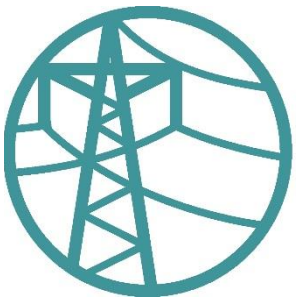

WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Katedra Informatyki i Automatyki

Techniki multimedialne

Post Mortem

Przemysław Szpara

L3

Rzeszów 2018

1. Podsumowanie projektu

Projekt został zakończony sukcesem. W zakładanym czasie powstała działająca aplikacja umożliwiająca symulacje Kinecte'a, generowanie chmury punktów zapisywanych w postaci mapy głębi oraz zapis tej mapy do pliku PGM. Czas przeznaczony na projekt został wykorzystany w 100%. Zaplanowane zadania zostały wykonane naruszając w niewielkim stopniu czas na nie przeznaczony. Dzięki silnikowi Unity możliwe było tak szybkie zakończenie projektu. Posiadające pełną gamę funkcjonalności narzędzie w znacznym stopniu przyczyniło się do szybkiej implementacji aplikacji.

2. Co się udało

Aplikacja została zakończona sukcesem w związku z tym można wymienić wiele elementów, których zostały pomyślnie wykonane. W dalszej części wymienione zostaną najważniejsze z tych elementów.

2.1 Dobór narzędzi

Niewątpliwie dobór narzędzi jest jednym z ważniejszych elementów, które przyczyniły się do sukcesu projektu. Język C# oraz silnik Unity nadały się idealnie do napisania aplikacji, która opiera się na grafice 3D. Język C# jest językiem obiektowym, który jest bardzo intuicyjny. Silnik Unity pozwala na implementację aplikacji właśnie z użyciem tego języka. Dodatkowo Unity posiada wiele funkcjonalności, które zostały wykorzystane przy tworzeniu aplikacji. Dzięki temu nie potrzebne było pisanie wielu linii kodu, a jedynie wywołanie jednej z wielu metod dołączonych do Unity.

Kolejnym dobrym narzędziem okazał się Doxygen tj. program do generowania dokumentacji kodu. Przy niewielkim wkładzie pracy możliwe było wygenerowanie dokumentacji kodu który zapisany został do pliku .html.

2.2 Generowanie chmury punktów

Algorytm generowania chmury punktów zawierał niewiele linii kodu ale działał poprawnie. Algorytm opierał się na liście kroków:

- dla co ustalonego piksela wyślij promień;
- jeżeli promień napotkał obiekt to do tablicy o odpowiednim indeksie wstaw odległość od obiektu do płaszczyzny kamery, a jeżeli promień nie napotkał obiektu na swojej drodze to do tablicy wstaw maksymalną wartość jaka jest w zasięgu promienia;

Większość wykorzystanych funkcjonalności takich jak: Ray, Raycast, RaycastHit należały do silnika Unity i zostały jedynie wykorzystane.

2.3 Generowanie mapy głębi

Mapa głębi powstawała na podstawie chmury punktów zapisanej w tablicy. Dla każdej klatki powstawała nowa mapa głębi. Mapa głębi wykorzystywała kolor biały dla bliskich odległości oraz czarny dla maksymalnej odległości do obiektu. Odległości były normalizowane do przedziału od 0 do 1. Następnie dzięki funkcji Lerp określany był kolor dla konkretnej odległości punktu. Otrzymywana mapa głębi wyświetlana była w prawym dolnym rogu ekranu.

2.4 Zapis do formatu PGM

Format PGM został zaprojektowany tak, aby był łatwy w nauczeniu oraz pisanie programów było bardzo przyjemne. Reprezentuje on obraz w skali szarości.

Każdy obraz PGM składa się z następujących elementów:

- „Magiczny numer” do identyfikacji formatu. Format PGM ma numer „P2”;
- Pusta przestrzeń(tabulacja, znak końca linii, spacja);
- Szerokość podana w kodzie ASCII;
- Pusta przestrzeń;
- Wysokość podana w kodzie ASCII;
- Pusta przestrzeń;
- Maksymalna wartość koloru szarego podana w kodzie ASCII;
- Pusta przestrzeń;
- Tablica wartości koloru szarego dla każdego piksela.

W aplikacji zapis do formatu PGM został napisany kierując się wyżej wymienionymi elementami.

3. Co się nie udało

Podczas projektu nie udało się dokładnie oszacować czasu potrzebnego na realizację konkretnych zadań. Błąd pomiędzy oszacowanym czasem potrzebnym na realizację zadania a rzeczywistym czasem wyniósł nawet 1,5 godziny. Jest to problem, który występuję w dużej ilości projektów.

4. Dalszy rozwój aplikacji

Dzięki użytym narzędziom nie powinno być problemów w dalszym rozwoju aplikacji. Język C# i Unity wydają się przez długi czas być wspierane przez producentów.

Pierwszą rzeczą jaką należałoby uczynić to możliwość zapisu mapy głębi do większej ilości formatów np. png, jpg. Format PGM jest łatwy w zapisie, ale nie jest tak popularny w użyciu jak wymienione formaty.

Kolejną rzeczą jest dodanie możliwości zmiany rozdzielczości kamery. Na obecnym etapie aplikacji, rozdzielczość jest domyślnie ustawiona przez silnik Unity. Podczas działania aplikacji powinno się też dać radę ustawić kąt pomiędzy wystrzeliwanymi promieniami.

Należałoby również poprawić poruszanie się po mapie. Obecny ruch kamery działa ale jest nieintuicyjny. Podczas większej ilości zmian orientacji kamera staje się „krzywa”.

Powinno być również dostępne wczytywania obiektów 3D na mapę podczas działania aplikacji podając koordynaty miejsca w którym miałby się znaleźć.