DOCUMENTACIÓN
PROYECTO DE IGV

Trabajo hecho por:

Rubén Sanabria Díaz (rsd00008) y Rubén Prieto García (rpg00035)

Índice

1. Explicación general del proyecto

Como proyecto de prácticas, hemos diseñado un juego del tipo “Tower Defense” en 3D. Este juego se basa en que existe un camino por el que aparecen enemigos que tratan de llegar hasta el castillo ubicado al final del camino, lo que implicaría que nos quitan una vida (si nos quitan 6 vidas, perderíamos la partida). Para evitar que los enemigos lleguen al castillo del final del camino, debemos construir defensas con un dinero inicial, en cada uno de los terrenos disponibles, las cuales quitarán vida a los enemigos hasta derrotarlos. Cuando se derrota a un enemigo, nos proporcionan dinero, con el cual podemos construir más defensas o mejorarlas.

En el castillo, podemos ver unos personajes haciendo alusión a los creadores del proyecto, estos personajes se llaman reyes.

El juego consiste en completar 3 oleadas, en las cuales los enemigos tienen más vida y velocidad conforme pasen dichas oleadas.

2. Requisitos pedidos

- Vistas de la escena
    - La escena se podrá visualizar desde varios puntos de vista (planta, alzado, perfil)
    - Se podrá seleccionar un terreno para acceder al menú de estructuras de dicho terreno

- Movimientos de la cámara (zoom, órbita, cabeceo, panorámica y desplazamiento)
    - Se podrá hacer zoom y girar la escena al gusto.
    - Nos podremos desplazar por la escena
    - Podremos realizar movimientos orbitales y arcball (cabeceo)

- Modelar objetos complejos en la escena
    - Las estructuras/defensas serán estructuras complejas (varios objetos con varias transformaciones)

- Realizar transformaciones de los objetos de la escena de manera interactiva
    - A la estructura Torre se le podrá aumentar y disminuir el nivel de altura con un desplazamiento vertical clickando en ella, ya que los enemigos podrán ser voladores

- Grafo de escena:
    - Los reyes podrán ser manipulados interactuando con sus partes móviles.

- Menús de funcionamiento:
    - Menú de estructuras: con el que podremos construir estructuras, venderlas, subirlas de nivel, ...

- Luces
    - Una luz puntual que alumbra la escena en general,
    - Una luz focal que podrá ser manipulada
    - Luces focales que alumbran las estructuras y cuyos colores (de los focos) depende del nivel de la estructura alumbrada

- Objetos con color y textura
    - Las estructuras/defensas y enemigos tendrán colores dependiendo del nivel de estos. Habrá 3 niveles:
        - verde: nivel 1
        - amarillo: nivel 2
        - rojo: nivel 3.
    - La carretera será de un color específico
    - Usaremos texturas para los laterales del terreno.
    - Usaremos materiales y seremos capaces de variar los coeficientes de los mismos.

- Sombreado:
    - la escena (objetos) tendrá un sombreado plano (flat) o un sombreado de Gouraud (smooth). Se podrá alternar entre un sombreado u otro.

- Animación
    - Los enemigos y estructuras tendrán animación para aportar realidad, además si ganas una partida se podrá ver a los reyes bailando
    - Hemos incluído música y sonidos para la victoria, derrota, muerte de un enemigo y disparos.

3. Explicación de cada una de las clases

igvEscena3d

- visualizar. En este método, lo que hacemos es ver si estamos en la tienda o no.
  Si no estamos en la tienda, cargamos componentes externos (luces, fijas, texturas, materiales, ...). Miramos todos los terrenos y vemos si hay estructuras construidas o no. Si hay estructuras construidas, la construimos y creamos el foco cuyo color depende del nivel de la estructura.
  Después, creamos terreno, castillo y corazones (indican las vidas que quedan).
  Ahora llegamos a la parte de crear enemigos y meterlos en el vector de enemigos, pero antes calculamos los tiempos que han pasado en función de diferentes eventos del juego (si hemos disparado, si hemos entrado en tienda, ...) y creamos enemigos si el tiempo calculado (el tiempo REAL (hacemos hincapié en el tiempo REAL puesto que si estamos en la tienda 8 segundos, saldrían 2 enemigos pero realmente no tienen que salir) que ha pasado entre enemigo y enemigo) es de 4 segundos y además si hay enemigos por crear (variable “numEnemigos” que indica los enemigos que quedan por ser creados). Si no se puede crear enemigos, es porque estamos en el principio (dejamos 4 segundos para prepararte) o porque ya se han creado todos los enemigos (se tendría que incrementar la oleada).
  Ahora, miramos si seguimos vivos (si nos quedan vidas). En tal caso, visualizamos los enemigos del vector que no hayan llegado al final (en tal caso, los eliminamos del vector). Si no seguimos vivos, reproducimos música de derrota.

Si estamos en la tienda, deshabilitamos todas las luces, controlamos el tiempo y creamos la tienda y el menu de estructuras. Saldremos de la tienda cuando escojamos alguna opción y activaremos las luces.

- pintar_ejes: pintamos los ejes.
- pintar_tubo: pintamos un tubo.
- crearFoco: creamos un foco para un terreno en concreto y dependiendo del nivel de la estructura que se vaya a crear, la luz tendrá un color u otro.
- cargarExternos: cargamos las cosas externas y fijas como texturas, luz puntual y material. En el caso del material, si este estaba activado, no se veía bien la escena, por lo que deberíamos poder decidir si poner o no el material. Como el material “una vez que se pone no se puede quitar”, decidimos hacer como un boton de seguridad controlado por la variable llamada “autorizarMaterial”. Esta variable, si está a false (por defecto), no se podrá activar material, pero si ponemos a true (mirar apartado4), es activará el material y se podrá alternar entre un material u otro.
- crearEnemigo: creamos el enemigo de forma aleatoria.
- visualizarVB: metodo que se usa para poder realizar la selección por buffer de color.
- controlColisiones: comprueba se ha ocurrido una colisión entre flecha/bala y enemigo.
- realizarAccionesMenu: se realiza la opción elegida (comprar, vender, mejorar) en el menú de estructuras.
- comprar: compramos alguna estructura
- vender: vendemos alguna estructura
- mejorar: mejoramos alguna estructura
- añadirEnemigo: añadimos enemigo al vector

- añadirEstrucura: añadir estructura al terreno
- quitarEstructura: quitarmos estructura del terreno
- cambiarColorTorres: cambiamos color de las torres a la hora de usar la selección por color para alargar la torre.
- corazones && crearCorazones: creamos corazones
- terreno: creamos terrenos
- castillo: creamos castillo
- rey: creamos rey
- crearTapa: creamos tapa del cilindro
- crearCilindro: creamos cilindro
- costadoIzquierda: creamos costado izquierdo del rey
- costadoDerecha: creamos costado derecha:del rey
- inferiorIzquierda: creamos inferior izquierdo del rey
- inferiorDerecha: creamos inferior derecha:del rey
- parteArriba: parte de arriba del rey.

igvCamara

Esta clase representa la camara virtual de la escena. Gracias a esta clase, podremos cambiar entre las distintas perspectivas (panoramica - perspectiva) y entre las distintas vistas (planta - alzado - perfil).
Métodos:

- orbital: implementamos movimiento orbital de la camara
- aplicar: aplicamos todos los valores de la camara
- zoom: realizamos zoom

igvPunto3d

Esta clase representa un punto en 3d que se puede usar para cualquier cosa que sea necesario. Se compone por 3 componentes (x,y,z).

igvColor

Esta clase representa un color que se puede usar para todo lo que necesitemos. Se compone de 4 valores:

- r: red
- g: green
- b: blue
- a: alfa

igvMaterial

Esta clase representa un material que se puede aplicar a la escena. Esta clase la usamos para poner un material sobre los modelos de la escena si el usuario quiere (al principio no está aplicada). Si se quieren aplicar los materiales, se podrá alternar entre diferentes materiales. Un material está compuesto por su componente ambiental (ka), componente difuso (kd), componente especular (ks) y exponente de Phong. Todos estos componentes o coeficientes se podrán modificar para cambiar la forma en la que se aplica el material (ver apartado 4).

igvTextura

Esta clase representa una textura que realmente es una imagen en 2d. Esta clase la usamos para poner una textura en los laterales del terreno a modo de césped.

igvFuenteLuz

Se encargará de controlar las distintas fuentes de luz que poseen nuestro juego, almacenando en cada fuente la posición, su identificado, las atenuaciones, los distintos colores (ambiental, difuso y especular), y si es focal o no. Nuestro juego posee 1 luz puntual fija que alumbrará a toda la escena, 6 focales, una para cada terreno, siempre y cuando tenga una estructura en él, y una focal con la que podamos interactuar.

igvInterfaz

- set_glutKeyboardFunc: Este método es el encargado de ir comprobando si pulsas una tecla en el teclado, en caso de ser así cada tecla tendrá su propia función.

- set_glutDisplayFunc: Con este método, en caso de haber seleccionado previamente algo con el ratón, podemos comprobar que se ha seleccionado para interactuar con dicho objeto, para ello hemos hecho una variación de color entre los objetos que queremos seleccionar para distinguir el color de dicho pixel. Esto hace que podamos distinguir entre los 6 terrenos que disponemos, o que al clicar una torre sepa el programa cuál es. Este método hace llamada a otras dos funciones de igvInterfaz que serían “buscarTerreno” y “buscarTorre” para localizar dichos objetos.

- set_glutMouseFunc: Método encargado de reconocer si se ha clicado el ratón, variando entre clic derecho, izquierdo y central, y si se está pulsando o soltando.

- set_glutMotionFunc: Se usa principalmente para ir realizando acciones mientras un botón del ratón se está manteniendo

pulsado, pudiendo aplicar un desplazamiento a nuestra torre en el eje y.

- animación: Gracias a este método podemos saber cuál es la animación que debe realizar en cada momento nuestro programa, ya que openGL solo permite realizar una sola animación a la vez.

- reproducirSonido: Método utilizado para reproducir los distintos sonidos que tiene nuestro juego.

- gradosLibActivos: Comprobación de los grados de libertad del rey para poder realizar la animación correspondiente.

- create_menu: Crea un menú si y sólo si nos encontramos en la tienda donde podremos visualizar nuestro dinero y las distintas acciones que podremos hacer con él en un terreno específico.

- reyIdleFunc: Animación del rey.

- enemigoIdleFunc: Animación de avance de los enemigos.

- balaCañonIdleFunc: Animación de disparo del cañón.

- flechaTorreIdleFunc: Animación de disparo de la torre.

Estructura

Esta clase es un clase padre de las clases cañón y torre, donde almacena los métodos y variables comunes a amba, como puede ser la posición, los precios de venta, compra o mejora, el daño y los niveles de la misma. La principal diferencia entre torre y cañón es que cuando hablamos del mismo nivel entre estructuras y enemigos los cañones matan a los enemigos de un solo disparo y las flechas de dos, en caso de tener mayor nivel una torre que un enemigo, lo mataría de un solo disparo.

Torre

Clase usada para la creación de las torres, heredada de estructura, esta almacena la altura, usada para el desplazamiento vertical, y la flecha que usará para eliminar a los enemigos.

Cañón

Clase usada para la creación del cañón, heredada de es estructura, esta almacenará las distintas acciones del cañón y la bala que usará para eliminar a los enemigos.

Enemigo

Esta clase se encarga de crear a los enemigos, almacenando el nivel, la vida, el dinero que ofrecen al matarlos y su posición. Además se podrá distinguir entre enemigos terrestres y voladores, estos últimos aparecerán en un rango aleatorio del eje y, pudiendo morir exclusivamente por disparos de torre.

TerrenoEstructuras

Con esta clase podemos crear los 6 terrenos, cada uno en su posición, donde podremos colocar en ellos las estructuras disponibles, cada terreno poseerá un menú de tienda propio.

MenuEstructuras

Clase creada a partir de un terreno, que mostrará las estructuras disponibles a situar, una vez se abra el menú, al hacer clic derecho en la pantalla mostrará el menú de openGL donde tenemos las distintas operaciones que podremos realizar, ya sea compra, venta o mejora

4. Interacción con la escena

- ‘1’: disparar estructura del terreno 1 (si hay)
- ‘2’: disparar estructura del terreno 2 (si hay)
- ‘3’: disparar estructura del terreno 3 (si hay)
- ‘4’: disparar estructura del terreno 4 (si hay)
- ‘5’: disparar estructura del terreno 5 (si hay)
- ‘6’: disparar estructura del terreno 6 (si hay)

- ‘p’: cambia tipo de proyección (paralela - perspectiva)
- ‘v’: cambia tipo de vista (alzado - perfil - planta)

- ‘-’: zoom in
- ‘+’: zoom out

- ‘n’: aumentar distancia del plano cercano (znear)
- ‘N’: disminuir distancia del plano cercano (znear)

- ‘x’: rotar la escena en el eje x positivo
- ‘X’: rotar la escena en el eje x negativo
- ‘y’: rotar la escena en el eje x positivo
- ‘Y’: rotar la escena en el eje x negativo
- ‘z’: rotar la escena en el eje x positivo
- ‘Z’: rotar la escena en el eje x negativo

- ‘b’: subimos sombrero
- ‘B’: bajamos sombrero
- ‘q’: bajamos brazo izquierdo
- ‘Q’: subimos brazo izquierdo
- ‘u’: movemos brazo izquierdo para fuera
- ‘U’: movemos brazo izquierdo para dentro

- ‘e’: bajamos brazo derecho
- ‘E’: subimos brazo derecho
- ‘<’: movemos brazo derecho para fuera
- ‘>’: movemos brazo derecho para dentro

- ‘h’: movemos pierna izquierda para atrás
- ‘H’: movemos pierna izquierda para adelante

- ‘g’: movemos pierna derecha para atrás
- ‘G’: movemos pierna derecha para adelante

- ‘m’: agachamos muñeco
- ‘M’: levantamos muñeco

- ‘P’: activamos/desactivamos ejes

- ‘W’: movemos cámara para arriba (eje -z)
- ‘S’: movemos cámara para abajo (eje z)
- ‘A’: movemos cámara para izquierda (eje -x)
- ‘D’: movemos cámara para derecha (eje x)
- ‘r’: movemos cámara para arriba (eje y)
- ‘R’: movemos cámara para abajo (eje -y)

- ‘w’: movemos cámara (arcball (cabeceo)) para arriba
- ‘s’: movemos cámara (arcball (cabeceo)) para abajo
- ‘a’: movemos cámara (orbital) para izquierda
- ‘d’: movemos cámara (orbital) para derecha

- ‘i’: movemos foco “morado” de la escena para arriba (eje -z)
- ‘k’: movemos foco “morado” de la escena para abajo (eje z)
- ‘j’: movemos foco “morado” de la escena para izquierda (eje -x)

- ‘l’: movemos foco “morado” de la escena para derecha (eje x)

- ‘o’:aumentamos coeficiente difuso
- ‘O’: disminuimos coeficiente difuso

- ‘f’: aumentamos coeficiente especular
- ‘F’: disminuimos coeficiente especular

- ‘t’: disminuimos valor de phong
- ‘T’: aumentamos valor de phong

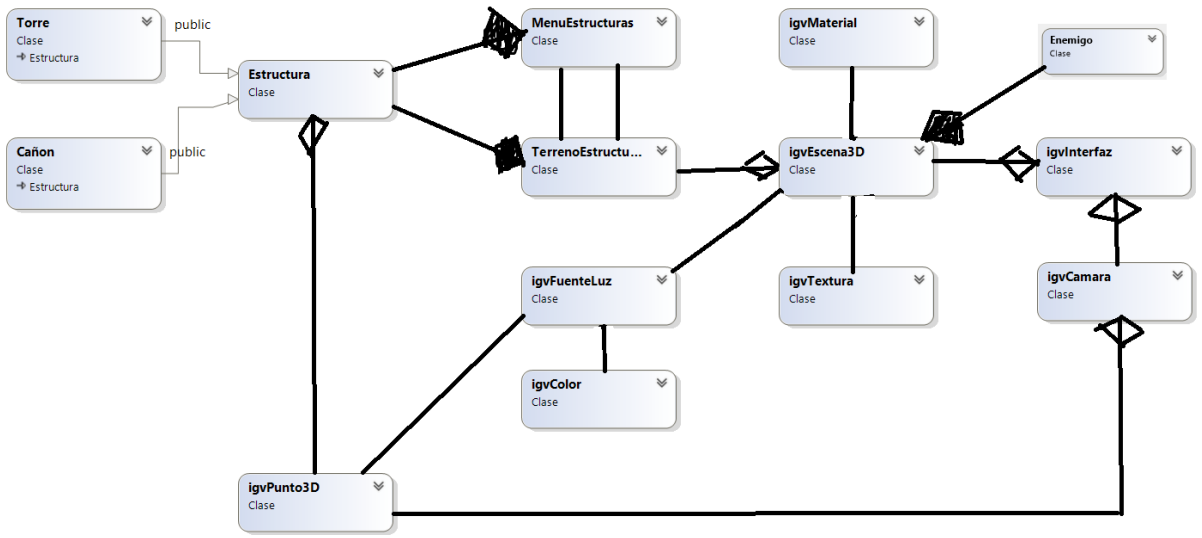
- ‘C’: cambiamos entre materiales
- ‘V’: autorizamos poner materiales (explicación en apartado 3)

- ‘J’: cambiamos entre tipo de sombreado

- Interacción con ratón:
    - al hacer click con el ratón a algún terreno, accedemos a la tienda, donde podremos comprar, mejorar y vender estructuras
    - al hacer click con el ratón en la torre y desplazar el ratón con el click derecho mantenido, la torre aumentará y disminuirá.
    - al hacer clic derecho en la tienda, abrimos un cuadro de texto con opciones, que será el menú de estructuras

5.Diagrama UML

Hemos realizado un diagrama de clases (diagrama UML) con el que podemos ver las relaciones entre las clases del proyecto.



6.Bibliografía

- https://stackoverflow.com/
- https://www.geeksforgeeks.org/
- http://www.aprendeaprogramar.com/mod/forum/view.php?id=337
- https://www.lawebdelprogramador.com/foros/Dev-C/index1.html
- https://community.khronos.org/
- https://learn.microsoft.com/