Instituto Tecnológico de Costa Rica

Semestre I

2021

DonCEy Kong Jr.: Documentación técnica

Lenguajes, compiladores e intérpretes

Tarea#3: Programación imperativa y orientada a objetos

Prof. Marco Rivera Meneses

José Alejandro Chavarría Madriz

2019067306

Natalia González Bermúdez

2019165109

Índice

Estructuras de datos desarrolladas

Arrays

C:

Para el caso del cliente se utilizaron arrays para el control de la interfaz gráfica. Específicamente dos array de tamaño MAX_ELEM que contienen las variables de tipo HWND con las imágenes que se mueven en la interfaz. Cada ciclo de refrescamiento de la interfaz son eliminadas las imágenes de HWND past[], los contenidos de HWND elements[] son cargados a past[] y las nuevas imágenes a elements[].
Adicionalmente se utilizaron "matrices" de char* para representar arrays de strings, que son utilizados al momento de parsear los mensajes del servidor. Una explicación a profundidad se da en la siguiente sección.

Java:

En el servidor los arrays se utilizan para guardar información que se debe mostrar en la interfaz gráfica. Se utilizan tres arrays distintos en la clase App que guardan los datos en los JSpinners de la interfaz: gameObj[] para guardar los objetos que se pueden eliminar o crear como enemigos azules, enemigos rojos o frutas, gameOpt[] para guardar las opciones que tiene el usuario con respecto a los objetos como eliminar o crear y por ultimo gameLiana[] que guarda las lianas en las que se pueden crear o eliminar dichos objetos en forma de enteros de 1-10.

Listas

Java:

Se utilizaron listas enlazadas en el servidor para guardar los objetos que se van creando. En la clase Juego se utilizaron para guardar objetos de tipo Cocodrilo, es decir enemigos rojos y azules, de tipo Fruta y de tipo Liana. Cada vez que el usuario crea un nuevo enemigo o una nueva fruta, se almacenan en la lista cocodrilos o frutas respectivamente con las funciones

crear_rojo, crear_azul y crear_fruta y se elimina los elementos de la lista con eliminar_cocodrilo y eliminar_fruta. En el caso de las lianas, se crean en el constructor de la clase Juego y se almacenan en la lista lianas.

Algoritmos desarrollados

Creacion y actualizacion de interfaz

El funcionamiento del cliente puede ser descrito mediante tres procesos individuales:

Inicialización de la interfaz

Mediante el uso de la librería Win32 API se crea una ventana, son cargados todos los elementos (botones, imágenes y un timer interno). Y una vez listos son desplegados al usuario. Allí se presentará al usuario una pantalla de selección entre jugador o espectador.

Inicialización del socket

Una vez seleccionado el modo de juego se lleva al usuario a la selección de servidor. Al escoger se define un flag que determinara el puerto al cual escuchará el socket. En ese instante el socket es creado. En caso de éxito de conexión enviará un primer mensaje identificándose como jugador o espectador, con el fin de que el servidor evalúe si es posible aceptarlo en el servidor, o si este ya se encuentra lleno o hace falta el jugador y se intenta conectar un tercer espectador.

Refrescamiento de la interfaz

Una vez verificada la conexión y con el visto bueno del servidor el flag de conexión es activado, lo que abre lugar a los ciclos de refrescamiento de la interfaz. Según el timer definido durante la primera etapa (en este caso definido para unos 50ms, dando origen a 24fps), cada vez que se termine se enviará al servidor un query pidiendo la información de la partida actual, el servidor responderá con un string que contiene la posición de todos los elementos de la partida. El cliente procederá entonces a eliminar todos los elementos pasados y mostrar las nuevas posiciones de cada elemento y aquellos que sean nuevos. Adicionalmente si es un jugador durante este periodo las teclas WASD y flechas del teclado están habilitadas y envían mensajes al servidor para que actualice su posición.

Parser de string lado cliente

Para mostrar cada elemento enviado por el servidor en la posición correcta se ha condensado esta información en un string de la siguiente forma:
“Vidas;Puntos;PlayerX,PlayerY;Color1,Enm1X,Emn1Y:Color2,Enm2X,Emn2Y:...;Fruta1X,Fruta1Y : Fruta2X , Fruta2Y : ...”

Primeramente el string es separado por cada “;” que hay. Es cargada, entonces, cada parte a un array de strings donde respectivamente se tienen: las vidas, los puntos, la posición del jugador (separada por una “,” (x,y)), los enemigos (separados por “:”) y las frutas (separados por :). Una vez hecho esto se comienza a recorrer el array. Los primeros dos elementos son cargados tal y como vienen a la interfaz. Para la posición del jugador el substring es separado en sus partes separadas por la “,”, cada parte corresponde a la posición, seguidamente son convertidas a un entero es cargada la imagen correspondiente del jugador (según su posición y ciclo de animación). Seguidamente tanto para los enemigos como para las frutas estos substrings son separados en cada parte dividida por un “:” y vueltos a separar después para la “,” así se extrae la posición y en caso de los enemigos el color también.

Para mayor claridad un ejemplo real de tal string seria:

LIVES:1 ; POINTS:2050 ; 50, 180 ; 1, 80 , 470 : 2 , 240 , 100 ; 20 ,110

Entonces tendremos:

- 1 vida
- 2050 puntos
- Jugador en (50,180)
- Un enemigo rojo en (80,470)
- Un enemigo azul en (240,100)
- Una fruta en (20,110)

Lógica del juego con la interfaz del servidor

Inicialización del servidor

Cuando el servidor inicia aparece una interfaz para controlar dos juegos de DonCEy Kong Jr. Cada uno se inicia con la clase Juego que tiene un jugador, los puntos de la partida, la velocidad de los enemigos dependiendo del nivel, una lista vacía de enemigos, una lista vacía de frutas y una lista con 10 lianas que se crean automáticamente con la función crear_lianas().

Creación y eliminación de objetos desde la interfaz

En la interfaz hay tres spinners para seleccionar como modificar el juego: Uno para decidir si se quiere eliminar o crear un objeto (spinner de opción), otro para decidir el tipo de objeto (spinner de objeto) y el último para seleccionar la liana en la que se quiere crear (spinner de liana). También hay un slider para indicar la posición de la liana en la que se quiere crear el objeto donde 0 es el inicio de la liana y 100 es el final. Por último hay un botón para confirmar la acción y enviarla al jugador.

Al crear una fruta en la liana X(spinner de liana) en la posición Y(slider) se llama a la función crear_fruta(X, Y) que convierte el valor de Y que va de 0-100 en un valor de Y relativo a la liana X, crea una fruta en esa posición y la agrega a la lista de frutas del juego. Al eliminar esa fruta se llama a la función eliminar_fruta(X, Y) que busca en la lista de frutas una que se encuentre en esa posición y la elimina.

Al crear un enemigo rojo o uno azul en la liana X en la posición Y se llama a la misma función crear_cocodrilo (Tipo, X, Y), el tipo es “1” si se quiere crear un enemigo rojo y “2” si se quiere crear uno azul. La función crear_cocodrilo llama a la función crear_rojo(X, Y) y crear_azul(X, Y) respectivamente que convierten el valor de Y a uno relativo a la liana, crea un nuevo objeto(Rojo o Azul) en esa posición y lo agrega a la lista de cocodrilos. Para eliminar el enemigo se llama la función eliminar_cocodrilo(Tipo, X) se busca, en la lista cocodrilos, un enemigo de ese tipo en esa liana y se elimina.

Manejo de respuestas del servidor

Mensaje default

Cuando el cliente le pide al servidor toda la información del juego al servidor se llama a la función game_string() de Juego que se encarga de construir el string principal del juego y llama a las funciones que mantienen al juego en movimiento. El string se construye con las vidas del jugador, los puntos del juego, la posición del jugador y las funciones cocodrilos_str() y frutas_str().

Para armar el string del juego primero se toman las vidas y los puntos, seguidamente se verifica si el jugador ganó con la función won() que si el jugador está en la misma posición que la llave, vuelve al jugador a su posición inicial, suma 1000 pts y aumenta la velocidad de los enemigos. Después de esto, se toma la posición del jugador y se llama a la función cocodrilos_str() que recorre la lista cocodrilos, verifica si hay una colisión con el jugador y se asegura de eliminar a los enemigos que se salen de la pantalla y después va formando un nuevo string con la posición actual de cada uno y llama a la función de cocodrilos move() para que estén moviéndose constantemente. Por último se llama a la función frutas_str() que verifica si hay una colision con el jugador para eliminar la fruta y sumarle puntos y si no va formando un string con las posiciones de las frutas. La unión de todos estos strings se envía al cliente cada 50ms para que actualice su interfaz.

Movimiento del jugador

Cuando el cliente especifica una tecla WASD se llama a la función de juego move_player(command) que se encarga, de verificar si el jugador puede realizar ese movimiento y si se puede llama a las funciones de jugador move_up(), move_down o move_sideways() que cambian la posición del jugador según la tecla que reciba.

Diagramas

Diagrama de arquitectura

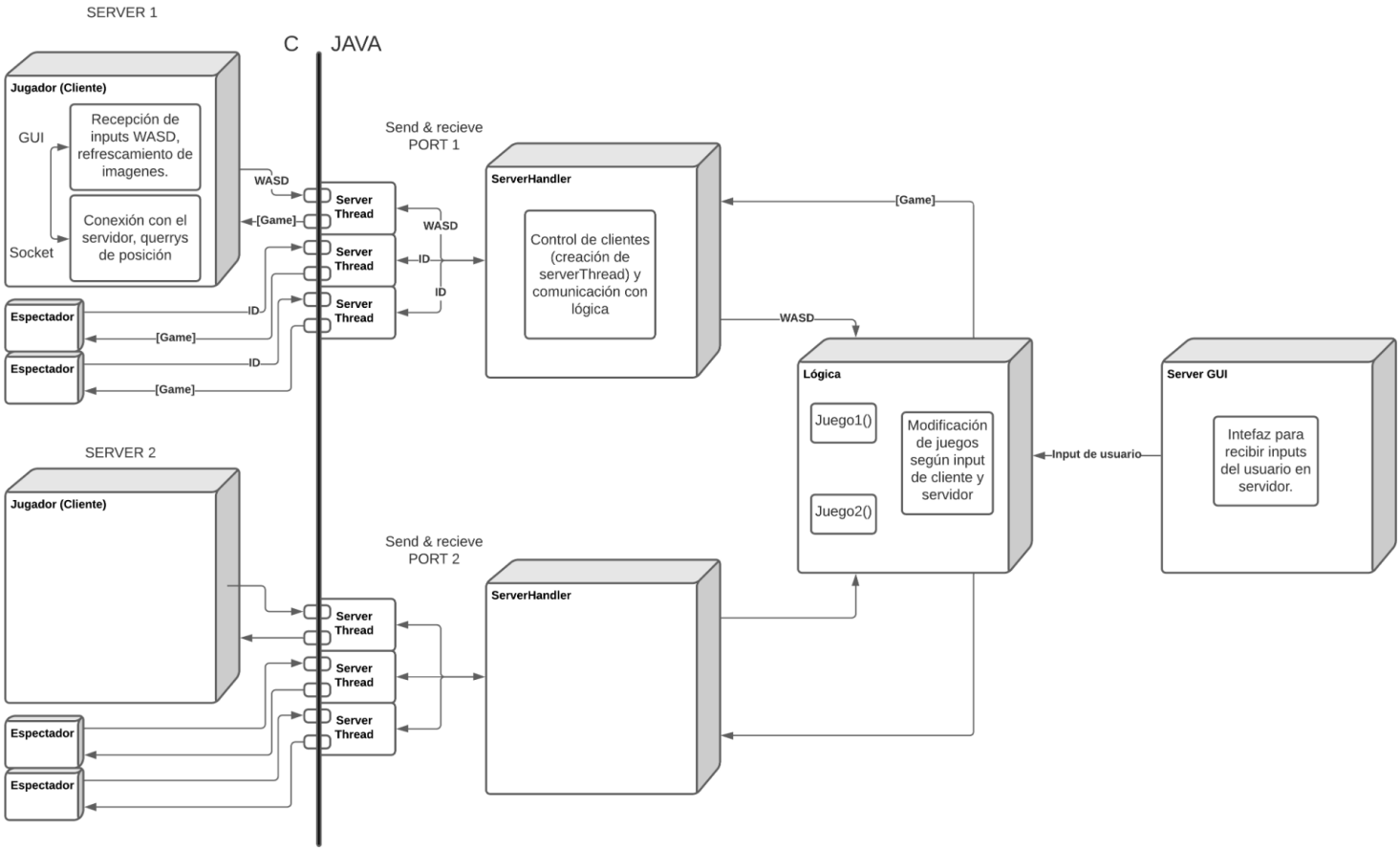
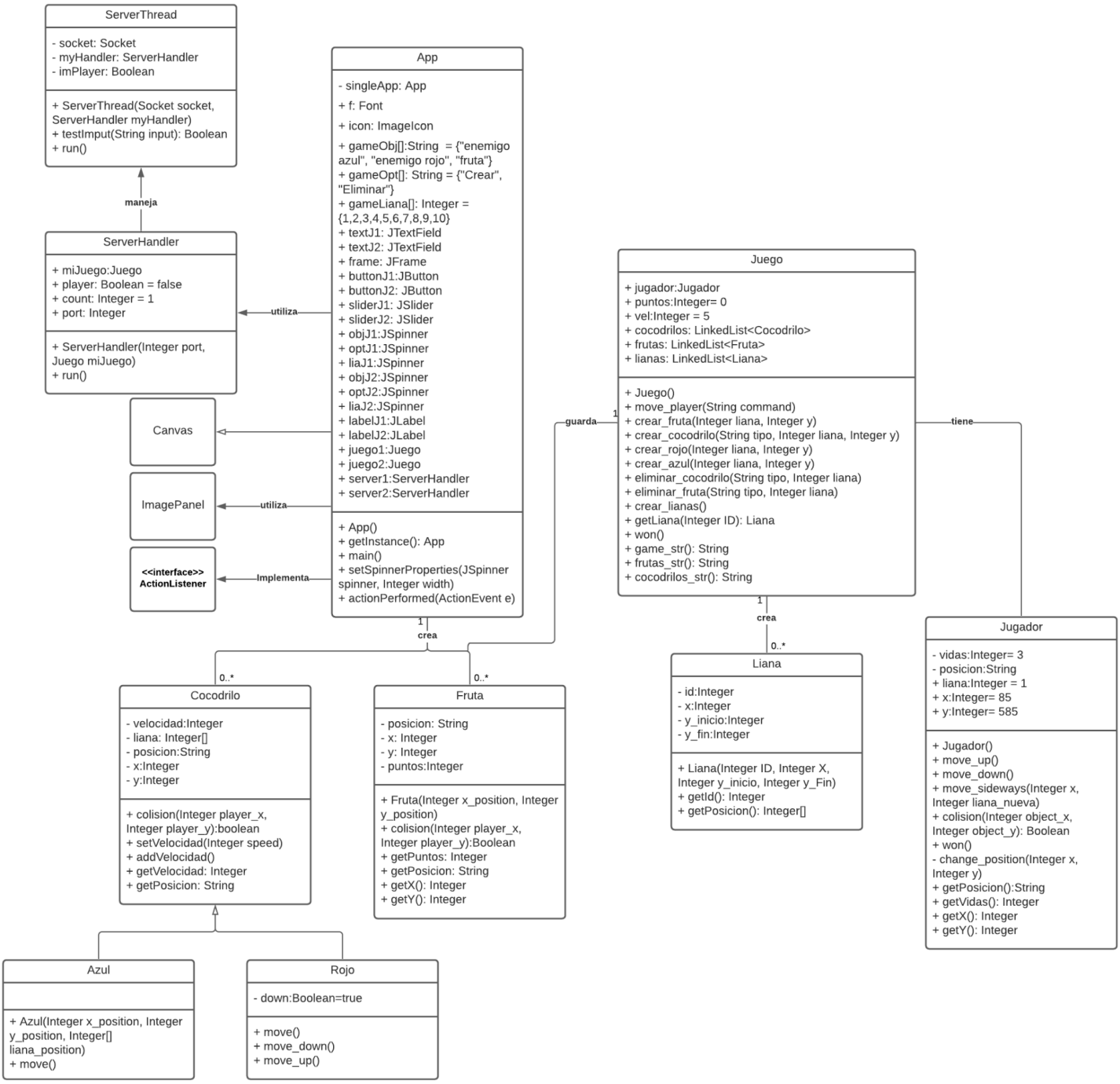


Diagrama de clases



Problemas sin solución

Utilizar imágenes .png sin fondo en Win32

Dados los métodos utilizados para crear y controlar imágenes implementados no fue posible lograr utilizar un formato de imagen diferente a .bmp, lo cual presenta el enorme inconveniente de que estas no poseen un fondo transparente, lo que impacta muy gravemente la calidad visual de la interfaz del cliente. A pesar de los extensivos intentos por implementar imágenes .png no se logró encontrar un método lo suficientemente rentable en tiempo y complejidad. Por lo que se tomó la decisión de mantener las imágenes .bmp en detrimento de la calidad visual.

Plan de actividades

Descripción de la Tarea	Duración Aproximada	Encargado	Fecha de entrega
Creación de un servidor en java	2h	Jose	15/4/21
Creación de sockets para conexión Java-C	2h	Jose	15/4/21
Implementación de 2 patrones de diseño	3h	Natalia	24/4/21
Creación de archivo de constantes	1h	Jose	15/4/21
Creación de la clase Juego	1h	Natalia	17/4/21
Creación de la clase Jugador	1h	Natalia	17/4/21
Creación de la clase Liana	1h	Natalia	17/4/21
Creación de la clase padre Cocodrilo	1h	Natalia	17/4/21
Creación de la clase hija Rojo	1h	Natalia	17/4/21
Creación de la clase hija Azul	1h	Natalia	17/4/21
Creación de la clase Fruta	1h	Natalia	17/4/21
Funcion de deteccion de colisiones	2h	Natalia	20/4/21
Implementación de interfaz gráfica de en C (con animación)	4h	Jose	18/4/21

Definición de entradas esperadas por la interfaz (C)	2h	Jose	17/4/21
Creación del cliente jugador	2h	Jose	21/4/21
Creación del cliente observador	2h	Jose	21/4/21

Problemas encontrados

El socket de Java no recibe mensajes de c

Al iniciar el socket en Java y mandar un string desde C este era completamente ignorado por el servidor. Sin embargo al leer la documentación del socket en Java se descubre detectara todo string finalizando en "\n". Se prueba entonces agregar "\n" al final de los strings enviados por el cliente y se logra la comunicación deseada.

Parpadeo al eliminar imágenes en Win32

Al eliminar y mostrar imágenes rápidamente se presenta un parpadeo importante, un cuadro blanco aparece en la interfaz cada vez que es eliminada la imagen. Para arreglarlo fue necesario agregar la sentencia case WM_ERASEBKGND: retornando true, dentro del loop principal de la ventana de win32.

Mantener dos partidas distintas al mismo tiempo

Para resolver la necesidad de dos partidas distintas se decidió crear una clase ServerHandler que crea un thread que activamente escucha solicitudes de conexión para un puerto específico. De recibir una solicitud se crea un ServerThread para manejar los mensajes de ese cliente en específico. ServerThread tiene una referencia al Serverhandler que lo genera, de esta forma El serverHandler puede ser modificado por las instancias de ServerThread y así ejercer el control sobre cuantas más solicitudes de cliente acepta o si alguno de los clientes se ha desconectado. Para este caso en particular se crean al inicializar la aplicación del servidor dos instancias de ServerHandler esculachando a los puertos 1108 y 802, cada uno capacitado para aceptar 3 clientes con un mínimo de 1 jugador por puerto.

Parsear el string que se recibe del server

Al comienzo para parsear el string se utilizaba el resultado de strtok pero por retornar un puntero al manipular el resultado de dicho puntero se perdía la referencia, por lo que se utilizan arrays

guardar todos los punteros creados al recorrer todos los resultados de strtok y luego operar sobre los array.

Mostrar las imágenes en la interfaz del servidor

Las imágenes de la interfaz del servidor no se lograban mostrar de la forma que se quería ya que en lugar de estar en el fondo se colocaban a la par del resto de componentes gráficos. Para resolver este problema se da ImagePanel derivada de la clase JPanel de java swing que dibujaba la imagen en el panel al crearlo.

Mantener la velocidad de los enemigos

Cuando un jugador gana una partida, la velocidad de los enemigos debe aumentar pero inicialmente solo aumentaba la velocidad de los enemigos existentes y no los nuevos que se creaban. Para resolver este problema se cambió la variable velocidad y se hizo estática.

Movimiento del jugador

El movimiento del jugador tenía varios problemas. Lo primero es que aunque su movimiento en X si estaba restringido para que solo se moviera de una liana a otra, su movimiento en Y le permitía moverse sin restricciones. Para resolver esto se agregó un atributo Liana a la clase Jugador para saber en cual liana se encuentra y se agregó una condición para que solo se moviera en el rango de inicio y final de dicha liana.

Otro problema era que al ganar aunque si se devolvía a la posición inicial, hacía unos movimientos inesperados al presionar las teclas WASD hasta que volvía a la normalidad después de presionarlas algunas veces. El problema fue que en la función para resetear la posición no se estaba cambiando el valor de la liana en la que se encontraba entonces el valor de X quedaba guardado de forma incorrecta.

Conclusiones

C:

- Es posible crear interfaces gráficas en windows el el lenguaje C utilizando la librería de windows Win32 API.
- Win32 es un API que a pesar de su avanzada edad (lanzada en 1993, pero con sus orígenes en 1985) aún mantiene su relevancia, pero es por esta misma característica puede resultar particularmente difícil de utilizar comparada con librerías mucho más nuevas.
- Es posible representar un array de strings como una matriz de o array de array tipo char*.
- Los punteros son una herramienta sumamente útil pero también son causantes de muchos errores.
- Crear estructuras permite organizar conjuntos de datos relacionados bajo un único nombre.
- Se puede mantener un archivo .h con todas las constantes y definiciones de funciones y estructuras para facilitar la organización y mantenimiento del código.
- Es posible crear sockets en windows utilizando la librería de C winsock2.

Java:

- La librería Java Swing es muy útil para crear interfaces gráficas en java por su gran variedad de componentes.
- Se puede utilizar la herencia para crear nuevos componentes gráficos utilizando la librería Swing.
- Se pueden utilizar listas y arreglos para almacenar datos de una partida.
- Con la programación orientada a objetos se puede programar la lógica de un juego de manera ordenada.

Recomendaciones

C:

- Si desea crear una interfaz gráfica en C para windows es posible utilizar la librería Win32, pero se ha de tener en cuenta su antigüedad, por lo que se recomienda disponer tiempo para poder comprender previamente el uso de la librería.
- Los punteros son una herramienta que debe usarse con cuidado. Es vital comprender su funcionamiento.
- Organizar bloques de datos relacionados en estructuras para mejorar la legibilidad del código.
- Utilizar archivos .h para las constantes. Esto con el fin de mayoir facilidad de mantenimiento y organización.
- Utilizar la librería de C winsock2 cuando se desea implementar sockets simples.

Java:

- Utilizar listas enlazadas de Java cuando se quiere almacenar datos sin tener que establecer un límite de elementos.
- Utilizar arrays cuando se quiere almacenar datos estáticos o una cantidad conocida de datos
- Tener bien definidas las clases y los objetos que se quieren crear para poder establecer una buena estructura.
- Realizar un código modular para facilitar cualquier cambio en el futuros
- Utilizar la librería Swing de Javax si se necesita hacer una interfaz gráfica simple ya que tiene componentes como botones, spinners, sliders, labels, etc.

Bibliografia

CodeJava. (2019). Java Socket Server Examples (TCP/IP).
https://www.codejava.net/java-se/networking/java-socket-server-examples-tcp-ip

Java2s. (s.f). Java Swing Tutorial - Java Swing Intro.
http://www.java2s.com/Tutorials/Java/Java_Swing/index.htm

Microsoft. (2018). Keyboard Input (Get Started with Win32 and C++).
https://docs.microsoft.com/en-us/windows/win32/learnwin32

Oracle. (s.f). Lesson: Using Swing Components.
https://docs.oracle.com/javase/tutorial/uiswing/components