Instituto Tecnológico de Costa Rica

Ingeniería en Computadores

Lenguajes, Compiladores e Intérpretes

Ice Climber

Estudiantes:
Jarod de la O - 2018147298
Mario Gudiño Rovira - 2017106391
Ignacio Carazo Nieto -2017090425
Joseph Jiménez – 2016133677

Profesor:

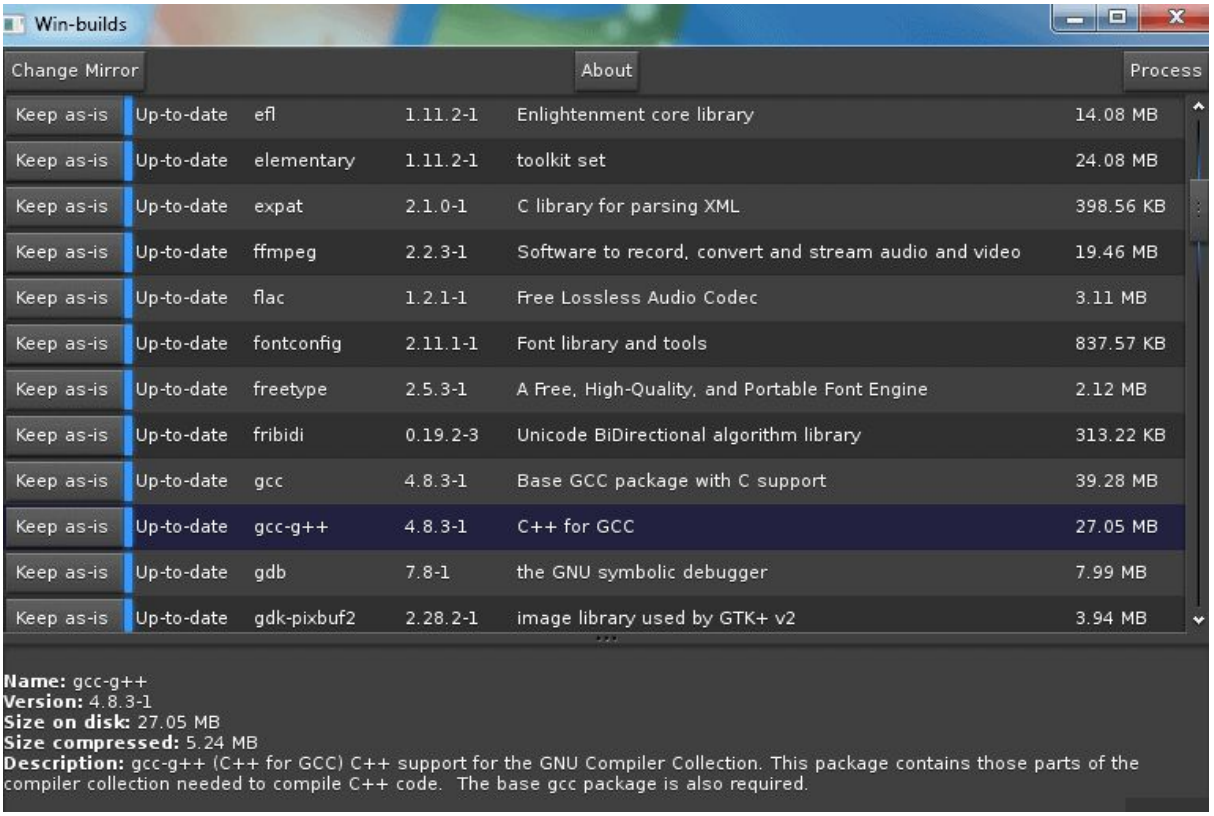
Marco Rivera Meneses

II Semestre 2020

Índice

Manual de Usuario

Para lograr crear el servidor por medio de **gcc serverListener.c -o serverListener -lws2_32 -ljson-c** se debe de descargar el winbuild el cual contiene el gcc y json-c que permite usar la biblioteca json-c



Para lograr usar el comando gcc, se debe hacer añadir lo instalado hasta el /bin/ a variables de entorno para sustituir el mingw. De ejemplo quedaría:
C:\Users\user\Documents\winbuild\bin

También es necesario crear instalar Link al tutorial:
https://www.youtube.com/watch?v=tuVjurLVPO4&ab_channel=NicolasBourr%C3%A9

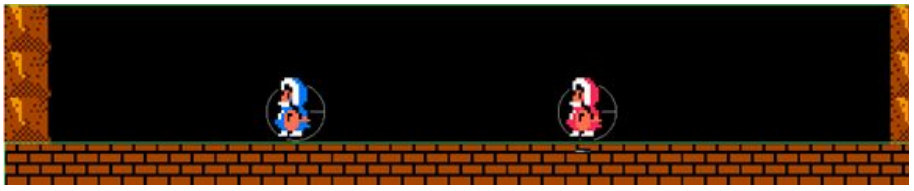
Ice Climber

Para poder utilizar el programa es importante primero hacer notar que para el lenguaje C se utilizó el IDE de JetBrains Clion (utilizando el compilador de gcc) y para el lenguaje de Java se utilizó IntelliJ. Se utilizó la librería de Json-c para la serialización y deserialización del Json para el lenguaje C, la cual fue instalada por medio de WinBuild 1.5.0 para evitar problemas de variables de entorno aun con la utiliza. La librería para desarrollo de videojuegos utilizada se llama LIBGDX y es necesario ver y seguir el siguiente tutorial para poder utilizar la librería en el proyecto.

Una vez iniciado el juego se tienen estas opciones donde se debe seleccionar la cantidad de jugadores y luego darle click a “PLAY”.

Jugadores

Si es seleccionado solo un jugador se utiliza al personaje Popo el cual puede moverse con las teclas W (salto), A (caminar izquierda), D (caminar derecha). Si se eligen dos jugadores el jugador 2 utiliza al personaje Nana la cual puede moverse con las teclas de las flechas. Flecha arriba (saltar), flecha izquierda (caminar izquierda), flecha derecha (caminar derecha). Si Popo o Nana caminan mas del limite de la pantalla a la izquierda o a la derecha pueden salir del otro lado de la pantalla



El puntaje de los jugadores puede verse arriba en la pantalla de la siguiente manera:

El puntaje puede aumentar de tres formas:

- Recogiendo frutas (varía dependiendo de la fruta)

- Eliminando enemigos (varía dependiendo del enemigo)
- Rompiendo bloques (50 pts cualquier bloque)

“Frutas”

Popo o Nana reciben puntos al entrar en contacto con alguna fruta

***Calabaza**: 600pts* ***Cebolla**: 300pts* ***Zanahoria**: 100pts*

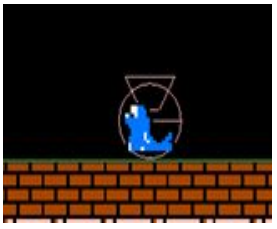


Enemigos

Si Popo o Nana saltan encima de un enemigo lo matan y reciben el puntaje respectivo. En caso de entrar en contacto con un enemigo de cualquier otra forma, el enemigo se “sacrifica” para bajarle la vida al jugador. El único enemigo que se mata de forma diferente es el pterodáctilo al que hay que golpear desde abajo.

***Pájaro:** 250pts* ***Oso:** 500pts* ***Yeti**: 300pts* ***Foca**:100pts*



***Pterodáctilo (jefe final):** 1500pts*

Estructuras de Datos desarrolladas

En los clientes se encuentran dos listas. Una de enemigos y otra de frutas. En estas se encuentran todos los enemigos y todas las frutas que actualmente existen en la pantalla. Es decir si hay dos osos y un yei en la pantalla, estos se encuentran en la lista “enemies” y de igual forma funcionaria con las frutas.

```
private Array<Enemy> enemies;

private Array<Fruit> fruits;
```

La función de estas listas es que a la hora de crear un enemigo o una fruta nueva estos se agregan a su lista correspondiente y la interfaz funciona de forma que solo se “dibuja” lo que hay dentro de estas listas. De esta forma facilita el manejo de lo que se encuentra en la pantalla.

En el manejo del servidor en la parte de los sockets utiliza la estructura para guardar información para la creación del socket.

```
WSADATA wsData;
SOCKET listening;
struct sockaddr_in server;
char* message;
int i;
```

```
//Llenando estructura sockaddr_in
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( hostshort: 8888);
```

También para el manejo de la estructura de en el server se utiliza un struct y los punteros el cual sirve para el parseo del mensaje entrante y también identifica el id proporcionado por el cliente.

```
struct json_object *parsed_json;
struct json_object *id;
size_t n_items;
int i;
```

Algoritmos Desarrollados

No se desarrollo un algoritmo específico a parte de la solución total planteada

Problemas Conocidos

Debido a circunstancias desconocidas y luego de una serie de debuggear sin encontrar una solución un problema conocido tiene que ver con las colisiones de objetos dentro del juego y al Popo o Nana pasar de un lado a otro de la pantalla. Si alguno de los personajes cruza de un lado a otro de la pantalla las colisiones con los objetos dentro de la pantalla dejan de funcionar, sin embargo si vuelven a hacer la misma acción 2 veces se arregla este error. Otro problema es en la pantalla del menú principal donde por diferencia de hardware más que todo de dimensiones de la pantalla de cada computador se desfasan los botones y se complica la selección de estos.

Plan de Actividades

Cliente, interfaz gráfica en java

Ignacio Carazo:

Crear jugador 1, crear jugador 2, como máximo 3 jugadores.

Deberá mostrar la puntuación del juego

Crear nivel de interacción entre los jugadores, dividido por varios pisos, conforme va subiendo el nivel más bajo debe perderse.

Permitir que los jugadores salten y rompan los bloques del piso superior.

Haziel:

El jugador puede perder vidas si no cae en el piso o si colisiona con algún enemigo.

Crear enemigos: focas, yetis, aves, hielos, osos polares

Crear último piso bonus:plataforma no rompible, recoger verduras,permitir saltar para agarrar las patas de un pterodáctilo que vuela de lado a lado del escenario, al llegar a la fase de bonus el juego le otorgara una vida y el juego inicia desde el principio, pero la velocidad con la que los obstáculos de desplazan es mayor.

Servidor (debe ser implementado en C)

Joseph:

La conexión entre el servidor y los clientes debe realizarse utilizando Sockets.

La creación de los Obstáculos será desde consola/app por un usuario y el decidirá qué tipo de obstáculo es.

Jarod:

El cliente Observador que se logre unir a una partida existente pero sólo podrá observar lo que sucede. Máximo 2 observadores por juego.
Yetis/Focas: salen en cada piso, una dirección de donde salen Izquierda o Derecha
Aves: Salen siempre de la izquierda, pueden subir o bajar niveles.
Hielos: Salen de un piso superior y caen en línea recta se debe especificar el piso del que sale y la ubicación en ese piso.
Creación de frutas/verduras: 4 frutas y solo se permite en la fase de Bonus, cada verdura tendrá un valor (naranjas 100, bananos 200, Berenjenas 300, Lechugas 400)

Problemas Encontrados

Se buscaba que se pudiese compilar el servidor c por medio del comando gcc pero a su vez vinculandolo con la biblioteca json-c para lograr realizar el desarrollo de las estructuras json, se intentó hacer una referencia directa en los parámetros del llamado en gcc hacia la biblioteca de diferentes maneras, se hizo uso del manejador de paquete vcpkg, se intentó vincular con cmake, se trato de instalar la librería faltante en mingw y así otros varios intentos con el fin de lograr usar la biblioteca. Como solución encontrada fue un paquete llamado winbuild, el cual viene con archivos g++, gcc y la biblioteca json-c que era necesaria para la compilación del proyecto en C. Como recomendación es buscar una especie de manejador de bibliotecas que facilite la vinculación de bibliotecas, en conclusión para hacer referencia a la biblioteca C si no se encuentra en mingw, se ocupa un manejador de bibliotecas o un idle que logre referenciarlas con mayor sencillez

Win-builds.org. 2020. *(Free) Software Building And Packaging For Windows [Win-Builds]*. [online] Available at: <http://win-builds.org/doku.php> [Accessed 19 December 2020].

Conclusiones y Recomendaciones

En conclusión el manejo de bibliotecas con lenguajes de bajo nivel puede ser un poco complicado por lo cual se recomienda el dominio de herramientas enlazadoras de bibliotecas con el compilador para evitar problemas varios

Para entender cómo funciona el paradigma de la programación orientada a objetos es necesario ver un como una colección de objetos que interactúan entre sí enviándole mensajes y cambiando su estado durante la ejecución esto implica determinar y caracterizar los diferentes objetos que intervienen en el problema, definir sus propiedades y métodos y ponerlos a interactuar.

Bibliografía Consultada

- Moon, S., 2020. *Winsock Tutorial - Socket Programming In C On Windows*. [online] BinaryTides. Available at: <https://www.binarytides.com/winsock-socket-programming-tutorial/> [Accessed 06 December 2020].
- Moon, S., 2020. *Handle Multiple Socket Connections With Fd_Set And Select On Linux*. [online] BinaryTides. Available at: <https://www.binarytides.com/multiple-socket-connections-fdset-select-linux/> [Accessed 09 December 2020].
- Science, C., 2020. *Swarthmore Academics*. [online] Cs.swarthmore.edu. Available at: <https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_C_libraries.html> [Accessed 12 December 2020].
- Hathibelagal, A., 2020. *JSON-C Tutorial: How To Parse JSON In C*. [online] Progur!. Available at: <https://progur.com/2018/12/how-to-parse-json-in-c.html> [Accessed 16 December 2020].
- Rai, A., 2020. *Java Gson + JSON Tutorial With Examples*. [online] Concretepage.com. Available at: <https://www.concretepage.com/google-api/java-gson-json-tutorial-examples> [Accessed 17 December 2020].
- Programming in Linux. 2020. *Json_Object_New_Object: Creating A New Json Object*. [online] Available at: <https://linuxprograms.wordpress.com/2010/08/19/json_object_new_object/> [Accessed 18 December 2020].
- Json.org. 2020. *JSON*. [online] Available at: <https://www.json.org/json-en.html> [Accessed 18 December 2020].
- Code School, B., 2020. *Libgdx Game Development With Android Studio*. [online] Youtube.com. Available at: <https://www.youtube.com/watch?v=a8MPxzkwBwo&list=PLZm85UZQLd2SXQzsF-a0-pPF6IWDDdrXt> [Accessed 19 December 2020].

Bitácora Digital

Estudiante	Actividad	Descripción	Tiempo(h)	Fecha
Joseph J	Comunicación Cliente-Servidor	Conexion socket del server que acepta varios Clientes	3	06/12/2020
Joseph J	Comunicación Cliente-Servidor	Envío de Data a varios clientes	2	10/12/2020
Joseph J	Comunicación Cliente-Servidor	Creación de Sockets en cliente	1	11/12/2020
Joseph J	Comunicación Cliente-Servidor	básico de lectura de datos json en servidor-cliente	4	17/12/2020
Joseph J	Comunicación Cliente-Servidor	básico de escritura de datos json en servidor-cliente	5	18/12/2020
Ignacio C	Interfaz Gráfica	Investigación sobre libreria para game development en Java	1	09/12/2020
Ignacio C	Interfaz Gráfica	Inicio de código para la interfaz gráfica. (menú principal, mapa principal, colisiones de popo y nana)	8	10-11/12/2020
Ignacio C	Interfaz Gráfica	Juego multijugador local. Factory de enemigos y de frutas. Colisiones con enemigos y con los bloques. Se inició el aumento de los scores y la colisión con frutas, también el paso de un lado de la pantalla a otra.	10	11-12/12/2020
Ignacio C	Interfaz Gráfica	Documentación interna del código. Paso de pantalla de un lado a otro de popo y nana. Colisiones con frutas y enemigos. Score y vidas de los personajes.	11	17-18/12/2020

Ignacio C	Interfaz Gráfica	Pantalla de game over y algunos otros arreglos dentro de la interfaz.	4	19/12/2020
Jarod	Servidor	Funciones de creación de Objetos para el juego	7	10/12/2020
Jarod	Servidor	Investigación sobre uso correcto de threads en C	3	12/12/2020
Jarod	Servidor	Investigación, corrección y instalación de librerías en windows (gcc y Json-c)	9	14/12/20
Jarod	Servidor	Documentación y estandarización relacionada al servidor	3	17/12/2020
Haziel	Cliente	Actualización de pintado de interfaz	2	10/12/2020
Haziel	Cliente	Manejo de datos analizados en el json	3	17/12/2020
Haziel	Comunicación Cliente-Servidor	Conexión de interfaz analiza json y sockets	4	18/12/2020