

Koszęcin, dnia 25.03.2020

Laboratorium ZTPGK

Prosta implementacja rendera terenu

Karol Kozuch
AEI Informatyka, IGT SII
Semestr: 1
E-mail: karokoz247@student.polsl.pl

1. Cel laboratorium

Celem niniejszego laboratorium jest zapoznanie się z metodami generacji i modyfikacji terenu trójwymiarowego w środowisku Unity.

2. Przebieg wykonania punktu 2.

Pierwszym krokiem wykonywania zadania było utworzenie prostego terenu używając narzędzi dostępnych w edytorze Unity. Zgodnie z instrukcją utworzono obiekt terenu. Następnie zainportowano paczkę z Asset Store, zawierającą podstawowy zestaw elementów, między innymi prostą wodę, tekstury terenu, flary i skybox.

Następnie, używając tekstur i prefabów z zainportowanej paczki, ustawiono zestaw tekstur dla stworzonego terenu. Utworzono na płacie terenu wyspę używając pędzla wysokości a na koniec nałożono tekstury pędzlem tekstur dla całego skrawka terenu. Na koniec dodano prostą wodę i przeskalowano wszystkie tekstury z rozmiaru 2x2 do 128x128.

3. Przebieg wykonania punktu 4.

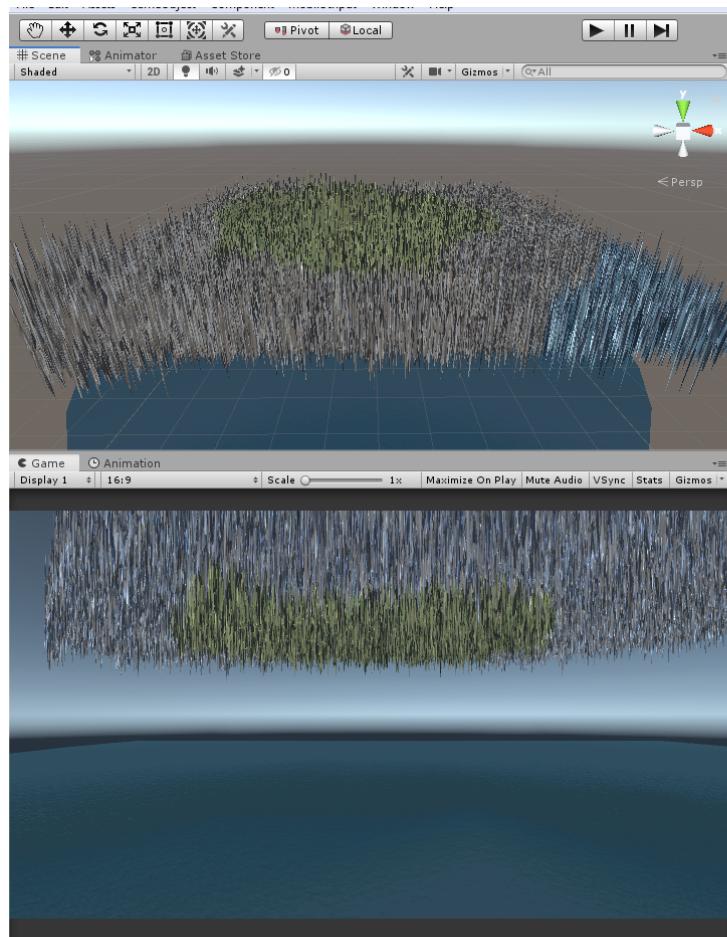
3.1. Metoda generacji terenu

Utworzono skrypty, do których wpisano kod zawarty w wymienionym w podtytule punkcie. Skrypty te trafiły jako komponenty do utworzonego w punkcie 2. terenu. Następnie uruchomiono symulację by zobaczyć efekty – został wygenerowany teren złożony ze wzniesień.

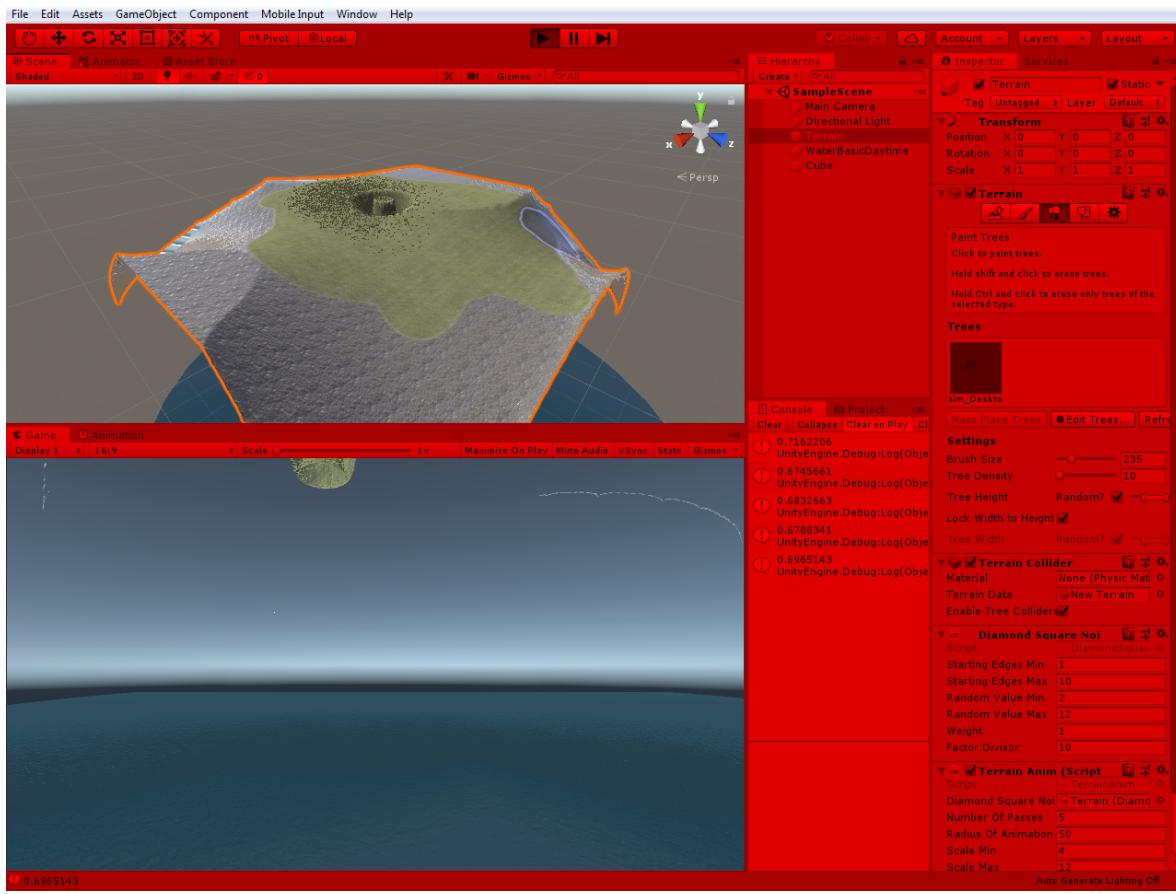
Do kodu rozpoczęto wprowadzać modyfikacje – pierwsza z nich polegała na próbie implementacji warstw w próbkowaniu szumu Perlina. Postanowiono, że dla każdego wierzchołka sekwencja losowania danych powtarzana będzie trzykrotnie, a dla każdej sekwencji wylosowany będzie dodatkowo inny współczynnik (rzeczywisty) z zasięgu [4.0f; 12.0f]. Wynik końcowy dla danego wierzchołka był dodatkowo dzielony przez 3. Dało to dość niespodziewane wyniki, widoczne na rysunku 1. Kilka kolejnych prób zakończyło się wynikami podobnymi do tego przedstawionego na rysunku 1. Z tego względu zdecydowano się na zaimplementowanie drugiego algorytmu, który działałby na tym samym skrawku terenu, a wpływ poszczególnych algorytmów definiowałby parametr wagi danego algorytmu. Po przeprowadzonych poszukiwaniach wybrano algorytm Diamond-Square (ang. diament-kwadrat). Algorytm przebiega następująco (założenie - krawędzie mapy zorientowane równolegle do odpowiadających im krawędzi ekranu):

1. Wylosuj wartości dla narożników fragmentu terenu.
2. Przejdz do fazy diamentu – wyznacz wartości w punktach przecięć prostych wychodzących z wyznaczonych już punktów, pod kątem 45°. Wartość wyznacz poprzez średnią wartości wysokości 4 najbliższych punktów i dodaj wartość losową.
3. Przejdz do fazy kwadratu – wyznacz wartości w punktach przecięć prostych równoległych do krawędzi ekranu, a wychodzących z wyznaczonych już punktów. Wartość wyznacz poprzez średnią wartości wysokości 4 najbliższych punktów (dla krawędzi mapy zapętl odczyt wartości) i dodaj wartość losową.

Kroki 2 i 3 są powtarzane $\lfloor \log_2 n \rfloor$ razy, gdzie n to ilość wierzchołków. Dla uproszczenia założono, że mapa jest w kształcie kwadratu o boku długości $2^n + 1$ wierzchołków. Z każdym kolejnym krokiem pętli wartość losowa jest zmniejszana o jeden rzad. Pierwsza próba generacji terenu została zaprezentowana na rysunku 2.



Rys. 1 – wielokrotne próbkowanie szumu Perlina z przemnożeniem przez losowe współczynniki dla każdej osobnej warstwy.

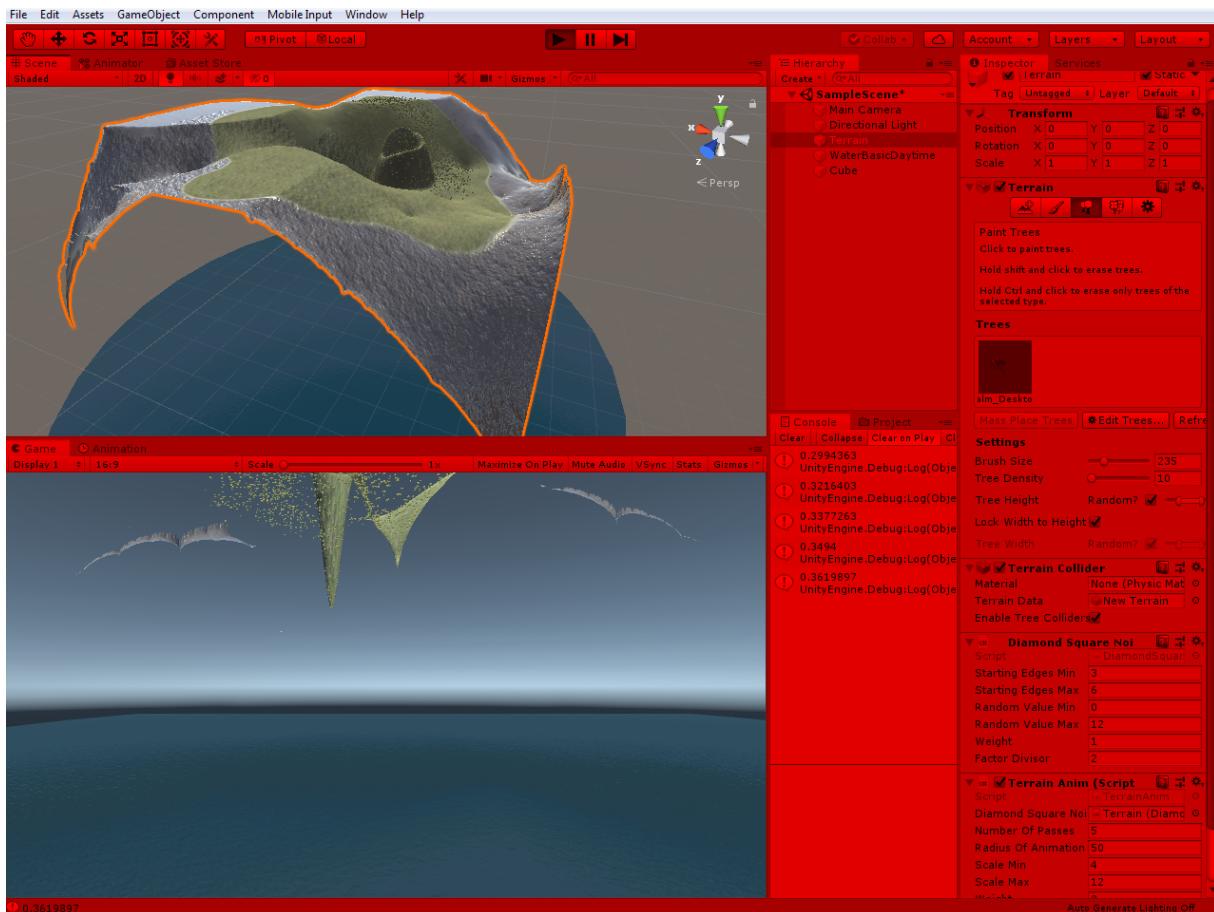


Rys. 2 – Pierwsza próba użycia algorytmu Diament-Kwadrat.

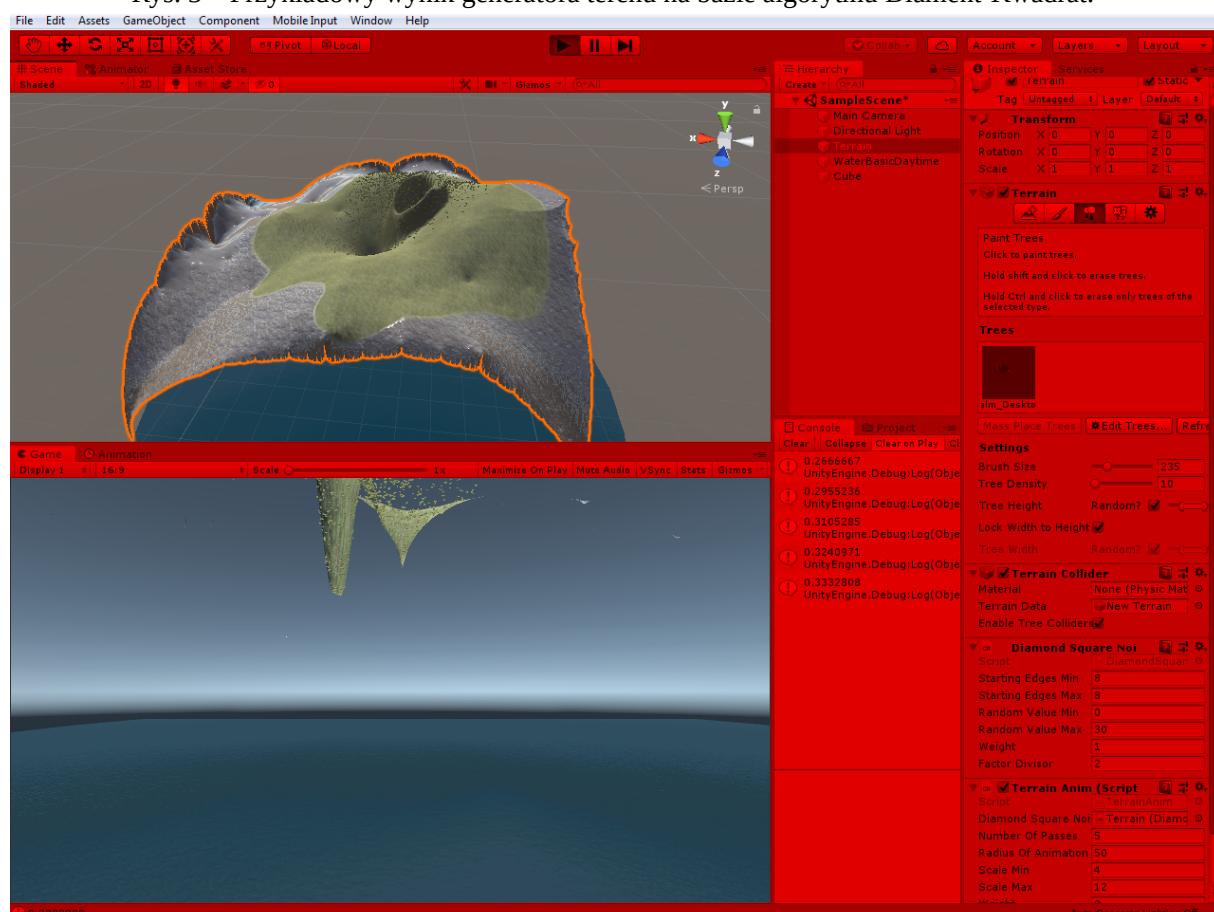
Konfiguracja działania algorytmu bazuje na 6 wartościach:

- minimalna i maksymalna możliwa wartość 4 narożników mapy,
- minimalna i maksymalna możliwa wartość losowa dodawana do obliczanych punktów,
- podstawa rzędu o który zmniejszana jest wartość losowa
- waga generatora

Więcej przykładów działania generatora znajduje się na rysunkach 3 oraz 4. Warto nadmienić, że dla rysunków 2-4 minimalne wartości są większe lub równe 0. Ustawienie takie ma tendencję do wymuszenia wygenerowania płaskowyzów bądź płaskich szczytów wznieśień.



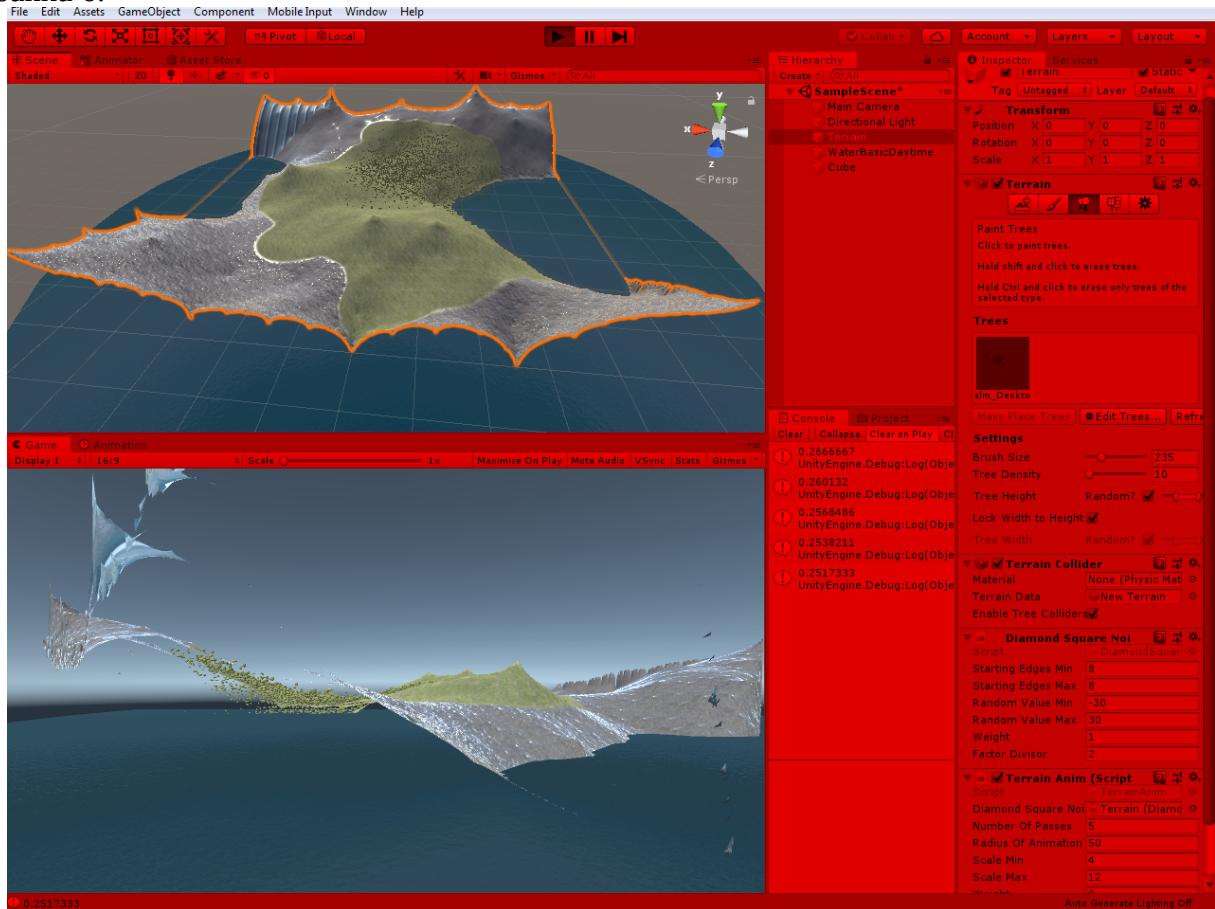
Rys. 3 – Przykładowy wynik generatora terenu na bazie algorytmu Diamant-Kwadrat.



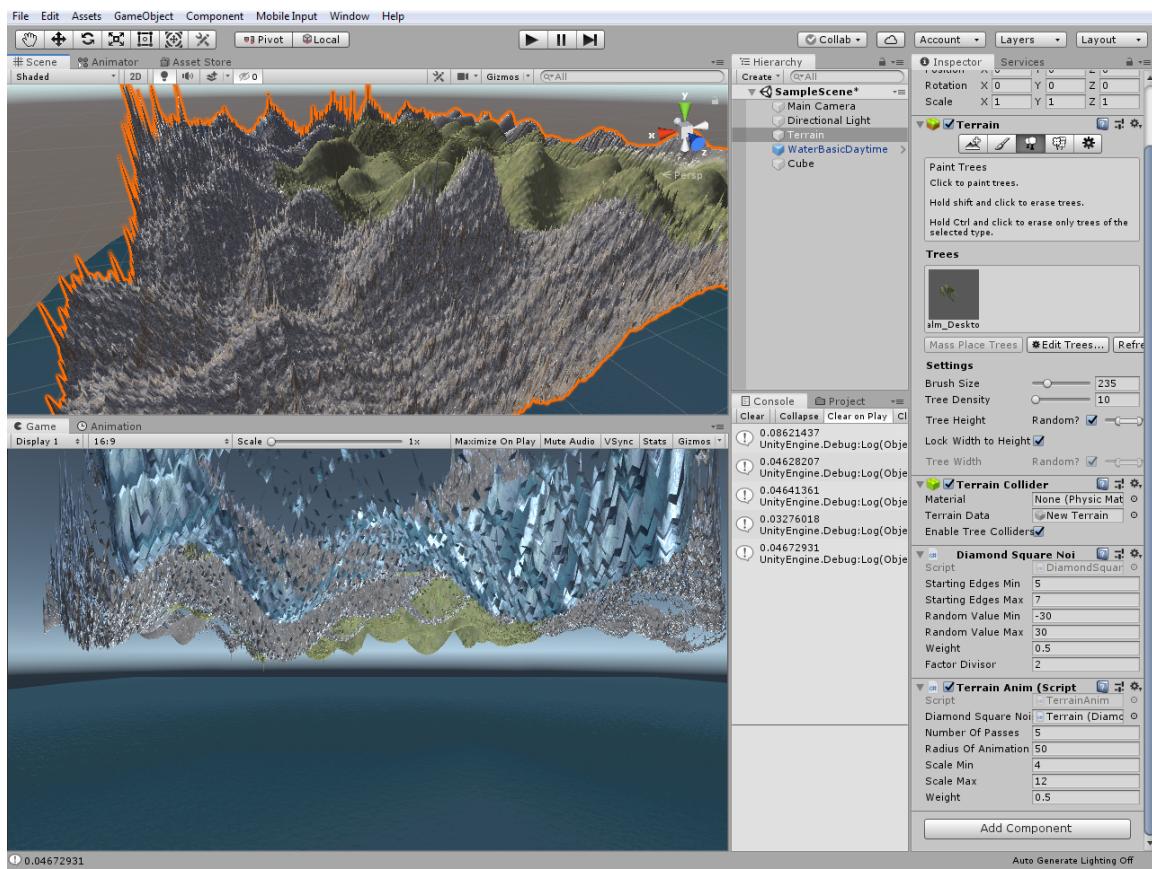
Rys. 4 – Inny przykład działania generatora Diamant-Kwadrat.

Szczególnie na rysunku 4 widać charakterystyczne wgłębienia w gruncie i poszarpaną krawędź mapy. W przypadku rysunku 2 parametr podstawy rzędu został ustawiony na wartość 10, co zwykle dawać dość łagodne zmiany terenu. Dla rysunków 3 i 4 wartość tę ustawiono na 2, co, jak widać, poskutkowało bardziej drastycznymi zmianami w ukształtowaniu widocznego fragmentu terenu. Na rysunku 5 znajduje się wynik generatora dla znacznie większego zasięgu wartości, w tym ujemnych (poprzednie przypadki oscylowały w granicach [0; 14]). W przypadku terenu widocznego na rysunku 5 zakres możliwych wartości losowych do wygenerowania wynosił [-30; 30]. Spowodowało to wygenerowanie terenu dużo bardziej zróżnicowanego niż w przypadku poprzednich ustawień, generujących głównie szczyty i płaskowyże.

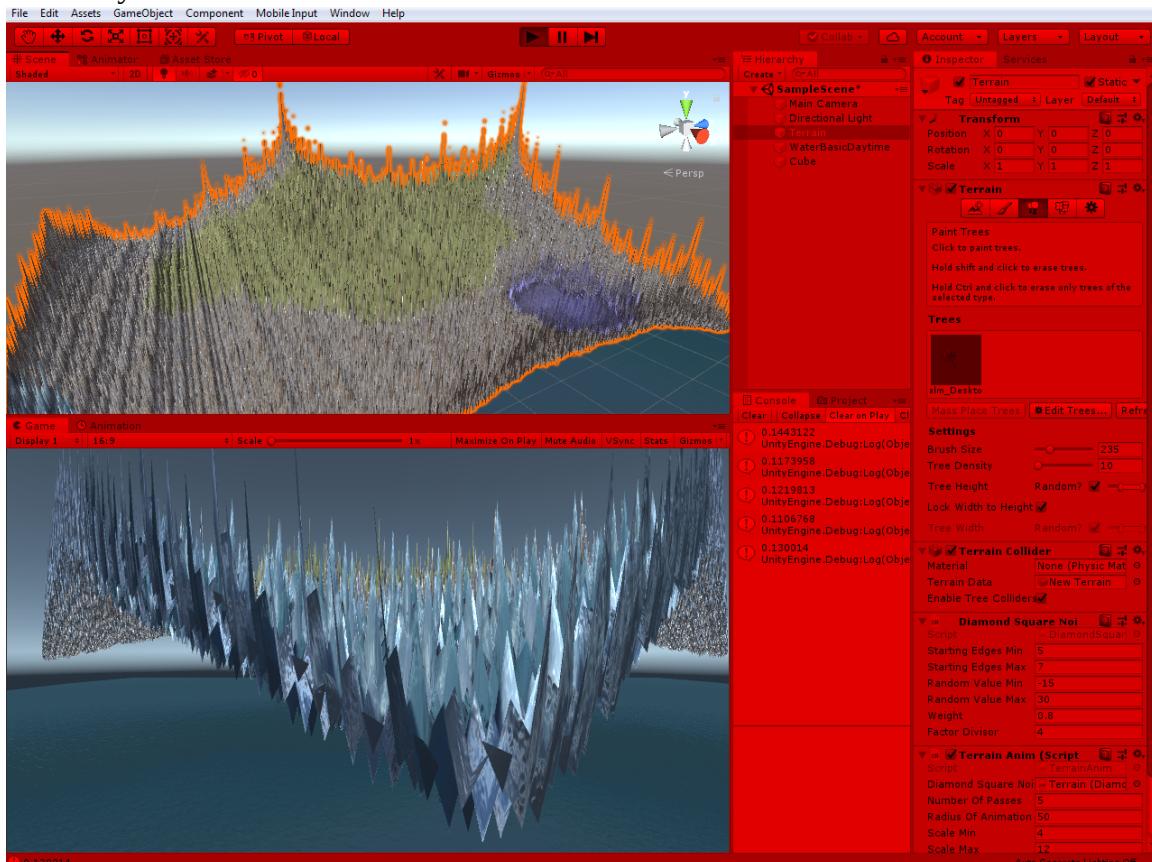
Na koniec podpięto ponownie generator bazujący na szumie Perlina oraz ustawiono wagę obydwoch generatorów na wartość 0.5. Uzyskany rezultat był dość nieoczekiwany i zaprezentowano go na rysunku 6:



Rys. 5 – Teren wygenerowany generatorem Diament-Kwadrat dla wartości losowych z przedziału [-30; 30].

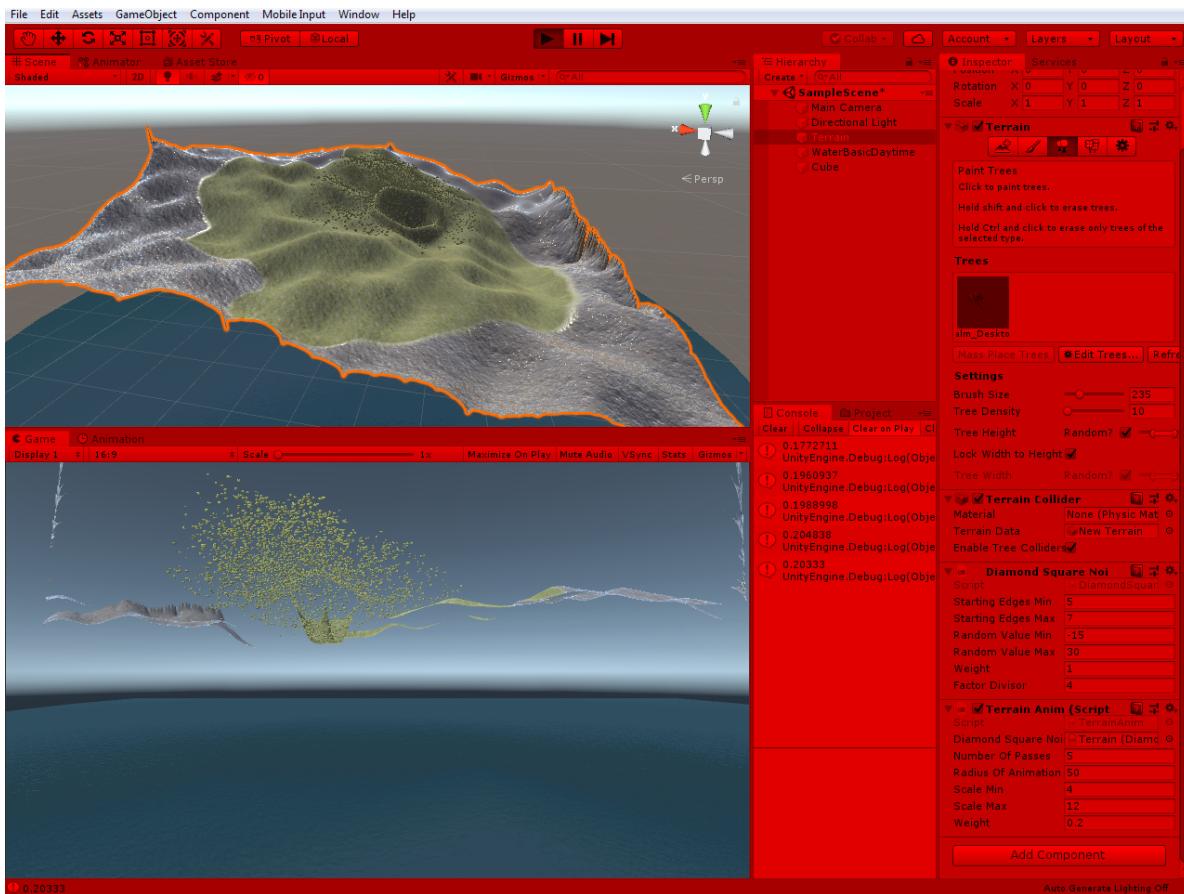


Rys. 6 – wynik generacji terenów przy użyciu szumu Perlina i Diamentu-Kwadratu przy wagach 0.5.
Dla próby, zmieniono wagę na 0.8 dla Diamentu-Kwadratu oraz 0.2 dla szumu Perlina. Wynik uzyskano na rysunku 7:



Rys. 7 – wynik generacji terenów przy użyciu szumu Perlina i Diamentu-Kwadrat przy wagach odpowiednio 0.2 i 0.8.

Po kilku testach okazało się, że algorytm Diamentu-Kwadratu działa dobrze tylko dla wagi 1. Dla wagi 0 generator ten jest wyłączony. Można więc uznać Diament-Kwadrat za generator głównego terenu, a szum Perlina wykorzystać do wygenerowania szczegółów w postaci pagórków. Wynik takiego postępowania zaprezentowano na rysunku 8:



Rys. 8 – zastosowanie łącznego szumu Perlina i Diamentu-Kwadratu z wagami odpowiednio 0.2 i 1.

3.2. Mapy wysokości

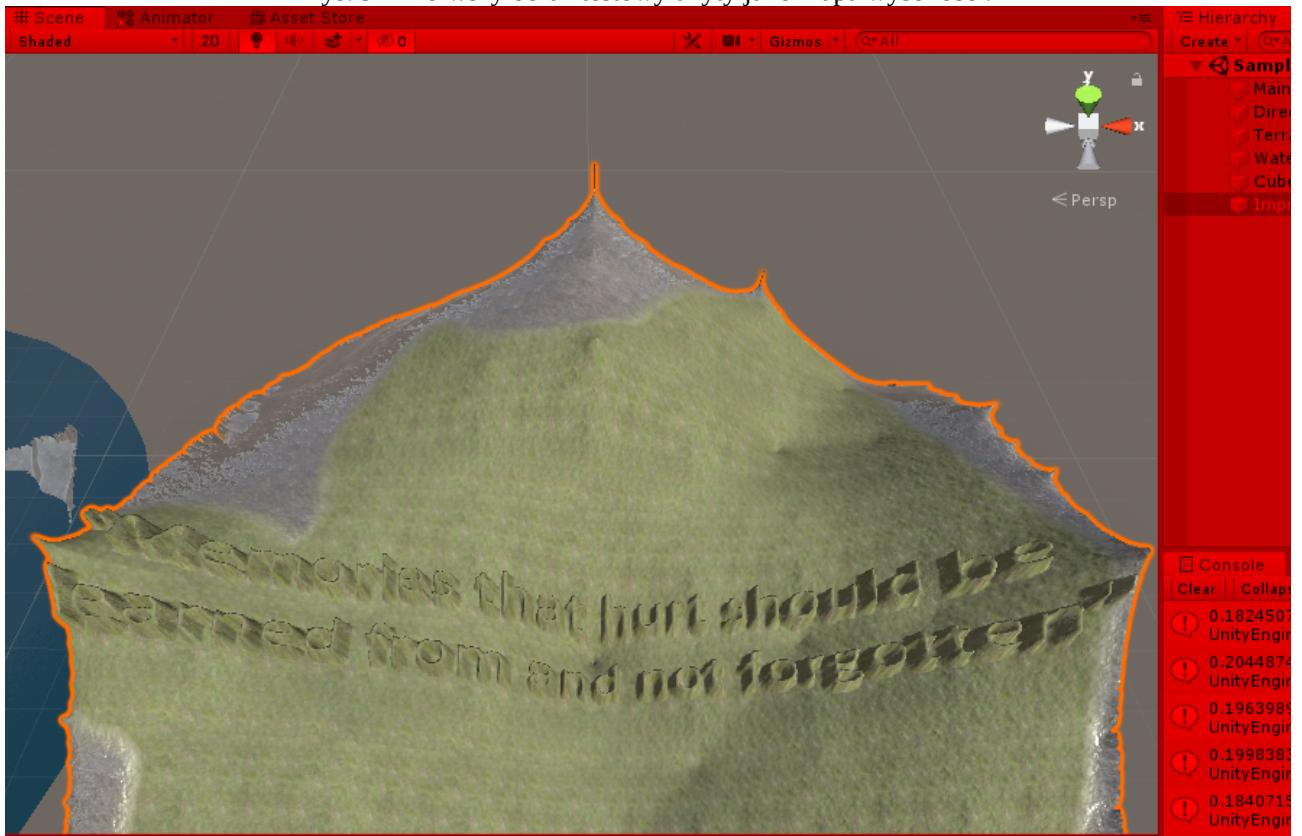
W kolejnym kroku napisano kod ładujący mapy wysokości. Mapą wysokości może być w tym przypadku dowolny obraz, nie ważne czy czarno-biały, czy kolorowy. Jedynymi warunkami dla obrazu są:

- Obraz winien znajdować się w folderze *Assets/Resources/Maps*
- Obraz powinien mieć rozszerzenie, które potrafi przetworzyć Unity – na przykład .tiff, .png, .jpg czy .bmp.

W załadowanym obrazie, każdy piksel odpowiada jednemu wierzchołkowi z terenu, na który oddziałyuje skrypt. Piksel konwertowany jest na pojedynczą wartość zmiennoprzecinkową pojedynczej precyzji poprzez uśrednienie wartości kolorów i przemnożenie wyniku przez kanał alfa (wszystkie wartości są znormalizowane, z przedziału [0; 1]). Przy załadowaniu obrazu znane są jego rozmiary – użyte są, by wyśrodkować obraz na mapie, natomiast jeżeli obraz jest za duży – jest on obcinany tak, by jego rozmiary pokryły się z rozmiarami mapy. Tak przygotowana mapa wysokości jest dodawana do odpowiednich punktów (a dokładniej ich wysokości), generując wybruszenie (lub wgłębianie) w terenie. Na rysunku 9 zaprezentowano obraz źródłowy, a na rysunku 10: wynik wprowadzonego rozwiązania.

"Memories that hurt should be learned from and not forgotten"

Rys. 9 – Pierwszy obraz testowy użyty jako mapa wysokości.

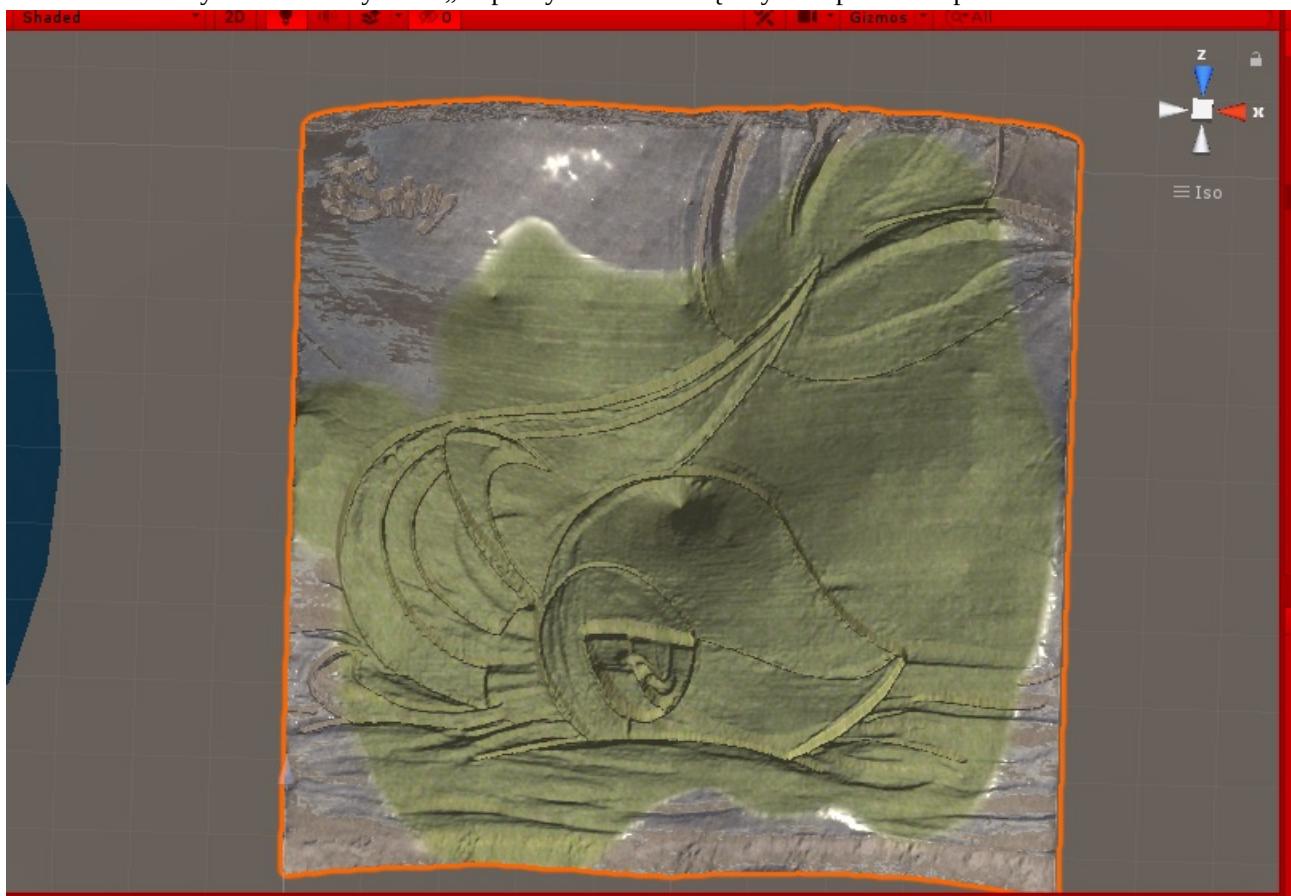


Rys. 10 – wynik przetwarzania rysunku 9 jako mapy wysokości.

Napisany skrypt był w stanie, po naprawieniu kilku błędów a między innymi nieprawidłowym odczytem wartości kolorów pikseli, przetwarzać proste, czarno-białe obrazy. Skrypt jednakże napisano w taki sposób, by był w stanie przyjąć jakikolwiek obraz. Postanowiono użyć obrazu o większym stopniu skomplikowania, z dużo szerszą gamą kolorów. Obraz ten zaprezentowano na rysunku 11, natomiast wynik działania skryptu – na rysunku 12 i 13. Rysunek 13 prezentuje obraz w perspektywie dla ukazania realnego widoku, natomiast 12 – izometryczny, dla łatwiejszego porównania z danymi ze źródła.



Rysunek 11 – wybrana „mapa wysokości” o większym stopniu skomplikowania.



Rys. 12 – izometryczny widok na fragment terenu zmodyfikowanego mapą wysokości wygenerowaną z rysunku 11.



Rys. 13 – widok z perspektywy na fragment terenu zmodyfikowanego mapą wysokości wygenerowaną z rysunku 11.

3.3. Animacja terenu

W celu stworzenia animacji terenu wzorowano się na fragmencie kodu dostarczonym wraz z instrukcją. W środowisku użytkownika definiuje, jaki fragment terenu będzie animowany (prostokąt o rozmiarze w wierzchołkach). Dwoma dodatkowymi parametrami jest amplituda oraz długość okresu. Skrypt, bazując na czasie symulacji oraz pozycji danego wierzchołka w terenie względem osi x, generuje animację przemieszczającej się fali. Pozycja animacji wyliczana jest tak, by zawsze była na środku fragmentu terenu. Jeżeli dobrany rozmiar jest za duży – ustawiany jest maksymalny rozmiar mapy. Działanie przedstawiono na rysunku 14:



Rys. 14 – fragment zaanimowanego terenu przy pomocy funkcji sinus o rozmiarze 20x20.

4. Źródła:

- [1] Rysunek 11 - „-Snivy-chan-”, użytkownik Umishaiii na stronie DeviantArt.com:
<https://www.deviantart.com/umishaiii/art/Snivy-chan-200017990>