Grafika 3D i systemy multimedialne

POKÓJ STRACHÓW

Autorzy:
Piotr Osipa, nr indeksu: 200756
Paweł Andziul, nr indeksu: 200648
Maciej Kiedrowski, nr indeksu: 200105
Joanna Piątek, nr indeksu: 199966

Grupa: Czwartek TN/13:15

Data oddania: 19.01.2017

Prowadzący: dr inż. Jan Nikodem

Ocena pracy:

Spis treści

1 Instrukcja obsługi

1.1 Uruchomianie aplikacji

Aplikacja jest uruchamiana za pomocom pliku o nazwie „DomStrachow.exe". Pliki znajdujące się w folderze „data" są niezbędne do jej prawidłowego działania.

1.2 Ustawienia graficzne

Aplikacja pozwala na wybór detali wyświetlanych elementów poprzez wybranie przygotowanych zestawów ustawień „ustawienia minimalne", „ustawienia maksymalne". Ustawieniem domyślnym jest ustawienie maksymalne.

1.3 Sterowanie

Sterowanie postacią odbywa się za pomocą kontrolera Xbox 360 lub za pomocą klawiatury lub myszki według tabeli 1 i 2.
//TODO wstawić tabelę

		Funkcja	
		Sterowanie postacią	Poruszanie kijem
Urządzenie	Klawiatura	W, S, A, D lub strzałki	Q, E, R, F
	Pad do Xbox360	Lewa gałka analogowa	Prawa gałka analogowa

Tabela 1: Sterowanie

2 Implementacja projektu

2.1 Świat

Świat widoczny na scenie zaprojektowany został z wykorzystaniem narzędzia „World Machine". Narzędzie to pozwoliło na zaprojektowanie własnego terenu, który następnie został zaimportowany do Unity jako mapa wysokości.

Zalesienie Świata
Zalesienie świata (tekstury ziemi, trawa i drzewa) zostały ręcznie naniesione z wykorzystaniem modeli pochodzących z Assetu „World Builder".

Ocean
Ocean został zaimportowany z paczki Unity Standar Assets (Basic Water) i naniesiony na scenę.

2.2 Pokój

Model pokoju stworzony został w narzędziu Blender. Model ten jest sześcianem o odwróconych normalnych ścian, co pozwoliło na oświetlenie jego wnętrza. W modelu tym wycięte zostało miejsce na okno, oraz został on podzielony na części przygotowane do nałożenia tekstury.

2.3 Biurko i okno

Modele biurka i okna zostały stworzone z wykorzystaniem narzędzia Blender.

Biurko
Biurko zawiera w sobie odpowiednio zeskalowane i połączone sześciany i bryły. Biurko posiada elementy stylistyczne wykonane przy pomocy narzędzia *Extrude*.

Okno
Okno zawiera w sobie modele szyby oraz ramki, która ją podtrzymuje. Przeźroczystość szyby uzyskana została poprzez zastosowanie odpowiedniego materiału „Transparent/Diffuse" w Unity

2.4 Zasłony

Model zasłony utworzony został w narzędziu Blender wykorzystując do tego gotową powierzchnię, zmodyfikowaną tak by wyglądała jak delikatny materiał. Model ten został zaimportowany do Unity, gdzie przypisano do niego odpowiednie parametry fizyki (*Cloth physics*).
Zastosowanie zasłon wymagało utworzenia zestawu colliderów w postaci kapsuł, dzięki czemu możliwa była interakcja firany z obiektami takimi jak kij, piłka czy postać.

2.5 Sterowanie postacią

Sterowanie postacią zostało zaimportowane z paczki Unity Standard Assets zawierających m.in. moduł FirstPersonCharacter posiadający gotowe skrypty upraszczające takie elementy jak poruszanie się postaci, obrót, skok. Asset ten posiada również ścieżkę dźwiękową zawierającą odgłosy kroków.

2.6 Poruszanie kijem

Element poruszania kijem został zaimplementowany poprzez modyfikację skryptu poruszania kamerą zawartego w assecie *FirstPersonCharacter*. Modyfikacja ta pozwoliła na poruszanie kija względem położenia obserwatora – kij znajduje się przed nim, w odpowiedniej odległości, pod odpowiednim kątem.
Poruszanie kijem niezależnie od kamery obserwatora polegało na translacji położenia kija względem położenia obserwatora, a następnie jego rotację.

2.7 Piłka

Model piłki utworzony został bezpośrednio w Unity jako gotowy obiekt sfery. Do obiektu dołączony został model odpowiadający za fizykę materiału. Model ten został wygenerowany w Unity, następnie został zmodyfikowany by lepiej oddawać właściwości fizyczne piłki. Piłka posiada specjalnie przygotowaną teksturę imitującą rzeczywisty wygląd piłki do gry w koszykówkę.

2.8 Wazon i jego rozbicie

W celu zademonstrowania możliwości interakcji i destrukcji otoczenia, w pomieszczeniu umieszczona została waza. Przy mocnym uderzeniu, upadku, rozpada się na mniejsze fragmenty, z którymi użytkownik może przeprowadzać dalszą interakcję.

2.8.1 Elementy

Waza
Waza została zamodelowania w środowisku *Blender*. Podstawowym kształtem dla wazy jest okrąg, z którego za pomocą narzędzia *Extrude* i skalowania w prosty sposób można utworzyć symetryczne, owalne przedmioty.
Krawędzie utworzonego modelu zostały wygładzone za pomocą narzędzia *Smooth* dostępnego w oprogramowaniu.

Fragmenty
Modele rozbitej wazy - zarówno całej, jak i już rozbitych elementów które podlegają dalszej destrukcji zostały również utworzone w narzędziu *Blender*. W tym celu skorzystano z dodatkowego narzędzia *Cell Fracture* dostępnego dla *blendera*. Dodatek ten pozwala na podział utworzonego modelu na n fragmentów za pomocą tesselacji Woronoja. Środki obszarów mogą zostać wygenerowane automatycznie, lub zostać wybrane przez użytkownika. Korzystając z narzędzia *Cell Fracture* użytkownik ma do dyspozycji szereg opcji, m.in. wybór maksymalnej liczby fragmentów na które zostanie podzielony model, czy wielkość odstępu pomiędzy nowo utworzonymi fragmentami. Z jednej strony pogarsza to wizualne wrażenia przy rozbiciu przedmiotu, jednakże pozwala ograniczyć ryzyko "wybuchu" szczątków na

wskutek błędu silnika fizycznego. Ma to szczególnie duże znaczenie przy podziale modelu na kilkadziesiąt lub kilkaset fragmentów.

2.8.2 Import do środowiska Unity

Po zaimportowaniu modelu do *Unity*, należy wygenerować *collidery*, poprzez zaznaczenie opcji *Generate Colliders* w Inspektorze modelu. Na poszczególnych fragmentach należy zaznaczyć opcję *Convex* w komponencie **Mesh Collider** w celu umożliwienia wykrywania kolizji pomiędzy tymi obiektami. Aby zminimalizować odbicia obiektu od podłoża po upadku, materiał powinien byś ustawiony na materiał z małym współczynnikiem odbicia, np. metal. Następnie każdemu fragmentowi należy dodać komponent **RigidBody**, który daje możliwość ustawienia takich cech obiektu jak masa czy opory powietrza wpływające na zachowanie obiektu na scenie. Tak przygotowany obiekt w postaci *prefabu* jest gotowy do użycia na scenie.

2.8.3 Skrypt

W celu symulacji zniszczenia obiektu w środowisku *Unity*, konieczne jest przygotowanie modeli przed i po zniszczeniu. W momencie wykrycia działania na obiekt siły wystarczającej do jego zniszczenie, należy podmienić model na scenie.
Dla wazy został utworzony skrypt *DesctrucibleVase.cs*, jeżeli w momencie kolizji z innym obiektem zostanie przekroczona uprzednio zdefiniowana wartość progowa, dynamicznie tworzona jest nowa instancji wazy posiadająca za model wazę zniszczoną. Następnie dla każdego fragmentu nowej fazy przypisywana jest rotacja i pęd istniejącego obiektu, po czym obiekt ten jest usuwany ze sceny i zastępowany nowo utworzonym. W ten sposób użytkownik nie widzi podmiany modelu, a jedynie efekt rozpadającej się na kawałki wazy. Ta sama technika może zostać wykorzystana przy dowolnym obiekcie w środowisku *Unity*.

2.9 Świeca

Kolejnym elementem prezentowanym na scenie jest świeca. Głównym powodem wyboru tego elementu była potrzeba zaprezentowania ognia, który jest umocowany na przesuwalnym obiekcie. Dzięki temu przesuwając święcę można zaobserwować ruchy płomienia.

2.9.1 Elementy

Wosk
Model części woskowej został stworzony w programie *Blender*. Do tego celu wykorzystano gotowy element *Cylinder*, którego górny fragment został zmodyfikowany tak, by przypomiał roztopiony wosk. Bardzo pomocna okazała się tu opcja *Proportional Edit*, dzięki której wystarczyło zaznaczyć jeden punkt na siatce obiektu, a po jego przesunięciu pobliskie punkty podążały w tym samym kierunku. Kolejnym krokiem było dodanie knotu świecy. Został on wykonowany z elementu *Cylinder* ukształtowanego następnie za pomocą narzędzia *Extrude*.

Płomień
Płomień świecy został utworzony w całości w środowisku *Unity*. Do tego celu wykorzystano gotowy element *Particle System*, czyli system cząsteczek. Domyślna konfiguracja tego komponentu nie przypomina ognia, lecz rozproszone cząstki wydobywające się z danego miejsca. Należało więc zmienić właściwości elementu, takie jak prędkość ruchu cząstek, ich

maksymalna ilość, kształt grupy cząstek w trakcie cyklu życia czy ich barwa. Płomień został dodany jako podelement do woskowej części świecy, by poruszał się razem z nią.

Źródło światła
W celu nadania płomieniowi realnego wyglądu zostało do niego dodane punktowe źródło światła. Jest ono gotowym elementem dostępnym w środowisku *Unity*.

2.9.2 Import do środowiska Unity

Po stworzeniu modelu wosku należy zaimportować go do środowiska *Unity*. Następnie konieczne jest wygenerowanie *colliderów* w taki sam sposób, w jaki odbyło się to dla wazy. Należy dodać także komponent **RigidBody** odzwierciedlający cechy fizyczne obiektu. Jak wspomniano wcześniej, utworzone w *Unity* elementy - płomień i światło - trzeba umocować na części woskowej tak, by stanowiły razem tzw. *prefab*, gotowy do ponownego użycia w programie.

2.9.3 Skrypty

Żeby osiągnąć pożądane zachowanie płomienia należało zadbać o następujące kwestie:

- w trakcie przesuwania świecy ogień powinien pochylać się w przeciwnym kierunku,
- moc światła powinna odzwierciedlać ilość cząsteczek płomienia.

Pierwsze zagadnienie rozwiązano za pomocą skryptu *FlameBehaviourScript.cs*. Dzięki dodaniu w *Unity* elementu **Rigidbody** możliwe jest pobranie aktualnej prędkości woskowej, niezmiennej części świecy. Na jej podstawie obliczany jest wektor prędkości, który powinien zadziałać na cząstki płomienia. Zostały tutaj uwzględnione także sytuacje, kiedy obiekt zostanie przesunięty ze zbyt dużą szybkością lub przewróci się. W takiej sytuacji ogień gaśnie. Kolejny skrypt - *CandleLightBehaviour.cs* zapewnia odpowiednią moc światła. Na bieżąco sprawdzana jest liczba aktualnie "żyjących" cząstek płomienia. Następnie obliczany jest jej stosunek do ich ilości maksymalnej. Taki ułamek zostaje przemnożony przez początkową, ustaloną moc. Dodatkowo, kiedy płomień gaśnie, światło zostaje wyłączone.