Projekt na zajęcia wstęp do programowania w C

Twórca Projektu: Maciej Pietrwicz

Indeks: 325022

Nazwa Projektu: Gra Tetris

Jako projekt na zaliczenie przedmiotu wykonałem grę Tetris. Gra została napisana w języku C przy pomocy dodatkowych bibliotek. Biblioteki te to:

-SDL (https://www.libsdl.org)-

-SDL_Image (https://www.libsdl.org/projects/SDL_image/)

-SDL_ttf (https://www.libsdl.org/projects/SDL_ttf/)

Przykładowe tutoriale co do instalacji bibliotek:

https://lazyfoo.net/tutorials/SDL/01_hello_SDL/index.php (SDL)

https://lazyfoo.net/tutorials/SDL/06_extension_libraries_and_loading_other_image_formats/index.php (SDL_Image, podobnie się ustawia SDL_ttf)

Cel gry:

Celem gry jest zdobycie jak największej ilości punktów. Punkty są zdobywane poprzez ustawianie losowo generowanych kształtów w taki sposób, aby jeden cały wiersz został uzupełniony. Wtedy punkty są naliczane, a bloki z danego wiersza usuwane. Za zapełnienie kilku wierszy naraz można uzyskać dodatkowe pkt. Gracz przegrywa jeśli jedna cała kolumna zostanie zapełniona blokami.

Dodatkowe utrudnienia:

W grze występują poziomy. Gracz po przekroczeniu pewnej ilości punktów awansuje na następny poziom. Przejście na kolejny poziom. wiąże się z przyśpieszeniem prędkości spadania bloków, a także z ilością zdobywanych pkt –większy poziom, łatwiej zdobyć pkt.

Sposób gry:

Gracz za pomocą strzałek na klawiaturze może kontrolować poruszanie się bloku w poziomie. Lewa strzałka –lewo, prawa strzałka –prawo. Za pomocą strzalki do dołu możliwe jest przyśpieszanie bloku. Strzałka do góry natomiast odpowiada za obracanie bloków.

Bloki:

W grze znajduje się 7 bloków o kształtach odpowiadającym literom.Litery te to: O,Z,S,T,L,J,I.Blok O odpowiadający za kwadrat z czterech bloków jako jedyny nie wykonuje obrotu.

Informacje podczas gry:

Podczas gry na ekranie widzimy główną planszę gry, oraz trzy napisy:- „Level" pod którym wyświetlany jest aktualny poziom. -„Next shape", pod który widzimy następny kształt, który zostanie wygenerowany- „Score", pod którym wyśietlana jest aktualna ilość punktów.

Struktura projektu:

Projekt podzielony jest na 4 pliki z rozszerzeniami pliku C oraz 3 pliki nagłówkowe.

Plikiem uruchamiającym projekt jest plik main.c.

```
initWindow();
startMenu();
exitGame();
```

Używa kolejno tych trzech funkcji zdefiniowanych w Game.c.

Funkcje te kolejno odpowiadają za stworzenie okna, w którym wyświetlana jest gra, uruchomienie funkcji odpowiadającej za główne menu gry oraz funkcja kończąca działanie programu.

Następnym plikiem jest Game.c. Są tu zdefiniowane wyżej podane funkcje oraz wiele innych dotyczących renderowania oraz zmian w siatce (po ang grid) , w której znajdują sie bloki.

```
bool emptySpace(int x, int y)
```

Funkcja ta odpowiada za sprawdzanie czy dane pole w siatce jest puste.

```
void initGrid()
```

Inicjuje siatkę w postaci tablicy.

```
void drawGrid()
```

Funkcja odpowiada za narysowanie i wyrenderowanie siatki.

```
int cordinatesToGridX(int x)
{
    return (x - TOP_LEFT_X) / BLOCK_SIZE;
}
int cordinatesToGridY(int y)
{
    return (y - TOP_LEFT_Y) / BLOCK_SIZE;
}

int gridXtoCordinates(int x)
{
    return TOP_LEFT_X + BLOCK_SIZE * x;
}
int gridYtoCordinates(int y)
{
    return TOP_LEFT_Y + BLOCK_SIZE * y;
}
```

Funkcje te odpowiadają za zamianę z wspolrzednych tablicy grid[][] na współrzędne okienka gry, lub na odwrót.

```
bool checkIfLost()
```

Funkcja sprawdza czy gracz nie przegrał gry, jak przegrał zwraca informacje, że należy zakończyć główną pętle gry.

```
bool checkEmptySpaces(int number, Block *block, int m, int t)
```

Funkcja sprawdza czy miejsca na siatce są puste dla nowych współrzędnych bloku.

```
void handleEvents(Block *block, int BlockSpeed, int level, int *rotateDelay)
```

Funkcja odpowiada za wykonywanie poleceń zgodnie z danymi wejściowymi od uzytkownika (odpowiednie przyciski z klawiatury)

```
void rendererFuncs(Block* currentBlock, int nextShape, int score)
```

Główna funkcja odpowiadająca za renderowanie

```
void clearAndUpdateGrid(int y)
```

Funkcja aktualizuje siatkę przy zapełnieniu jednego wierszu.

```
int getPoints(int Level, Block *block, int score) {
```

Funkcja sprawdza czy gracz zapełnił jeden wiersz siatki(lub więcej, maks 4) wtedy dostaje odpowiednią ilosc punktów.

```
int levelUp(int score)
```

Funkcja aktualizuje lewel gdy odpowiednia ilość punktów zostanie zdobyta.

```
}
int startGame(const int frameDelay)
```

Funkcja uruchamiająca grę. Znajduje tu się główna pętla gry.

Następnym plikiem w moim projekcie jest BlocksManager.c

W tym pliku zdefiniowane są funkcje dotyczące bloków.

Mamy tu również zdefiniowaną strukture.

```
typedef struct Block
{
    SDL_Texture* blockTexture;
    SDL_Rect cordinates[4];
    int whichTexture;
    int fallSpeed;
    int state;
}Block;
```

Zawiera ona kolejno: teksturę pojedyńczego kwadratu bloku, współrzędne bloków, numer tekstury, prędkośc poruszania się bloku oraz, w którym momencie obrotu jest blok.

```
char S2[] = ".......00..0x..........";
char S1[] = ".......0....x0....0......";

char I1[] = ".......0....x....0....0..";
char I2[] = "..........00x0..........";
char I3[] = "...........0x00.........";

char Z1[] = "...........0x....00......";

char Z2[] = ".......0...0x...0........";

char O[] = "............x0...00......";

char J1[] = ".......0....x...00.......";
char J2[] = "...........0x0....0......";
char J3[] = ".......00...x....0.......";
char J4[] = "......0....0x0...........";

char L1[] = ".......0....x....00......";
char L2[] = "........0..0x0...........";
char L3[] = "......00....x....0.......";
char L4[] = "...........0x0..0........";

char T1[] = "...........0x0...0.......";
char T2[] = ".......0....x0...0.......";
char T3[] = ".......0...0x0...........";
char T4[] = ".......0...0x....0.......";
```

Są tu również zdefiniowane tablice char, które posiadają odpowiedni układ „.”,”0” i „x”. Użyłem tego typu tablic, aby rotować bloki oraz je tworzyć. Te tablice można interpretować jako tablice 5x5 gdzie w okół środka, którym jest „x” za pomocą „0” stworzone są kształty bloków. Następnie za pomocą funkcji

```
int createShape(Block* block, char shape[], bool isRotate)
```

Tworzony jest odpowiedni kształt interpretując owe tablice char na siatkę (grid).

```
void RotateBlock(Block* block)
{
```

Funkcja ta za pomocą createShape() rotuje blokiem.

Mamy tu również funkcje odpowiadające za poruszanie bloków:

```
void moveAllBlocksLeft(Block* block)
{
    for (int i = 0; i < 4; i++)
    {
        block->cordinates[i].x -= BLOCK_SIZE;
    }

}

void moveAllBlocksRight(Block* block)
{
    for (int i = 0; i < 4; i++)
    {
        block->cordinates[i].x += BLOCK_SIZE;
    }

}
```

Oraz za ich wyświetlanie:

```
void showBlocks()
```

```
void chooseShape(Block* block, int Texture)
```
- odpowiada za wybranie odpowiedniego kształtu do stworzenia.

```
void setBlockSize(Block* block)
```

Ustawia rozmiary bloku na odpowiednie.

```
void spawnBlock(Block* block, int shape, int level)
```

Funkcja tworzy nowy blok.

Już ostatnim plikiem w projekcie jest TexturesManager.c

Mamy tu podane funkcje:

```
SDL_Texture* loadTexture(const char* filename);
Text createText(TTF_Font* GameFont, const char* text, int x, int y);

void loadBlocksTextures();
void loadTextsTextures();
void destroyTextsTextures();
void destroyBlocksTextures();
```

Odpowiadają one za ładowanie oraz niszczenie tekstur oraz tekstów.

Znajduje tu się również struktura

```
typedef struct Text
{
    SDL_Texture* textTexture;
    SDL_Rect position;
}Text;
```

, która zawiera w sobie teksturę danego tekstu oraz współrzędne na ekranie.