

Landscape Generator

Inżyniera Oprogramowania

Artur Bednarczyk, Dawid Grajewski, Tomasz Januszek
Politechnika Śląska
Wydział Matematyki Stosowanej
Informatyka, semestr V

28 listopada 2018

Spis treści

1	O projekcie	3
1.1	Zespół	3
1.2	Temat	3
2	Projekt	3
2.1	Plany i pomysły	3
2.2	UI/UX	3
2.2.1	Zawartość	3
2.2.2	Projekty UI	3
3	Teoria	4
3.1	Losowość	4
3.2	Algorytmy	4
3.2.1	Szum Perlina	4
4	Narzędzia	4
4.1	Kontrola wersji	4
4.2	Zarządzanie zespołem	5
4.3	Środowisko	5
5	Aplikacja	5
5.1	Architektura	5
5.2	Struktury danych	5
5.3	Schemat graficzny struktury systemu	6
5.4	Komunikacja między modułami	6
6	API	6
6.1	Perlin	6

1 O projekcie

1.1 Zespół

Artur Bednarczyk, Dawid Grajewski, Tomasz Januszek.

1.2 Temat

Generowanie realistycznych krajobrazów 3D Generowanie w języku wysokiego poziomu (nie w generatorach typu Unity) losowych krajobrazów z uwzględnieniem zadanych parametrów: stromizny terenu, poziomu wody, kolorów na danej wysokości lub obszarzem wizualizacja i symulacja prze-mieszczania kamery.

Bonus: dodanie roślinności (drzewa, krzewy - co najmniej 3 rodzaje) o zadanej częstości i miejscu występowania.

2 Projekt

2.1 Plany i pomysły

2.2 UI/UX

2.2.1 Zawartość

UI generatora pozwoli na ustalenie parametrów zgodnie z którymi zostanie wygenerowany krajobraz. Parametry te to:

- Stromizna terenu
- Poziom wody
- Górzystość
- Gęstość dodatkowych elementów
- Położenie dodatkowych elementów

2.2.2 Projekty UI

includegraphics

3 Teoria

3.1 Losowość

Celem projektu jest wygenerowanie losowego krajobrazu, więc sama losowość jest bardzo ważna. Każdy kolejny krajobraz powinien być inny, a prawdopodobieństwo wystąpienia dwóch podobnych obrazów bardzo niskie. Takie efekty możemy uzyskać dzięki algorytmowi jakim jest Szum Perlina"

3.2 Algorytmy

3.2.1 Szum Perlina

Bazę dla naszego sposobu generowania danych potrzebnych do wyświetlenia zróżnicowanego terenu stanowi szum Perlina. Jest to jeden z typów szumu gradientowego utworzony przez Kena Perlina już w 1983 roku dla potrzeby tworzenia realistycznych grafik komputerowych. Algorytm składa się z trzech kroków

- Pierwszym krokiem jest zdefiniowanie wielowymiarowej siatki jednostkowych wektorów rozpatrywanego gradientu. W naszym przypadku są to wartości losowe z zakresu $(0, 1)$ liczb rzeczywistych. Dla jednowymiarowego przypadku byłyby dostępne jedynie wartości -1 albo 1
- Kolejno iteruje się po podawanych punktach. Punkt wpada do pewnej komórki wygenerowanej siatki. Następnie wyliczany jest iloczyn skalarny między punktem a wektorem każdego z rogów komórki (a więc ich odległość), po czym zapisane zostają w pamięci.
- Przeprowadzona zostaje interpolacja dla każdej pary punktów z uwzględnieniem funkcji wygładzającej.

Wynikowo otrzymujemy wielowymiarową macierz (lub tensor) zawierający wartości zamknięte w pewnych granicach. Jest możliwe nakładanie na siebie wielu takich macierzy generowanych dla różnych częstotliwości siatki w celu uzyskania różnych ułożeń lub skupień wartości.

4 Narzędzia

4.1 Kontrola wersji

Do zarządzania kodem i wersjami projektu wykorzystujemy narzędzie Git. Korzystamy z platformy GitHub jako repozytorium dostępnego online. Do-

bór narzędzi służących do korzystania z repozytorium to sprawa indywidualna każdego członka zespołu, ponieważ nie ma ona wpływu na sam projekt.

4.2 Zarządzanie zespołem

Trello - Kanban Board - to tutaj rozpisujemy zadania i przydzielamy je sobie, określamy również terminy i planujemy.

4.3 Środowisko

Visual Studio 2015

Do wyświetlenia terenu wykorzystujemy zestaw funkcji API wspomagający generowanie grafiki – DirectX wraz z DirectX Software Development Kit używany z C#.

5 Aplikacja

5.1 Architektura

Utworzony silnik graficzny jest wzorowany na istniejących już popularnych silnikach do gier takich jak Unity, Cry czy Unreal Engine. Opiera się on na trzech głównych składowych:

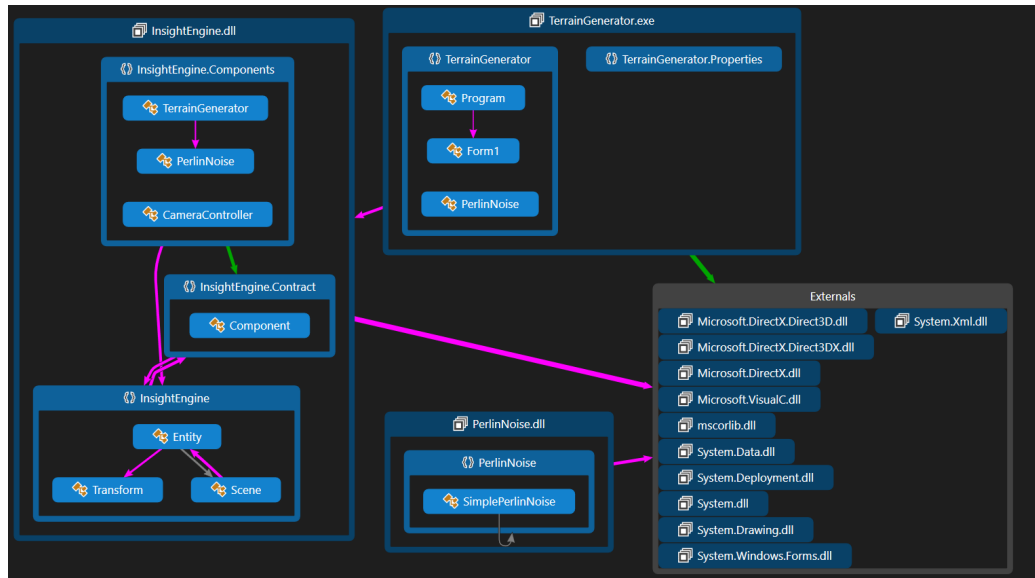
- scena - obiekt nadrzędny, to w nim rozgrywa się akcja.
- encja - umieszczona w scenie. Jest to pojedynczy agent w scenie, np. kamera, teren czy postać.
- komponent - to składowa encji - odpowiada za część logiki encji (przykładowo, encja "player" może mieć komponenty takie jak "moveComponent" czy "shootComponent")

Silnik jest zbudowany zgodnie z wzorcem kompozytu - encja składa się z komponentów, ale pojedynczy komponent może zawierać kolejne, zagnieżdżone komponenty. Pozwala to na rekurencyjne wykonywanie metod w obiektach o tym samym interfejsie.

5.2 Struktury danych

Dane o wygenerowanym terenie będą przechowywane w pliku .json. Umożliwi nam to zapisanie i późniejsze odtworzenie terenu.

5.3 Schemat graficzny struktury systemu



5.4 Komunikacja między modułami

IN PROGRESS

6 API

6.1 Perlin

Tworzymy obiekt **PerlinNoise**, który wymaga podania rozmiaru i wymiarów. Pobieranie wartości za pomocą metody: `CalculatePerlin(x,y);`