Gliwice, 06.03.2018 r.

Projekt z Grafiki Komputerowej

Dokument projektowy

Tytuł projektu:
Gra 2D typu Tower Defence

Prowadzący:
dr Ewa Lach

Skład sekcji:
Karol Marszałek
Błażej Moska
Kamil Janas

1. Treść zadania

Gra typu Tower Defence 2D zrealizowana z wykorzystaniem jezyka C++ oraz bibliotek graficznych umozliwiajacych tworzenie gier komputerowych w tym języku. Gra opiera się na popularnym modelu gier typu Tower Defence polegającym na obronieniu pewnego kluczowego punktu na mapie, na przykład zamku, przed nadchodzącymi przeciwnikami. Przeciwnicy poruszają się po ściśle określonej ścieżce na mapie, na pozostałych polach użytkownik może ustawiać wieże eliminujące przeciwników w drodze do zamku. Rozgrywka kończy się wygraną gracza jeżeli pokona on wszystkich przeciwników lub jego porażką jeżeli przeciwnicy zniszczą zamek.

2. Analiza zadania:

1. **Podstawy teoretyczne problemu:**
2. **Wykorzystane zagadnienia grafiki komputerowej:**
   1. Przekształcenia afiniczne
   2. Shadery
   3. Wykrywanie kolizji
3. **Wykorzystane biblioteki i narzedza programistyczne:**
   1. Visual Studio jako rozbudowane narzedzie programistyczne zostanie wykorzystane w celu implementacji projektu z wykorzystaniem jezyka C++.
   2. Biblioteka SFML - biblioteka graficzna do jezyka C++ ułatwiajaca korzystanie z mechanizmów do tworzenia grafiki komputerowej w wykorzystywanym środowisku programistycznym. SFML wprowadza podejście obiektowe do programowania efektów graficznych co pozwala na lepszą integrację warstwy graficznej programu z resztą programu napisaną z wykorzystaniem podejścia programowania zorientowanego obiektowo. Dodatkowo biblioteka SFML zapewnia multiplatformowość tworzonych z jej wykorzystaniem programów.Jest ona zorientowana w celu optymalizacji grafiki dwuwymiarowej, w związku z czym nie wspiera ona mechanizmów do tworzenia grafiki trójwymiarowej. Jest to ograniczenie, ktore nie wpływa na specyfikację projektu ze względu na wybór implementacji grafiki dwuwymiarowej. Biblioteka SFML nie jest przystosowana do prostego tworzenia graficznego interfejsu użytkownika w przeciwieństwie do, na przykład, bibiotelki QT, jednak jej możliwosci pozwalają na tworzenie intuicyjnego interfejsu gry komputerowej.
4. **Algorytmy, struktury danych, ograniczenia specyfikacji**
   1. Struktury danych:
      1. Klasa opisujaca mapę zrealizowana wewnętrznie w postaci tablicy jednowymiarowej.
      2. Klasa umożliwiajaca przechowywanie historii wyników gracza zapisywanych na dysku twardym użytkownika, wczytywana do aplikacji przy jej uruchomieniu do listy.
      3. Klasa przechowujaca konfigurację klas graficznych, w tym zbiór tekstur wykorzystywanych przez grę w tablicy.
      4. Klasa przechowująca informacje o znajdujących sie na mapie przeciwnikach implementowana z wykorzystaniem tablicy jednowymiarowej.
      5. Klasa przechowująca informacje o znajdujących sie na mapie wieżach ustawionych przez użytkownika w postaci tablicy jednowymiarowej.
5. **Algorytmy:**
   1. Algorytm wyboru celu wieży na mapie na podstawie jej zasiegu oraz położenia przeciwników na mapie.
   2. Algorytm wyznaczania trajektorii pocisku.
   3. Algorytm wyznaczania trasy przejścia przeciwników po mapie.

6. **Ograniczenia specyfikacji:**
    1. Rozmiar planszy - minimalnie: 250x250 px, maksymalnie 500x500 px.
    2. Ograniczenie grafiki do grafiki dwuwymiarowej.
    3. Gra przeznaczona jest dla jednego uzytkownika bez rozróżnienia uzytkowników lokalnych.

3. Plan pracy

1. Stworzenie dokumentacji. - 1 dzień
2. Przygotowanie środowiska programistycznego oraz repozytorium. - 1 dzień
3. Stworzenie projektu odpowiedzialnego za grafikę oraz działającej pustej aplikacji graficznej. - 1 dzień
4. Stworzenie klasy przechowującej globalne ustawienia grafik, między innymi ścieżki tekstur. - 1 dzień
5. **Stworzenie zestawu klas do obslugi mapy gry - kamien milowy :**
    1. Stworzenie klasy implementującej mape gry w formie tablicy. - 1 dzień
    2. Stworzenie klasy rysującej mape na podstawie tablicy. - 1 dzień
    3. Integracja klas wymienionych w podpunktach 1), 2). - 1 dzień
    4. Implementacja reguł ścieżek przechodzenia planszy przez przeciwników w przypadku rozgałęzienia ścieżek. - 2 dni
    5. Przetestowanie działania mapy. -1 dzień
6. **Stworzenie zestawu klas do obsługi wież oraz przeciwników na mapie. - kamien milowy:**
    1. Osbługa wiez na planszy - umieszczenie, wykrywanie kolizji. - 1 dzień
    2. Osbługa przeciwników na planszy - rysowanie przeciwników oraz przejście po planszy zgodnie z regulami zdefiniowanymi na planszy. - 2 dni
    3. Wykrywanie kolizji pomiedzy przeciwnikami oraz odczytanie reguł przejścia po planszy w przypadku rozgałęzień ścieżek. - 2 dni
    4. Przetestowanie działania stworzonych funkcjonalnosci oraz ich integracji z wczesniejszymi etapami projektu. - 2 dni
7. **Stworzenie zestawu klas odpowiedzialnych za interakcje wiez oraz przeciwników (wyznaczanie celu oraz trajektorii pocisku) - kamien milowy:**
    1. Implementacja algorytmu wyboru celu dla wiezy. - 1 dzień
    2. Narysowanie pocisków na mapie. - 1 dzień
    3. Implementacja algorytmu obliczającego trajektorie pocisków. - 2 dni
    4. Implementacja kolizji pocisku oraz przeciwnika. - 1 dzień
    5. Implementacja usuwania pokonanych przeciwników z mapy. - 1 dzień
    6. Przetestowanie działania stworzonych funkcjonalnosci oraz ich integracji z wczesniejszymi etapami projektu. - 2 dni
8. Stworzenie zestawu klas zarządzających pojedyncza rozgrywka (rozpoznanie konca gry oraz warunkow poczatkowych). - 1 dzień
9. Osbługa statystyk gracza. - 1 dzień
10. Przetestowanie działania programu. - 2 dni

4. Wstępny podział pracy między osoby w zespole:

1. Stworzenie dokumentacji - Karol Marszałek, Kamil Janas, Błażej Moska
2. Punkty 2, 3 - Karol Marszałek
3. Punkt 4, 5 - Kamil Janas
4. Punkt 6, 1) Błażej Moska; 2) Kamil Janas; 3,4) Karol Marszałek
5. Punkt 7, 1) Kamil Janas, 2) Błażej Moska, 3) Karol Marszałek
6. Punkt 8, 1) Błażej Moska, 2) Karol Marszałek, 3) Kamil Janas, Karol Marszałek,
7. 4) Błażej Moska, 5) Karol Marszałek, Błażej Moska
8. Punkt 9, Karol Marszałek
9. Punkt 10, Błażej Moska
10. Testowanie działania kolejnych kamieni milowych oraz finalne testowanie programu : Karol Marszałek, Błażej Moska, Kamil Janas.

5. Link do repozytorium

https://github.com/karmar1995/GK.git