**Propuesta Juego Rick and Morty:**

Se plantea inicialmente seguir cronológicamente con la historia expuesta en el capítulo, donde se tienen dos historias pasando al mismo tiempo y cada una pasando a segundo plano cuando se requiere. En esta historia jerry trata de colocar las luces de navidad en la casa y apuesta a que no saldrá lastimado en el proceso, Rick le dispara un rayo que lo hace menos denso que el aire lo que le permite flotar de la misma forma le lanza otro rayo pero esta vez a sus zapatos haciéndolos “más pesados” de esta forma ahora puede saltar muy alto.Jerry queriendo presumir de su nueva habilidad quiero jugar  un partido de básquetbol con un grupo de personas que se encontraban en una cancha y al intentar hacer una jugada pierde un zapato y sale disparado hacia el cielo, Rick le ofrece ayuda para llevarlo a casa y el orgullo de Jerry no lo deja aceptar, una vez finalizado esto Rick y Morty salen al espacio donde  Morty es mordido por una serpiente espacial(la cual es vida inteligente y perteneciente a una civilización) la cual mata, el sentimiento de culpa lo invade vuelve a la tierra compra una serpiente y la sustituye haciéndola parecer como si fuera la que mató inicialmente, esto desarrolla una guerra y viajes en el tiempo de la sociedad de serpientes inteligentes contra Rick y Morty por lo que deben exterminarlas para solucionar el problema.

La historia del videojuego adaptará el cómo Jerry regresa a casa sin ayuda y la batalla de Rick y Morty contra la sociedad inteligente y futurista de serpientes.

Se propone como primer nivel, el trayecto que recorre Jerry apenas sale disparado al cielo una vez que perdió el zapato hasta que llega a un bar en el bosque.

Como segundo nivel la batalla de Rick y Morty contra la sociedad avanzada e inteligente de serpientes del futuro.

Como tercer nivel el trayecto de Jerry desde el bar hasta su casa.

Entre las clases que pretendemos usar para la elaboración del video juego se plantea usar las siguientes:

**Generales:**

* **Menú.**
* **interfaz gráfica.**

**Niveles Jerry:**

* **Avatar.**

**como métodos se proponen:**

* **Agarrar objetos.**
* **Movimientos.**
* **Objetos.**
* **Tiempo.**
* **Peso.**
* **Enemigos.**

**Nivel Rick y Morty:**

* **Personajes.**

**Como métodos se proponen:**

* **Agarrar objeto.**
* **Movimiento.**
* **Enemigos.**
* **Tiempo.**
* **Vida.**

Entrando a mayor detalle en los niveles además del inicio.

el primer nivel consiste en Jerry quien evite perderse en el espacio, en este momento Jerry se encuentra flotando en el cielo del cual aparecerán objetos de arriba, en este nivel debes evitar que los objetos golpean a Jerry, para eso tendrá los movimientos limitados a izquierda y derecha de la pantalla, si los objetos golpean a Jerry este recorre cierta distancia hacia arriba, no todos los objetos harán que Jerry suba, algunos lo harán descender.

Se piensa que para los objetos habrá algunos que lo hagan ascender más rápido que otro, lo mismo para lo que le hacen ganar “peso”.

habrá un tiempo en este nivel, por ahora se propone dos minutos, si logras no perder en ese tiempo se considera completado, se piensa que a medida que el tiempo llegue a cero la cantidad de objetos que te hacen ganar “peso” sea menor que al inicio y la cantidad de objetos que te hacen perder “peso” sea mayor.

En este nivel como se puede intuir, solo habrá interacción entre Jerry y los objetos, los cuales modifican la “variable” peso.

en el segundo nivel será una “batalla” entre el personaje (Rick o Morty aún no se sabe cuál) contra serpientes, en este nivel el personaje se podrá mover en cuatro direcciones (sujeto a cambios radicales) además de poseer un arma para su defensa.

En este nivel habrá un tiempo por el cual debes mantener vivo el personaje. las serpientes (por ahora se tiene pensado que salgan de tres direcciones) a cierta distancia “brincaran” o acercará de forma instantánea al personaje haciéndole un daño x, no hay forma de recuperarse de dicho daño, además se hará que ciertas veces cuando seas atacado por las serpientes haya una posibilidad de terminar envenenado, lo cual hará bajar la vida progresivamente por un corto periodo de tiempo, cabe mencionar que a medida que el tiempo sea menor mayor será la cantidad de enemigos.

Aunque el personaje se mueva en cuatro direcciones se planea limitar el espacio donde se puede mover, ya que si se pudiera mover por toda la pantalla dificulta el movimiento de los enemigos, ya que estos tendrían que perseguir al personaje para hacerle daño.

las interacciones en este nivel se limitan a los enemigos con las balas (estas harán un daño fijo) y el personaje principal, los demás objetos visibles serán planos.

El tercer nivel será similar al primero, pero los movimientos serán en horizontal, además del valor que se pierde será la vida del personaje cuando éste colisiona con los objetos, los cuales vendrán del lado derecho de la pantalla, la tasa de objetos para la curación será menor que la del primer nivel.

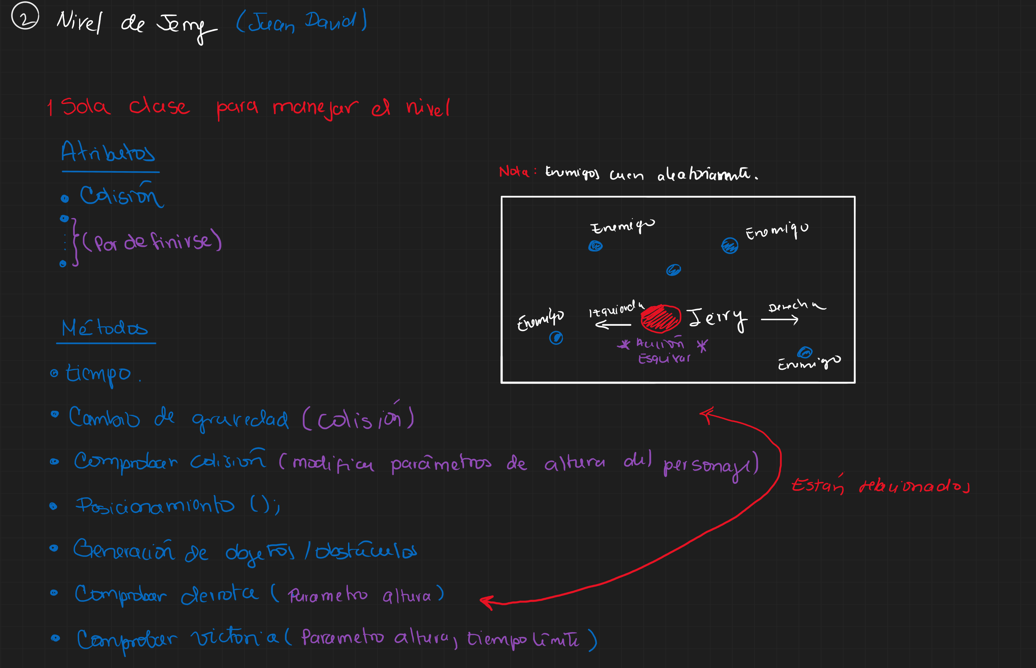
También habrá un tiempo en este nivel, el cual cumple con las mismas propiedades que en los anteriores niveles.

Los objetos que interactúan entre sí son los mismos que en el primer nivel.

Por recomendación del docente solo se van a trabajar dos niveles, cada uno con distintos enfoques en cuanto a movilidad, por ende, se elimina el tercer nivel propuesto al inicio.

el primer nivel.

nivel Jerry según Juan David.



nivel Jerry según Gabriel.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente





Nivel Rick and Morty según ambos:



Texto

Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**Implementación:**

Por recomendaciones del docente Anibal se reduoe el numero de niveles para no tener tantas fisicas de movimiento o lanzamientos repetidas.

**Nivel 1:**En el nivel 1 se crearon  dos clases las cuales se manejan a partir del mainwidows,los nombres de las clases son:  
**bolita** y **personaje\_nivel\_1** cada una con sus respectivos atributos y métodos.

**personaje\_nivel1:**es la clase que representa al personaje principal jugable por el usuario, hereda de QGraphisItem y QObject, donde QGraphisItem necesita que definamos unos métodos virtuales.

1. **Atributos:**

* **pos\_x:**almacena la coordenada en ‘x’ del personaje en la escena.
* **pos\_y:**almacena la coordenada en ‘y’ del personaje en la escena.
* **velocidad:** la velocidad con la que el personaje se va a mover en la escena al presionar su respectiva tecla direccional.
* **ancho:**el ancho de la imagen que se va a cargar en un mapa de pixeles para añadirla en la escena.
* **largo:**el largo de la imagen que se va a cargar en un mapa de pixeles para añadirla en la escena.
* **mapa\_personaje:** es el mapa de pixeles donde se va a cargar la imagen/sprite correspondiente al personaje.

1. **Métodos:**

**public:**

**personaje\_nivel1**():constructor.

**personaje\_nivel1**(int x,int y):constructor sobrecargado.

    QTime timed

    ~**personaje\_nivel1**():destructor.

**QRectF boundingRect**()const: método virtual que se necesita redefinir.

**void paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget = nullptr)**: método virtual que se necesita redefinir.

**movimiento\_derecha()::**incrementa la pos\_x con la velocidad.

**movimiento\_izquierda():**decrementa la pos\_x con la velocidad.

**movimiento\_arriba():**decrementa la pos\_y con la velocidad para subir la posición del personaje en la escena(recordemos que eje y de la escena está invertido).

**reinicio\_posicion(int x, int y)**;

**getPos\_x()** const:métodos de acceso de atributos.

**getPos\_y()** const:métodos de acceso de atributos.

**Bolita:** es la clase que representa los obstáculos del enemigo, esta clase hereda también de :QGraphisItem y QObject.

1. **Atributos:**

* **double pos\_x**
* **double pos\_y:**
* **double radio:**
* **double movi\_y:**
* **double movi\_x;**
* **double vel\_y:**

1. **Métodos:**

**public:**

**bolita():** constructor.

**bolita(double x,double y):**constructor sobrecargado.

**//ABSTRACTOS**

**QRectF boundingRect()const:**método virtual que se necesita redefinir.

**void paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget = nullptr):** método virtual que se necesita redefinir.

**//SETTERS:** Métodos para setear valores de los atributos.

**void setPos\_x(double newPos\_x);**

**void setPos\_y(double newPos\_y);**

**void setMovi\_y(double newMovi\_y);**

**void setMovi\_x(double newMovi\_x);**

**void setVel\_y(double newVel\_y);**

**void setAngulo(double newAngulo);**

**void colocar\_posiciones(double x, double y);**

**//GETTERS:** Métodos para obtener valores de los atributos.

**double getPos\_x() ;**

**double getPos\_y() ;**

**double getMovi\_y();**

**double getMovi\_x();**

**double getVel\_y() ;**

**double getAngulo() const;**

**double getRadio() const;**

**Nivel 2:** El nivel 2 está compuesto por 3 clases, la clase de **personaje, enemigos, proyectil personaje**,las cuales todas heredan de QGraphisItem y QObject, por lo tanto hay funciones virtuales que se tienen que redefinir.

1. **personaje:** esta clase representa al personaje principal jugable del videojuego del nivel 2.

**private:**

**//atributos:**

* **int pos\_x;**Posición del personaje en X;
* **int pos\_y;**Posición del personaje en Y;
* **int largo=125;**Largo de la imagen a cargar en el mapa en pixeles.
* **int ancho=125;**Ancho de la imagen a cargar en el mapa en pixeles.
* **int velocidad=40;** velocidad o incremento para la posición del personaje en cualquier eje.
* **int vida=3;** vida del personaje.
* **QPixmap\*mapa\_sprite=new QPixmap(":/Morty/sprites/Morty/MortyDerecha1.png");**

**public:**

**//CONSTRUCTORES:**encargado de inicializar la clase y setear las posiciones inicial en ‘x’ y ‘y’ del personaje.

**Personaje();**

**Personaje(int x,int y);**

**//DECLARACION DE METODOS QUE ERAN ABSTRACTOS**

* **QRectF boundingRect()const;**
* **void paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget = nullptr);**

**//SETTER**

* **void setVida(int newVida);**

**//MOVIMIENTOS;**

* **void Mov\_up();**
* **void Mov\_down();**
* **void Mov\_left();**
* **void Mov\_rigth();**

**//GETTERS**

* **QPixmap \*get\_mapa\_sprite();**
* **int get\_pos\_x();**
* **int get\_pos\_y();**
* **int getVida();**

**//DESTRUCTOR**

* **~Personaje();**

1. **enemigos:**Esta clase representa el enemigo del nivel 2, este enemigo no es jugable, y se mueve siguiendo al personaje principal en el tiempo.

**private:**

* **int pos\_x:**Posición del personaje en X;
* **int pos\_y:**Posición del personaje en Y;
* **int largo:**Largo de la imagen a cargar en el mapa en pixeles.
* **int ancho:**Ancho de la imagen a cargar en el mapa en pixeles.
* **velocidad:**velocidad o incremento para la posición del personaje en cualquier eje.
* **QPixmap \*mapa\_sprite\_en:**

**public:**

**//MÉTODOS**

**//CONSTRUCTOR**

**Enemigos():**

**Enemigos(int x,int y):**

**//MÉTODOS ABSTRACTOS HEREDADOS DE QGRAPHICSITEM:** estos metodos existen para crear los rectángulos donde se van a cargar los mapas de píxeles.

* **QRectF boundingREct() const:**
* **void paint(QPainter\*painter,const QStyleOptionGraphicsItem \*option,QWidget \*widget = nullptr).**

**//MOVIMIENTOS**

* **void Move\_up();**
* **void Move\_down();**
* **void Move\_left();**
* **void Move\_rigth();**

**//SETTERS**

* **void set\_posiciones(int x,int y);**

**//GETTERS**

* **int get\_pos\_y();**
* **int get\_pos\_x();**
* **int get\_velocidad();**

**//PERSECUCION PERSONAJE**

* **void persecucion\_personaje(int personaje\_x,int personaje\_y);**

**//DESTRUCTORES**

* **~Enemigos();**Elimina todo lo que tiene que ver con memoria dinámica una vez instanciada la clase.

1. **proyectil\_personaje:** esta clase representa el proyectil que va a lanzar el personaje para impactar con el enemigo.

**private:**

**//ATRIBUTOS**

* **double pos\_x:**Posición del personaje en X;
* **double pos\_y:** Posición del personaje en Y;
* **double ancho=32:**Ancho de la imagen a cargar en el mapa en pixeles.
* **double largo=32:**Largo de la imagen a cargar en el mapa en pixeles.
* **double velocidad\_x=35:**Velocidad inicial en x para que cumpla condiciones de movimiento parabólico.
* **double velocidad\_y=28:** Velocidad inicial en y para que cumpla condiciones de movimiento parabólico.
* **doubletiempo=1**
* **double y\_final**
* **char direccion\_disparo:**
* **QPixmap \*mapa\_proyec\_per=new QPixmap(":/Proyectil/sprites/Proyectil/Bola\_de\_fuego.png"**
* **QTimer \*timer=new QTimer();**

**public:**

**//METODOS**

**//CONSTRUCTOR**

* **Proyectil\_personaje();**
* **Proyectil\_personaje(int x,int y,char direccion);**

**//DECLARACIONES DE FUNCIONES QUE ERAN ABSTRACTAS**

* **QRectF boundingRect()const;**
* **void paint(QPainter \*painter, const QStyleOptionGraphicsItem \*option, QWidget \*widget = nullptr);**

**//MOVIMIENTO**

* **void set\_posiciones(double x,double y);**

**//DESTRUCTOR**

* **~Proyectil\_personaje();**

**//GETTERS**

* **QTimer \*getTimer() const;**

**private slots:**

* **void Mov\_up();**
* **void Mov\_down();**
* **void Mov\_left();**
* **void Mov\_rigth();**

**NOTA:** en este caso se dan valores iniciales a las velocidades para que puedan hacer un movimiento parabólico, en el que el proyectil no supere los 200 píxeles recorridos en el eje x y una altura máxima de 40 píxeles en y.En el repositorio se anexará como se hicieron dichos cálculos.