Propuesta de proyecto

Juan Daniel González Puerta

jdaniel.gonzalez@udea.edu.co

Dept Ingeniería de telecomunicaciones

Universidad de Antioquia

Justificacion

A lo largo del semestre se cubrieron objetivos muy especificos los cuales fueron escalando poco a poco mostrando su interes por converger en un punto donde el estudiante se sintiera plenamente consiente de su aprendizaje, sin embargo este sentimiento de aprendizaje debe de manifestarse en un producto final que muestre todos estos conocimientos adquiridos de manera condensada y armonica, es por todo esto que se propoone el desarrollo de un videojuego en Qt ya que estos por más simples que sean necesitan el desarrollo de los temas vistos, tales como interfaz grafica, programacion orientada a objetos, simulacion de sistemas fisicos, carga y guardado de datos y la interaccion entre todos estos elementos y aunque no es estrictamente necesario el desarrollo de un videjuego para mostrar estas capacidades si es la mejor manera de aplicar todos estos conceptos de manera armonica.

Motivación

Ciertamente el desarrollo de un videojuego es más complejo de lo que podía pensar, esto lo descubrí en el laboratorio 5 “PACMAN”, donde se debía desarrollar un juego similar a Pacman y sus mecánicas, pero no solo era crear unos condicionales y métodos para el juego, era crear todo desde 0, desde el mapa hasta los objetos, desde la estética hasta las interacciones, lo cual fue un gran desafío para mí y tuve que reforzar mis conocimientos de programación y adquirir habilidades nuevas tales como edición de imágenes, sonido, razonamiento y creatividad. Fue un nuevo mundo para mí, sin embargo al final del día fue

copiar la idea de otro a menor escala, pero fue el primer paso en lo que es la búsqueda de nuevo conocimiento y habilidades, ciertamente me emociona pensar en que desafíos trae este proyecto para mí, como mejorare las habilidades que ya poseo y que habilidades nuevas voy a adquirir, y si se me permite adentrarme hacía una perspectiva más emocional, diré que lo que más me motiva es el desarrollo y aprendizaje que este trae y no solo el afán de entregar un producto por una nota.

Desafios a afrontar

- imaginar un juego con todos sus elementos y mecánicas que funcionaran de manera orgánica no es para nada fácil ya que en la mente suena todo muy bien pero no suele funcionar en la realidad, al menos no sin hacer cambios.
- La implementación de los sistemas físicos es algo que me tiene un poco preocupado ya que si estos sistemas no se aplican de manera correcta el juego no dará esa sensación de haberse realizado bien, por más que el resto de elementos funcionen bien.
- el nivel de dificultad siento que será laborioso ya que tendré que probarlo muchas veces para saber que no sea demasiado fácil o difícil y quede fuera de su categoría.
- Una de las dificultades que se me presentan es el tiempo que hay para el desarrollo y que en este momento no cuento con demasiado tiempo ya que trabajo y tengo otras materias.
- El correcto orden de los elementos y su apropiada interacción con el juego.

Descripción

El juego consta de un personaje o dos depende si se quiere jugar individual o multijugador, el cual dispara balas a unos enemigos que caen de manera aleatoria y con velocidades diferentes, contara con 3 niveles diferentes los cuales serán fácil medio y difícil, pero el jugador solo contara con 3 vidas para poder superarlo todo, puede guardar su partida para continuar después pero si pierde la vida empezara desde cero, la idea del juego es completarlo y obtener la mayor cantidad de puntos posibles, una de las mecánicas será que el jugador empezara con un 100% de probabilidad de que el disparo salga bien y cada vez que dispare se restara 1% a esa probabilidad, es decir que empieza con 100% de probabilidad de que el disparo salga bien y si dispara le quedara 99% de probabilidad de

que dispare bien, si el disparo sale bien saldrá la bala normal pero si sale mal la nave explotara y perderá una vida, la idea es que el jugador piense que puede disparar mucho por la baja probabilidad que tiene sin embargo como es una probabilidad acumulada desde el 95% hay bastante probabilidad de que el disparo salga mal.

Las clases que voy a usar serán:

- Objeto gráfico: serán todos los objetos que ponga en pantalla y a los cuales le agregare su imagen. Esta clase será heredada por todos los elementos.
- Cuerpo: será el cuerpo principal de los personajes y estos contaran con posición, tamaño, vidas y puntaje, heredara la clase objeto grafico para poder dibujarlo.
- Enemigo: serán los enemigos que caen y estos tendrán ciertas características como vida, posición, velocidad.
- Balas: serán los objetos que interactuaran con los enemigos destruyéndolos y haciéndoles daño, también pueden interactuar con el personaje principal dañándolo.
- Puntaje: esta clase llevara el puntaje del usuario principal.
- Vidas: esta clase llevara las vidas del usuario.
- Porcentaje: esta clase llevara el porcentaje de las balas. Si estas salen bien o no.
- Daño enemigo: serán algunos daños extras que le quitaran velocidad al usuario dejando el piso con más fricción.
- Porcentaje: esta clase llevara el porcentaje de las balas. Si estas salen bien o no.

Clases

***Cuerpo**: esta clase me permite crear un cuadrado y dibujarlo, con una posición y un tamaño.*

Atributos:
Posx=double
Posy=double
Ancho=doublé
Largo=doublé
Velocidad=int
Métodos:
Cuerpo()
double getPosx()
double getPosy();
void setPosx(double x);
void setPosy(double y);
QRectF boundingRect() const;
*void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);*

void mover_derecha();
void mover_izquierda();

Enemigo: *esta clase me creara los enemigos con una posición, y un tamaño, además de características especiales como velocidades y ángulos iniciales.*

Atributos:
Posx = double
Posy = double
Ancho= double
Largo = double
Vel = double
Velx= double
Vely = double
G = double
Delta = double
Pi = double
Angulo = double
Dir = int

Métodos:
enemigo();
double getPosx();
double getPosy();
double getVelx(); //funcion para tomar la velocidad en x.
double getVely(); //funcion para tomar la velocidad en y.
void setVelx(double vx_);
void actualizarposicion_derecha();
void actualizarposicion_izquierda();
void rebotepiso();
void setVel(double vel_);
double getVel();
double getAngulo();
void setAngulo(double Angulo);
void actualizarvelocidad();
void setDir(int dir_);
int getDir()
QRectF boundingRect() const
*void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);*

puntaje: esta clase me permite llevar la cuenta de el puntaje

Atributos:
Posx=double
Posy=double

Ancho=doublé
Largo=doublé
Puntaje = int

Métodos:
Puntaje()
double getPosx()
double getPosy();
void setPosx(double x);
void setPosy(double y);
QRectF boundingRect() const;
*void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);*
Actualizarpuntaje();
int getpuntaje();

vidas: esta clase me permite llevar la cuenta de las vidas

Atributos:
Posx=double
Posy=double
Ancho=doublé
Largo=doublé
Vidas=int

Métodos:
vidas()
double getPosx()
double getPosy();
void setPosx(double x);
void setPosy(double y);
QRectF boundingRect() const;
*void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);*
Actualizarvidas();
Int getvidas();

balas: esta clase me permite llevar la cuenta de las balas que disparara el personaje principal y el porcentaje de que un disparo salga hacia adelante.

Atributos:
Posx=double
Posy=double
Ancho=doublé
Largo=doublé
Velocidad = double
Porcentaje = double

Métodos:

balas()
double getPosx()
double getPosy();

void setPosx(double x);
void setPosy(double y);
void setVel(double vel_);
QRectF boundingRect() const;
*void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);*
Moverbalas();
double getPorcentaje();
void setPorcentaje(double porcentaje)

obstaculos: esta clase me permite llevar la cuenta de las balas que disparara el personaje principal

Atributos:
Posx=double
Posy=double
Ancho=doublé
Largo=doublé
Viscosidad=double

Métodos:
obstaulos()
double getPosx()
double getPosy();
void setPosx(double x);
void setPosy(double y);
QRectF boundingRect() const;
*void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);*
double getViscosidad();

Música: esta clase me permite reproducir música

Atributos:

Sonido= Qmediaplayer

Cancion = string

Métodos:
Reproducirmusica(string canción_);
Seleccionarcancion(string canción_)