Politechnika Wrocławska

Interaktywna grafika komputerowa

Sprawozdanie z laboratoriów

Autor:
Maciej Szaflik 195925

Prowadzący:
Dr inż. Tomasz Kapłon

3 maja 2016

1 Założenia projektowe

Zadaniem na postawionym na laboratoriach było wykonanie 30 sekundowej animacji, przedstawiającej postać ludzką. Animacja nie powinna była składać się z jednego zapętlonego ruchu, wykonywanego przez dłuższy okres. Dodatkowo powinna zawierać elementy interaktywne, takie jak obracanie kamery umożliwiające obejrzenie modelu z każdej strony. Zakończony projekt został udostępniony w następujących miejscach:

- Repozytorium : https://github.com/MaciejSzaflik/HexagonalThings
- Strona: http://maciejszaflik.github.io/HexagonalThings/index.html

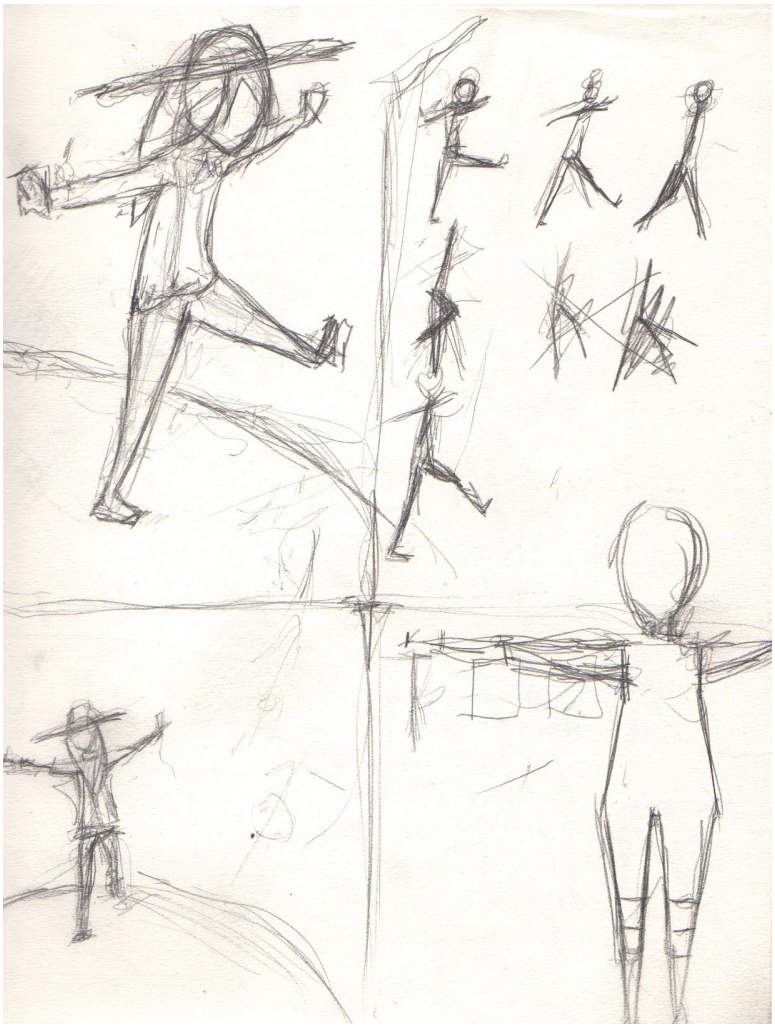
W celu wykonania projektu postanowiono wykorzystać następujące narzędzia:

- Three.js - jako biblioteka do renderowania WebGL
- Blender - wykonanie oraz animacja modelu postaci

Biblioteka Three.js została wybrana z jednego ważnego powodu. Umożliwiała ona łatwą prezentacje wyników pracy, bez konieczności instalacji dodatkowego oprogramowania, poprzez wstawienie animacji na statyczną stronę internetowa. Jedyne co jest wymagane to przeglądarka obsługująca WebGL. Możliwość prostego dostępu do projektu przeważyła ponad większą wydajnością oraz dodatkowymi możliwościami OpenGL'a.

2 Faza projektowa

Pierwszym krokiem w celu wykonania animacji było zaprojektowanie modelu, oraz akcji przez niego wykonywanych. Postanowiono wykonać postać chodzącego człowieka. Do wykonania były dwie animacje - chodzenia, oraz machania (do wykorzystania gdy postać znajduje się w stanie spoczynku). Elementem interaktywnym miała być możliwość przemieszczania się postaci poprzez klikanie. Dodatkowo należało sporządzić proste animacje tła. Ze względu na małą ilość czasu oraz ograniczone umiejętności w zakresie modelowania postanowiono wykonać animacje w stylistyce *low-poly*.



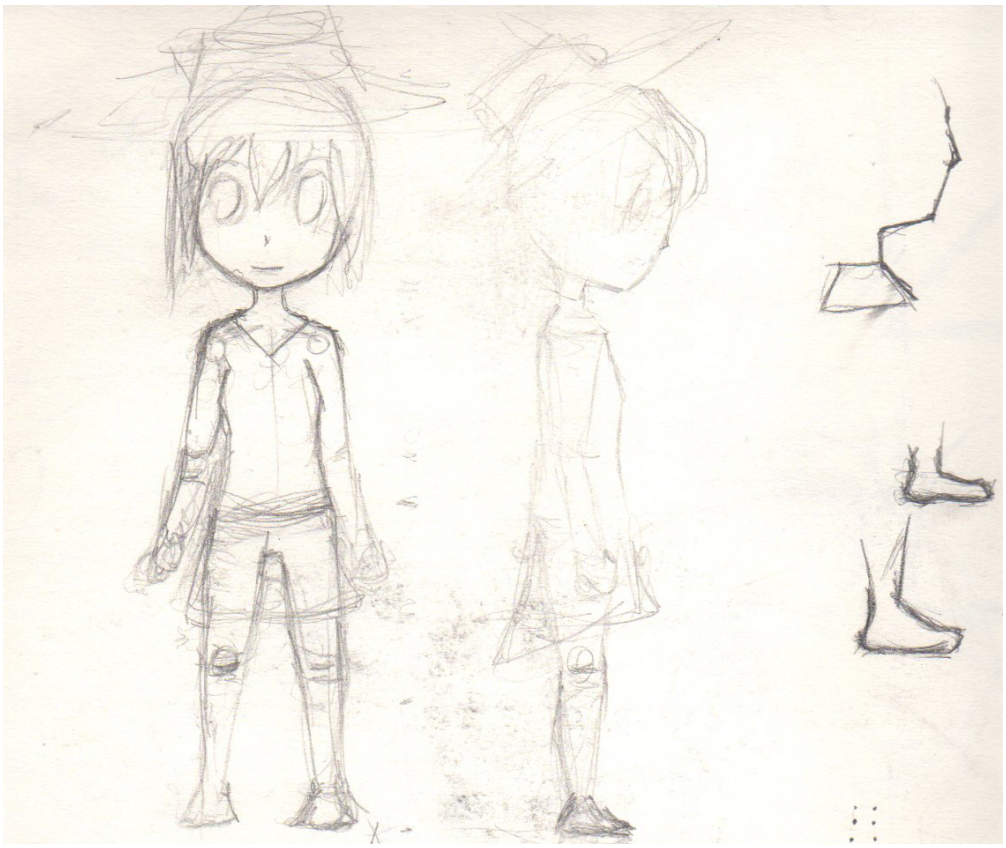
Rysunek 1: Wstępny projekt postaci, oraz animacji

3 Wykonanie

Projekt powstawał etapami. Głównym tego powodem była konieczność sprawdzenia możliwości środowiska programistycznego. Podstawowym wymaganiem był import modelu oraz wspieranie odtwarzania animacji szkieletowych (opartych na kościach). Tym samym po wykonaniu każdego z elementów, był on wdrażany do głównego projektu w celu sprawdzenia poprawności działania.

3.1 Modelowanie

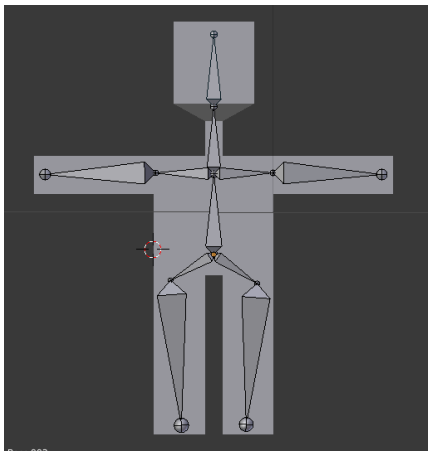
W celu wykonania modelu został sporządzony prosty szkic modelu z dwóch rzutów - przodu i boku. Umożliwiło to łatwiejszą prace nad modelem niż próba wykonania wszystkiego z głowy. Jako że rysunek był bardzo schematyczny, pasował on do wybranej stylistyki - mała ilość detali.



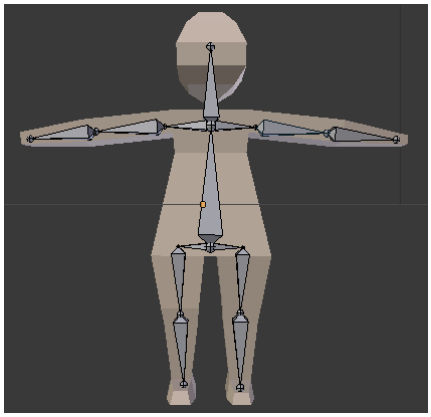
Rysunek 2: Projekt modelu postaci

Wybraną techniką modelowania było : *Box Modeling*. Polegało to na stworzeniu podstawowego kształtu (w tym wypadku kostki), i jego modyfikowanie poprzez sub dywizje oaz wyciąganie, skalowanie oraz rotacje *face'ów'*. W celu zwiększenia ilości detalu dodawane były *edge loops*, czyli kolejne podziały. Technika ta wydawała się znacznie prostsza niż ręczne dodawanie krawędzi. Dodatkowo łatwo było osiągnąć zamierzany cel - model *low-poly*, co przy wykorzystaniu narzędzi takich jak *sculpting*, byłoby bardzo problematyczne.

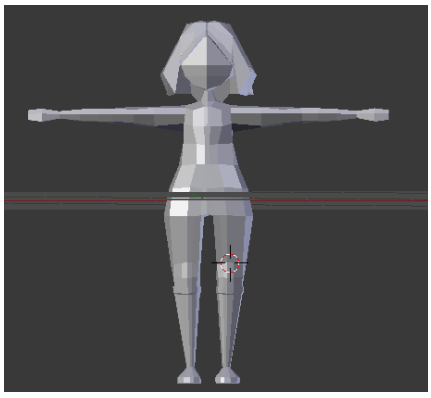
Na kolejnych rysunkach, zostaną przedstawione kolejne etapy powstawiania modelu. Może powodować zdziwienie fakt, że niektóre z nich, mimo iż są dalekie od ukończenia posiadają kości. Jest to spowodowane wcześniej wspomnianym faktem - każdy z modeli był testowany na stronie internetowej, w celu przetestowania czy framework może zostać wykorzystany do wykonania projektu.



Rysunek 3: Wyjściowy kształt



Rysunek 4: Zmiana sylwetki



Rysunek 5: Zagęszczenie siatki, dodanie detalu, zmiana sylwetki aby bardziej przypominała ludzką

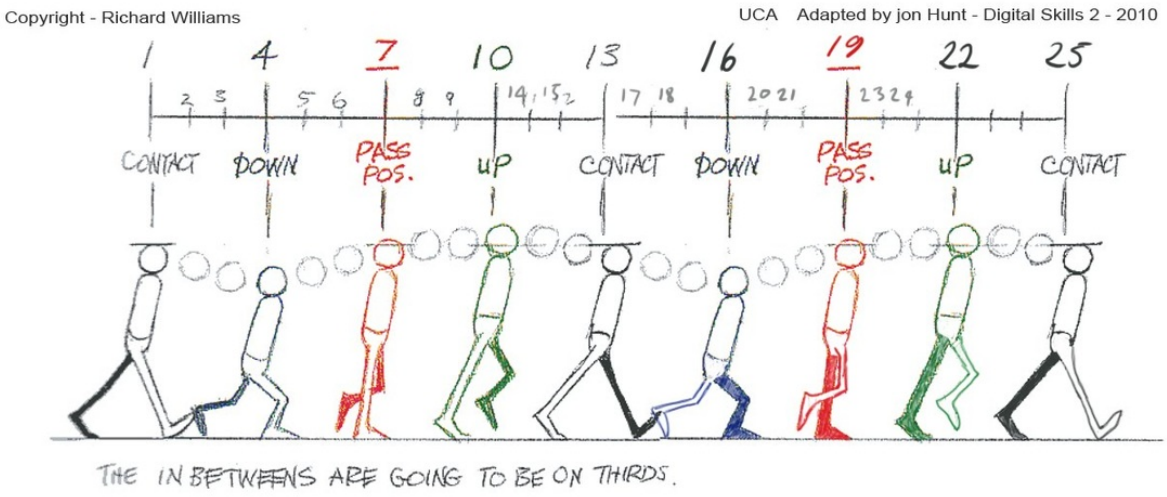
Rysunek 6: Efekt końcowy, Vertext'y zostały pokolorowane, oraz została dodana czapka

4 Animacja postaci

Animacja postaci w całości została wykonana w Blenderze. Stworzono standardowy szkielet ludzki z małą dokładności. Uznano że mniejszą liczba kości będzie łatwiej kontrolować i będzie prostsza do opanowania. Armatura została przypisana do modelu przy pomocy automatycznego przyznawania wag do wierzchołków. Duża cześć elementów: ręce, stopy i głowa nie przypisały się poprawnie, przez co trzeba było wykorzystać narzędzie do malowania wag.

Do stworzenia animacji wykorzystano technikę klatek kluczowych. Została wybrana odpowiednia referencja [7], przedstawiająca kolejne klatki. Armatura postaci została dopasowana do kolejnych klatek, zaś przekształcenia pomiędzy nimi zostały pozostawione do automatycznej interpolacji.

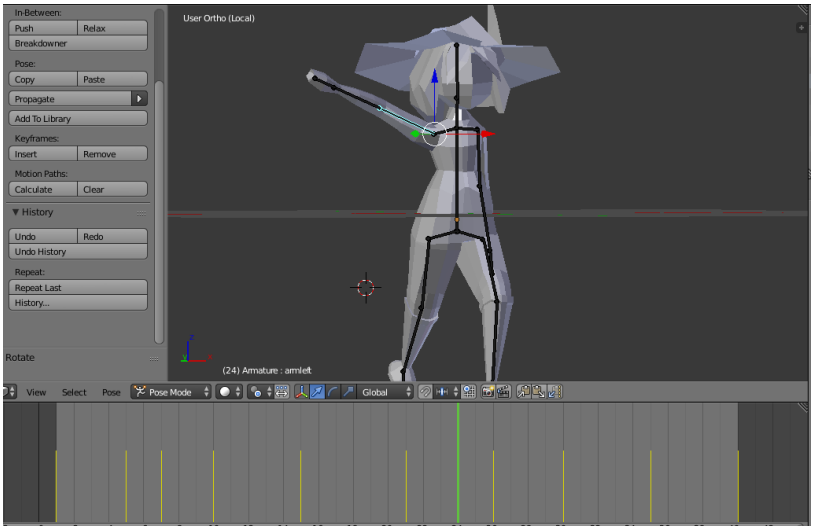
Wykonane zostały dwie animacje - chodzenia i machania. Ta pierwsza okazała się o wiele bardziej problematyczna. Animacja która która poruszała jedynie kończyny była bardzo nienaturalna, i powodowała że postać wydawała się ślizgać po powierzeni. Aby temu zapobiec dodano, zgodnie z referencja, ruch pionowy całej postaci.



Rysunek 7: Referencja wykorzystana przy animacji chodzenia

Problemem okazało się zapętlenie animacji. Różne pozy na początku i końcu animacji powodowały widoczne przeskoki, co wyglądało bardzo nienaturalnie. W celu rozwiązania tego problemu starano się dopasować końcową pozycje animacji aby zawsze zbiegała ona pozy początkowej. Blender umożliwiał skopiowanie całej pozy co znacznie ułatwiło to zadania.

Pułapka było animowanie postaci przy włączonej kamerze perspektywicznej [8]. Podczas nadawania rotacji Blender automatycznie dostosowywał oś obrotu co powodowało niespodziewane ruchy, w kierunkach zupełnie innych niż zakładane. Rozwiązanie było bardzo proste - należało animować postać przy włączonej kamerze ortograficznej w odpowiednim rzucie. Dzięki temu podczas nadawania rotacji kością działo się to tylko w jednej, określonej przez rzut osi.



Rysunek 8: Tworzenie animacji przy wykorzystaniu techniki klatek kluczowych w blenderze. Niebieskie linie na dole reprezentują kolejne zapisane pozycje kości pomiędzy którymi interpolowane są wartości podczas animacji

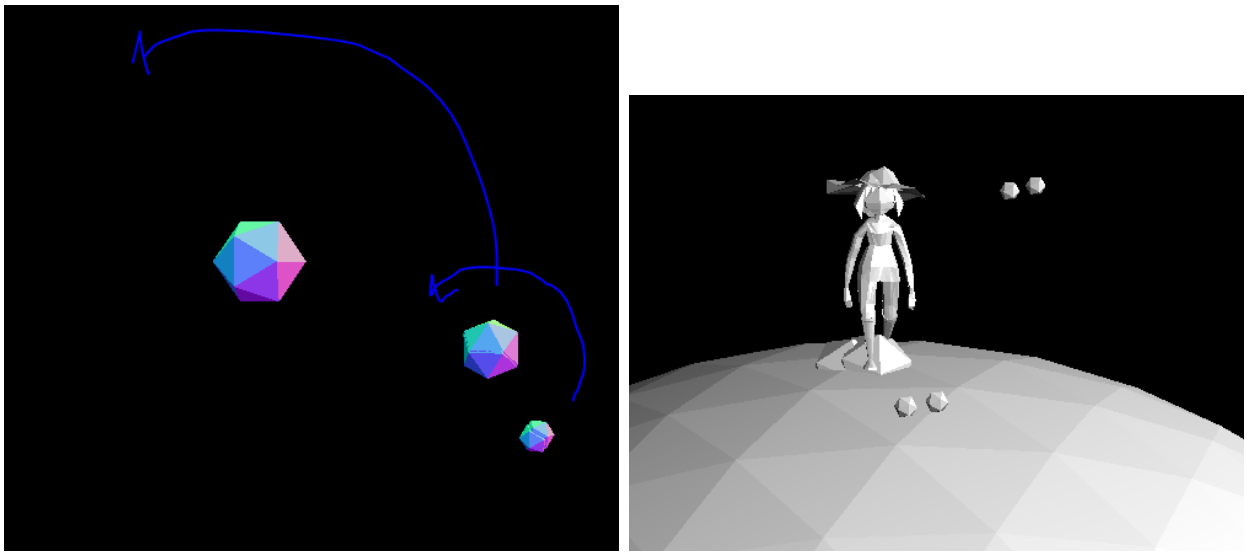
5 Otoczenie

Głównym pomysłem na otocznie były planety zawieszone w przestrzeni, krążące wokół siebie i umiejscowieniu stworzonego modelu na jednej z nich. Ta część projektu miała zostać wykonana przy wykorzystaniu jedynie kodu. Na scenie miało się znajdować kilka świateł - dwa orbitujące wokół głównej planety, a jedno znajdujące się zawsze blisko głównego modelu.

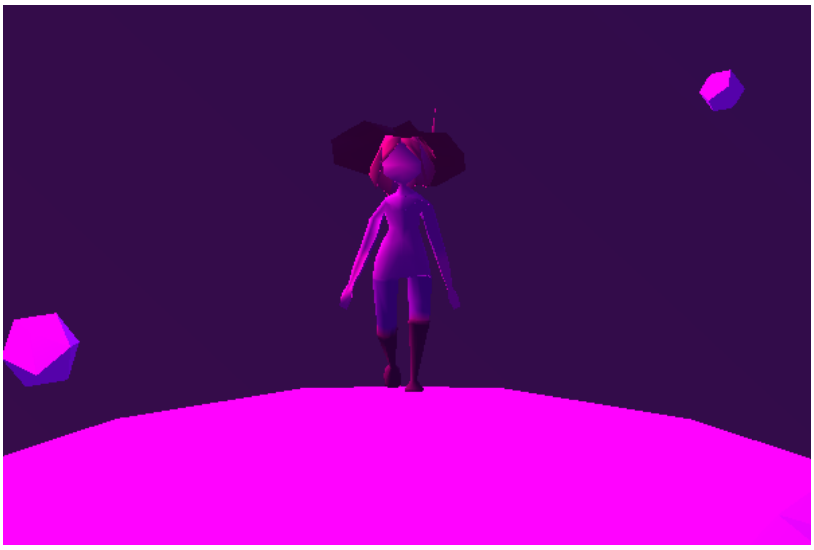
Animacje otoczeniu są proste - polegają na przemieszczaniu się w równomierny sposób całych obiektów, bądź też ich skalowaniu. W głównej pętli renderującej znajduję się obiekt zarządzający trwającymi transformacjami, z którego można usuwać i dodawać potrzebne animacje.

5.1 Planety

Planety to proceduralnie generowane dwudziestościany foremne. Kształt ten to zostać wykorzystane do równomiernego ich zniekształcaniu, ale ze względu na czas temat ten został porzucony. Aby wprawić planety w ruch, stworzone transformacje ruchu po okręgu względem innego obiektu, oraz rotacje w określonej osi. Na scenie końcowej znajduje się około 20 obiektów orbitujących względem głównej planety, oraz 80 orbitujących względem nich.

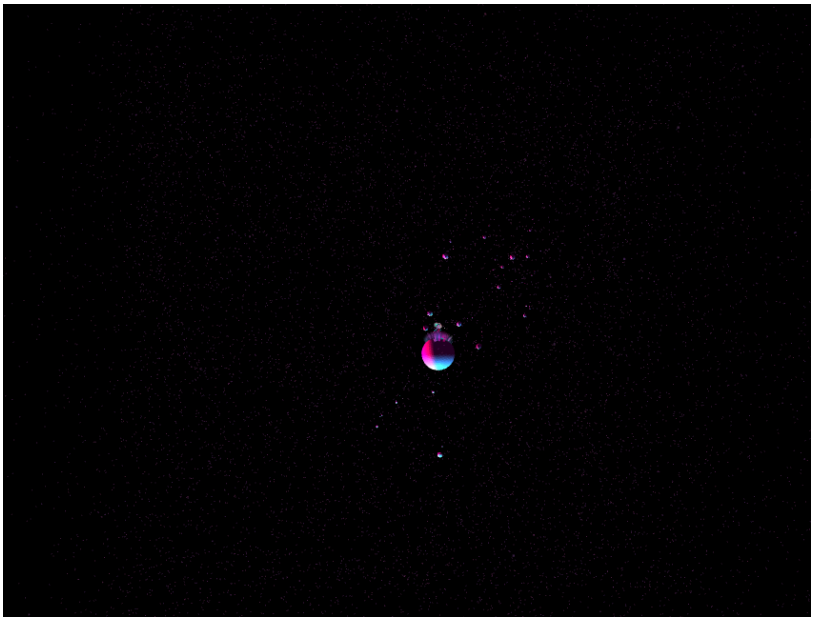


Rysunek 9: Dodawanie kolejnych elementów sceny



Rysunek 10: Kolory

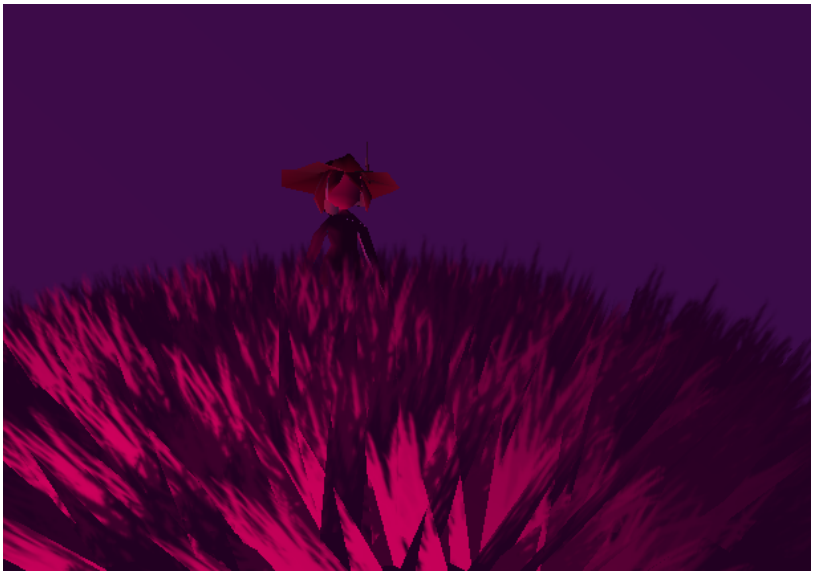
Dodatkowym elementem nie interaktywnym otoczenia są *gwiazdy*. Są one stworzone przy pomocy dwóch chmur cząsteczek. Chmury te obracają się względem siebie, nadając wrażenie że główne elementy sceny znajdują się ich wnętrzu. Początkowe obawy o wykorzystanie cząsteczek, zostały szybko rozwiane po paru testach z których wynikało, że utworzenie nawet do 300000 punktów nie wpływało znacząco na wydajność, dopóki cząsteczki są połączone w jedną geometrie. W tej chwili na scenie znajduje się około 10000 cząsteczek.



Rysunek 11: Chmura punktów udająca gwiazdy

5.2 Trawa

W celu uatrakcyjnienia głównej planety postanowiono dodać do niej trawę. Próbowano wykorzystać pomysły zawarte w [2], ale niestety nawet to podejście wymagało wykorzystania zbyt wielu obiektów, oraz nie było przystosowane do tak dziwnej sytuacja, jaką jest stawianie elementów na powierzchni kuli. Dodatkowo obiekty musiały wykorzystywać kanał alpha co stanowiło dodatkowe obciążenie dla wydajności. Postanowiono dostosować je do zaistniałego problemu i w celach optymalizacyjnych trawa znajduje się tylko w pobliżu głównej postaci. Dzięki temu liczba obiektów jednoczenie widocznych znacznie spadła. Dodatkowo doszedł pomysł dodania dodatkowej animacji. Wraz z ruchem postaci, trawa nie porusza się wraz z nią, lecz rośnie tam gdzie ona idzie i maleje tam gdzie jej nie ma.



Rysunek 12: Początkowa wersja trawy

5.3 Interaktywność

Istnieje możliwość interakcji z stworzoną sceną. Z menu bocznego możemy wybrać parę opcji (część to testy developerskie), takich jak zmiana kolorów świateł, zmiana aktualnie odtwarzanej animacji. Kamera może być poruszana względem sceny klawiszami ‘wasd‘. Po wybraniu z menu bocznego *tryToLock* i zezwoleniu przeglądarce na zablokowania się kursora możliwe jest obracania kamery przy pomocy myszki.

Rysunek 13: Finalna scena

6 Napotkane problemy

6.1 Eksport modelu

Największym problemem było połączanie dwóch środowisk jakie stanowią Blender i Three.js. Na całe szczęście do dyspozycji użytkowników został napisany specjalny eksporter umożliwiający zapis modelu do pliku JSON. Ma to swoje wady i zalety. Plik wynikowy jest zrozumiały dla człowieka i umożliwia testową edycje parametrów takich i jak zmienne materiałów czy czas trwania animacji. Jednoczenie przez swoją tekstową naturę jest on całkowicie nie skompresowane, czego wynikiem jest dość duży plik wynikowy, nawet dla tak małego modelu i krótkich animacji, jak te wykorzystane projekcie.

Jako że nie istniej praktycznie żadna dokumentacja co robią koleje przełączniki w eksporterze, trzeba było zastosować metode prób i błędów w celu uzyskania wszystkich wymaganych danych z Blendera.

6.2 Poruszanie się postaci

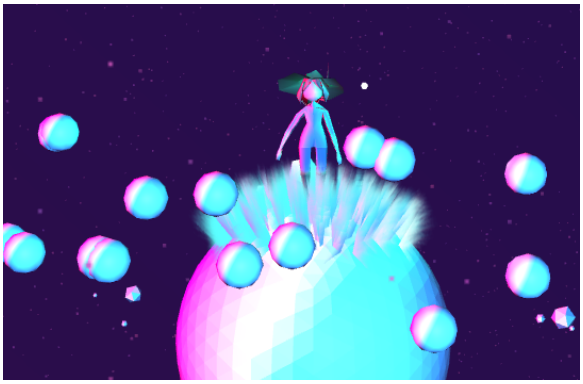
Poruszanie się postaci po kuli sprawiło wiele kłopotów [14]. O ile obliczenie wektora normalnego do powierzchni planety, jest bardzo proste (normalna do powierzchni), to wymuszenie odpowiedniej rotacji na postaci było co najmniej problematyczne. Słaba znajomość przekształceń na quaterinionach doprowadziła do połowicznego rozwiązania. Postać co prawdę stała przy powierzchni, ale podczas poruszania się jej obrót w osi równoległej do normalnej terenu był źle obliczany - postać poruszała się tyłem, bądź bokiem.

Rysunek 14: Problemy przy obrocie postaci.

Rozwiązaniem okazało się wykorzystanie funkcji w frameworku *lookAt* z połączaniem poprawnego ustawiania wektora up dla postaci.Polega to na tym że *lookAt* ustawia rotacje postaci tak aby jej wektor forward wskazywał na dany punkt - zapewniało to obrót w kierunku ruchu. Ustawienie wektora up powodowało poprawne ustawienie dolnej części modelu na powierzchni sfery.

6.3 Wydajność

WebGl jest technologią która jest zawarta w przeglądarce internetowej. Ma to swoje duże zalety - możliwość odpalania na wielu urządzeniach, ale i wady. Jedną z wad która stała się problematyczna podczas tworzenia projektu - wydajność. Duża ilość obiektów (nawet nie animowanych) znacznie zwiększa czas renderowania. Dużym obciążeniem są nawet obiekty nie przeliczające światła. Ze względu na krótki czas przeznaczonego na projekt nie rozwiązano tego problemu w pełni. Jednym ze środków na organicznie liczby renderowanych obiektów było ich grupowanie aby stanowiły jedną całość. Drugim było wyłącznie renderowania obiektów aktualnie nie potrzebnych.



Rysunek 15: Testowanie wydajności.

Literatura

[1] *Dokumentacja frameworku THREE.js*, `http://threejs.org/docs/` (dostęp 3 maja 2016).

[2] *GPU Gems - Chapter 7. Rendering Countless Blades of Waving Grass*, `http://http.developer.nvidia.com/GPUGems/gpugems_ch07.html` (dostęp 3 maja 2016).

[3] *Podstawy Blender*, `https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro` (dostęp 3 maja 2016).