Generowanie terenu

opis plików:

„StartMenu.cs” – skrypt startowy programu, rozpoczyna pracę od wyświetlenia menu wyboru mapy.

„Form1.cs” – skrypt zawiera klasę Form1 tworzącą menu wyboru mapy. Klasa zawiera wszelkie potrzebne funkcje do odczytu pożądanych danych z plików graficznych zawierających mapy wysokościowe terenu.

„TerrainGenerator.cs” – skrypt zawiera klasę TerrainGenerator odpowiedzialną za generację trójwymiarowego terenu w Unity na podstawie danych odczytanych z mapy wysokościowej terenu.

„PerlinNoise.cs” – skrypt zawiera zbiór algorytmów szumu wykorzystanych przy losowym wstawianiu drzew i innych detali na utworzonym terenie.

Najważniejsze klasy i funkcje programu:

class StartMenu

Klasa jest przypisana do pustego obiektu *„Start Object”* na startowej scenie projektu. Program w fazie uruchomieniowej wywołuję metodę *„Start()”* tej klasy, w której jest tworzony nowy obiekt klasy Form1 implementujący menu wyboru mapy.

Klasa dodatkowo zawiera zmienną całkowitą *„m_mapSize”* i tablicę *„m_grayLevels”*, do których po wyborze mapy wysokościowej mają zostać zapisane rozmiar mapy i wysokości poszczególnych pixeli w postaci poziomu szarości w przedziale <0.0 ; 1.0>.

W metodzie *„Start()”* klasy dodatkowo jest wywoływana funkcja *„Object.DontDestroyOnLoad(this)”*, aby obiekt, do którego jest przypisany nie został usunięty, aby nie utracić odczytanych danych z mapy.

class Form1

Klasa jest odpowiedzialna za tworzenie menu wyboru mapy. Zawiera wszelkie potrzebne funkcje do odczytu pożądanych danych z plików graficznych zawierających mapy wysokościowe terenu.

Klasa umożliwia odczyt danych z map wysokościowych w formatach grafiki rastrowej:

1) *.jpg, *.jpeg, *.bmp, *.png, *.gif, *.tiff
2) oraz surowym *.raw

Mapy mogą być zarówno czarno-białe, jak i kolorowe, gdyż odczytane dane o kolorze piksela w modelu RGB przeliczamy na informację o jasności koloru (poziom szarości). Należy jednak wziąć pod uwagę, że nie będzie w takim przypadku rozróżniane przypisanie na mapie wysokości do różnych kolorów (zielony, czerwony, niebieski), a jedynie do jasności koloru. Wymogiem natomiast jest, aby wysokość i szerokość mapy były jednakowe i rozmiar ten był potęgą dwójki. Ograniczenia te zostały

dodane, gdyż Unity takie ograniczenia narzuca, aby poprawnie zostały ustawione wysokości terenu na podstawie przekazanych danych.

Wybór mapy i odczyt danych następuje po wywołaniu metody *„int readHeightmap(string fileName)”*.

Do odczytania plików rastrowych wykorzystano klasy *„Bitmap”* i *„Pixel”* oraz ich metody udostępnione w bibliotekach języka C#. Do odczytu jasności pixela wykorzystano metodę *„GetBrightness()”*, która zwraca już gotową informację o jasności pixela.

Do plików w formacie surowym *.raw trzeba było podejść w inny sposób. Format ten przechowuje w sposób bezpośredni bajt, po bajcie (a dokładniej 1,5 bajta po 1,5 bajcie) kolory pikseli, jednak nie posiada żadnego nagłówka z dodatkowymi informacjami i klasa Bitmap nie oferuje możliwości pobrania danych z obrazka w takim formacie. Dane z pliku *.raw program przepisuje bezpośrednio do bufora bajtowego, a następnie jasność piksela jest przeliczana z jego składowych RGB przy pomocy wzoru zaczerpniętego z Internetu (na końcu dodatkowo dzielenie przez 255, aby uzyskać jasność w skali <0.0 ; 1.0>):

```
m_grayLevels[x, y] = (float)((buffer[bufferIdx] * 0.3) + (buffer[bufferIdx + 1] * 0.59) + (buffer[bufferIdx + 2] * 0.11)) / 255;
```

W obu przypadkach w rezultacie zwracane są takie same (a przynajmniej prawie takie same) dane dla dwóch takich samych obrazów zapisanych w innym formacie.

Odczytane dane można również zapisać do pliku tekstowego i z takiego pliku je później odczytać.

class TerrainGenerator

Klasa jest odpowiedzialna za generację trójwymiarowego terenu w Unity na podstawie danych odczytanych wcześniej z mapy wysokościowej terenu.

Sporo miejsca by zajęło opisanie wszystkich funkcjonalności użytych do doprecyzowania generowanego terenu, ale warto nadmienić co najważniejsze. W Unity generacja terenu z poziomu skryptu składa się z dwóch części, skonfigurowanie pożądanego terenu z wykorzystaniem odpowiednich metod klasy *„TerrainData”* oraz utworzenie terenu przy pomocy klasy *„Terrain”* na podstawie skonfigurowanych danych w klasie *„TerrainData”*.

Aby ustawić pożądane wysokości terenu, potrzebne dane odczytane z mapy wysokościowej terenu są pobierane z obiektu *„Start Object”*, w którym wcześniej je zapisano i przekazywane do metody *„SetHeights(...)”* klasy *„TerrainData”*.

Moduł gry

Airplane

class AirplaneMovement

Rotate() – obracanie samolotu pod wpływem wciśniętych klawiszy strzałek.
SetSpeed() – zwiększanie prędkości lotu samolotu (lewy przycisk ctrl) oraz zmniejszanie (lewy alt).
Move() – poruszanie samolotem. Zrobione za pomocą przesunięcia (translacji) o zadany wektor).

class AirplaneController

InitializePosition(Terrain) – inicjalizacja początkowej pozycji samolotu. Ustawienie na środku mapy (powyżej jej wysokości w tym punkcie – wysokość jest zwiększana o wartość zmiennej *StartHeightDistanceBetweenTerrainAdnAirplane).*

Camera

class CameraSwitcher

Update() – zmiana kamery w przypadku wykrycia wciśnięcia jednego z przycisków (F1 – F4).
DisableAllCameras () – wyłączenie wszystkich kamer. Funkcja pomocnicza wywoływana w celu wyłączenia kamer przed załączeniem nowej (zmiany kamery na inną).

Texts

class GameOverTextController

ShowGameOverText () – wyświetlenie informacji o końcu gry.

class TimeTextController

UpdateTimeText(string) – wyświetlenie czasu gry podanego w parametrze.

class ScoreTextController

AddScore(int) – dodanie do wyniku gracza punktów (przekazanych jako parametr) i zaktualizowanie tekstu.
UpdateScoreText () – zaktualizowanie tekstu z wynikiem.

GameController

class GameController

Update () – wykrywanie czy podczas zatrzymania gry użytkownik wcisnął klawisz restart (‘R’). Wywołanie aktualizacji wyświetlanego czasu.
EndGame() – zatrzymanie gry.
UpdateDisplayedTime() – zaktualizowanie informacji o czasie jaki upłynął od momentu rozpoczęcia gry.

Gates

class ActiveGateCollisionDetector

OnCollisionEnter (Collision) – wykrywanie czy samolot przeleciał przez bramkę. Skrypt dołączany do aktywnej bramki (dołączony do prefaba). W przypadku poprawnie rozpoznanej kolizji użytkownik jest nagradzany punktem oraz jest wyświetlana kolejna aktywna bramka (w przypadku braku bramek do wyświetlenia następuje koniec gry) – sytuacja obsługiwana przez skrypt GatesController.

class GatesController

Klasa odpowiedzialna za całą obsługę działania bramek. Podczas inicjalizacji wywołuje odpowiednią metodę z klasy Factory.

GenerateGatesRandomly (Terrain) – wygenerowanie bramek w losowych lokalizacjach na mapie. W rzeczywistości następuje wywołanie skryptu *GatesPositionsFactory*.
GenerateGatesWithPartiallyRandomPosition (Terrain, Vector3) – to samo co powyższa metoda z tą różnicą, że wywołuje inną funkcję z klasy *GatesPositionsFactory* (odpowiedzialną za generowanie bramek w pozycjach powiązanych ze sobą). Jako drugi parametr jest przekazana pozycja samolotu.
ShowNextActiveGateOrEndGame () – funkcja, która niszczy aktywną bramkę, przesuwa indeks (używany do poruszania po listach ze wszystkimi zainicjalizowanymi bramkami i ich pozycjami) oraz niszczy kolejną nieaktywną bramkę tworząc w jej miejsce bramkę aktywną. W przypadku gdy bramki się wyczerpały (użytkownik przeleciał przez wszystkie) następuje koniec gry – wywołanie metody *EndGameWhenUserWon*.
EndGameWhenUserWon () – zakończenie gry w przypadku wygranej użytkownika. W rzeczywistości wywołanie odpowiedniej metody ze skryptu *GameController*.
GetPositionOfCurrentActiveGate () – metoda zwracająca pozycję aktywnej bramki. Używana przez skrypt *ArrowController*.

class GatesPositionsFactory

Klasa odpowiedzialna za wygenerowanie pozycji (i ich zwrócenie) dla wszystkich bramek. Może generować pozycje w pełni losowo jak i tylko częściowo.

InitializeGatePositionRange(Terrain) – prywatna funkcja używana do zainicjalizowania zmiennych danymi terenu. Uzupełnia zmienne odpowiedzialne za przechowywanie informacji o zakresie pozycji

dostępnym dla bramek (bierze pod uwagę rozmiar terenu, jego pozycję oraz zdefiniowaną odległość od granic terenu).
GetGateRandomPosition(Terrain) – prywatna funkcja używana do pobrania pojedynczej pozycji (w pełni losowej) dla przekazanego terenu. Wartości współrzędnych X i Z są losowane za pomocą metody .Net (System.Random.NextDouble()) w granicach przechowywanych w zmiennych zainicjalizowanych funkcją *InitializeGatePositionRange*. Wysokość bramki nad terenem (współrzędna Y) jest wyliczana funkcją dostępną w Unity - terrain.SampleHeight. Dodawana jest losowa liczba ze zdefiniowanego przedziału (aby bramka była nad poziomem terenu).

GetGatePositionAccordingToThePassedPosition(Terrain, Vector3) – funkcja podobna do opisanej powyżej z tą różnicą, że nowo wygenerowana pozycja jest powiązana z poprzednio wygenerowaną i przekazaną jako drugi parametr (Vector3). Generowanie współrzędnej X odbywa się za pomocą losowania liczby z dostępnego zakresu, która jest dodawana do współrzędnej X poprzedniej współrzędnej. Gdy nowa współrzędna wykracza poza zakres (ten sam co w poprzedniej funkcji), to jest nadpisywana liczbą wygenerowaną w ten sam sposób co w poprzedniej metodzie. Współrzędna Z jest liczona w trochę inny sposób – pobierana jest losowa liczba (z ustalonego zakresu dla tej operacji), która jest dodawana do współrzędnej parametru. Losowa liczba jest nieujemna, w związku z czym przesunięcie następuje ciągle w przód. W momencie gdy współrzędna wykroczy poza zakres mapy, mnożnik *_moveForward* jest mnożony przez -1. Dzięki temu kolejne wylosowane liczby będą ujemne a bramki będą „się przesuwały w tył" (aż do momentu ponownego wyjścia poza zakres). W przypadku wyjścia poza zakres losowanie jest powtarzane.

GetGateRandomPositions (Terrain) – funkcja, która wywołuje dwie metody z tej samej klasy wspomniane wcześniej: *InitializeGatePositionRange* oraz *GetGateRandomPosition* (w pętli dla wszystkich bramek). Jest interfejsem do losowania pozycji w pełni losowych.

GetGateRandomPositions (Terrain, Vector3) – wywołuje dwie metody z tej samej klasy wspomniane wcześniej: *InitializeGatePositionRange* oraz *GetGatePositionAccordingToThePassedPosition* (w pętli dla wszystkich bramek). Jest interfejsem do losowania pozycji zależnych od siebie. Drugim argument jest pozycja samolotu.

class GatesFactory

Klasa generująca bramki dla przekazanych pozycji.

InstantiateGatesForLocations (Vector3[], GameObject, GameObject) – metoda tworząca bramki. Pierwszy argument to tablica pozycji bramek, drugi to obiekt aktywnej bramki, trzeci – nieaktywnej. Dla każdego elementu tablicy (dla każdej pozycji) tworzy instancję bramki. Najpierw jednak losuje (w pełni losowo) kwaternion (funkcja *GetRandomQuaternion)*, który odpowiada za rotację bramki. Ostatnim etapem jest sprawdzenie, czy kąt przechylenia bramki (eulerAngles.X) nie jest uznawany za poziomy (funkcja *GateIsRotatedHorizontally*). Jeżeli bramka jest w poziome (lub bliska tego) to następuje jej przesunięcie w górę – aby użytkownik miał możliwość przelotu przez nią od dołu.

GateIsRotatedHorizontally (Vector3) – funkcja sprawdzająca czy bramka (jej kąty rotacji przekazane w parametrze) jest obrócona w poziome (lub bliska tego).

GetRandomQuaternion () – metoda losująca całkowicie losowy kwaternion.

Inne

class ArrowController

Update () – aktualizowanie rotacji strzałki względem aktualnie aktywnej bramki. Strzałka wskazuje ciągle aktywną bramkę, czyli kierunek w którym użytkownik powinien podążać.

class Constants

Klasa zawierająca w postaci zmiennych nazwy obiektów wykorzystywanych w symulatorze.

class CollisionDetector

OnCollisionEnter(Collision) – obsługa kolizji z samolotem. Skrypt dołączany do terenu. W przypadku wykrycia kolizji gra jest zatrzymywana i wyświetla się odpowiednia informacja.