**Estructura del informe:**

* **Objetivos**
* **Ideas**
* **Procedimientos**
* **Explicación de funciones**
* **Finalización**

**Objetivos:**

**-Crear software de cualquier tipo (juego, app de escritorio, app web, algún tipo de servidor, etc…) que implemente los conocimientos adquiridos en clase de informática 1 en lenguaje de programación C.**

**-Los conocimientos mínimos que hay que emplear en el software son:**

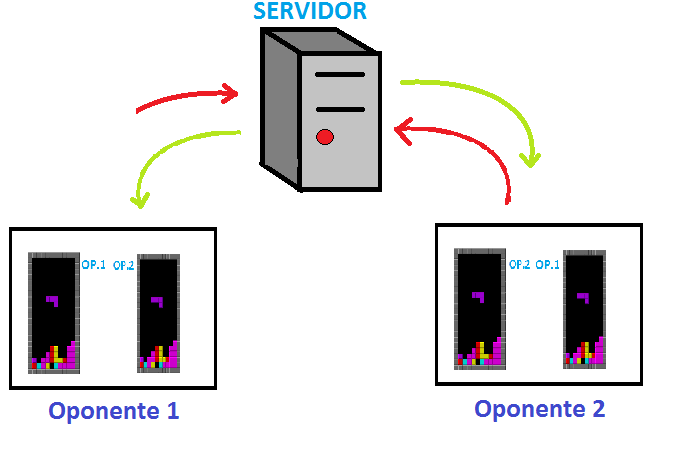
* **Uso de variables y su aritmética**
* **Tipos y tamaños de datos**
* **Funciones**
* **Prototipos**
* **Archivos de cabecera**
* **Operadores relacionales y lógicos**
* **Operadores de autoincremento**
* **Proposiciones (switch, while, do while, for…)**
* **Conversiones de tipo de datos**
* **Variables Punteros y sus operadores**
* **Álgebra de punteros**
* **Funciones con punteros como argumentos**
* **Puntero a función**
* **Strings**
* **Argumentos del main**
* **Estructuras**
* **Reserva dinámica de memoria[ malloc() , free() ]**
* **Y lo más importante:**
* **Listas**
* **Socket**
* **Librerías graficas (Open CV, Allegro en nuestro caso...)**

**Ideas:**

**Un poco de historia: Nuestra idea fue basarnos en el famoso juego de puzzle llamado Tetris que fue lanzado por primera vez el 6 de junio de 1984 por Alekséi Pázhitnov que trabajaba para el Centro de Computación Dorodnitsyn de la Academia de Ciencias de la Unión Soviética en Moscú ,interesante. Volviendo al tema de nuestra idea…**

**Queremos crear una versión del Tetris donde se puedan jugar por streaming en una red LAN, viendo tu propio campo de juego mientras también ves el campo de juego de tu oponente, teniendo información de tu nombre, todo en tu misma pantalla.**

**Tú oponente ve exactamente lo mismo que tu, los datos de ambas partidas se pasaran mediante un mini servidor que recogerá dichos datos y los enviara de un oponente a otro y viceversa.**

**Para hacer el servidor nos concentramos más que nada en el uso de funciones socket, donde el servidor habrá que iniciarlo primero y quedara esperando a que inicien ambos clientes que jugarán entre ellos, una vez que ambos clientes estén iniciados, el servidor permitirá que inicie la partida LAN.**

**Diagramas de bloques del funcionamiento**

**Multijugador**

**Carga y guardado de datos**

**Gráficos**

**Manejo de piezas y basura**

**Puntaje**

**CAMPO**

**Campo:**

**Creamos una matriz llamada campo que es el pilar fundamental del juego. En esta matriz se guarda los valores que representan cada uno de los segmentos de colores, las piezas , el marco , los lugares "vacios", la basura, etc.**

**Manejo de piezas y basura:**

**Realizamos funciones como "rotar\_pieza()" o "print()" que se encargan de imprimir borrar mover o rotar modificando las posiciones de las piezas , basura, etc..en el campo. Esto se hace para que dichas modificaciones se plasmen en el campo , ya que el campo es fundamental y en él se almacenan todas las posiciones y sus respectivos colores.**

**Puntaje:**

**Se guardara el puntaje record con el nombre del jugador. Si el record es superado, se reemplaza este con el nuevo puntaje y nombre del jugador.**

**Carga y guardado de datos:**

**Utilizamos listas y archivos para obtener la información necesaria para la creación de las piezas ya que en los archivos se encuentran las posiciones de cada segmento ,el nombre y color de cada pieza. Además cargamos de un archivo el puntaje record para comparar con el puntaje actual y reemplazarlo en caso de que lo supere.**

**Gráficos:**

**Utilizamos la librería para videojuegos Allegro 5.2.1.1. Lo utilizamos para crear un display con los gráficos de las piezas , el campo ,el puntaje. Además, usamos las funciones del teclado para los controles.**

**Multijugador:**

**Utilizamos sockets y procesos. Un proceso se encarga de ejecutar el juego mientras el otro se encarga de sockets para el multijugador .Dichos procesos se intercomunican mediante FIFO transmitiendo mediante el servidor la información del campo y el nombre del jugador contrario .**

**Problemas encontrados en el proyecto:**

**Tuvimos bugs gráficos y violaciones de segmento. Pudimos corregir estos errores, detectándolos y programando con más eficacia. Por ejemplo los bugs gráficos eran debidos a la carga de imágenes en muchas funciones diferentes que se usaban todo el tiempo. En vez de esto, cargamos las imágenes una única vez en el main y las pasamos como parámetros a las funciones. También tuvimos problemas pensando cómo hacer la parte de sockets .**

**Beneficios encontrados a lo largo del desarrollo del proyecto:**

**Aprendimos a controlar el manejo de gráficos y eventos de teclado.**

**Aprendimos como funciona un sistema multijugador mediante el uso de un servidor.**

**Aprendimos que la eficiencia en la programación es importante.**

**Aprendimos a documentar con Doxygen y a utilizar Git.**

**Conclusiones:**

**Realizar un sistema cliente servidor no es fácil, que en estos proyectos es muy importante prestarle atención a la eficiencia del código en la parte grafica, ya que esto consume muchos recursos de la computadora. Y que hacer un juego tampoco es fácil. Pero quitando todo esto, aprendimos y asentamos muchos conocimientos y utilizarlos en un proyecto grupal.**

**Bibliografía y links**

[**https://www.allegro.cc/manual/5/**](https://www.allegro.cc/manual/5/)

[**es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html**](es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html)

**Material subido por los profesores al campus virtual.**