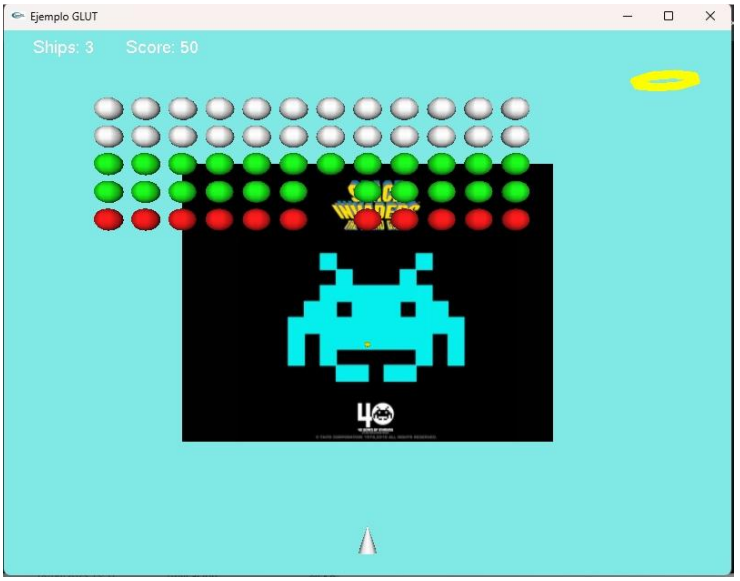
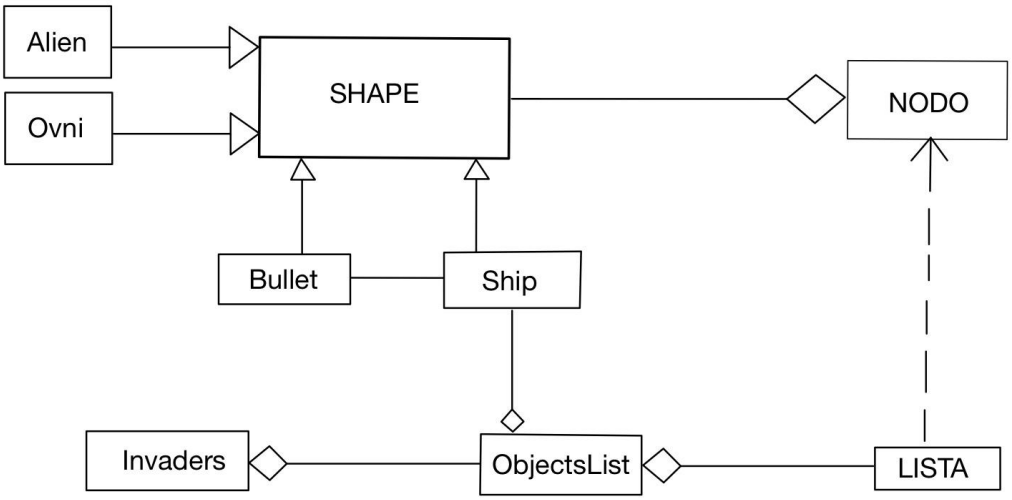
TRABAJO 2 PROGRAMACIÓN DE SISTEMAS: PROGRAMACIÓN ORIENTADA A OBJETOS



Trabajo 2 Programación de sistemas: Space Invaders.
Lola Alberte Tapia 20003
Adela Zapatera Argüello 20364
Jaime Pérez Pérez 20259

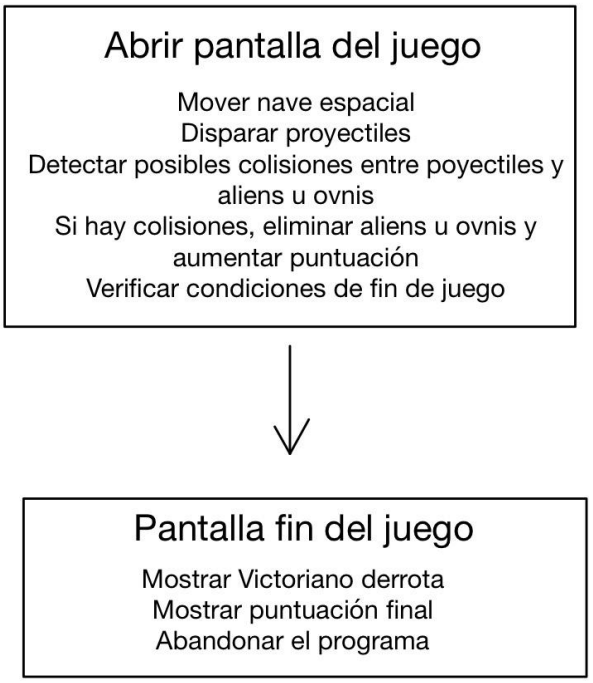
ÍNDICE

Diagrama de clases del juego:



Para definirlo, hemos tenido en cuenta que la clase Shape es la clase padre, de la que derivan las clases Alien, Ovni, Bullet y Ship. Estas dos últimas están relacionadas mediante asociación, ya que un método de la clase Ship devuelve un valor de tipo Bullet*.
La clase Nodo tiene una relación de agregación con Shape (esta última pertenece a Nodo) y de dependencia con la clase Lista (la Lista está formada por nodos). Lista y Ship son clases pertenecientes a Object_List, por lo que mantienen una relación de agregación (Lista como clase protegida y Ship como clase privada).
Por último, Invaders es una clase friend de ObjectsList, conocido como un tipo de agregación.

Diagramas de interacción de las fases del juego más interesantes:



Descripción general de funciones y estructuras de datos más representativas:

Para el correcto funcionamiento de nuestro código, se han desarrollado una serie de funciones y clases. De ellas, destacaremos las consideradas como más importantes, mencionando su finalidad e implementación de manera general, puesto que su análisis detallado se puede realizar haciendo uso del propio código del programa.

Mencionaremos la clase lista, implementada de manera previa a la realización de este trabajo, así como de la clase OVNI, o la ObjectsList.
Para entenderlas correctamente, es esencial tener una visión general del funcionamiento del código. El programa consta de 4 elementos esenciales, los Aliens (clase Alien), el OVNI (análoga al Alien), la ship (el jugador), y los proyectiles disparados por la ship (bullets).

Mediante la clase Lista ya mencionada, almacenaremos y controlaremos todos estos elementos, recogidos en la variable general del programa, worldObjects. Esta lista será recorrida por nodos, definidos en el fichero de cabecera nodo.h.

Las funciones más importantes de esta clase Lista son su destructor, y la funciones ins_cabeza y remove. El destructor elimina todos los nodos de la lista, guardando antes de cada eliminación, la dirección de memoria del siguiente nodo. Por su parte, ins_cabeza introduce un elemento en la primera posición de la lista.
Finalmente, la función remove de esta clase trata de eliminar un elemento concreto de la lista, obtenido por referencia. En su funcionamiento, existen 4 distintos casos:

1. La lista está vacía
2. El primer elemento de la lista es el que se desea eliminar
3. El caso general, el elemento a eliminar se encuentra en cualquier posición media de la lista
4. No se encuentra ningún elemento que contenga la información buscada, deseada de eliminar.

Otra de las secciones de código más relevantes a mencionar es la función operator+, contenida en las clases Alien y Ovni.

Esta función nos permite conocer la distancia (usando la función mydistance, definida dentro del fichero commonstuff) entre el Alien u ovni correspondiente y un elemento shape, pasado por parámetro (el bullet). Esta función será esencial para poder controlar cuando el proyectil disparado por el jugador impacta con algún alien o algún ovni, permitiendo eliminarlos.

Continuando, mencionaremos la clase ObjetsList, desarrollada íntegramente por nosotros, y cuyo funcionamiento es esencial para el desarrollo de la aplicación. Dentro de esta clase, se encuentran, principalmente, su constructor, encargado de generar todos los aliens, y la función Collisions, que mediante el operator+, regula cuándo impactan los proyectiles disparados por el jugador con los alienígenas correspondientes.

Dentro del constructor de la clase ObjectsList, uno de los aspectos a tener en cuenta son los puntos de inicio de los bucles for necesarios para recorrer fila por fila, y columna por columna, todas las posiciones en las que existirá un alienígena. Para solventar este aspecto correctamente, hay que tener en cuenta que el centro de la pantalla se considera el origen de coordenadas. Por tanto, empezaremos situando aliens en la columna -6

Por su parte, la función collisions comprueba, como ya se ha mencionado, si existe colisión entre un alienígena y el proyectil (Bullet) disparado por el jugador, asegurándose de si existe un proyectil, y si existe, comprobando de si se choca contra un alienígena o contra un OVNI, calculando la distancia.

Si el impacto es contra un alienígena, se suma un número constante de puntos en función del tipo de alien, mientras que si es contra un OVNI, los puntos obtenidos pueden ser 100, 200 ó 300, de manera aleatoria. Finalmente, se eliminan los aliens u OVNIs disparados y los proyectiles correspondientes.

En este método se usa la operación de cast dinámico para ver de qué tipo es el objeto theShape en el elemento de la lista que se está evaluando en ese momento.

Finalmente, otra de las secciones importantes de nuestro código es la implementación de la función win. Gracias a ella, el juego termina una vez se eliminan a todos los alienígenas, evitando que la pantalla quede vacía una vez ocurra dicho evento.

Esta función win() , perteneciente a la clase Invaders, devuelve el número de elementos de la lista worldobjects. Se utiliza en el main para comprobar que has ganado. Esto ocurre cuando win()

retorna 1, es decir, cuando has eliminado a todos los aliens y solo queda theShip. Finalmente, se muestra por pantalla el mensaje “YOU WON!”.

Descripción de pruebas realizadas para comprobar el correcto funcionamiento del algoritmo.

En primer lugar, se ha probado a variar la velocidad de los aliens porque la establecida inicialmente no nos permitía eliminar todos los aliens y no era posible comprobar el correcto comportamiento del juego al disparar a todos los alienígenas.
Para que esto sea posible, hemos reducido la velocidad ALIEN_SPEED como prueba, dándonos cuenta de que, con el código original, al eliminar todos los aliens el programa se quedaba bloqueado. Para evitar este error hemos creado una variable llamada win en el main que se activa al eliminarlos a todos con el uso de la función win(), comentada anteriormente. Una vez activada esta variable win, se muestra por pantalla un mensaje al ganar el juego y se sale del programa correctamente.
En la versión definitiva del programa se ha restablecido la velocidad ALIEN_SPEED a su valor original.
Otra de las pruebas realizadas en este programa ha sido la regulación de la posición más baja a la que pueden llegar los aliens, regulando la función get Lowest. En una primera instancia, el valor mínimo de esta función era superior a la posición de aparición de los Aliens, causando esto que el programa finalizara instantáneamente. Para solventarlo, simplemente cambiamos el valor de la posición más baja de los aliens, asegurándonos de que se trataba de la misma posición en la que se encuentra el The Ship del jugador.
Además hemos hecho pruebas con los tiempos de aparición de los OVNIs para que no aparecieran demasiados ni demasiado seguidos, como nos ocurría al principio.
También se ha probado a cambiar las posiciones tanto de los aliens como de los OVNIs para que no colisionen en ningún momento, y hemos dejado que dicho OVNI recorra toda la pantalla sin ser disparado para asegurarnos que desaparezca de manera correcta.

Finalmente, la prueba más importante realizada, ha sido asegurar el correcto funcionamiento del código en distintos sistemas operativos, como windows o Linux. Para asegurarnos del funcionamiento de ambos, se ha comentado en el código las operaciones y las sentencias a realizar, asegurándonos, de que si se realiza en windows, la librería glut.h está guardada en la carpeta adecuada, y de que se incluye la librería windows.h , así como de prescindir de la unistd.h, solamente necesaria para operar en Linux y osX

GUÍA DE USO DEL PROGRAMA EJECUTABLE

El uso del ejecutable depende fundamentalmente del sistema operativo en el que nos encontremos. Si se trata de Windows, debemos asegurarnos de tener los archivos relacionados con la librería openGL guardados correctamente y en la ubicación adecuada. Si, por ejemplo, se emplea osX o Linux, dichas librerías ya vienen instaladas por defecto.

En cuanto al funcionamiento del ejecutable como tal, se controla haciendo uso del teclado. La nave del jugador será movida y controlada utilizando las flechas del teclado, y los proyectiles se disparan con la barra espaciadora.
Si se desea salir del juego en cualquier momento, bastará con presionar la tecla escape (27). El juego terminará por sí solo cuando los alienígenas lleguen hasta la posición inferior de la pantalla (choquen con la fila en la que se encuentra la Ship), o si se elimina a todos los alienígenas. El mensaje mostrado por pantalla será distinto en ambos casos, siendo Game Over si los alienígenas llegan hasta abajo del todo, o YOU WON, si se elimina a todos ellos, caso en el que la puntuación será 2500 puntos.
Además, cada cierto tiempo, aparecerá en la pantalla un Ovni. Se trata de una especie de alienígena especial, dado que no desciende su posición en el eje vertical, y la puntuación obtenida al eliminarlo es aleatoria, será 100, 200 ó 300 puntos.

Descripción del reparto de roles del equipo:

La realización de este trabajo se ha llevado a cabo dividiendo las tareas a realizar de la manera más equitativa, y acorde al tiempo disponible para su realización.
Los roles de nuestro grupo han sido los siguientes: de manera general, el código ha sido desarrollado de manera conjunta entre los tres integrantes del grupo, poniendo ideas en común, y comprobando el funcionamiento del mismo de manera conjunta. Aprovechando la disponibilidad de distintos sistemas operativos, Lola se encargó de asegurar el funcionamiento del código en osX, usando un ordenador Mac. Jaime fue el encargado de asegurar el funcionamiento del código en windows, asegurándonos, en ambos casos, de la compilación del código y, en el caso de windows, de la correcta ubicación de las librerías y funciones glut.
Finalmente, el desarrollo de la memoria, así como la implementación de la clase OVNI, y la confirmación del correcto funcionamiento de ambos sistemas empleando el escritorio virtual ha sido realizada por Adela.

PROPUESTA DE MEJORA Y VALORACIÓN PERSONAL

Aún faltan múltiples funcionalidades que podrían haber sido implementadas en este programa, como pueden ser la existencia de obstáculos, o de múltiples proyectiles. Una propuesta de mejora es, evidentemente, la implementación de las extensiones mencionadas, o de múltiples otras.

Por último, en cuanto a la valoración personal, consideramos que el trabajo realizado es acorde a los requisitos pedidos, y los conocimientos adquiridos en este curso. Como acabamos de mencionar, no se han implementado todas las funcionalidades posibles, pero el funcionamiento del programa es correcto.
En líneas generales, estamos satisfechos con el trabajo realizado , con el resultado final obtenido y con la manera de trabajar en la que se ha realizado este segundo trabajo de la asignatura de Programación de Sistemas.