Instituto Tecnológico de Costa Rica

Lenguajes, Compiladores e Intérpretes

I Semestre 2021

Grupo:

Freddy Armando Fallas Garro.

Harold Jose Espinoza Matarrita.

Mathiw Dario Rojas Jimenez

Profesor:

Marco Rivera Meneses

Bitácora

11/06/2021
Se realiza una reunión entre los integrantes del grupo para definir las funciones, en la cual Harold y Armando se enfocarán en la parte lógica de java y Mathiw en la parte del juego en C

15/06/2021
Se investiga la librería Allegro para C, para utilizarla en la creación del juego

17/06/2021
Se investiga acerca de la conexión por sockets en java y se crea un archivo inicial con métodos de prueba

20/06/2021 - 22/06/2021
Se descarta la librería de Allegro pues pese a estar desarrollada en C necesita de un main .cpp para poder ser ejecutada. Posteriormente se busca la librería Raylib como nueva opción para crear el juego. De parte de Java se crean las clases: Jugador, Alien, Bunker, Director con sus respectivos geters y seters, además de funciones propias de cada clase. También se implementa el patrón de diseño constructor en la creación de los aliens.

23/06/2021
Se cambia la librería utilizada para el desarrollo de sockets, pasando a utilizar winsock32, posteriormente se readaptan los sockets, cambiando el envío de datos, por lo que se agrega un “\0” en el caso de los datos enviados por el servidor y un ‘\n’ para el mensaje enviado por el cliente. Luego se añadió la lectura y escritura de .txt de los nuevos sockets.

24/06/2021
Por parte del cliente se agregó la función de escribir los datos que mantienen al juego funcionando para posteriormente poder enviarlos al servidor por medio de los sockets. Por parte del servidor se empieza la creación del lector de los datos recibidos por los sockets, en este último se lee los datos del txt y se almacenan en un arreglo y los elementos de este se manipulan mediante índices.

25/06/2021
Se continua con el desarrollo de la lectura de los datos por parte del servidor, se realiza el procesamiento para ver qué índice corresponde a qué dato, posteriormente se implementa el uso de los datos para actualizar los elementos del juego.
Por parte del cliente se desarrolla una función que leerá los datos almacenados en el archivo recibido del servidor, e interpreta su contenido para actualizar las variables que manejan el juego.

Plan de Actividades

Desarrollo de la conexión por sockets. Responsable: Armando Fallas. Fecha límite de entrega 22/06/21.

Implementación de patrones de diseño. Responsables: Harold Espinoza, Armando Fallas. Fecha límite de entrega 23/06/2021.

Creación de la interfaz gráfica. Responsable: Mathiw Rojas. Fecha límite de entrega 23/06/21.

Creación de las clases para la lógica del juego Responsables: Harold Espinoza, Armando Fallas. Fecha límite de entrega 24/06/2021.

Crear el cliente que observa la partida del primer jugados. Responsable: Mathiw Rojas. Fecha límite de entrega 25/06/21.

Transmisión de los datos correspondientes para el funcionamiento del servidor y el cliente. Responsables: Mathiw Rojas, Armando Fallas, Harold Espinoza. Fecha límite de entrega 25/06/2021.

Estructuras de datos

aliens[5][8]:
Se desarrolla una matriz para el almacenamiento de los aliens de tamaño 5x8 teniendo 5 filas de 8 aliens cada una, y así cada celda contiene un alien.
alien[0][0] = alien1

bunkers[4]:
Se utiliza un arreglo de 4 elementos para almacenar los bunkers, en esta lista el primer elemento corresponde al bunker 1 iniciando desde la derecha y así sucesivamente con el resto de bunkers :
bunkers[0] = bunker1

Datos[]
Se implementa un arreglo de tipo string que almacenará los datos recibidos del archivo .txt, en este arreglo cada elemento corresponde a un dato leído del archivo .txt
Datos[i] = “posx”

Algoritmos desarrollados

Inicio del juego y escritura de datos por parte del servidor
Mediante la conexión de sockets el servidor espera un cliente a la cual enviarle los datos de inicio del juego, cuando la conexión logra ser realizada con éxito el servidor llama a la clase nivel la cual es la encargada de iniciar todos los parámetros necesarios, cuando esto

termina se escriben los datos en un txt con un formato el cual separa los datos en líneas; los datos llevan el siguiente orden línea a línea: línea 1 posición en el eje X del jugador seguido de su vida, de la línea 2 a la 41 se escriben la posición de los 40 aliens, de la línea 42 a la 45 la vida de los bunkers, de la 46 a la 55 la posición de las balas enemigas (tanto de las que se deben mostrar en pantalla cómo las que están fuera de ella), y de la 57 a la 59 la posición de las balas del jugador (nuevamente tanto de las que están dibujadas cómo las que no).

Lectura de datos por parte del servidor
Los datos al llegar al servidor son escritos en un .txt, para interpretar estos datos desde el servidor se define una variable String[] Datos en el cual se hará un split en los datos del .txt para separarlos por cada espacio que se encuentre, así tendremos en cada índice de la variable Datos los datos de posiciones, puntuación y entre otros datos enviados por el cliente.

Lectura de datos por parte del cliente
El cliente está constantemente leyendo los datos del archivo “DatosR.txt”, una vez la lectura del archivo es almacenada en una variable de tipo FILE, como el cliente ya conoce el orden que llevan los datos, lee línea a línea por medio de ciclos (a excepción del primer dato) para poder separar el texto del los datos que ocupa el juego, cuando ya los logra separar se lo asigna a la variable del juego correspondiente.
Además al caber la posibilidad de que el archivo no este en el formato correcto por algún error con los tiempos de escritura y lectura, el algoritmo de lectura revisa que los datos estén correctos, en el caso de no ser así, deja los datos del juego sin modificar y vuelve a leer el archivo esperando que haya cambiado

Escritura de datos por parte del cliente
Primero se crea una variable cadena de tipo char que pueda almacenar 1500 caracteres, esta se va llenando poco a poco con los datos del juego con el formato ya mencionado anteriormente. Cuando ya se introdujo todos los datos dentro de la variable cadena, esta es enviada a una función la cual la escribirá en el archivo “DatosE.txt”.

Menú inicial del juego
Cuando el juego se inicia se dibuja sobre la pantalla un menú en texto el cual le da la opción al usuario de jugar o de observar. El usuario interactúa con este menú moviéndose de arriba a abajo y para seleccionar una opción presiona la tecla “Enter” o la tecla “Espacio”. El juego para detectar qué opción seleccionó el jugador, cada vez que se mueve hacia arriba coloca una variable llamada “menuTextSelected” en 0 y cada vez que se mueve hacia abajo en 1, luego al presionar enter el juego verifica el valor de la variable mencionada y si es 0 le carga al usuario el juego, en cambio si es 1 le dibuja en la pantalla al jugador los datos que estén almacenados en el archivo “DatosR.txt”.

Actualizar posicion del jugador
El juego está constantemente leyendo si se pulso alguna de las teclas de movimiento (“a” o “flecha izquierda” para moverse a la izquierda, y “d” o “flecha derecha” para moverse a la derecha), si este es el caso se mueve dentro de las variables del juego la posición en el eje x del jugador. Si el jugador se encuentra cerca de los bordes de la pantalla y se intenta mover más allá de estos, su posición será calculada para ser situado justo en ese borde.

Actualizar posición de los aliens
Los aliens iniciaran moviéndose de izquierda a derecha con una velocidad determinada (esta velocidad se incrementa tres veces dependiendo de la cantidad de aliens que queden vivos. La primera se ejecuta al quedar la mitad de los aliens, la segunda al quedar 10 y la última cuando solo queda 1 alien), al acercarse a algún borde se cambia la dirección de movimiento de estos y se les hace descender mientras estén a un rango de 10 posiciones de alguno de los bordes laterales.

Patrones desarrollados

Patrón Builder

Este patrón fue montado para la construcción de los distintos aliens dentro del juego, debido a que cuentan con parámetros distintos como el pulpo otorga 40 pts por su muerte, el cangrejo otorga 20 pts por su muerte y el calamar otorga 10 pts. Para iniciar la construcción del patrón se creó una interfaz llamada Builder en el cual se colocan los siguientes métodos mostrados en la imagen.

```
package ConstructorAlien;

/**
 * @author Armando Fallas
 * @version 0.1
 */
public interface Builder {
    void setPuntosMuerte(java.lang.Integer PuntosMuerte);
    void setposx(java.lang.Integer posx);
    void setposy(java.lang.Integer posy);
    void setvida(java.lang.Integer vida);
}
```

A partir de esta interfaz se crea la clase constructor del Alien en la cual se sobreescribe los métodos de la interfaz del builder y retorna el objeto creado del alien como en la siguiente imagen.

```
public Alien getResultado(){
    return new Alien(vida, PuntosMuerte, posx, posy);
}
```

Se tendrá una clase director la cual será la encargada de asignar los atributos a los objetos que se vayan a crear de los aliens, acá es donde tendremos los 3 tipos distintos de aliens mostrados como en la siguiente imagen.

```java
/**
 *
 * @author Armando Fallas
 * @version 1.0
 */
public class Director {

    public void Calamar(Builder builder){
        builder.setPuntosMuerte(10);
        builder.setposx(0);
        builder.setposy(0);
        builder.setvida(1);
    }

    public void Cangrejo(Builder builder){
        builder.setPuntosMuerte(20);
        builder.setposx(0);
        builder.setposy(0);
        builder.setvida(1);
    }

    public void Pulpo(Builder builder){
        builder.setPuntosMuerte(40);
        builder.setposx(0);
        builder.setposy(0);
        builder.setvida(1);
    }

}
```

Al crear un nuevo alien se debe realizar de la siguiente manera:

```java
Director director = new Director();

AlienBuilder builder = new AlienBuilder();
AlienBuilder builder2 = new AlienBuilder();
AlienBuilder builder3 = new AlienBuilder();

director.Calamar(builder);
director.Cangrejo(builder2);
director.Pulpo(builder3);
```

Se crean 3 variables builder a partir de la clase del constructor del Alien, cada builder corresponde a los 3 distintos aliens, a partir del director se llama a la función del alien que se desee crear y se les pasa el constructor, para obtener el alien simplemente se deberá llamar al builder de la siguiente manera: builder.getResultado() y lo almacenamos en la estructura de datos de nuestra conveniencia.

Patron Fachade

Este patrón se implementa mediante la clase “nivel” en la que se crean métodos y atributos para la manipulación de los elementos del juego. Para ello dentro de la clase se instancia al jugador, los bunkers y se implementa un método para instanciar la matriz de aliens, además se implementan funciones como:“UpdatePlayer()", “UpdateAliens()”,”getPPuntaje()”, entre otras las cuales son utilizadas para manipular los objetos instanciados previamente. Con esto basta con crear un objeto de tipo “nivel” para generar los elementos necesarios del juego. Esto se puede hacer de la siguiente manera:

```java
Nivel nivel = new Nivel();
```

Así con este objeto creado es posible manipular los elementos sin tener contacto directo con ellos, utilizando los métodos que proporciona la clase nivel, por ejemplo si se desea actualizar la posición y el puntaje del jugador se haría de la siguiente manera:

```
nivel.UpdatePlayer(PosicionX, PosicionY, Puntaje);
```

También si se desea crear un nuevo nivel con su matriz de aliens respectiva se hace de la siguiente manera:

```
nivel.CreateNewLvl();
```

Esto hace que sea más simple poder manipular los datos mediante un solo objeto, a diferencia de tener la necesidad de crear los objetos diferentes por separado y manipularlos uno por uno de forma individual.

Problemas sin solución

En la versión a dia de 25/06/2021 no se ha encontrado ningún problema que no se haya podido implementar una solución.

Problemas encontrados

1) Incompatibilidad entre la librería raylib usada para la interfaz gráfica con la librería usada en los sockets (sys/socket) al momento de compilar la aplicación, debido a que raylib debe compilar usando mingw pero los sockets deben compilar mediante cygwin y no había opción para compilar ambas cosas usando solo un compilador, por lo que se debió cambiar la librería de los socket en el cliente sustituyendo por la librería winsock que si es posible compilar usando mingw.

2) Los datos enviados del cliente hacia el servidor no lograban ser leídos y también se caía el servidor, mediante el debugger encontramos la línea que origina el error mostrada en la siguiente imagen:

   ```
   java.lang.String info = in.readLine().toString();
   ```

   Buscando en internet encontramos que el string que reciba el servidor al final debe tener “\n” debido a que el .readLine() debe encontrarlo para encontrar el final del string a leer, por lo que revisando en el código del cliente le asignamos la siguiente línea de código:

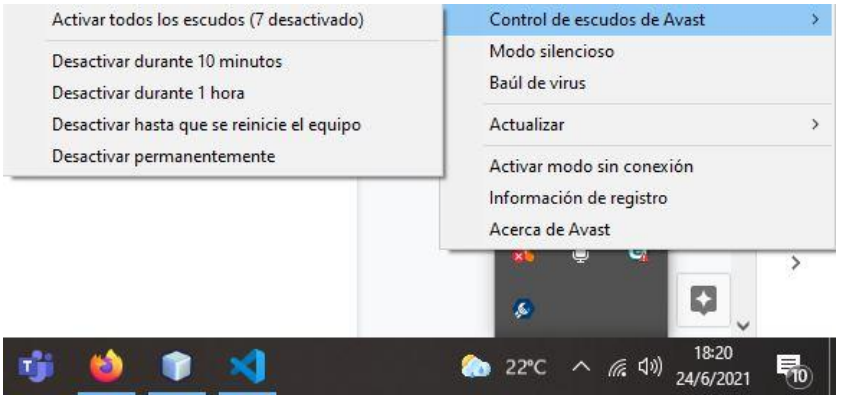
   ```
   data[i] = '\n';
   ```

   Ya con esto le asignamos al final de los datos el “\n” y el servidor ya logra interpretarlo.

3) Al momento de iniciar el servidor el cliente no lograba realizar la conexión con el socket, la solución fue sencilla pero muy difícil de percibir, ya que el error no era del código sino del antivirus instalado en el ordenador donde se compile dichos códigos, ya que el antivirus detecta que la conexión no es segura y no permite realizarla, en este caso el antivirus el cual originó el problema fue Avast por lo que simplemente se debe desactivar por un tiempo determinado para que así no logre intervenir en la conexión, los pasos a desactivar el antivirus Avast es la siguiente:

    1) Presionar la pestaña de iconos ocultos en la barra de tareas de windows

    2) Dar click derecho en el icono de Avast, presionar en el control de escudos y escoger por cuanto tiempo desactivar el antivirus..



4) A la hora de intentar escribir los datos en el .txt para el envío de datos por parte del cliente de C cuando se llamaba a la función de guardado ‘escribirTxt()’, cuando se intentaba obtener el dato de alguno de los elementos ocasionaba que el juego se cerrara de manera abrupta, para solucionarlo se juntan todos los datos en un char ‘cadena’ y posteriormente se llama a la función de guardado “escribirTxt()” pasándole como parámetro la variable ‘cadena’ con los datos necesarios, además se añade un sleep(0.05) para que la función tenga el tiempo necesario para escribir los datos

5) En el cliente al manejar todos los datos leídos en una sola variable de tipo “char*”, no se logró identificar mediante la función strtok() cuales caracteres almacenados le corresponden cual dato del juego. Esto se soluciono moviendo una variable archivo de tipo FILE línea a línea después de leer el archivo. Al tener posicionada esta variable en una fila, un vector tipo char de 32 posiciones copia su contenido para que ahora pueda ser procesado con la función strtok() la cantidad de veces necesaria.

6) A la hora del cliente observador leer los datos, puede que estos se encontraran en el formato incorrecto, lo cual provocaba un cierre del observador porque la vida del jugador se cambiaba a 0. Esto se soluciono agregando una línea que verificará que el primer token obtenido por la función strtok() sea diferente de null.

Conclusiones

En la elaboración de una interfaz en C está muy limitado en cuanto a librerías ya que muchas están enfocadas en c++ por lo cual no hay tanto recursos de donde poder elegir, en el caso de esta tarea ya se tenía una base montada usando la librería de Allegro ya que se suponía que funcionaba en el lenguaje de c pero al momento de compilarlo no funcionaba en c y solo en c++. No es un lenguaje en donde sea fácil crear interfaces gráficas.

Para el manejo y escritura de datos se debió elegir entre un txt o Json, se eligió usar un txt debido a la sencillez de sus métodos para escribir y leer datos además de que no es necesario instalar una librería, lo cual sucede con Json que a pesar de contar con un manejo de datos mucho más ordenado a un txt se tuvo una inseguridad con respecto a la instalación de su librería además de poder encontrar incompatibilidad con el compilador usado y así evitar errores en el desarrollo.

El uso de un patrón de diseño como el de “Fachade” puede hacer que se genere un código más simple y ordenado ya que solo es necesario manipular un objeto, en vez de manipular varios y distintos objetos de manera diferente lo que se puede hacer tedioso y complicado de manejar.

Recomendaciones

Compilar en windows utilizando mingw y revisar con anterioridad que las librerías estén correctamente instaladas, además revisar que la variable de mingw esté agregada al path.

A la hora de trabajar archivos en c los cuales se componen de diferentes datos en diferentes líneas, los cuales se desean convertir a un tipo de dato diferente de “char*”, se recomienda leer linea a linea del archivo original por medio del comando “fgets (char * str, int num, FILE * stream)”, para luego tratar los datos mediante el comando de “strtok(char * str, char * delim)”.

Manual de usuario

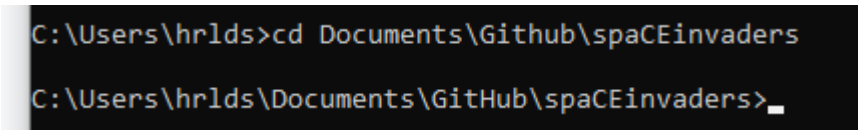
Requisitos del sistema

- Debe contarse con el SO Windows
- Tener instalado mingw gcc para poder compilar en gcc
- Tener Visual Studio Code para la manipulacion de codigo en C
- Se debe contar con NetBeans 8.2 o alguna extensión en Visual Studio Code para la manipulación de código en java
- Es necesario tener instalada la librería “Raylib” para la compilación del programa
- Es necesario contar con la librería winsock32

Compilación del Juego

Primero abra la consola de comandos, para ello desde la barra de búsqueda de windows ingrese “cmd.exe” y presione “enter”.

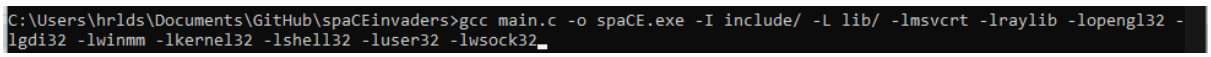
Una vez en el cmd diríjase a la ruta donde tiene guardado el juego, para ello utilice el comando cd + ruta. Ejemplo:



Posteriormente ejecute el siguiente comando para realizar la compilación del juego:

gcc main.c -o spaCE.exe -I include/ -L lib/ -lmsvcrt -lraylib -lopengl32 -lgdi32 -lwinmm -lkernel32 -lshell32 -luser32 -lwsock32

Ej:

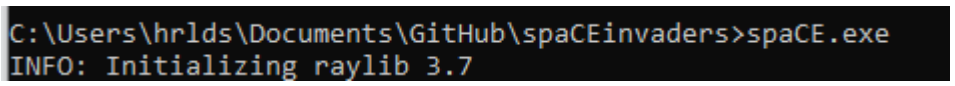


Para poder ejecutar este comando recuerde que debe tener añadida el directorio en la variable de entorno path.

Ademas, es importante que cuente con la librería Raylib para el proceso de compilación, de lo contrario podría generar errores.En caso de no contar con la librería diríjase a la página web https://www.raylib.com y proceda a realizar la descarga e instalación de la librería. También debe tener en cuenta la librería winsock32 en caso de tampoco contar con la misma, deberá seguir el proceso de instalación dependiendo de su equipo y la versión de mingw que tenga instalada.

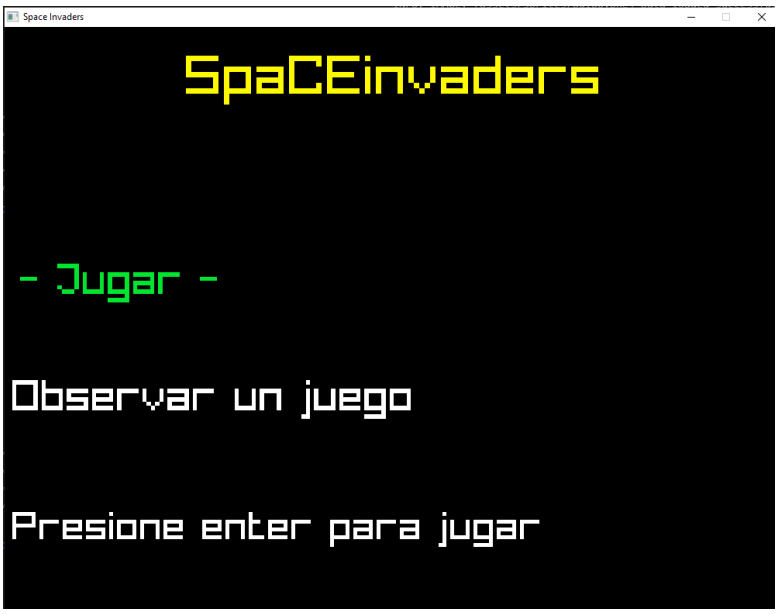
Ejecución del juego

Primero ejecute el archivo “servidor.exe” para la ejecución del servidor.
Posteriormente ejecute el ejecutable que creo en el proceso de compilación con el siguiente comando en la consola



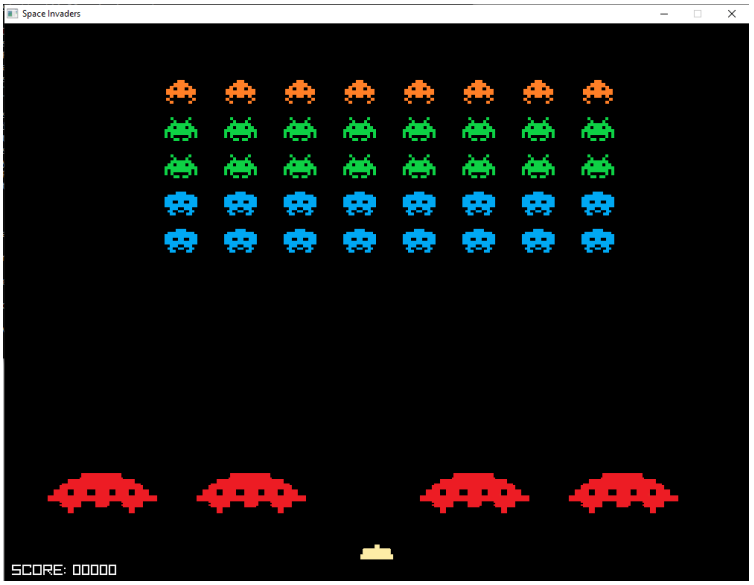
o bien inicie el ejecutable que se le proporciona en la carpeta con el nombre de “spaCE.exe”

Con esto se abrirá la pantalla principal del juego



Para poder iniciar el juego seleccione Jugar, y luego presione la tecla “enter”, con ello iniciara la partida.

Para poder observar una partida seleccione Observar un juego, y luego presione la tecla “enter”, con ello se mostrará en pantalla la partida de otro jugador.



Para poder jugar utilice las teclas:

a,d / ←,→ : para poder desplazarse de izquierda a derecha respectivamente

w / ↑ / space : para poder disparar a las naves enemigas.

Bibliografía

Raylib. (2017). Raylib. https://www.raylib.com/

Moon, S. (2020, 25 julio). *Winsock tutorial - Socket programming in C on windows*. BinaryTides. https://www.binarytides.com/winsock-socket-programming-tutorial/