STRESZCZENIE

ABSTRACT

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

1. ORGANIZACJA

1.1. Główne etapy projektu

1. Wybór i analiza konkretnego kontekstu historycznego.
2. Syntetyczny opis modelu postrzegania przestrzeni na podstawie dzieł pisanych, architektury i sztuki.
3. Przegląd rozwiązań stosowanych w grach strategicznych z wybranego okresu oraz dodatkowo mechanizmów z innych gier, które mogłyby być zaadoptowane na potrzeby projektu.
4. Opracowanie fabuły, selekcja postaci i wydarzeń, a także określenie zakresu autonomii świata gry oraz możliwości modyfikowania go przez gracza.
5. Opracowanie szczegółowej koncepcji i projektu gry, w tym projekt mechanizmów zarówno tych pozwalających graczowi pozyskiwać informacje o świecie gry i zdarzeniach w nim zachodzących, jak również tych poprzez które gracz będzie wywierał wpływ.
6. Implementacja poszczególnych funkcjonalności gry.
7. Testowanie, weryfikacja założeń i walidacja.
8. Stworzenie dokumentacji przeprowadzonych prac.

Przewidywany termin zakończenia prac nad projektem to grudzień 2023 roku.

2. WSTĘP I CEL PRACY

Celem projektu jest zaprojektowanie oraz zaimplementowanie gry real-time strategy osadzonej we wczesnym średniowieczu i przeznaczonej dla jednego gracza. Kluczowe dla niej będzie uwzględnienie mechanik, które umożliwią użytkownikowi podejmowanie decyzji w zbliżony sposób to tego wykorzystywanego w tamtych czasach. Dawniej decyzji strategicznych nie podejmowano na podstawie precyzyjnych map. Ludzie zmuszeni byli budować przestrzenny obraz istniejącej sytuacji w toku dyskusji z naocznymi świadkami, takimi jak dowódcy czy podróżnicy. Z punktu widzenia projektu, uwzględnienie tych różnic jest kluczowe, gdyż umożliwi to graczowi wczucie się w realia epoki. Bliski pożądanemu efektowi jest sposób w jaki zrealizowano dowodzenie drużyną w grze RPG „Pillars of Eternity". Niestety, te mechaniki w tego typu grach zwykle są bardzo ograniczone i nie pozwalają na zarządzanie większymi grupami wojsk, czy budowę baz, czyli zagadnienia podstawowe z punktu widzenia gier RTS. Z tego powodu celem pracy jest opracowanie projektu gry strategicznej czasu rzeczywistego dla jednego gracza, w której dowodzenie przebiegałoby w sposób możliwie zgodny z realiami historycznymi.

3. PRZEGLĄD ROZWIĄZAŃ STOSOWANYCH W GRACH

3.1. System dialogów w grze Mass Effect 3

W Mass Effect 3 system dialogowy jest integralną częścią rozgrywki, pozwalając graczom na prowadzenie rozmów z różnymi postaciami w trakcie gry. System dialogów w Mass Effect 3 wykorzystuje interfejs oparty na kole dialogowym. To koło przedstawia graczom wiele opcji odpowiedzi podczas rozmów, zwykle podzielonych na kategorie według ich ogólnego tonu lub intencji. Dostępne opcje często obejmują wybory, dyplomatyczne, agresywne, konfrontacyjne oraz opcje neutralne lub śledcze. Podczas niektórych rozmów lub przerywników filmowych gracze mogą przerwać trwającą rozmowę, szybko wybierając określoną opcję dialogową. Te opcje przerywania pozwalają graczom podjąć natychmiastowe działania lub podjąć decyzje na miejscu, często wpływając na wynik sytuacji lub relacje postaci z innymi. Ogólnie rzecz biorąc, system dialogowy w Mass Effect 3 został zaprojektowany tak, aby zapewnić graczom bogate i wciągające doświadczenie w opowiadaniu historii, pozwalając im kształtować narrację poprzez wybory i interakcje z olbrzymią gamą postaci. System oferuje różnorodne opcje odpowiedzi, dynamiczne rozmowy i konsekwencje, przyczyniając się do fascynującej i rozgałęzionej narracji gry.

3.2. Model sztucznej inteligencji przeciwników w grach Warcraft III i StarCraft II

Sztuczna inteligencja przeciwników w grach takich jak Warcraft III lub StarCraft II, przede wszystkim w trybie kampanii, jest odpowiedzialna za kontrolowanie wrogich jednostek w celu zaoferowania graczowi wyzwania. Głównym zadaniem AI jest zasymulowanie strategicznych decyzji i wydajne zarządzanie zasobami. AI podejmuje decyzję na podstawie predefiniowanych zasad i algorytmów. Analizuje sytuację, w której się znajduje, biorąc pod uwagę siłę swojej własnej armii, siłę armii gracza oraz specjalne zdolności jednostek i środowisko, w którym toczy się gra. Ta analiza pozwala komputerowi na podejmowanie strategicznych decyzji jak na przykład, kiedy atakować, bronić się, eksplorować oraz rozszerzać swoje terytorium. W tych grach sztuczna inteligencja może przybrać jedną z kilku wariantów wynikających z poziomu trudności. Wyższe poziomy dają przeciwnikowi przewagę takie jak wydajniejsze zbieranie zasobów lub szybsza produkcja jednostek.

W grze Warcraft III w trybie kampanii zachowanie przeciwników jest zaprojektowane z myślą o zanurzeniu gracza w fabularnej opowieści, jednocześnie prezentując wciągające wyzwania związane z rozgrywką. Akcje wykonywane przez sztuczną inteligencję są dostosowane do celów danej misji, co pozwala na dopasowanie do obowiązującej narracji. Początkowo przeciwnik konstruuje i rozbudowuje swoją bazę, w celu zgromadzenia odpowiedniej liczby zasobów, szkolenia jednostek i prowadzenia badań. AI strategicznie rozmieszcza budynki i struktury obronne, aby ochronić swoją fortecę przed najazdami gracza. Misje kampanii często też zawierają oskryptowane wydarzenia lub walki, które dodają głębi rozgrywce. Podczas tych starć wroga sztuczna inteligencja może zachowywać się w specjalny sposób, kontrolując potężne jednostki, do których gracz normalnie nie ma dostępu lub inicjując działania, które popychają narrację do przodu. Te wyreżyserowane wydarzenia tworzą niezapomniane chwile i jeszcze bardziej wciągają gracza w fabułę kampanii. Zachowanie wroga w kampanii jest zróżnicowane i obejmuje różnorodne cele misji i scenariusze. Gracze mogą napotkać wrogów, którzy preferują agresywne ataki, inni skupiają się na strategiach obronnych lub specjalizują się w taktyce hit and run. Sztuczna inteli-

gencja dostosowuje proces podejmowania decyzji do konkretnych wymagań misji, często wykorzystując ukształtowanie terenu, synergię jednostek i scenariusze wydarzeń, aby rzucić wyzwanie umiejętnościom gracza. Ogólnie rzecz biorąc, zachowanie wrogów w kampanii Warcraft III ma na celu zapewnienie dynamicznego i wciągającego doświadczenia. Gracze muszą wykorzystywać myślenie strategiczne, zarządzanie zasobami i efektywny skład jednostek, aby przezwyciężyć różnorodne strategie stosowane przez wrogą sztuczną inteligencję.

4. SYSTEM DIALOGÓW

System dialogów jest podstawową metodą, którą gracz będzie wykorzystywał, aby pozyskać informacje o świecie oraz celach misji. Gracz może inicjować konwersacje z postaciami niezależnymi, po czym zostaną mu zaproponowane opcje sposobu prowadzenia rozmowy. W zależności od wybranych opcji dialogowych gracz może się spodziewać różnych konsekwencji.

Rysunek 4.1: Kadr z gry Fallout 3 przedstawiający przykładowy dialog

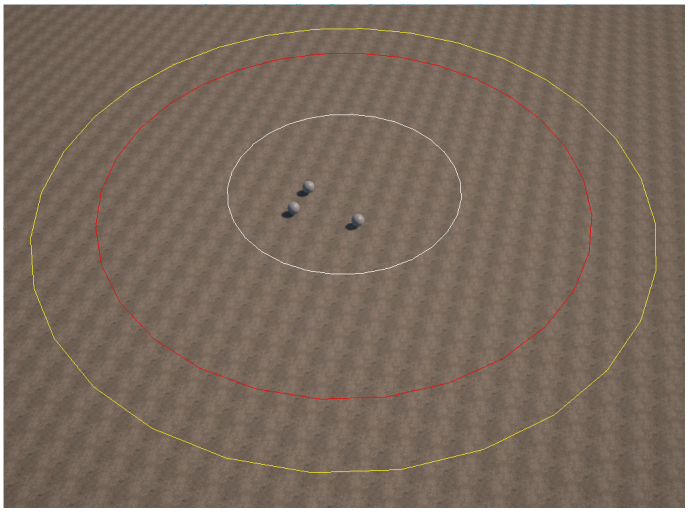
Dialogue Editor autorstwa Grasshop Dev jest prostym narzędziem pozwalającym na szybkie dodawanie i modyfikację dialogów. Zawiera zestaw elementów ułatwiających wdrożenie systemu do projektu oraz udostępnia struktury danych wykorzystywanych do tworzenia interfejsu użytkownika. Podczas rozmowy z postaciami niezależnymi gracz będzie mógł pozyskać informację o geografii świata, możliwych zagrożeniach oraz zadaniach do wykonania. Podobne systemy występują w grach takich jak Pillars of Eternity oraz w grach z serii Mass Effect.

5. SZTUCZNA INTELIGENCJA PRZECIWNIKÓW

W przypadku implementacji mechanizmów sztucznej inteligencji przeciwników będziemy się inspirować trybami kampanii w grach Warcraft III oraz Starcraft II. Na mapie będą rozsiane punkty, w których będą pojawiać się przeciwnicy. Tak długo, jak drużyna gracza jest poza zasięgiem, wrogowie pozostają nieaktywni. Aktywni przeciwnicy zachowują się zgodnie z ich archetypem (klasą postaci) oraz pozostają aktywni tak długo, jak drużyna gracza jest w zasięgu. Zdezaktywowani przeciwnicy wracają do swojego oryginalnego stanu. Gracz będzie napotykał tego typu obozowiska przede wszystkim w trakcie eksploracji świata. Innym planowanym przykładem implementacji sztucznej inteligencji przeciwników jest model, w którym jednostki wroga poruszają się z punktu początkowego w stronę bazy gracza. Jeśli podczas swojej podróży napotkają drużynę gracza, wtedy niezwłocznie zmieniają swój cel ataku. Będziemy wyróżniać trzy archetypy jednostek w zależności od sposobu walki (bliski zasięg, średni zasięg, daleki zasięg). Postacie walczące na bliski zasięg mają na celu podejście w stronę najbliższego przeciwnika i wykonać atak. Jednostki średnio zasięgowe w momencie, w którym najbliższy przeciwnik jest odpowiednio daleko, wykonują atak dystansowy, w przeciwnym wypadku zachowują się tak jak jednostki walczące w zwarciu. Postacie dalekodystansowe dokonują ataków dystansowych w kierunku najbliższego przeciwnika, natomiast uciekają, gdy ten podejdzie zbyt blisko.

Przeciwnicy są kontrolowani poprzez jeden obiekt przydzielający cele każdemu przypisanemu wrogowi. W normalnym trybie wrogowie poruszają się w sposób losowy w obrębie wyznaczonej przestrzeni (biały okrąg). Kiedy przyjazne jednostki znajdą się w wystarczającej odległości (czerwony okrąg), przeciwnicy obiorą sobie za cel jedną z nich. Po opuszczeniu przez drużynę gracza wyznaczonego obszaru (żółty okrąg) wrogowie wracają do poruszania się w sposób losowy w obrębie białego okręgu.

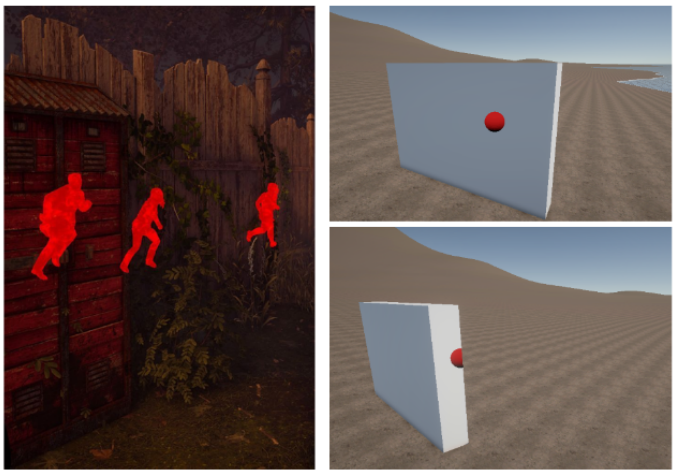
Nawigacja przeciwników została zrealizowana poprzez wbudowany w silnik Unity system NavMesh. Pozwala nam on na łatwe wyznaczenie powierzchni, po której mogą poruszać się postacie niekontrolowane przez gracza oraz realizuję zadanie wyznaczania ścieżki dla tych postaci.



Rysunek 5.1: Obraz przedstawia zasięgi odpowiednich regionów.

6. WIDZENIE PRZEZ ŚCIANY

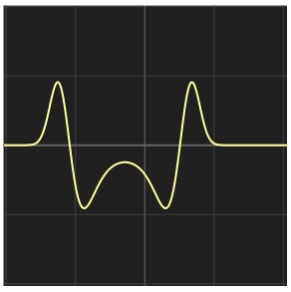
Do umiejętności wykorzystywanych przez gracza będzie należeć zdolność widzenia przeciwników oraz innych istotnych obiektów przez przeszkody. Gracz po wciśnięciu przycisku przez krótki okres będzie w stanie zobaczyć sylwetki przeciwników znajdującymi się w jego polu widzenia. Grafika przedstawia rozwiązanie zawarte w grze Dead by Daylight. Markery nie poruszają się za celem lecz pojawiają się i pozostają w tym samym miejscu przez czas trwania animacji.



Rysunek 6.1: Po lewej stronie przykład z gry Dead by Daylight. Po prawej zachowanie shadera w przypadku przysłanianiu markera przez przeszkody.

Efekt został osiągnięty poprzez zmodyfikowanie potoku renderowania w taki sposób, że w zależności od wartości w buforze głębi jest wykorzystywany inny shader. W tym przypadku, jeżeli sfera jest przysłonięta przez ścianę jest ona narysowana w przeciwnym wypadku jest uruchamiany pusty shader.

Po naciśnięciu przycisku E następuje zagranie animacji opisanej wzorami $w(t, offset) = 1.1 \times 2.1^{-\left(\frac{(\sin(t)+1-0.4-\text{offset})^2}{0.02}\right)}$ oraz $w(t,0) - w(t,-0.2) + w(t,-1) - w(t,-1.2)$. Od podanych funkcji zależy przeźroczystość, jak i natężenie efektu Fresnela.



Rysunek 6.2: Wykres przedstawiający funkcję opisującą zachowanie efektu Fresnela w animacji markera

Listing 6.1: Fragment shadera odpowiedzialny za animację

```
fixed4 frag (v2f i) : SV_Target
{
```

```
    float t =  6.2 * _Progress - 0.6;
    fixed4 pattern = tex2D(_PatternTex, i.uv + _Speed *t);
    float fresnelInfluence = dot(i.worldPos, i.viewDir);
    float saturatedFresnel = saturate(1 - fresnelInfluence);

    float g = w(t, 0) - w(t, -0.2) + w(t, -1) - w(t, -1.2);
    float4 color = pow(saturatedFresnel, g * _FresnelPow) * (_Color * _ColorIntensity) * p
    color.a *= dot(i.worldPos, i.viewDir);
    return color;
}
```

7. WPROWADZENIE DO DZIEDZINY

7.1. Przegląd silników

Podstawą tworzenia gier jest silnik graficzny. Stanowi serce kodu, odpowiadając za interakcję poszczególnych elementów. Dostarcza podstawowe narzędzia dzięki którym łatwiej można dokonywać zmian w grze. Wybranie odpowiedniego silnika przed rozpoczęciem pracy ma kluczowy wpływ na proces wytwórczy i efekt końcowy.

Obecnie dostępnych jest wiele silników, a każdy z nich posiada różne możliwości. W celu uproszczenia wyboru zdecydowaliśmy się zawęzić listę do trzech pozycji. Są to Godot, Unity oraz Unreal Engine.

Pierwszy z nich jest w pełni darmowym silnikiem open source. Posiada prosty i intuicyjny interfejs, a w internecie tworzonych jest przez społeczność wiele samouczków. Nie posiada on jednak oficjalnej dokumentacji oraz jest zdecydowanie mniej popularny od pozostałych dwóch.

Kolejny silnik, Unity jest określany jako przyjazny dla początkujących. Posiada bogatą dokumentację oraz jest dostępnych dużo samouczków stworzonych przez jego społeczność. Unity świetnie się nadaje do tworzenia gier 3D. Silnik ten jest dostępny w wersji bezpłatnej oraz oferującej więcej możliwości wersji płatnej. Co więcej, posiada możliwość rozszerzenia o dodatkowe narzędzia dostępne w Assets Store.

Ostatni z silników jest najbardziej kojarzony z grami AAA. Cechuje go zaawansowana grafika, która umożliwia wytwarzanie fotorealistycznych gier. Korzystanie z niego jest darmowe, a opłata w wysokości 5

Poniższa tabela przedstawia porównanie wymienionych silników w istotnych, z punktu widzenia projektu, aspektach.

Silnik	Unity	Unreal Engine	Godot
Popularność	duża	duża	mała
Elastyczność	duża	duża	duża
Obsługa	dobra	dobra	dobra
Język	C#	C++	C#, C++, GDScript
Baza wiedzy	dokumentacja, samouczki	dokumentacja, samouczki	samouczki, fora

Finalnie zdecydowaliśmy się na implementację gry w Unity, ponieważ jest to silnik, który najlepiej odpowiada wymaganiom projektu.

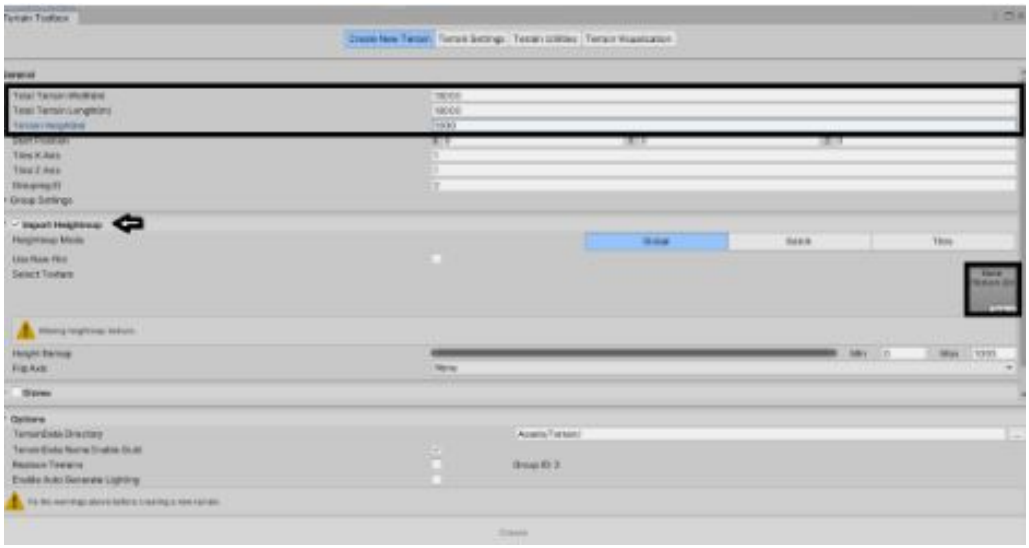
8. TECHNOLOGIE, ALGORYTMY I NARZĘDZIA

8.1. Modelowanie terenu

Do utworzenia terenu do gry wykorzystaliśmy zasób 3D World Building udostępniany przez Unity. Umożliwia on automatyczne generowanie terenu na podstawie heightmapy - monochromatycznego obrazu reprezentującego model wysokościowy. Kolor czarny reprezentuje najniższe punkty, natomiast kolor biały - najwyższe.

Rysunek 8.1: Przykładowe heightmapy

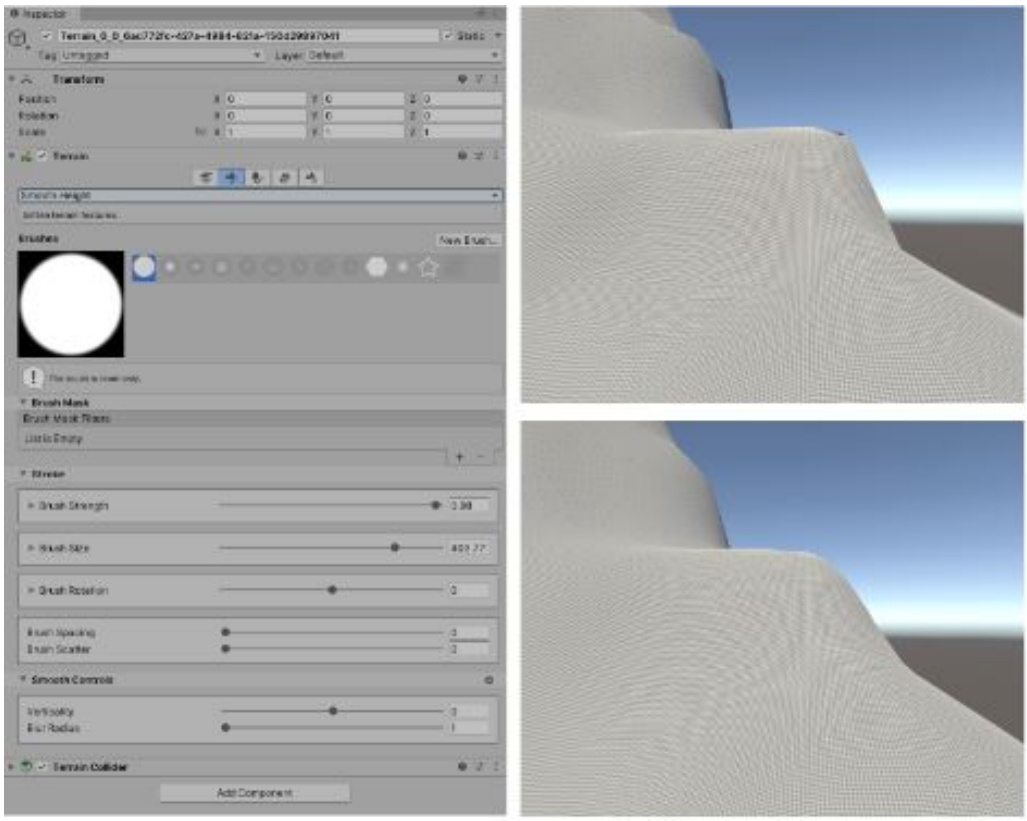
Wygenerowanie terenu umożliwia narzędzie Terrain Toolbox, które można uruchomić wybierając z menu Window -> Terrain -> Terrain Toolbox. Pozwala ono na ustawienie podstawowych parametrów takich jak długość, szerokość oraz wysokość terenu. Należy również zaznaczyć checkbox Import Heightmap oraz załączyć obraz z modelem wysokościowym. Na koniec trzeba nacisnąć przycisk Create, co spowoduje dodanie do sceny wygenerowanego terenu.



Rysunek 8.2: Widok na panel narzędzia Terrain Toolbox z zaznaczonymi wymienionymi sekcjami.

Tak utworzony teren, chociaż już jest grywalny, posiada ostre i postrzępione krawędzie, które nie wyglądają zbyt estetycznie. Wygładzenie ich poprawi wygląd terenu i sprawi, że będzie on bardziej realistyczny. Do tego służy narzędzie Smooth Height dostępne w inspektorze terenu. Powoduje ono

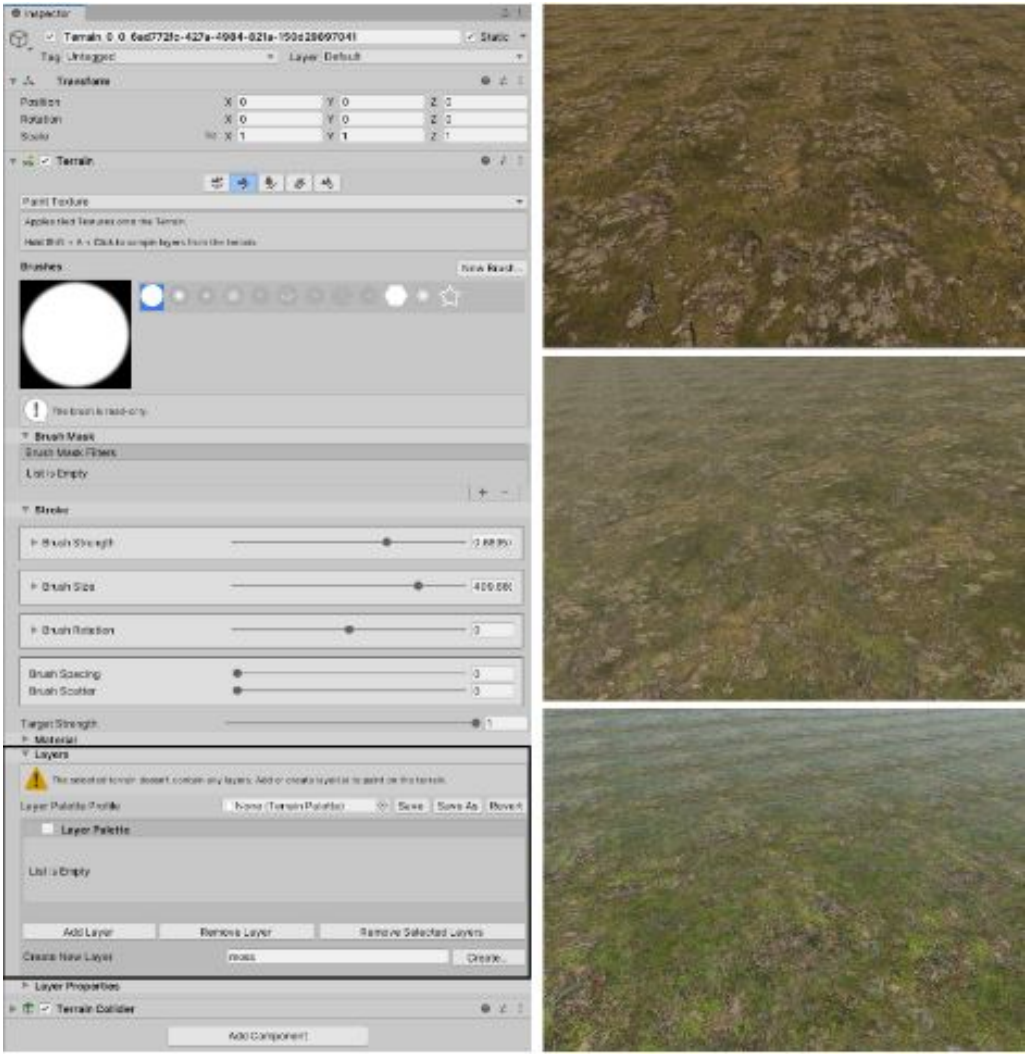
uśrednienie pobliskich płaszczyzn, co pozwala na usunięcie nagłych zmian terenu i w rezultacie wygładzenie go.



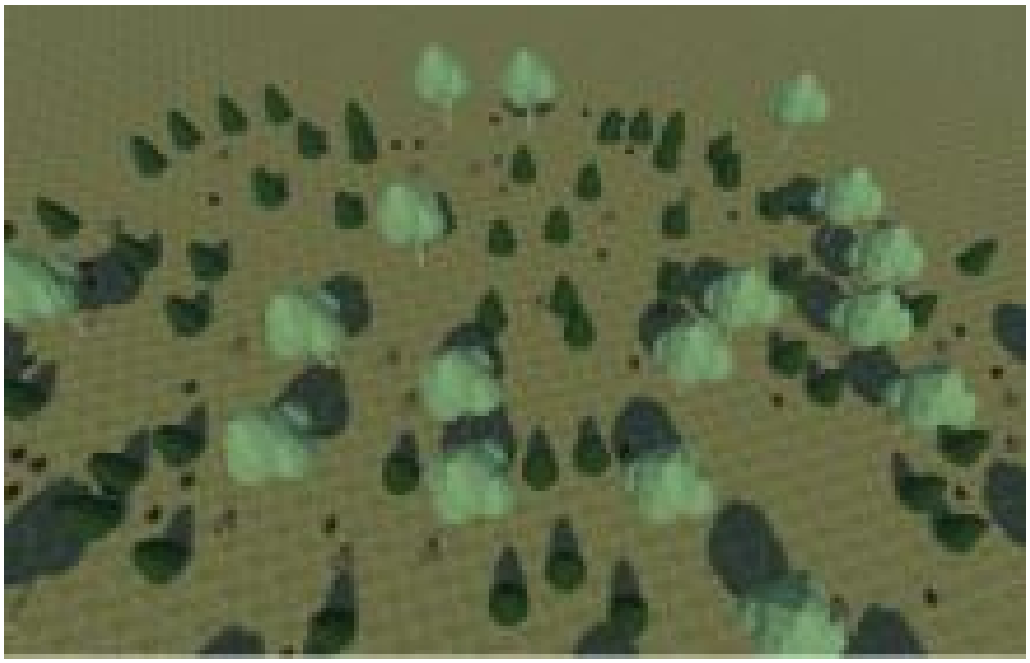
Rysunek 8.3: Widok panelu inspektora oraz terenu przed (górny) i po (dolny) zastosowaniu narzędzia Smooth Terrain.

Kolejnym krokiem jest nałożenie tekstur. Służy do tego narzędzie Paint Texture. Umożliwia ono dodanie warstw, którymi będzie można pokolorować teren. Warstwa znajdująca się najwyżej jest uznawana za domyślną i jej tekstura zostanie nałożona na cały teren. Pozostałe warstwy natomiast stanowią swego rodzaju paletę kolorów, którymi można pomalować teren za pomocą pędzla, który można wybrać w zakładce Brushes. W zakładce Stroke można ustawić podstawowe parametry pędzla takie jak Brush Size oraz Brush Strength. Pierwszy parametr odnosi się do rozmiaru pędzla, a co za tym idzie obszaru na który dana tekstura zostanie nałożona, natomiast drugi pozwala na określenie w jakim stopniu nakładany materiał zakryje już nałożony.

Kolejną opcją udostępnianą przez Unity jest możliwość automatycznego ustawienia obiektów na mapie w losowy sposób, dzięki narzędziu Paint Trees. Do udostępnianych przez nie parametrów należą między innymi Brush Size, działający analogicznie jak poprzednio, oraz Tree Density, które definiuje średnią liczbę drzew umieszczanych na zdefiniowany obszar. Obok przedstawiono przykładowy rezultat. Wykorzystano do tego paczkę LowPoly Trees and Rocks dostępną w Unity Assets Store.



Rysunek 8.4: Widok panelu inspektora z wybranym narzędziem Paint Texture oraz efektu przed nałożeniem 2 tekstury (górny zrzut), po nałożeniu tekstury gdy Brush Strength wynosi 0.26 (środkowe zdjęcie) oraz gdy wynosi 0.67 (dolne).



Rysunek 8.5: Widok na teren z drzewami.

Rysunek 8.6: Widok na inspektor z włączonym narzędziem Paint Trees.

8.2. Mechanizm budowania

Jednym z istotnych elementów w grach real-time strategy jest tworzenie baz i budowanie fortyfikacji. Mechanizm ten stanowi urozmaicenie rozgrywki i wprowadza dodatkowe aspekty możliwe do uwzględnienia w planowaniu strategii. Dla wielu gier RTS jest wręcz nieodłącznym elementem, który umożliwia graczowi tworzenie i rozwój nowych jednostek, produkcję zasobów, umacnianie swojej pozycji oraz zwiększanie swojej potęgi.

Rozpatrując ten mechanizm konieczne jest uwzględnienie podstawowych ograniczeń związanych z ukształtowaniem terenu oraz rozmieszczeniem już istniejących obiektów i elementów scenerii. Czynniki te mają ogromne znaczenie, gdyż zignorowanie ich może doprowadzić do błędów oraz niepożądanych efektów i w rezultacie do obniżenia jakości rozgrywki. Dodatkowo, kluczową kwestią są zasoby wymagane do wybudowania danego obiektu. Ich istnienie powoduje, że gracz jest w pewien sposób ograniczony i nie może tworzyć budowli w dowolnej ilości.

Jednym z błędów, którym należy przeciwdziałać jest nakładanie się na siebie obiektów. Taka sytuacja może powstać w wyniku braku rozpatrywania przez grę położenia elementów terenu i istniejących budowli w trakcie umieszczania na mapie nowych budynków przez gracza. W efekcie może dojść do sytuacji, że obiekt zostanie wybudowany w miejscu zajmowanym przez inną część scenerii. Gra powinna przeciwdziałać temu zjawisku, uniemożliwiając graczowi tworzenie nowych budynków w miejscu, które w nawet najmniejszym stopniu narusza obszar zajęty przez już istniejący element.

Inną sytuacją, której należy unikać jest umożliwienie graczowi wybudowanie budynku w sposób fizycznie niemożliwy. Przykładem może być ulokowanie obiektu na zbyt stromym zboczu albo na nieodpowiednim gruncie, co w rzeczywistym świecie byłoby niedopuszczalne. Również w tym przypadku gra nie powinna umożliwić graczowi wykonanie takiej akcji. W przeciwnym razie obiekt mógłby zostać umieszczony w miejscu do którego postać gracza nie będzie mieć dostępu i nie będzie mógł z niego korzystać. W efekcie zużyje on bezsensownie swoje zasoby tracąc je bezpowrotnie, co może negatywnie wpłynąć na jego opinię o grze.

Inspiracją do implementacji tego mechanizmu jest gra Warhammer 40,000: Dawn of War. Chociaż realia tej gry znacząco różnią się od tych w których zostanie osadzona fabuła tworzonej przez nas gry, to stanowi świetny przykład pożądanego efektu. Udostępnia ona możliwość podglądu z trzeciej osoby przed wybudowaniem wraz z walidacją położenia. Jeśli miejsce w którym gracz chce postawić budynek jest poprawne tzn. nie nachodzi na inne obiekty oraz teren jest odpowiedni to pokazywany widok jest podświetlany na zielono, w przeciwnym razie - na czerwono. Jest to zbliżony efekt, który chcielibyśmy uzyskać dla widoku pierwszoosobowego.

Aktualnie gra umożliwia graczowi przełączenie się w tryb budowania poprzez naciśnięcie klawisza Tab, który od razu wyświetli podgląd bazowego obiektu. Widok budynku przemieszcza się przed graczem oraz odpowiednio obraca się razem z nim. Efekt poruszania się podglądu został uzyskany za pomocą poniższych wzorów:

$$x = r \times sin(\alpha)$$

$$y = 0$$

$$z = r \times cos(\alpha)$$

gdzie r to odległość środka obiektu od postaci gracza, natomiast α to kąt o jaki jest on obrócony względem osi y. W przypadku rotacji alpha została przypisana do rotacji obiektu względem tej osi.

Podgląd budynku będzie widoczny do czasu, aż gracz go umieści naciskając prawy lewy myszy, bądź wychodząc z trybu edycji naciskając klawisz Escape.

9. PROJEKT SYSTEMU

10. BADANIA

11. PODSUMOWANIE

WYKAZ RYSUNKÓW

WYKAZ TABEL