JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 17 (2020) 412–416
Received: 21 September 2020
Accepted: 20 October 2020

Comparison of methods and tools for generating levels of details of 3D models for popular game engines

Porównanie metod i narzędzi generowania poziomów szczegółowości modeli 3D dla popularnych silników gier

Tomecki Michał*

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract
Computer games have come a long way since their inception using increasingly advanced graphics. With so many models present on the screen at the same time, simplified models are needed that will be replaced when they are further away from the camera in the game. The aim of the study was to compare the tool for automatic generation of model detail levels in Unity and Unreal Engine. The article presents the results of the equipment load test and the time needed for generation, as well as a survey checking the opinion of people.

Keywords: OD; 3D models

Streszczenie
Gry komputerowe przeszły długą drogę od momentu swojego powstania korzystając z coraz bardziej zaawansowanej grafiki. Przy tak dużej ilości modeli obecnych jednocześnie na ekranie potrzebne są uproszczone modele, które będą podmieniane gdy będą dalej od kamery w grze. Celem badania było porównanie narzędzia służącego do automatycznego generowania poziomów szczegółowości modeli w Unity oraz Unreal Engine. W artykule zostały zaprezentowane wyniki testu obciążenia sprzętu i czasu potrzebnego do generacji oraz ankiety sprawdzającej opinię ludzi.

Słowa kluczowe: poziomy szczegółowości modeli; modele 3D

*Corresponding author
Email address: **michal.tomecki@pollub.edu.pl** (M.Tomecki)

1. Wstęp

Niniejszy dokument opracowano dla autorów Gry komputerowe to obecnie jeden z najpopularniejszych sposobów wykorzystania komputerów przez młodych ludzi na całym świecie. Przeszły one długą drogę od momentu swego powstania. Zaczynając od gry Pong po raz pierwszy uruchomionej w rok 1972 przez firmę Atari poprzez gry typu Minecraft do gier takich jak Wiedźmin 3 czy wszystkim znane gry z serii Grand Theft Auto [1]. Z prostych gier w dwóch wymiarach mających kilku bitową grafikę dotarliśmy do gier w trzech wymiarach z grafiką tak realistyczną, że można ją pomylić z rzeczywistością..

Na samym początku swojej historii gry były pisane przez bardzo wąskie grono osób. Wraz z rozwojem języków programowania pisanie gier stawało się coraz łatwiejsze. Obecnie na rynku dostępne są darmowe silniki gier takie jak Unity Game Engine oraz Unreal Engine. Upraszczają one znacząco pracę osoby tworzącej daną grę poprzez udostępnianie wiele różnych potrzebnych funkcjonalności [2]. W środowisku Unreal Engine dostępne są tzw. schematy(blueprints) [3]. Pozwalają one na programowanie gier używając graficznego interfejsu wykorzystującego prostokąty połączone ze sobą liniami. W ten sposób potencjalny twórca gry nie musi znać nawet języka programowania i dalej być w stanie stworzyć swoją wymarzoną grę. Z powodu tak łatwego dostępu do możliwości tworzenia swojej własnej gry, twórcy muszą sprawić aby ich gra się w jakiś sposób wyróżniała od konkurencji na rynku. Wielu z nich wykorzystuje do tego wspaniale wyglądające modele graficzne. Niestety obciążają one znacząco system w trakcie wyświetlania ich na ekranie. W przypadku, w którym będzie ich dużo jednocześnie obecnych na ekranie gra może nie działać płynnie, a tego osoby, które w nią grają nie chcą widzieć. Dlatego większość twórców używa algorytmów teselacji. Służą one do dynamicznego podmieniania modeli obiektów obecnych w świecie gry [7]. Wraz z oddalaniem się kamery od obiektu jego model zostaje podmieniony na bardziej uproszczony. Pozwala to na zmniejszenie obciążenia systemu, co umożliwia płynniejsze działanie gry na danym systemie.

Większość obecnych na rynku silników gier posiada w mniejszym lub większym stopniu dostęp do narzędzia służącego do automatycznego generowania poziomów szczegółowości modeli 3D. Jedyne co twórca gry musi zrobić to wskazanie wybranego obiektu, dla którego mają zostać wygenerowane poziomy szczegółowości oraz ile ich ma być, jeśli mu nie pasuje domyślna ilość. Narzędzia te same przypiszą wygenerowane modele do wybranego modelu oraz wybiorą odległość, przy której mają one zostać podmienione. Nasuwa się jednak pytanie czy te modele wygenerowane przez różne środowiska będą na takim samym poziomie jakości czy może jednak będą się od siebie znacznie różnić.

2. Siliniki gier

Obecnie na rynku znajduje się wiele różnych silników gier. Są bardzo rozbudowanymi środowiskami służącymi do pisania głównie gier. Mają one zestaw narzędzi znacznie ułatwiający pisanie nowym w tej dziedzinie użytkownikom. Jedne z najpopularniejszych silników to Unity Game Engine oraz Unreal Engine.

Unity Game Engine jest to silnik służący do tworzenia gier zarówno w 2D jak i w 3D. Powstał on w 2005 roku jako silnik do pisania gier na system OS X. Następnie został rozbudowany i obecnie wspiera pisanie gier na ponad 25 różnych platform. Skrypty używane do kontrolowania rzeczy, które dzieją się wewnątrz niego na początku jego istnienia były pisane zarówno w stworzonym przez firmę Microsoft języku C# jak i w zmodyfikowanym języku JavaScript zwanym przez użytkowników UnityScript [4]. W roku 2017 UnityScript stracił wsparcie i od tego czasu jednym językiem używanym wewnątrz Unity jest C#.

Elementy używane wewnątrz silnika, takie jak skrypty, obiekty, pliki graficzne itd są umieszczone wewnątrz folderu "Assets" [5]. Programista może sam je tam ręcznie umieścić lub pobrać z Unity Asset Store. Jest to wbudowany w Unity sklep gdzie można znaleźć zarówno płatne jak i bezpłatne elementy wykonane przez innych użytkowników.

Unreal Engine jest to silnik gier wyprodukowany przez przedsiębiorstwo Epic Games [6]. Podobnie jak Unity Game Engine jest to duże, rozbudowane środowisko do tworzenia gier. Po raz pierwszy środowisko na rynku pojawiło się w 1998 roku. Wstępnie był on tylko wykorzystywany w grach typu FPS(First Person Shooter). Wraz z rozbudową silnika zwiększono jego zastosowanie, nie był już on ograniczony tylko do jednego typu gier. Podobnie jak Unity Game Engine umożliwia on tworzenie gier na wiele różnych platform. Unreal Engine został napisany w języku C++, ale skrypty kontrolujące co się dzieje wewnątrz gry mogą być pisane nie tylko w tym języku. Dostępne są również tak zwane schematy(blueprints). Pisanie w nich odbywa się w sposób graficzny poprzez połączenie liniami prostokątów, które odpowiadają wcześniej automatycznie wygenerowanym funkcjom takim jak np "Skok". Jest to bardzo duże ułatwienie dla nowych w tej dziedzinie użytkowników. Przez brak konieczności wcześniejszego znania jakiegokolwiek języka programowania jest on dość popularnym rozwiązaniem .

3. Narzędzia służące do automatycznego generowaniapoziomów szczegółowości modeli 3D

Praktycznie każdy obecny na rynku silnik gier posiada mniej lub bardziej rozbudowane narzędzia służące do automatycznego generowania poziomów szczegółowości modeli 3D. Wyjątkami nie są Unity Game Engine ani Unreal Engine. Oba silniki są jednymi z najpopularniejszych na rynku, więc bardzo dziwny by był brak takiej funkcjonalności. W pierwszym z nich jest to realizowane przez społeczność użytkowników tego środowiska. We wbudowanym w nie Unity Asset Store można znaleźć wiele płatnych jak i bezpłatnych narzędzi, które w różnym stopniu dokładności i jakości wygenerują odpowiednie mniej szczegółowe modele dla wybranych obiektów. Nie jest to jedyne miejsce gdzie można takie narzędzia znaleźć, innym miejscem, w którym jest ich dość sporo jest platforma GitHub. W przeciwieństwie do swojego poprzednika twórcy Unreal Engine sami napisali narzędzie do automatycznego generowania poziomów szczegółowości modeli 3D. Jest ono wbudowane w silnik i dostępne w oknie ze szczegółami wybranego modelu.

Jednym z popularniejszych bezpłatnych narzędzi dostępnych w środowisku Unity jest AutoLOD. Dostępne z platformy GitHub narzędzie jest bardzo rozbudowane jak na projekt niekomercyjny. Funkcja generowania poziomów szczegółowości modelu jest dostępna z trzech różnych miejsc w programie. Dzięki obecnej w silniku Unity możliwości dodawania opcji do pasków narzędzi, AutoLOD wygląda jak opcja dostępna domyślnie w środowisku, a nie jak dodatek. Wartą wspomnienia funkcją jest też możliwość usuwania wcześniej wygenerowanych poziomów szczegółowości dla wybranego modelu. Więcej szczegółowych ustawień znajduje się w menu dostępnym z paska narzędzi. Umieszczone są tam między innymi takie opcje jak wybór ilości wygenerowanych poziomów szczegółowości jak i maksymalna ilość poligonów.

Jako, że w silniku Unreal Engine jest domyślnie wbudowane narzędzie służące do generowania poziomów szczegółowości użytkownik nie musi nic szukać. W momencie otworzenia okna ze szczegółami wybranego obiektu po prawej stronie ekranu będą widoczne dostępne ustawienia. W sekcji dotyczącej poziomów szczegółowości jest dostępne kilka gotowych ustawień przygotowanych przez twórców, jeśli użytkownik nie chce nic zmieniać, to wystarczy tylko wybranie gotowego zestawu, a narzędzie automatycznie wygeneruje odpowiednią ilość poziomów szczegółowości i wybierze pozostałe potrzebne ustawienia. Są one łatwo dostępne w tej samej sekcji na wypadek gdyby domyślne wartości nie były wystarczające.

4. Testy i ich metodologia

W celu dokonania prawidłowej oceny obu wcześniej wymienionych narzędzi zostały wybrane dwa sposoby testowania każdego z nich. Przygotowano dziesięć różnych obiektów 3D. Każdy z nich miał inną ilość werteksów, trójkątów i inne elementy charakterystyczne, na których było można zauważyć zmiany wywołane przez zmianę poziomu szczegółowości tego modelu. Dla każdego obiektu zostały wygenerowane dodatkowe trzy uproszczone modele. W poniższych tabelach (Tabela 1 oraz 2) przedstawiono ilość werteksów każdego modelu w obu środowiskach. Jak widać modele w środowisku Unreal Engine mają ich więcej niż te w Unity oraz każdy następny poziom szczegółowości ma prawie dokładnie o połowę mniej werteksów. W przypadku uproszczonych modeli w Unity Game Engine stosunek ilości werteksów uproszczonego modelu do bardziej szczegółowego jest różny dla każdego poziomu. Dla konsekwencji w obu środowiskach zostały one ustawione do

zmieniania się kiedy są w takiej samej odległości od kamery.

Tabela 1: Ilość werteksów modeli w środowisku Unreal Engine

Model	LOD0	LOD1	LOD2	LOD3
M1	2053134	1026564	513282	256638
M2	321088	160540	80268	40134
M3	418653	209322	104658	52326
M4	569505	284751	142374	71187
M5	691357	345675	172836	86415
M6	175896	87948	43974	21984
M7	1697952	848976	424488	212244
M8	445219	222608	111302	55647
M9	1289151	644592	322302	161148
M10	248610	124302	62148	31074

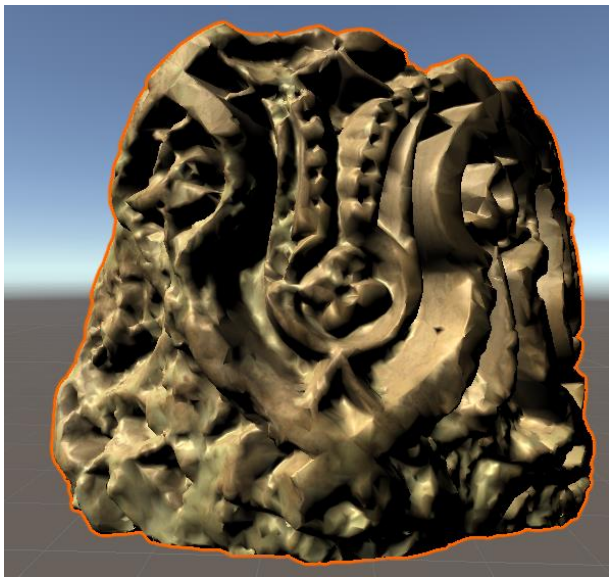
Tabela 2: Ilość werteksów modeli w środowisku Unity

Model	LOD0	LOD1	LOD2	LOD3
M1	356320	187377	100575	54585
M2	55039	28668	15158	8101
M3	73935	39818	21979	12262
M4	107261	60470	35080	20461
M5	123449	67193	37643	21339
M6	30762	16449	8949	4935
M7	288443	148270	77256	40480
M8	78990	42798	23861	13308
M9	218740	112210	58505	30883
M10	43234	23005	12550	6903

Pierwszy z nich polegał na porównania czasu potrzebnego na wygenerowanie takiej samej liczby poziomów szczegółowości tego samego modelu w obu środowiskach. Poza wspomnianymi wcześniej mierzonymi aspektami została sprawdzona i porównana liczba werteksów oraz trójkątów we wszystkich wygenerowanych modelach w obu środowiskach.

Drugi sposób służący do porównania obu narzędzi polegał na wykonaniu ankiety zawierającej zrzuty ekranu z wygenerowanymi poziomami szczegółowości modeli z obu środowisk. Następnie ankieta ta została wykorzystana do zbadania nią grupy ludzi i poznania ich opinii na temat aspektu wizualnego każdego modelu.

W ankiecie były umieszczone wszystkie dziesięć modeli w kolejności. Najpierw zostały pokazane wszystkie poziomy szczegółowości danego modelu obecne tuż przy kamerze w celu dokładnego pokazania wygenerowanego modelu (Rysunek 1). Następnie te same modele umieszczono w odległości, w której by były widoczne w grze (Rysunek 2). Na koniec z każdej serii zrzutów ekranów były umieszczone animacje pokazujące oddalanie się i przybliżanie do danego modelu. Pytania, które zadawano ankietowanym składały się z ogólnego pytania o wybranie środowiska, w którym ich zdaniem modele wyszły lepiej oraz z pytania o widoczność przez nich przejść między poszczególnymi poziomami szczegółowości każdego modelu na dołączonych animacjach.



Rysunek 1: Jeden z wykorzystanych w ankiecie modeli LOD3 tuż przy kamerze w Unity Game Engine

Rysunek 2: Jeden z wykorzystanych w ankiecie modeli LOD0 w odpowiedniej odległości od kamery w Unreal Engine

5. Wyniki testów

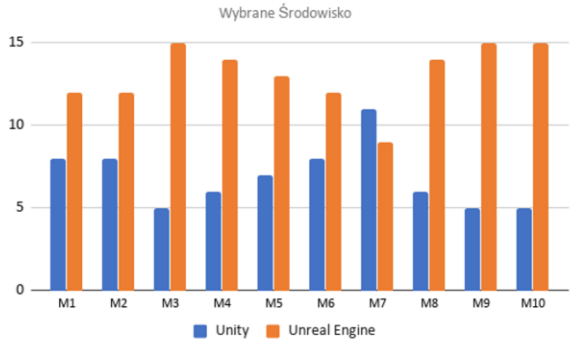
Pierwszy z przeprowadzonych testów, sprawdzający czas i obciążenie dał dość niespodziewane wyniki. Zaczynając analizę wyników od Unreal Engine można zauważyć, że podczas generowania poziomów szczegółowości poszczególnych modeli, dla obiektów, które miały więcej werteksów, narzędzie obecne domyślnie w tym silniku potrzebowało więcej czasu niż dla tych modeli, które miały mniej werteksów. Podobną obserwację można było zauważyć również na obciążeniu sprzętu, który był mocniej wykorzystywany podczas pracy nad bardziej skomplikowanymi obiektami.

Dokonując analizy tego samego testu w środowisku Unity Game Engine dostaje się niespodziewany rezultat. Otóż wykorzystując narzędzie AutoLOD do generowania poziomów szczegółowości modeli, dokonano obserwacji, że w momencie wybrania przycisku odpowiedzialnego za rozpoczęcie generacji nie pojawia się żadne okienko z paskiem pokazującym postęp generacji, jak to jest obecne w drugim środowisku, lecz tworzone są siatki z mniejszą szczegółowością od razu dostępne do użytku. ustawione W tym samym czasie do modelu zostaje dołączony odpowiedni komponent odpowiedzialny za zmianę poziomów szczegółowości gdy ten będzie się znajdował w odpowiedniej odległości od kamery. Wykonanie tego samego testu ponownie w celu sprawdzenia obciążenia sprzętu, na którym wykonywano badania, pokazało, że wykresy wyświetlające wydajność komputera nie zmieniają się.

Przeprowadzona ankieta została wypełniona przez dwadzieścia osób z czego osiemnaście z nich to byli mężczyźni, a reszta kobiety. Szesnaścioro z nich to były osoby pomiędzy 20 a 25 rokiem życia. Wyniki dla pierwszego pytania po zestawie zrzutów ekranów z wygenerowanymi modelami zostały przedstawione poniżej (Tabela 3).

Tabela 3: Wyniki ankiety sprawdzającej preferowane środowisko

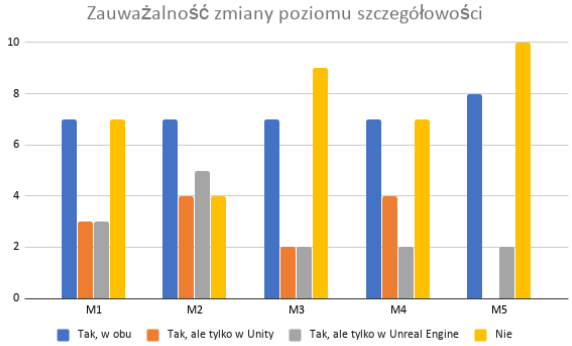
Model	Unity (osoby)	Unreal Engine (osoby)
M1	8	12
M2	8	12
M3	5	15
M4	6	14
M5	7	13
M6	8	12
M7	11	9
M8	6	14
M9	5	15
M10	5	15
Średnia	6,9	13,1



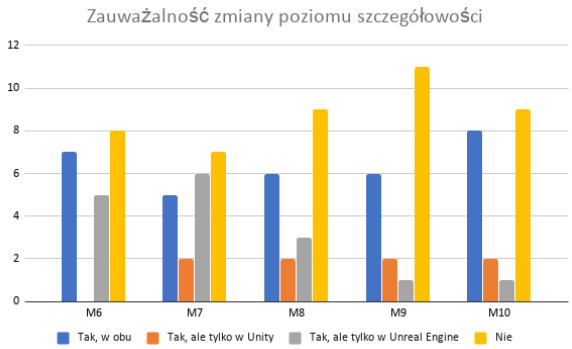
Rysunek 3: Wykres przedstawiający wybrane przez badanych środowisko

Jak to jest widoczne na powyższym rysunku (Rysunek 3) zdecydowanie większa część ludzi wybrała środowisko Unreal Engine. Ponadto tylko jeden model (M7) był uważany za lepszy w środowisku Unity Game Engine. W przedstawionej wcześniej tabelce (Tabela 1) poza dokładnymi wynikami ankiety widać również wyliczoną na ich podstawie średnią(ostatni rząd). Dzięki niej wiadomo, że średnio 65.5 % badanych wybierało Unreal Engine.

Drugim elementem badanym w przeprowadzonej ankiecie była widoczność zmian poszczególnych poziomów szczegółowości danych modeli na dołączonych po zestawie zrzutów ekranu gifach. Ankietowani wybierali czy widzieli/nie widzieli zmiany w obu środowiskach albo czy tylko w jednym z nich. Na zamieszczonych poniżej rysunkach (Rysunek 4 i 5) zostały przedstawione wyniki tych pytań.



Rysunek 4: Wykres przedstawiający odpowiedź ankietowanych na zauważalność zmian poszczególnych poziomów szczegółowości dla modeli 1-5



Rysunek 5: Wykres przedstawiający odpowiedź ankietowanych na zauważalność zmian poszczególnych poziomów szczegółowości dla modeli 6-10

6. Wnioski

W obecnym świecie gier komputerowych, który cały czas się rozwija nowe produkcje dużych firm zawierają coraz więcej modeli o coraz lepszej jakości. W celu utrzymania takiej samej płynności gry jak te poprzednie, studia gier muszą pracować nad optymalizacją swoich produkcji. Dzięki temu ich gra będzie dostępna dla większej liczby użytkowników. Jednym ze sposobów w jaki mogą tego dokonać jest zastosowanie teselacji. Dynamiczne podmienianie modeli na te o mniejszej szczegółowości znacząco poprawia płynność gry. Ręczne tworzenie takich modeli może zająć bardzo dużo czasu i zasobów, więc korzystanie z narzędzi, które automatycznie wygenerują te modele może być dobrym pomysłem. Niekomercyjne narzędzia, które zostały wcześniej przetestowane mogą być dobrym punktem startowym dla mniejszych programistów. Badania pokazały, że mimo szybszego oraz praktycznie nieodczuwalnego na sprzęcie generowania tych poziomów szczegółowości korzystając z narzędzia AutoLOD w środowisku Unity Game Engine, ankietowani woleli wygląd modeli, które zostały wykonane przez wbudowane w środowisko Unreal Engine narzędzie. Biorąc pod uwagę odpowiedzi ankietowanych na drugie pytania w przeprowadzonej ankiecie, większość ludzi nie zauważa różnic między mniej i bardziej szczegółowymi modelami. Dzięki tej informacji można przyjąć, że uwaga potencjalnego gracza nie będzie odciągana przez zmieniające się modele obecne w dalszej odległości od niego.

Literatura

[1] P. Glancey, The Complete History of Computer and Video Games. EMAP IMAGES 1996.

[2] A. Okita, Learning C# Programming with Unity 3D, Wyd. CRC Press Taylor & Francis Group, Nowy Jork 2015.

[3] Wszystkie potrzebne informacje o blueprintach dostępnych w Unreal Engine, https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html [06.09.2020].

[4] Krótkie wyjaśnienie Level of Details oraz spsosób ich tworzenia w Unreal Engine, https://docs.unrealengine.com/en-US/Engine/Content/Types/StaticMeshes/HowTo/LODs/index.html[06.09.2020].

[5] T. Norton, Learning C# by Developing Games with Unity 3D Beginner's Guide, Wyd. Packt Publishing, Birmingham 2013.

[6] W. Goldstone, Unity Game Development Essentials, Wyd. Packt Publishing, Birmingham 2009.

[7] A. Cookson, R. DowlingSoka, C. Crumpler, Unreal Engine w 24 godziny. Nauka tworzenia gier. Helion 2016.