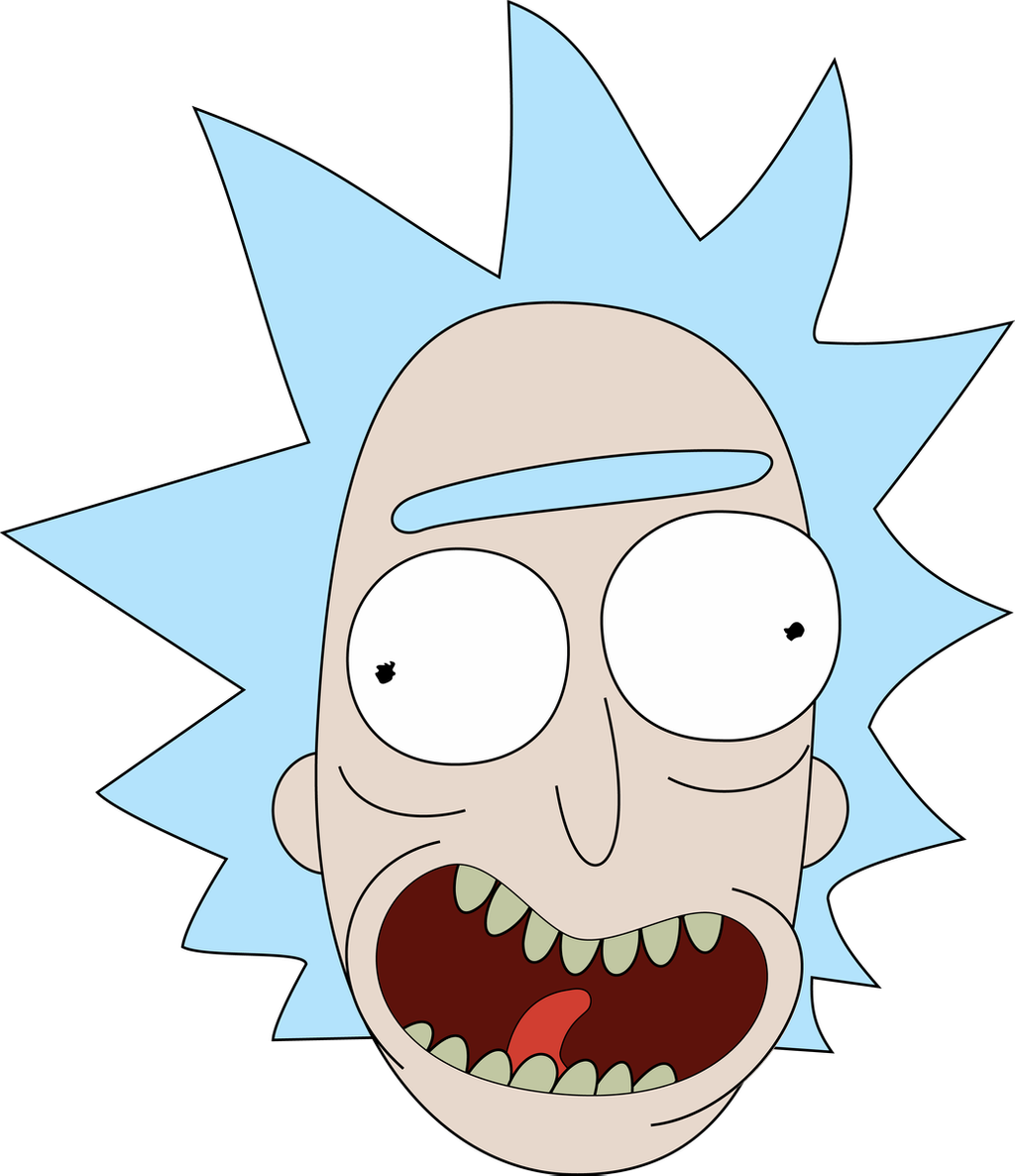












**Planeación**

Vamos a tener en cuenta cinco clases:

1. Clase Modelo, que va a ser quien se encargue de que cada imagen en el juego (personaje, enemigo u objetos inanimados) tenga propiedades físicas. Esto es, que sea capaz de describir tiros parabólicos, de caer por la gravedad o de rebotar. Va a poseer entonces atributos dimensionales de cada objeto y métodos que, en base a modelos físicos, calcule posiciones y efectos en la trayectoria.
2. Clase Escenario, que va a ser quien se encargue de generar el ambiente del juego. Esto es, generación de objetos de interfaz gráfica para la construcción de obstáculos, paredes, plataformas, entre otros. Va a poseer atributos de posición y recursos para cargar el escenario. Qt ofrece ya esta clase, llamada MainWindow.
3. Clase Usuario, que va a ser el objeto que se maneja con comandos por consola (el que controla el jugador). Este hereda de la clase Modelo para incluir sus físicas y además posee métodos para funcionar con eventos de teclado y mouse
4. Clase Personaje, que va a ser el objeto que realice acciones predeterminadas para atacar al jugador. Este hereda de la clase Modelo, y posee métodos para programar su comportamiento.
5. Clase Arma, que va a ser quien funcione como objeto que produce daño hacia los enemigos. También hereda de la clase Modelo.

Para el videojuego, la clase Escenario es la que inicia el nivel, donde luego se crea el personaje con el arma y simultáneamente se generan los enemigos. Para determinar el comportamiento de cada objeto, en términos de código, sobrecargamos los constructores para ofrecer una determinada forma en cuanto se instancien y así empiecen a funcionar en el ambiente. De esta manera, los métodos hacen la mayoría del trabajo, y en los códigos de control se verifican las condiciones de los objetos para definir si el nivel ha terminado. De ser así, el escenario cambia y con él las condiciones según lo necesitemos.

A modo de pseudocódigo, tendríamos la siguiente estructura para un nivel:

| Escenario(“cuál escenario va a cargarse”)//por dentro sus métodos ya saben qué imágenes y obstáculos cargar según la opción  //agregar el timer para el escenario  Usuario(“posición donde comienza”)//por dentro sus métodos ya saben qué imagen cargar y cómo inician sus valores. Dentro se instancian objetos de la clase  //timer que determina los movimientos de Morty  Arma y se define qué arma tiene a la mano  //timer para las trayectorias de las armas  Personaje(“posición donde va a generarse”, “cantidad”, “habilidad”)//de esta manera el objeto va a generar tantos como se indiquen, sus atributos están predefinidos para morir y desaparecer si reciben suficiente daño y su habilidad definirá si cumplen alguna dinámica en específico  //timer para las acciones de los personajes  //parar el timer del escenario cuando se acaben los enemigos |
| --- |

Se tienen como estimación los siguientes métodos:

1. Clase Modelo:
2. Atributos:

i. posicion(x, y)

ii. densidad(anchoObjeto,altoObjeto)

iii. gravedad(g)

iv. vel(spx,spy)

1. Métodos:

ii. calcularColision

iii. calcularRebote

iv. calcularSalto

v. Constructor general que inicialice los atributos

1. Clase Escenario//cosas gráficas
2. Clase Usuario : public Clase Modelo
3. Atributos extra:

i. Salud

1. Métodos:

i. Moverse

ii. Disparar contra Morty

1. Clase Usuario : public Clase Modelo
2. Atributos extra:

i. Salud

ii. Arma

1. Métodos:

i. Constructor con sobrecarga

ii. Manejo por consola y teclado

1. Clase Arma : public Clase Modelo

* La implementación final del código está guardada en un repositorio:

<https://github.com/LsBhrqz/Proyecto-final_Informatica-2.git>

* También, por lo frenética de la situación de desarrollo, hicimos un repositorio secundario:

<https://github.com/LsBhrqz/nuestro.git>

**Problemas de desarrollo afrontados y conclusiones**

Para este proyecto, podemos hablar de tres aspectos fundamentales que marcan la diferencia a la hora de trabajar: el conocimiento, la disponibilidad y la agilidad algorítmica. Respecto al primero, fue todo un proceso de investigación el descubrir cómo funcionan los objetos que posee Qt para su manejo de interfaces gráficas, el saber cómo usarlas, cuáles necesitamos, qué ingresar y qué modificar para que toda la escenografía y las imágenes se generen.

Luego de eso, el poder manipular esos datos en conjunto con los modelos y esquemas que teníamos preparados, lo cual fue todo un proceso de ensayo y error; el ver cómo no se deben conectar las entradas, saber con qué tipo de dato funcionan los métodos, el tener en claro el funcionamiento de objetos creados por nosotros mismos y su manejo de memoria en los ámbitos. Al final, se logra la habilidad de poder presentar muchas ideas, y con ello, la agilidad algorítmica se pone a prueba, para saber cuál solución implementar para que el programa no sufra algún desborde de memoria, o para mejorar la eficiencia en tiempos de ejecución, manejar errores, declarar funciones o atributos necesarios en la clase adecuada, todo para lograr un programa completo, funcional y sofisticado.

Sin embargo, el factor de la disponibilidad es el determinante de la calidad de nuestro trabajo. Se tuvieron muchos problemas de ámbito personal, entre ellos el que el integrante David no posee computadora para trabajar en el proyecto, por lo que sólo habían unas horas específicas para que pudiera avanzar en el desarrollo. Además, se cruzaron la mayoría de las responsabilidades académicas para los tiempos de desarrollo, por lo que la distribución del tiempo para trabajar en el proyecto fue tanta como se pudo, pero no como lo ameritaba. También, sobre la marcha, se fueron probando las ideas que surgían a medida que se aprendía algo nuevo, y no tuvimos la oportunidad de pulir el código tanto como hubiéramos querido.

Aún así, deja un gran deseo de mejorar y muchos conocimientos el proyecto presentado, aunque no esté completamente terminado, son grandes experiencias como programadores y grandes lecciones como estudiantes y profesionales para lograr mejores resultados. Estamos seguros de que, teniendo más tiempo y mejores condiciones para el desarrollo del juego, hubiéramos podido generar un resultado satisfactorio, no sólo con todas las ideas propuestas, sino también con mecánicas más complicadas y mejores detalles.