Jose Luis Jimenez Urbano
Junio de 2014
Programación

Índice

1.-Introducción

Este es el manual técnico de la práctica de Tetris y el programa de gestión/administración. En este manual vamos a tratar diferentes aspectos, desde la configuración de nuestro IDE para la compilación del código fuente, pasando por la explicación de las variables, procedimientos y funciones que utiliza nuestro código.

2.-Obteniendo el código fuente

Podemos conseguir el código fuente de este juego, "forkeando" el repositorio inferior directamente desde GitHub.

Este juego, así como su código está licenciado bajo licencia MIT.

https://github.com/ESAT-jimenezur/Tetris

3.-Dependencias y Requisitos Previos

Para poder compilar con éxito este juego, necesitamos varias cosas.

- Un IDE para C++ (Por ejemplo, CodeBlocks)
- Un compilador de C++ (Nosotros hemos usado GCC)
- Configurar nuestro compilador con las librerías incluidas en la carpeta “Libs”
- Configurar nuestro compilador con la librería “iJos.h”, descargable desde aquí: https://github.com/ESAT-jimenezur/Programacion/tree/master/Libs

4.- Compilando

Una vez configurado lo anterior, deberíamos poder compilar y ejecutar sin problemas.

5.- Estructuras de Datos

5.1.-Variables Globales

Estas son las variables globales, declaradas en la parte superior de “tetris.cpp” y usadas durante el resto del programa.

```
bool game_alive = 1;
```
Esta variable la utilizamos para saber si el juego esta “vivo”, es decir, si hemos perdido o si seguimos jugando.

```
const int anchoVentana = 150;
const int altoVentana      = 100;
```
Utilizamos estas variables para definir el tamaño de la pantalla.

```
const int gameInterface_size_height = 20;
const int gameInterface_size_width      = 10;
const int gameInterface_margin = 2;
```
Estas variables son utilizadas para definir el “area” de juego, el alto, ancho y los márgenes.

```
const int starting_poing_x = 5;
const int gameInterface_size_width      =  2;
```
Utilizamos estas dos variables para definir donde (x e y) soltamos las piezas o fichas.

```
const int gameFicha_ascii_model  =  178;
```
Este el tipo de ficha que utilizamos. Es un carácter ASCII (cuadrado)

```
int CREDITOS      =  0;
int CURRENT_USER[20]      = “NO USER”;
```
Utilizamos estas variables para almacenar los créditos del usuario, y el nombre de usuario. Este por defecto está a “NO USER”, y se sobrecarga con el que iniciemos sesión, al igual que los créditos, se sobrecargan de 0 a los que el usuario tenga.

```
const int PUNTOS = 0;
const int PUNTOS_LINEA = 10;
const int TETRIS = 100;
```
La primera variable son los puntos de la partida actual. Las otras dos son los puntos que nos da realizar una linea completa o un TETRIS (4 lineas completas)

```
const int game_area[23][12];
```
Este es el array de juego (Es muy grande, no lo pongo)
Además de esto, también utilizo las estructuras personalizadas del usuario (Fecha, Lugar y Jugador)

5.2.- Declaración de métodos y funciones

En el juego se definen de forma adelantada todas las funciones a utilizar, por dos motivos:

- Poder utilizarlas sin importar el orden de llamada.
- Tenerlas todas estructuradas de forma clara y ordenada.

Estas son las funciones utilizadas.

void game_init();
Este método es el que inicializa algunas cosas del juego, como la semilla aleatoria, el random del tiempo y el alto-ancho de la ventana

void setColors(**int** texto, **int** fondo);
Esta función toma 2 parámetros, el color del fondo, y el color del texto, lo único que hace es ejecutar la función "colorTexto" pasándole los parámetros anteriores. La función "colorTexto" se encuentra en la librería iJos.h

void resetColors();
Esta función es como la anterior, pero se encarga de resetear los colores a los "default" que utilicemos

void drawGameArea();
Esta función es la encargada de pintar el area de juego, así como las fichas y los márgenes

void insertFicha(**int** posX, **int** posY, **int** tipoFicha);
Esta función, es la encargadad de pintar cada ficha en uso. Toma 3 argumentos, la posición en X e Y, y la ficha que es (para pintarla)

void gameLoop();
Este es el bucle principal de juego, en este método, se comprueban los tiempos para realizar la gravedad, se capturan las teclas para realizar el movimiento, se giran las fichas...

int getFicha(**int** posX, **int** posY);
Este método devuelve una ficha aleatoria. Este método también llama a checkLines y a checkGameLost

void checkLines();
Esta función es la encargada de mirar si hay lineas completas, asignar puntos y bajar las fichas hacia abajo.

void updatePoints();
Esta función es la que se encarga de actualizar los puntos en el panel lateral

void drawCredits();
Esta función es como la anterior, pero pinta los créditos del jugador.

void drawCurrentUser();
Esta función pinta en pantalla el nombre de usuario actua

void drawHighScores();
Esta función pinta en el lado lateral las puntuaciones de otros jugadores.

void checkGameLost(**int** posX, **int** posY);
Esta funcion es la encargada de ver si el juego ha finalizado o continua, se llama antes de generar nuevas ficha

void clearFichaSide(**int** posX, **int** posY, **int** side, **int** tipoFicha);
Esta funcion limpia los laterales de la ficha, al moverla. Recibe como argumentos la posicion, el lado a limpiar y el tipo de ficha que es.

void clearOnRotate(**int** posX, **int** posY, **int** tipoFicha);
Esta función es como la anterior, pero solo limpia cuando la ficha rota.

bool testCollisionBottom(**int** posX, **int** posY, **int** tipoFicha);

bool testCollisionLeft(**int** posX, **int** posY, **int** tipoFicha);

bool testCollisionRight(**int** posX, **int** posY, **int** tipoFicha);
Estas 3 funciones checkean las colisiones en los 3 lados (izquierda, derecha y abajo). Si hay alguna ficha o suelo o pared, la ficha se detiene y se genera una nueva.

bool login_access();
bool login();
Estos dos métodos se utilizan para mostrar el login y realizar la consulta a la base de datos, para ver si el jugador exite. Cuando login() devuelve true, el juego se inicia.

void game_finished();
Esta función es la que se llama cuando el juego acaba, esta guarda los puntos y muestra el mensaje de “partida perdida”

6.-Cambios de la versión

1.01 prealpha (06/2014)

- En esta versión, se han integrado las principales funcionalidades del juego

7.-Bugs

Aquí se indicarán los diferentes tipos de bugs encontrados en el juego, para reparar en futuras actualizaciones:

Si se encuentra algún tipo de error, se recomienda abrir un nuevo “ticket” en el repositorio. (https://github.com/ESAT-jimenezur/Tetris/issues)

- Al rotar una ficha, no se comprueba lo que hay alrededor, por lo tanto se puede llegar a borrar información de las posiciones adyacentes (fichas o paredes)
- Cuando finaliza la partida, a la hora de guardar, el bucle falla y no sale nunca. Se ha comentado la parte de “Guardar” puntuación.
- Las máximas puntuaciones no aparecen ordenadas de mayor a menor

8.- Bibliografía y Recursos

Para la realización de este proyecto, se han consultado diversas fuentes, y se han utilizado numerosos recursos, los más destacables son:

Además, se ha utilizado la plantilla de Documentación del Stratego, para una mejor integración y consolidación del trabajo.

8.1.-Bibliografía

- http://es.wikipedia.org/wiki/Tetris
- http://stackoverflow.com/

8.2.-Recursos

- https://github.com/ (Control de versiones)
- CodeBlocks (IDE)
- GCC (Compilador de C)
- OpenOffice (Editor de textos libre y gratuito)
- Y, esta vez, por la época, mucho Monster fresquito, que el café ya no pega! ;)