AKADEMIA NAUK STOSOWANYCH W TARNOWIE

Gra platformowa w 3D

Stanisław Mendel 35684

Wiktor Warmuz 35243

AKADEMIA NAUK STOSOWANYCH W TARNOWIE

Spis treści

1. Wstęp – Cel projektu……..3
2. Implementacja ……..4
3. Testowanie ……7
4. Proponowane dalsze możliwości rozbudowy……8
5. Instrukcja obsługi……9
6. Instrukcja kompilacji……10

WSTĘP – CEL PROJEKTU

Celem tego projektu jest stworzenie prostej platformowej gry 3D przy użyciu biblioteki OpenGL w języku C++. Gra będzie obejmować kontrolę postaci gracza, poruszanie się w trójwymiarowym środowisku i interakcję z platformami oraz innymi obiektami.

Głównym celem projektu jest zaprojektowanie i implementacja gry, która pozwoli graczowi eksplorować trójwymiarowy świat oraz pokonywać różne przeszkody. Gracz będzie sterować postacią, poruszając się po platformach, skacząc, unikając przeszkód oraz wykonując inne akcje. Do realizacji projektu wykorzystamy bibliotekę OpenGL, która zapewni nam możliwość renderowania grafiki 3D i manipulowania obiektami w trójwymiarowym środowisku. OpenGL jest potężnym narzędziem do tworzenia gier i aplikacji graficznych, oferującym szeroki zakres funkcji do renderowania obrazów, zarządzania teksturami, oświetlenia i wiele więcej. W trakcie realizacji projektu będziemy dbać o optymalizację kodu, aby zapewnić jak najlepszą wydajność gry. Oczekuje się, że ten projekt pozwoli nam zgłębić wiedzę na temat programowania grafiki 3D, pracy z biblioteką OpenGL i rozwiniemy umiejętności projektowania i implementacji gier. W rezultacie powstanie prototyp gry, która po udoskonaleniu dostarczy rozrywki i satysfakcji dla graczy.

IMPELENTACJA

Organizacja programu:

- Plik nagłówkowy o nazwie: „Header.h”
- Plik z funkcjami o nazwie: „Function.cpp”
- Plik główny o nazwie: „Main.cpp”

Wybór środowiska:

- Język programowania: C++
- Środowisko programistyczne: Visual Studio 2022
- Biblioteki: OpenGL, GLUT, Assimp, STB_Image
- System operacyjny: Windows 10

Ciekawe fragmenty kodu:

```
// Aktualizacja pozycji kostki
if (cubeIsJumping)
{
    cubeY += 0.1f;
    if (cubeY >= tmpCubeY + JUMP_HEIGHT + FLOOR_Y)
    {
        cubeIsJumping = false;
        cubeIsFalling = true;
    }
    glutPostRedisplay(); // Odświeżenie ekranu po skoku
}
else if (cubeIsFalling)
{
    cubeY -= 0.1f;
    if (cubeY <= startCubeY)
    {
        cubeY = FLOOR_Y + 0.5f;
        cubeIsFalling = false;
    }
    glutPostRedisplay(); // Odświeżenie ekranu po opadaniu
    tmpCubeY = cubeY;
}
```

Kod ten bazując na zmiennych typu bool tj. cubeIsJumping oraz cubeIsFalling pozwala na „płynne” skakanie modelu.

Poniższy kod tworzy ściany w jednym z poziomów i nakłada na nie teksturę o nazwie „bricks-small.jpg”.

```
// Wyczyszczenie bufora koloru i głębokości
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        // Ładowanie nowej tekstury dla ściany
        unsigned char* wallData = stbi_load("Tekstury\\bricks-small.jpg", &width, &height,
&channels, 0);

        // Generowanie nowej tekstury dla ściany
        unsigned int wallTextureID;
        glGenTextures(1, &wallTextureID);

        // Ustawienie nowych parametrów tekstury dla ściany
        glBindTexture(GL_TEXTURE_2D, wallTextureID);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE,
wallData);

        // Zwalnianie pamięci
        stbi_image_free(wallData);

        // Lewa ściana
        glEnable(GL_TEXTURE_2D);

        glBindTexture(GL_TEXTURE_2D, wallTextureID);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-FLOOR_X, 0.0f, -FLOOR_Z);
        glTexCoord2f(0.0f, 5.0f); glVertex3f(-FLOOR_X, 0.0f, FLOOR_Z);
        glTexCoord2f(5.0f, 5.0f); glVertex3f(-FLOOR_X, FLOOR_HEIGHT, FLOOR_Z);
        glTexCoord2f(5.0f, 0.0f); glVertex3f(-FLOOR_X, FLOOR_HEIGHT, -FLOOR_Z);

        // Prawa ściana
        glBindTexture(GL_TEXTURE_2D, wallTextureID);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(FLOOR_X, 0.0f, -FLOOR_Z);
        glTexCoord2f(0.0f, 5.0f); glVertex3f(FLOOR_X, 0.0f, FLOOR_Z);
        glTexCoord2f(5.0f, 5.0f); glVertex3f(FLOOR_X, FLOOR_HEIGHT, FLOOR_Z);
        glTexCoord2f(5.0f, 0.0f); glVertex3f(FLOOR_X, FLOOR_HEIGHT, -FLOOR_Z);
        // Ściana na wprost
        glBindTexture(GL_TEXTURE_2D, wallTextureID);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-FLOOR_X, 0.0f, -FLOOR_Z);
        glTexCoord2f(0.0f, 5.0f);  glVertex3f(FLOOR_X, 0.0f, -FLOOR_Z);
        glTexCoord2f(5.0f, 5.0f); glVertex3f(FLOOR_X, FLOOR_HEIGHT, -FLOOR_Z);
        glTexCoord2f(5.0f, 0.0f); glVertex3f(-FLOOR_X, FLOOR_HEIGHT, -FLOOR_Z);

        // Ściana z tyłu
        glBindTexture(GL_TEXTURE_2D, wallTextureID);
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-FLOOR_X, 0.0f, FLOOR_Z);
        glTexCoord2f(0.0f, 5.0f);  glVertex3f(-FLOOR_X, FLOOR_HEIGHT, FLOOR_Z);
        glTexCoord2f(5.0f, 5.0f); glVertex3f(FLOOR_X, FLOOR_HEIGHT, FLOOR_Z);
        glTexCoord2f(5.0f, 0.0f); glVertex3f(FLOOR_X, 0.0f, FLOOR_Z);
        glEnd();
```

TESTOWANIE

Renderowanie grafiki: Upewniono się aby modele 3D, oświetlenie oraz tekstury zostały wyrenderowane i aby wyglądały zgodnie z dotychczasowymi naszymi założeniami.

Ruch postaci: Upewniono się aby wszystkie funkcje dotyczące poruszania się były prawidłowo obsługiwane.

Wydajność: Dotychczasowym największym problemem z wydajnością były wysokiej rozdzielczości tekstury – po zmianie na tekstury o mniejszej rozdzielczości wydajność znacznie wzrosła.

PROPONOWANE DALSZE MOŻLIWOŚCI ROZBUDOWY

Dalszymi możliwościami rozbudowy mogą być między innymi:

-Dodanie większej ilości poziomów, które będę wykorzystywały wszystkie zaimplementowane mechaniki.

-Dodanie lepszych tekstur dla obiektów i modeli.

-Dodanie animowanych obiektów np. lawa.

-Zaimplementowanie muzyki.

-Zaimplementowanie animacji ruchu.

-Dodanie fabuły.

-Dodanie cieni.

W – poruszanie postacią do przodu

A – poruszanie postacią w lewo

S – poruszanie postacią do tyłu

D – poruszanie postacią w prawo

SPACJA – skok (dostępne dla wszystkich poziomów poza poziomami ze schodami)

E – umożliwia chwycenie obiektu aby go ciągnąć (postać musi się stykać z obiektem aby go przesunąć)

T – wchodzenie po drabinie (dostępne tylko przy drabinie)

R – schodzenie po drabinie (dostępne tylko przy drabinie)

SPOJLER – JAK PRZEJŚĆ KONKRETNE POZIOMY

Poziom 1 – Należy ustawić skrzynię w wyświetlonym obiekcie który jest wysunięty do przodu, następnie postacią należy stanąć na obiekt bliżej kamery. Przejdź przez drzwi.

Poziom 2 – Należy przejść obok obiektów (UWAGA! – po wejściu w lawę umierasz i gra się resetuje)
Poziom 3 – Wejdź po schodach do następnego pomieszczenia.

Poziom 4 – Podobnie jak w poziomie 2 tylko tym razem obiekty się poruszają

Poziom 5 – Wejdź po drabinie po prawej, następnie będąc na szczycie drabiny należy użyć skoku aby zebrać punkt. Przejdź przez odblokowane drzwi aby zresetować grę.

AKADEMIA NAUK STOSOWANYCH W TARNOWIE

INSTRUKCJA KOMPILACJI

KOMPILACJA W VISUAL STUDIO 2022

Domyślne ścieżki plików nagłówkowych

GLUT: C:\Lib\freeglut\include

ASSIMP: C:\Program Files\Assimp\include

STB_IMAGE: C:\Lib

Domyślne ścieżki bibliotek

GLUT:C:\Lib\freeglut\lib

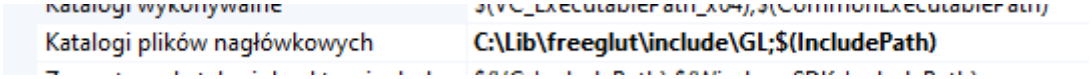
ASSIMP: C:\Program Files\Assimp\lib\x64

Otwórz projekt „OpenGL - Game Cube3D” przy pomocy VISUAL STUDIO 2022.

Kliknij prawym przyciskiem w oknie Solution Explorer na nazwę projektu i kliknij w „Właściwości”.

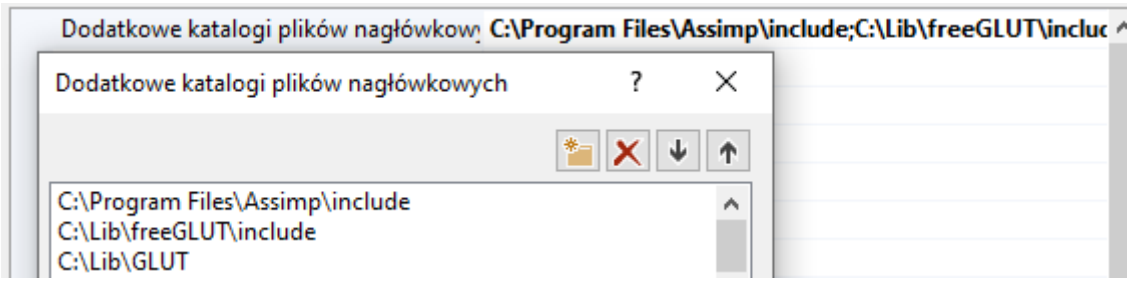
Następnie należy dodać:

- W „Katalogi VC++” – „Katalogi plików nagłówkowych” – (ścieżka do pliku nagłówkowego glut) np.:



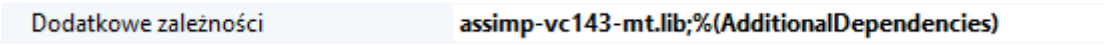
- W „C/C++” – „Dodatkowe katalogi plików nagłówkowych” oraz w „Wszystkie opcje” – ścieżkę do plików nagłówkowych Assim oraz

glut np.:



-W „Konsolidator” – „Dodatkowe katalogi biblioteki” – ścieżki do bibliotek assimp oraz glut np.:

-W „Konsolidator” – „Dane wejściowe” – dokładną nazwę biblioteki assimp np.:



Następnie należy wybrać plik „main.cpp” w oknie Solution Explorer i otworzyć go. W ostatnim kroku skompiluj program przy pomocy lokalnego debugera windows.