Grafika Komputerowa - Projekt

Gra horrorowa - Slenderman

Środowisko: Unity 3D
Wersja: 2020.3.1f1

Autorzy:
Karolina Kachelska
Kacper Flis

1. Treść zadania

Gra horrorowa z sieciową kooperacją do 4 graczy. Wzorowana na grze “Slender” z 2012 roku. Celem gry jest zebranie 8 kartek i ucieczka przed ścigającym nas Slender Manem (sztuczna inteligencja). Akcja ma miejsce w nocy, w lesie. Kartki pojawiają się losowo w predefiniowanych lokacjach. Każdy gracz dysponuje własną latarką. Po znalezieniu kartki można ją podnieść. Każda kartka zwiększa poziom trudności gry. Slender Man staje się tym szybszy i bardziej agresywny im bliżej końca gry są gracze. Dotknięcie Slender Mana powoduje, że gracz zostaje wyeliminowany i staje się obserwatorem do końca gry. Koniec gry następuje, gdy wszyscy gracze zostaną wyeliminowani (porażka), lub gdy zebrane zostaną wszystkie kartki (zwycięstwo). Rozgrywce towarzyszą efekty audiowizualne, które nadają jej upiorny klimat.

2. Analiza zadania

● Podstawy teoretyczne problemu

Projekt wykorzystuje implementację dedykowanego serwera. Serwer jest osobną aplikacją, również działającą w Unity. Aby rozpocząć grę serwer musi być włączony, a klienci poprzez wpisanie odpowiedniego adresu IP mogą się z nim połączyć. Komunikacja odbywa się poprzez wymianę pakietów poprzez protokoły TCP i UDP. Każdy pakiet zawiera unikalne id, dzięki któremu wiadomo jaką funkcją przetworzyć przesłane w nim dane.

● Wykorzystywane zagadnienia grafiki komputerowej

1. UNI - Praca z środowiskiem Unity 3D
2. COL - Wykrywanie kolizji
3. ANI - Animacja komputerowa
4. OSW - Modele oświetlenia
5. PAR - Efekty cząsteczkowe

● Wykorzystywane biblioteki i narzędzia programistyczne

Wykorzystane zostanie środowisko Unity 3D w wersji 2020.3.1f1 oraz gotowe już assety.

https://assetstore.unity.com/packages/3d/environments/urban/the-shed-10303
https://assetstore.unity.com/packages/vfx/particles/environment/hail-particles-pack-62038
https://assetstore.unity.com/packages/2d/textures-materials/nature/grass-flowers-pack-free-138810
https://assetstore.unity.com/packages/templates/packs/free-horror-game-kit-108847
https://assetstore.unity.com/packages/2d/textures-materials/nature/terrain-textures-pack-free-139542
https://assetstore.unity.com/packages/3d/vegetation/trees/conifers-botd-142076
https://assetstore.unity.com/packages/2d/textures-materials/floors/yughues-free-sand-materials-12964
https://www.cgtrader.com/free-3d-models/character/fantasy/nvjob-slender

4. Specyfikacja zewnętrzna

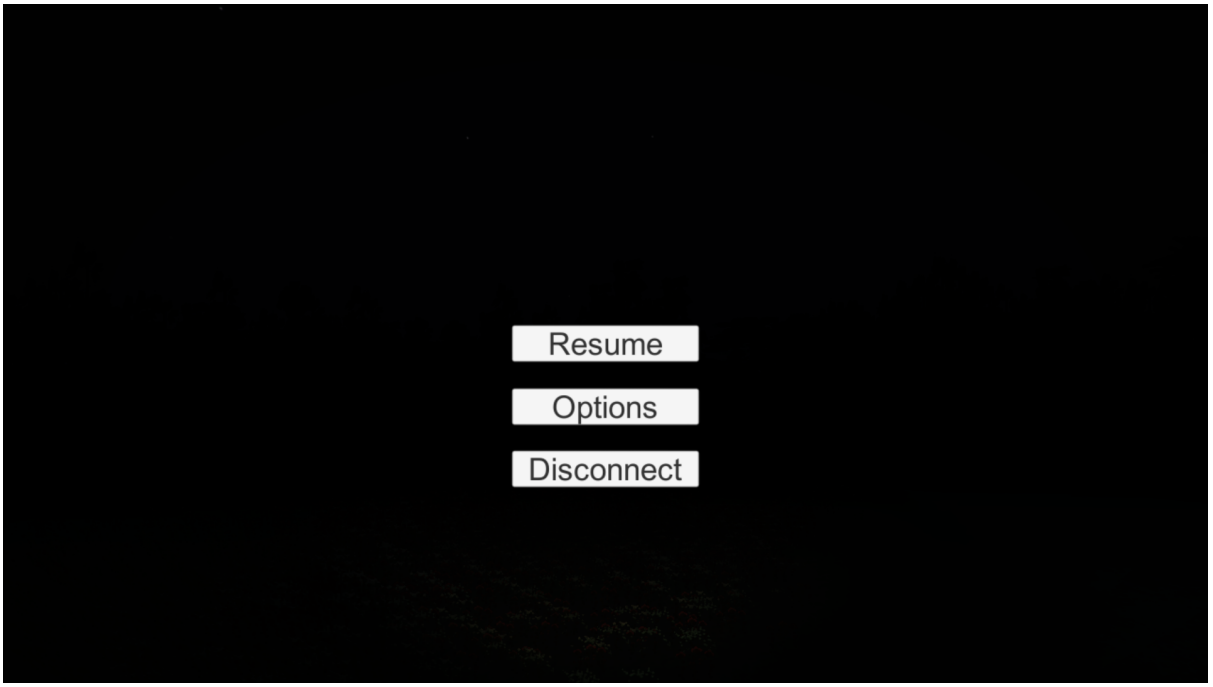
Początkowo należy wpisać adres IP serwera do którego gracz chce się podłączyć. Kolejno należy wpisać swój nick i zgłosić gotowość w lobby w celu rozpoczęcia rozgrywki. Została również dodana opcja ustawień w których można ustawić głośność i czułość myszy.

Gra polega na zebraniu 8 kartek rozstawionych losowo w lesie. Slenderman zaczyna poszukiwania ofiary dopiero po zebraniu pierwszej kartki. dla każdego gracza w lobby pozycja Slendermana i kartek jest taka sama.

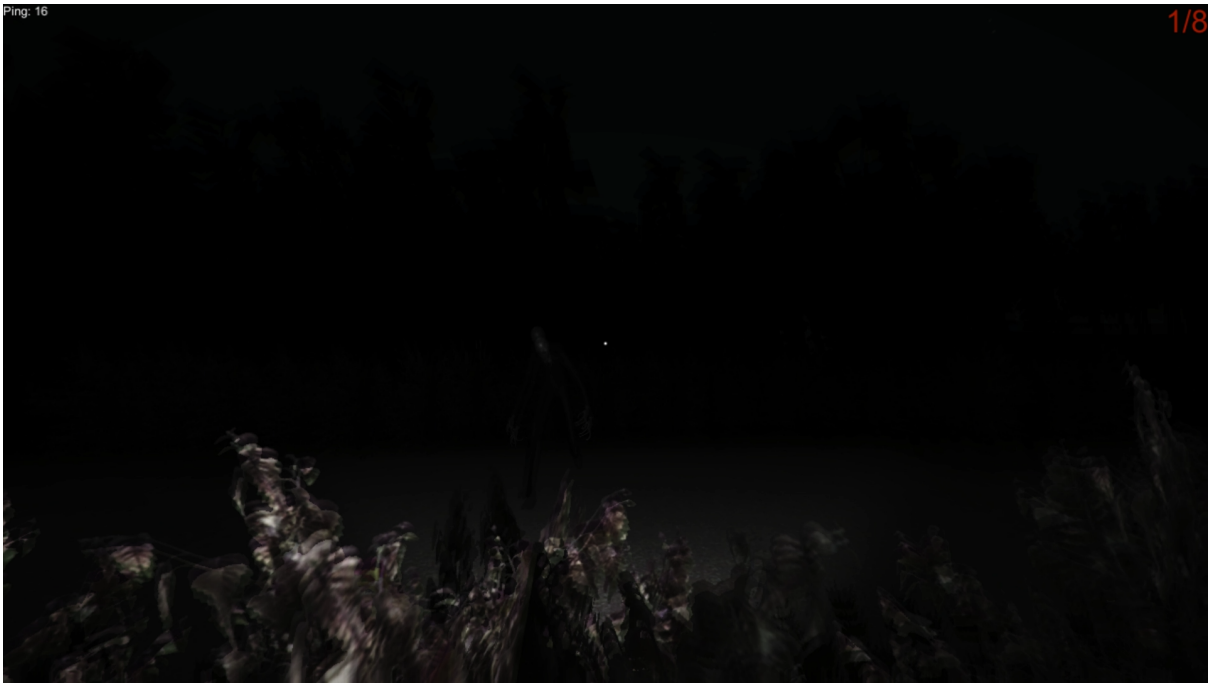
Slenderman goni ofiary mu najbliższe jak również może je zmienić w każdym momencie - pod warunkiem, że pojawi się gracz bliższy mu niż ścigany. Gracz może zauważyć, że Slenderman się zbliża gdyż nakłada się na cały ekran dodatkowy efekt (zielono-fioletowy ekran z szumami wizualnymi). Z każdą kolejną zebraną kartką Slenderman robi się agresywniejszy - tym samym trudniej od niego uciec.

Po zabiciu gracza może on dalej uczestniczyć w grze jako obserwator.
Wygrana następuję jeżeli wszystkie 8 kartek zosatnie zebrane, w przeciwnym wypadku - gdy Slenderman zabije wszystkich graczy gra kończy się ich przegraną.

5. Przykład działania




Ping: 8
0/8
Press [E] to collect


Ping: 16
1/8

6. Specyfikacja wewnętrzna

Client.cs

Klasa obsługująca klienta, posiada 2 klasy UDT i TCP, które służą do obsługi serwera. Posiada ona funkcje **public void SendIntoGame(string _playerName)**, która zajmuje się wysyłaniem gracza do gry.

NetworkManager.cs

Posiada ona dużo krótkich metod takich jak: **public Player InstantiatePlayer(), public void StartGame(), public void RestartGame(), public void OnNotePickup(int _noteId), public void CheckReady(), public void OnPlayerReady(int _playerId, bool _ready) , private void PickNoteSpawns()**, które służa do ogólnej obsługi rozgrywki.

SlenderAI.cs

Klasa do obsługi Slendermana, zajmuję się całym ruchem i animacją postaci.
public void findNewTarget() - metod znajdująca nowy obiekt polowania. Jest ona wywoływania w metodzie Update().
public void setPlayerPosition(int _playerId, Vector3 _playerPosition), public void removePlayerFromTargets(int _playerId) - metody obsługujące interakcje między Slendermanem i Graczem. Zajmują się one przekazaniem wiadomości do serwera, że dany gracz nie istnieje.

ServerSend.cs

Klasa posiadająca metody umożliwiające przekazanie informacji na serwer w zależności od wybranego rodzaju (TCP bądź UDP, jak również istnieje możliwość wysłania informacji do jednego klienta, bądź wszystkich).

ServerHandle.cs

public static void WelcomeReceived(int _fromClient, Packet _packet) - metoda łącząca klienta z serwerem.

public static void PlayerMovement(int _fromClient, Packet _packet) - metoda obsługująca ruch gracza i wysyłająca dane na serwer.

public static void CollectNote(int _fromClient, Packet _packet) - metoda obsługująca zebranie kartek.

Player.cs

Klasa zajmująca się poruszaniem postaci.

UIManager.cs

Klasa zajmująca się obsługą GUI, aktywuje kolejne ekrany menu. Zajmuje się również organizacją graczy w lobby.

7. Testowanie i uruchamianie

Program został przetestowany przez nas. Dzięki temu wykryliśmy dużo błędów, które należało poprawić.

8. Wnioski

Najtrudniejszym etapem pracy było przekazywanie wszystkich danych na serwer, ponieważ tematyka gier multiplayer była dla nas czymś nowym. W celu napisania projektu serwera i klienta posłużył nam poradnik na YouTubie : https://www.youtube.com/watch?v=uh8XaC0Y5MA&list=PLXkn83W0QkfnqsK8I0RAz5AbUxfg3bOQ5

Praca nad systemem serwer-klient spowodowała również najwięcej błędów i najbardziej wpłynęła na czas pisania projektu.

Początkowym utrudnieniem było również same środowisko Unity, które nie umożliwiało darmowegej synchronizacji projektu powyżej 1GB. Spowodowało to napisanie innej, nowej mapy, która jednak nic nie zmieniła w tej kwestii. Końcowo postanowiliśmy zapłacić za wyższy transfer dla naszej wygody i lepszej współpracy, gdyż pierwsze 2 tygodnie pisaliśmy projekt w dwóch oddzielnych projektach i planowaliśmy dopiero na sam koniec je połączyć.

Pozytywnie natomiast zaskoczyły nas darmowe assety, których grafika może nie była powalająca, ale pozwoliły nam na zbudowanie pełnej gry dokładnie takiej jaką zaplanowaliśmy.

Realizacja projektu była bardzo przyjemna, pomimo tego, że pracowaliśmy w zespole dwuosobowym, pokazała nam ona jak pracować w zespole z podziałem na wykonaną pracę.