

Politechnika Poznańska
Wydział Informatyki
Instytut Informatyki

Praca dyplomowa inżynierska

**PROJEKT I IMPLEMENTACJA GRY KOMPUTEROWEJ
Z WYKORZYSTANIEM TECHNOLOGII MICROSOFT DIRECTX 11**

Krzysztof Marciniak, 106574
Piotr Przybysz, 106602
Mikołaj Szychowiak, 106580
Ryszard Wojtkowiak, 106609

Promotor
dr inż. Witold Andrzejewski

Poznań, 2015 r.

Tutaj przychodzi karta pracy dyplomowej;
oryginał wstawiamy do wersji dla archiwum PP, w pozostałych kopiach wstawiamy ksero.

Spis treści

1	Wstęp	1
1.1	Cel i zakres pracy	1
2	Przegląd zagadnień teoretycznych	2
3	Przegląd narzędzi	3
3.1	Własny silnik graficzny w języku C++	3
3.2	Unreal Engine 4	3
3.3	Unity	4
3.4	OGRE	4
3.5	CryEngine 3	4
3.6	Unreal Engine 3	4
3.7	Unreal Development Kit (UDK)	4
3.8	Autodesk Maya	4
3.9	Autodesk 3D Studio Max	4
3.10	Blender	4
3.11	Adobe Flash - Scaleform	4
4	Praca własna	5
4.1	Projektowanie	5
4.1.1	Projekt silnika graficznego	5
4.1.2	Modele 3D	5
4.1.3	Projekt poziomu	5
4.2	Implementacja	5
4.2.1	Wykorzystanie technologii DirectX 11 w UDK	5
4.2.2	Zarządzanie projektem	5
5	Podsumowanie	6
6	Literatura	7
7	Dodatki	8
	Literatura	9

Rozdział 1

Wstęp

1.1 Cel i zakres pracy

Rozdział 2

Przegląd zagadnień teoretycznych

Rozdział 3

Przegląd narzędzi

Jedną z najważniejszych kwestii podczas tworzenia gry komputerowej jest prawidłowy dobór narzędzi, ponieważ decyduje to nie tylko o komforcie pracy, ale także o jakości końcowego produktu. Podczas tego procesu szczególny nacisk powinien zostać położony na doborze silnika graficznego, między innymi ze względu na dużą dywersyfikację narzędzi należących do tej kategorii oraz mnogość funkcjonalności przez nie oferowanych.

Ze względu na specyfikę tego rozdziału, w kolejnych punktach opisane zostaną przetestowane rozwiązania wraz z krótkim podsumowaniem w formie listy jego zalet i wad. Warto w tym miejscu również wspomnieć, iż wszystkie analizowane narzędzia oferują możliwość wykorzystania DirectX 11 API, choć niekoniecznie bezpośrednio.

3.1 Własny silnik graficzny w języku C++

Proces doboru narzędzi rozpoczęto od zaprojektowania i stworzenia własnego silnika graficznego w języku C++ w wersji 11 z wykorzystaniem środowiska Microsoft Visual Studio 2012 na platformę Windows. Pozwoliło to nie tylko na praktyczne wykorzystanie umiejętności nabytych podczas uczestnictwa w zajęciach z Inżynierii Oprogramowania, ale także poznać w praktyce wykorzystanie Microsoft DirectX API w wersji 11 na najniższym dostępnym poziomie. Tworzenie tego rodzaju oprogramowania wymaga jednak nie tylko odpowiedniej ilości czasu, ale także dobrego zaprojektowania interakcji między klasami oraz zrozumienia wielu zagadnień z zakresu grafiki komputerowej, w większości takich, które wykraczają poza program przedmiotu Grafika Komputerowa i Wizualizacja. Z ostatnich dwóch powodów (oraz faktu, iż tworzenie silnika nie było tematem pracy inżynierskiej), najpierw ograniczono rozwój oprogramowania do jednej osoby, a następnie zrezygnowano z wykorzystania go w pracy, uzasadniając tę decyzję wysoką czasochłonnością wytwarzania owego narzędzia.

Zalety:

- większe możliwości w zakresie wykorzystania DirectX API
- lepsza znajomość możliwości oferowanych przez oprogramowanie
- brak kosztów

Wady

- czasochłonność
- wysoki próg wejścia (znajomość m.in. C++ oraz DirectX API)
- konieczność wytworzenia edytora poziomów i dodatnia odpowiednich funkcjonalności

3.2 Unreal Engine 4

Unreal Engine 4 (UE4) jest jednym z najpopularniejszych silników graficznych dostępnych na rynku, co jest złożeniem wielu czynników. Pierwszym z nich jest z pewnością niska cena płatnej licencji (subskrypcja miesięczna to koszt x\$) oraz darmowy dostęp dla studentów zarówno w ramach licencji edukacyjnej (należy w tym wypadku zgłosić chęć wydania licencji w ramach

przedmiotu prowadzonego na uczelni) jak i w ramach GitHub Developers Pack (należy jedynie potwierdzić studencki adres e-mail oraz wykorzystać [ang. *redeem*] licencję dostępną na odpowiedniej podstronie serwisu GitHub w formie kodu [ang. *serial code*]). Drugim jest jakość generowanych (renderowanych) obrazów – wykorzystanie algorytmu Voxel Cone Tracing [na pewno ten?] (algorytm rozwiązywania zagadnienia globalnego oświetlenia w czasie rzeczywistym) pozwala uzyskać niemal fotorealistyczną grafikę, jednak kosztem wysokich wymagań sprzętowych. Unreal Engine, zarówno w wersji 3 jak i 4, oferuje dostęp do kodu źródłowego w języku C++, co – po poznaniu API udostępnianego przez twórców – pozwala szybko i wygodnie rozwijać logikę gry. Umiejętność programowania nie jest jednak wymagana do tego ze względu na obecność mechanizmu blueprintów (znaleźć tłumaczenie), który pozwala tworzyć kod z wykorzystaniem bloków oferowanych bezpośrednio w silniku graficznym (sprawdzić czy dokładnie tak jest). Ze względu na wysokie wymagania sprzętowe (brak możliwości uruchomienia na komputerach laboratoryjnych oraz komputerach 75% zespołu) ostatecznie odrzucono to rozwiązanie.

Zalety:

- tania (darmowa) licencja
- wygodny edytor i dostęp do API w języku C++
- niemal fotorealistyczna grafika

Wady:

- konieczność poznania API
- wysokie wymagania sprzętowe

3.3 Unity

3.4 OGRE

3.5 CryEngine 3

3.6 Unreal Engine 3

3.7 Unreal Development Kit (UDK)

3.8 Autodesk Maya

3.9 Autodesk 3D Studio Max

3.10 Blender

3.11 Adobe Flash - Scaleform

Rozdział 4

Praca własna

Proces twórczy został podzielony na dwa główne etapy: projektowanie i implementację. Ponieważ tworzenie gry wymaga przygotowania oraz doboru odpowiedniego środowiska, etapy te zostały poprzedzone analizą problemu oraz burzą mózgów, na której zarysowały się wstępne wymagania funkcjonalne. Zdefiniowane zostały również główne wymagania pozafunkcyjne, takie jak docelowa platforma obsługująca grę. Wymagania te zostały szerzej opisane w załączniku 1 – Game Design Document (GDD). Pomysły zebrane podczas burzy mózgów zostały poddane wnikliwej analizie, co pozwoliło na stworzenie projektu. W celu zapewnienia spójnej wizji gry wśród członków zespołu, postanowiono skonstruować dokument, zawierający opis wszystkich decyzji podjętych podczas tworzenia projektu oraz wyjaśnienia dotyczące wszystkich wymagań projektowych – Game Design Document, stanowiący jednocześnie załącznik do niniejszej publikacji.

4.1 Projektowanie

4.1.1 Projekt silnika graficznego

4.1.2 Modele 3D

4.1.3 Projekt poziomu

4.2 Implementacja

4.2.1 Wykorzystanie technologii DirectX 11 w UDK

4.2.2 Zarządzanie projektem

W celu sprawnej organizacji pracy w zespole wykorzystano metody zarządzania projektami.

Początkowo użyto modelu kaskadowego. Jego zaletą jest sekwencyjność, pozwalająca oddzielić procesy analizy problemu, projektowania, implementacji oraz testowania i późniejszego utrzymania projektu. Metoda Waterfall, wykorzystująca ten model, nie jest jednak pozbawiona wad. Dotyczy one głównie dużych projektów, a więc nie miały miejsca w przypadku oprogramowania tworzonego w czteroosobowym zespole programistów. Korzystając z tej metody oszczędza się czas na planowaniu, faza ta zajmuje zaledwie 25% czasu pracy nad projektem. Rezultat końcowy zostaje ustalony jeszcze przed rozpoczęciem implementacji, podobnie jak poszczególne przyrosty implementacji. Każdy przyrost miał trwać 2 tygodnie i zawierał określone zadania. Rezultatem końcowym był produkt posiadający wartość biznesową, w tym przypadku - w pełni działająca gra. Co istotne, wartość biznesową produkt miał zyskać dopiero w przedostatnim przyroście.

W niedługim czasie po zaplanowaniu prac nad projektem okazało się, że metoda ta nie jest wystarczająca, zaczęło się pojawiać opóźnienie w pracach, które mogło spowodować pogorszenie jakości produktu. Zdecydowano się więc skorzystać z metod zwinnych. Metodyki Agile charakteryzują się stałą jakością, a sterowane są zakresem. W omówionej wcześniej metodzie Waterfall zakres był stały, zmieniała się jedynie jakość, a chęć utrzymania wysokiej jakości powodowała opóźnienie względem planu.

Ostatecznie zastosowano metodykę zwinną zbliżoną do metody Scrum. Zadania podzielono na część dotyczącą rozpoznania danego zagadnienia i część implementacyjną. Realizowane były tygodniowe sprinty (przyrosty). Na koniec każdego z nich otrzymywany był prototyp posiadający pewną funkcjonalność. Ze względu na wcześniejszy nieudany eksperyment z metodą Waterfall

wartość biznesową projekt zyskał dopiero po kilku przyrostach. Oznacza to, że pierwsze efekty nie były satysfakcjonujące, jednak prezentowały postęp prac. Każdy sprint rozpoczynał się spotkaniem zespołu, na którym omówiono pozostałe zadania oraz zaplanowano jakie funkcjonalności będą implementowane w kolejnym przyroście. W efekcie udało się zachować jakość wykonania, modyfikując nieznacznie zakres dostępnych funkcjonalności.

Rozdział 5

Podsumowanie

Rozdział 6

Literatura

Rozdział 7

Dodatki

Literatura



© 2015 Krzysztof Marciniak, Piotr Przybysz, Mikołaj Szychowiak, Ryszard Wojtkowiak

Instytut Informatyki, Wydział Informatyki
Politechnika Poznańska

Skład przy użyciu systemu \LaTeX .

Bib \TeX :

```
@mastersthesis{ key,  
  author = "Krzysztof Marciniak \and Piotr Przybysz \and Mikołaj Szychowiak \and  
Ryszard Wojtkowiak",  
  title = "{Projekt i implementacja gry komputerowej z wykorzystaniem technologii  
Microsoft DirectX 11}",  
  school = "Poznan University of Technology",  
  address = "Pozna{\n}, Poland",  
  year = "2015",  
}
```