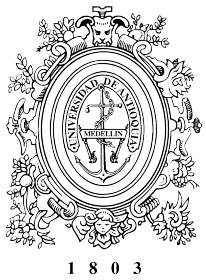


UNIVERSIDAD DE ANTIOQUIA

Facultad de Ingeniería

Descripción y Modelamiento del Juego

Juan Carlos Murillo Florez
Sebastian Zuluaga Correa

Informatica 2

JS GAMES CORP.

Profesores: Augusto Enrique Salazar Jimenez , Jonathan Ferney Gomez Hurtado

5 de abril de 2022

Índice

1. Introducción 2

2. Librerías propias 2

3. Posibles clases a utilizarse 3

4. Funcionamiento detallado del juego 7

5. Conclusión 7

6. Referencias 7

Resumen

El propósito de este informe es conocer un poco mas a detalle las caracteristicas y objetos que contiene nuestro juego y su respectiva planeacion y modelacion para su posterior implementación con las herramientas y conocimientos obtenidos de la programacion orientada a objetos (POO).

1. Introducción

En el presente informe de Informática 2, referente al parcial 2 sobre la planeación y modelamiento de las clases a usar en el proyecto final (juego desde cero). Además de dar a conocer las capacidades de los integrantes para modelar y para la programación orientada a objetos con las herramientas y conocimientos dados en el curso de informática 2

2. Librerías propias

- ✓ **Lógica:** La idea con esta librería es crear aquí los datos que de una u otra manera utilizaremos en las diferentes clases; aquí los definiremos y luego crearemos funciones que nos permitan devolver la dirección de cada una de estas cuando alguna otra clase invoque estas funciones, por ejemplo, crearemos un vector de un tipo de datos llamado obstáculos y luego brindaremos la dirección de este vector para que las diferentes clases puedan modificar su contenido.

- ✓ **Archivos:** La idea con esta librería es realizar toda la gestión de los archivos de texto que utilizaremos para guardar y cargar los estados y avances de 3 usuarios, con sus respectivos niveles, dinero, vehículos, entre otros datos, además de suministrarnos los datos que solicitemos durante el funcionamiento del juego.

3. Posibles clases a utilizarse

Figura 1. Personaje/Carro

FUENTE: Autores

✓ **Personaje/Carro**: La idea con esta clase es manejar todas las funcionalidades del carro. Aquí tendremos diferentes funciones que nos permiten realizar todo el control de este y además la interacción con los demás objetos ubicados en el escenario.Por medio de la funcion propia de QT llamada collidesWithItem() podremos saber si el personaje esta colisionando con otros obetos, ya que esta nos retorna positivo si colisiona o negativo si no lo hace, de esta manera haremos la logica para que si el objeto (si es un objeto estatico en el escenario) esta justo en frente, el personaje no pueda atravesarlo y deba hallar la manera de superar dicho obstaculo, ya sea saltando o usando llantas o resortes. En el caso del personaje debe hacer maniobras extremas y tener cuidado para no sufrir daños en el vehiculo. Si no es un objeto estatico en el escenario este se moverá al ser impactado por el personaje y le transmitirá energina de movimiento debido a la tercera ley de Newton (Acción-Reacción).

Figura 2. Containers

FUENTE: Autores

✓ **Containers**: En esta clase modelaremos un objeto tipo contenedor con sus respectivas características como inclinación, puntos de colisiones, ubicaciones predeterminadas en los niveles. Con la funcion de colisiones (collidesWithItem()) tambien podremos saber si el personaje esta colisionando con un obeto tipo container, como esta nos retorna positivo si colisiona o negativo si no lo hace, de esta manera haremos la logica para que si el objeto (y es un objeto estatico en el escenario) esta justo en frente, el personaje no pueda atravesarlo y deba hallar la manera de superar dicho obstaculo, ya sea saltando o usando llantas o resortes. En el caso de los containers, si estan inclinados el jugador podrá usarlos como rampa, de ser asi su movimiento cambiará y describirá una recta paralela a la superficie del container (esto se logra apartir de la ecuacion de la recta y su pendiente debe ser la misma que la del container).

Figura 3. Vehiculos NPC

FUENTE: Autores

✓ **Vehículos NPC:** Esta clase se encarga del modelamiento de ciertos carros que estarán por el mapa de forma predeterminada los cuales realizan un movimiento en dirección contraria a la del jugador cuando están en el capo de visión del jugador, además si el jugador colisiona a cierta velocidad con uno de estos vehículos NPC este explotará. Mediante la funcion de colisiones (collidesWithItem()) tambien podremos saber si este objeto es colisionado por el personaje, ya que esta nos retorna positivo si colisiona o negativo si no lo hace, de esta manera haremos la logica para que si este objeto es colisonado a cierta velocidad y/o con cierta fuerza, este llame a una funcion de la clase que se llama animacion de explosion y luego queda el objeto destruido como se puede ver en la figura 3.

Figura 4. Cajas

FUENTE: Autores

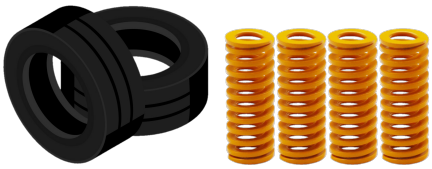
✓ **Cajas**: Esta clase se encarga del modelamiento de cajas que estarán repartidas por el camino con el fin de obstaculizar la trayectoria del jugador, con sus respectivas características como puntos de colisiones, ubicaciones, animación de destrucción, entre otras. Como en el objeto anterior se hace uso de la función de colisiones para saber si el personaje choca con este objeto tipo cajas, ya que si este es colisionado a cierta velocidad y/o con cierta fuerza, este deberá llamar a la funcion de la clase encargada de la animacion de destrucción para posteriormente removerlo de la escena.

Figura 5. Minas y Pinchos

FUENTE: Autores

✓ **Minas y Pinchos**: Esta clase se encarga del modelamiento de minas explosivas que estarán repartidas por el camino con el fin de obstaculizar la trayectoria del jugador o destruirlo en caso tal de que este colisione con uno de estos objetos, con sus respectivas características como puntos de colisiones, ubicaciones, animación de destrucción, entre otras.Como en los objetos anteriores tambien haremos uso de la función de colisiones (collidesWithItem()) para saber saber si este objeto es colisionado por el personaje, si el personaje pisa una mina nos retorna positivo y si el llevaba menos de la mitad de la vida o de la barra del estado del vehículo , se debe llamar la funcion de muerte de la clase personaje que se llama muerte personaje ya que las minas causan un daño de la mitad de la vida, o si el personaje toca los pinchos este morirá en el instante sin importar la cantidad de vida, y deberá reintentar el nivel desde el principio.

Figura 6. Llantas/Resortes



FUENTE: Autores

✓ **Llantas/Resortes**: Esta clase se encarga del modelamiento de resortes o llantas saltarinas que estarán repartidas por el camino con el fin de darle un impulso de salto extra al jugador creando así el efecto resorte o movimiento armónico simple, como todas las anteriores esta también contara con sus respectivas características como puntos de colisiones, ubicaciones, animación de destrucción, entre otras. Como hemos hecho para todos los objetos anteriores, debemos hacer uso nuevamente de la funcion de colision pero anidada con otras condiciones, ya en este caso se hacen cosas diferentes porque se aplicara el impulso extra si el jugador colisiona al objeto por arriba, si este lo llega a colisionar por los lados este sera devuelto una cierta cantidad de pixeles.

Figura 7. Monedas

FUENTE: Autores

✓ **Monedas**: Esta clase se encarga del modelamiento de monedas que estarán repartidas por el camino las cuales el jugador podrá recoger para aumentar su puntaje y así poder desbloquear y/o personalizar vehículos, con sus respectivas características como puntos de colisiones, ubicaciones, incremento y acumulación de dinero, entre otras. Para las monedas la cosa es un poco mas cencilla, ya que si el personaje colisiona con una de ellas de bebe imcrementar el contador que indica la cantidad de monedas que lleva el jugador y actualizar en la base de datos el saldo actual, por demás cabe recalcar que en este caso tambien se hace uso de la funcion de colisiones.

Figura 8. Escenario

FUENTE: Autores

✓ **Escenario/Nivel:** La idea con esta clase es crear cada uno de los escenarios o niveles que estarán presentes en nuestro juego de acuerdo al nivel de dificultad donde haya avanzado el usuario, a medida que se avance de nivel en el juego, esta clase creará un escenario, donde habrá diferentes obstáculos y/o amenazas que se ubicarán en el escenario de manera semi aleatoria, con sus respectivas características, comportamiento y/o posición de los ejes de coordenadas x y. En esta clase se carga una imagen que sera el boceto o la base donde se cargaran los objetos que conformaran el nivel, se debe sobrecargar el constructor y depende del parametro cree un nivel u otro, esto con el fin de que hayan varios niveles con sus respectivas diferencias de dificultades, pero todos con las mismas propiedades o con muchas en comun, es decir, haciendo uso de las mismas o de algunas funciones propias de la clase nivel.

4. Funcionamiento detallado del juego

El juego consiste en un camión monstruo que deberá recorrer un camino con diferentes obstáculos donde también encontrará monedas que podrá recoger para aumentar su puntaje y desbloquear y/o personalizar vehículos, el usuario tendrá el control del camión mediante algunas teclas predeterminadas, con las cuales deberá demostrar su destreza para superar los desafiantes caminos repletos de obstáculos y amenazas para mantener la integridad del vehículo con la menor cantidad de daños posibles, ya que cada vez que alguna parte del carro (diferente a los parachoques y la parte de abajo del carro) colisione con otros objetos o el suelo, este se verá reflejado en la barra que indica el estado del vehículo con el decremento de la misma, si esta llega a su mínimo el vehículo explotara y el jugador perderá. En el camino el jugador podrá encontrarse diferentes obstáculos como containers, cajas, vehículos y también diferentes amenazas como minas, pinchos, entre otros.

5. Conclusión

En conclusión, se hara un uso exhaustivo de la funcion de colisiones para determinar mucha informacion y dependiendo de esta, tomar decisiones. Además que se detalló el planeamiento inicial respecto a la elaboracion del juego, teniendo en cuenta principalmente el modelo final de juego que se desea obtener, sin embargo, durante el proceso de implementacion se pueden llegar a presentar cambios a la hora de implementar el modelamiento de las clases u objetos de acuerdo a las necesidades que se evidencien.

6. Referencias

1. Logosímbolo UdeA
2. Algunos Sprites e imagenes fueron extraidos de un juego:
Sitio web = Y8, Nombre del juego = Monster Truck,
title = `https://es.y8.com/games/monster`$_t ruck_h tml$