Dariusz Maciejewski

Camera Fighter –Gra komputerowa

Promotor: prof. nzw. dr hab. Przemysław Rokita

Pracownia problemowa magisterska 1 - raport -

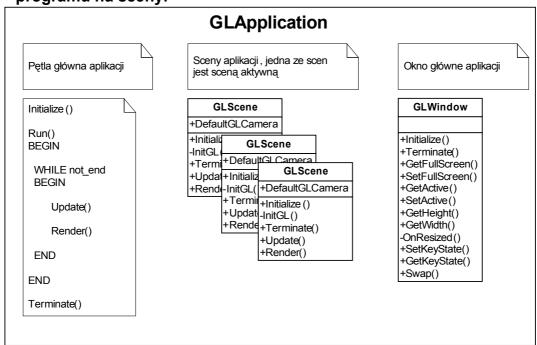
Opis zadania:

Camera Fighter ma być grą umożliwiającą przeprowadzanie wirtualnych pojedynków. Postacie będą się poruszały zgodnie z ruchami gracza, zarejestrowanymi przez dwie kamery.

Moim zadaniem jest stworzenie silnika 3D, który będzie napędzał całą grę.

Zakres prac wykonanych w pierwszym semestrze:

1. Uniwersalny szkielet aplikacji OpenGL, wspierający logiczny podział programu na sceny.



Klasy wchodzące w skład frameworku:

GLApplication - główna klasa zarządzająca aplikacją.

Initialize() — inicjalizuje obiekt prywatny typu GLWindow (skutkuje to utworzeniem okna aplikacji) i inicjalizuje scenę startową, przekazaną w parametrze metody.

Run () — uruchamia pętlę główną programu, wewnątrz niej wołane są metody Update () i Render ().

Update () – wysyła żądanie przetworzenia zdarzeń i odświeżenia zawartości bieżacej sceny.

Render () – wysyła żądanie wyrenderowania bieżącej sceny.

Terminate() – czyści scenę i okno, a następnie kończy działanie aplikacji.

```
GLWindow — reprezentuje okno OpenGL.

Initialize() — tworzy okno OpenGL.

Terminate() — niszczy okno OpenGL.

GetFullScreen() — informuje czy okno jest w trybie pełnoekranowym.

SetFullScreen() — zmienia tryb na pełnoekranowy lub okienkowy.

GetActive() — sprawdza czy okno jest aktywne.

SetActive() — zmienia stan aktywności okna.

GetHeight() — zwraca wysokość okna.

GetWidth() — zwraca szerokość okna.

SetKeyState() — ustawia stan podanego klawisza (na wciśnięty lub nie).

GetKeyState() — zwraca stan podanego klawisza.

Swap() — zamienia bufory okna.
```

friend GLWindowProc() - procedura przetwarzająca zdarzenia okna.

GLScene – scena reprezentuje stan w jakim znajduje się silnik 3D. Każda scena jest autonomiczna, więc jej działanie nie zależy od innych scen. Przykładowe sceny jakie możemy spotkać to: menu, tryb rozgrywki, przerywnik filmowy.

```
Initialize() — inicjuje OpenGL, tworzy zasoby, etc.

Terminate() — zwalna zasoby wykorzystywane przez scenę.

Update() — obsługuje zdarzenia i logikę sceny.

Render() — renderuje scenę.
```

GLCamera *DefaultGLCamera — domyślna kamera wykorzystywana w scenie.

2. System sterowania kamerami z zastosowaniem matematyki kwaternionów oraz własnej metody obliczania obrotów.

GLCamera – utożsamia wirtualną kamerę. Definiuje położenie oraz kierunek patrzenia kamery. Udostępnia również metody umożliwiające jej przemieszczanie i obracanie.

```
SetGLCamera() -ustawia kamerę na podanej pozycji.
Aim() - ustawia punkt patrzenia OpenGL na pozycji kamery.
Move() - przemieszcza kamerę w trybie 'człowiek' - ruchy w poziomie są
równoległe, a w pionie prostopadłe do płaszczyzny świata.
Shift() - przemieszcza kamerę w trybie 'rakieta' - ruchy w poziomie są
równoległe, a w pionie prostopadłe do płaszczyzny patrzenia.
Rotate() - obraca kamerę w okół statywu.
Orbit() - obraca kamerę w okół celu.
```

Obroty kamery są liczone z wykorzystaniem dwóch metod – uniwersalnej metody kwaternionów oraz metody autorskiej – trzy razy szybszej, ale umożliwiającej przeprowadzanie tylko obrotów, wokół prostych równoległych do jednej z osi.

3. **Klasa bazowa dla obiektów 3D**, oraz dwa demonstracyjne obiekty dziedziczące z tej klasy – arenę i skrzynię.

Object3D – klasa bazowa dla obiektów występujących w aplikacji.

SetPosition() - ustawia obiekt na podanej pozycji.

GetPosition() - zwraca pozycję obiektu.

SetRotation() - ustawia obrót obiektu.

GetRotation() - zwraca obrót obiektu.

Render() — modyfikuje macierz obrotów i przesunięć OpenGL zgodnie z wartościami pozycji i obrotu obiektu, woła metodę RenderObject() i przywraca oryginalną macierz obrotów i przysunięć.

RenderObject() — metoda wirtualna — w niej przebiega właściwy proces renderowania obiektu.

Obiekty te zostały wykorzystanie w scenie demonstracyjnej, wykorzystywanej do testowania aplikacji.



Plany na kolejny semestr:

W najbliższym semestrze planuję rozszerzyć framework aplikacji o klasę singleton InputManager. Ma ona pośredniczyć między zdarzeniami wejścia a aplikacją. Będzie ona pozwalała na mapowanie zdarzeń wejścia na wewnętrzne kody wykorzystywane przez logikę aplikacji, dzięki czemu użytkownik będzie mógł dowolne zmieniać powiazania klawiszy.

Ponadto planuję utworzenie managera zasobów i bazującego na nim managera tekstur.

Zamierzam dalej rozwijać bibliotekę obiektów, utworzyć managera świateł oraz rozpocząć pracę nad logiką gry.

Podsumowanie:

W trakcie ostatniego semestru rozpocząłem naukę OpenGL oraz stworzyłem podstawy przyszłej gry. Niestety z powodu licznych, pracochłonnych projektów z innych przedmiotów nie mogłem poświęcić pracy magisterskiej tyle czasu, ile bym chciał. Mimo to jestem zadowolony z efektów jakie osiągnąłem. Liczę że w najbliższym semestrze uda mi się osiągnąć znacznie więcej niż przedstawiłem to w swoich planach.

Literatura:

'Perełki programowania gier – tom 1' pod redakcją Marca DeLoura, Wydawnictwo Helion 2002

The Red Book http://fly.srk.fer.hr/~unreal/theredbook/

The Blue Book http://www.rush3d.com/reference/opengl-bluebook-1.0/index.html

NeHe Productions http://nehe.gamedev.net/

Jerome Lessons http://jerome.jouvie.free.fr/OpenGl/Lessons.php

Jerome Tutorials http://jerome.jouvie.free.fr/OpenGl/Tutorials1-5.php

Tulane.edu tutorial http://www.eecs.tulane.edu/www/Terry/OpenGL/Introduction.html#Introduction