Kierunek: ***Informatyka***

2024/2025

Seweryn Dutka

PRACA INŻYNIERSKA

Projekt i implementacja gry survivalowej

Promotor pracy:

mgr inż. Tomasz Gądek

Tarnów, *2025*

Spis treści

1. Wstęp

W dzisiejszych czasach branża gier komputerowych rozwija się w zawrotnym tempie, oferując użytkownikom zaawansowane technologicznie i realistyczne doświadczenie rozgrywki. Wśród gatunków gier szczególnie popularnych w ostatnich latach znajdują się gry survivalowe, które łączą elementy przetrwania i eksploracji, angażując graczy i stawiając ich przed wymagającymi wyzwaniami.

1.1. Motywacja

Wybór tematyki gry survivalowej był podyktowany jej popularnością wśród graczy oraz unikalnym połączeniem wyzwań i swobody eksploracji. Projekt stanowił okazję do połączenia wiedzy teoretycznej zdobytej podczas studiów z praktycznymi umiejętnościami programowania i projektowania. Realizacja gry umożliwiła poszerzenie kompetencji w zakresie tworzenia interfejsów użytkownika oraz implementacji zaawansowanych mechanik rozgrywki.

Dodatkowym motywatorem była chęć zaprojektowania gry, która dostarcza angażującej rozrywki, wykorzystując przy tym nowoczesne technologie i narzędzia dostępne na rynku. Projekt pozwolił również na zgłębianie zagadnień integracji systemów wizualnych oraz dźwiękowych, kluczowych dla współczesnych produkcji gier.

1.2. Cel pracy

Głównym celem niniejszej pracy jest opracowanie oraz wdrożenie gry komputerowej o tematyce survivalowej, która łączy w sobie mechanikami przetrwania i eksploracji otwartego świata. Projekt ma na celu wykonanie funkcjonalnego produktu, który będzie spełniał wymagania współczesnych graczy zarówno pod względem estetyki, jak i technicznych aspektów działania. W ramach pracy została przeprowadzona analiza wymagań, zdefiniowany stos technologiczny oraz przeprowadzone testy funkcjonalności i wydajności gry. Ostatecznym rezultatem jest grywalny prototyp.

1.3. Zakres pracy

Zakres niniejszej pracy obejmuje następujące etapy:

1. **Analiza biznesowa**
   - Określenie wymagań funkcjonalnych w tym kluczowych mechanik gry (przetrwanie, eksploracja, tworzenie przedmiotów).
   - Określenie wymagań niefunkcjonalnych (technicznych i jakościowych).
2. **Wybór i zastosowanie technologii**
   - Unreal Engine 5 jako platforma do implementacji gry.
   - Quixel Bridge, Fab, Blender do zdobywania i modelowania 3D.
3. **Projektowanie i implementacja funkcjonalności**
   - Implementacja systemu HUD (ang. *Head-Up Display*), ekwipunku i powiadomień o surowcach.
   - Mechaniki eksploracji, zbierania surowców i tworzenia przedmiotów.
4. **Projektowanie świata gry**
   - Zaprojektowanie otwartego świata z realistyczną topografią, roślinnością i animacjami postaci.
5. **Testowanie**
   - Testy manualne sprawdzające mechaniki gry.

Realizacja tych etapów doprowadziła do opracowania funkcjonalnego prototypu gry survivalowej, gotowego do dalszego rozwoju.

2. Analiza biznesowa

Analiza biznesowa jest kluczowym etapem w projektowaniu każdego produktu, w tym także gier komputerowych, ponieważ pozwala na określenie głównych celów projektu, grupy docelowej oraz wymagań, które wpływają na kształt finalnego produktu.

W przypadku gry survivalowej analiza biznesowa obejmuje szczegółowe określenie założeń funkcjonalnych i niefunkcjonalnych, aby finalna wersja gry była dopracowana pod kątem zarówno rozgrywki, jak i stabilności oraz kompatybilności technicznej.

Rozgrywka w grze survivalowej opiera się na mechanikach eksploracji i przetrwania w otwartym świecie, pełnym zagrożeń i zasobów. Gracz wciela się w postać, której głównym celem jest przetrwanie jak najdłużej w trudnych, naturalnych warunkach. W miarę rozgrywki musi gromadzić zasoby, takie jak drewno, kamień czy pożywienie, by zaspokoić podstawowe potrzeby postaci i przygotować się na niebezpieczeństwa, które mogą pojawić się w każdej chwili. Dodatkowym elementem rozgrywki jest możliwość tworzenia przedmiotów oraz budowy struktur, co pozwala graczowi na lepsze przystosowanie się do środowiska.

2.1. Wymagania niefunkcjonalne

Założenia niefunkcjonalne dotyczą wymagań technicznych i jakościowych, które wpływają na wydajność, dostępność oraz długoterminowe wsparcie gry. W projektowanej grze survivalowej niefunkcjonalne wymagania obejmują:

Tabela 2.1. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne	**Wymagania techniczne i jakościowe gry**
Kompatybilność z systemem Windows	Gra powinna działać poprawnie na urządzeniach z systemem Windows 10 lub nowszym.
Tryb offline	Gra powinna umożliwiać rozgrywkę offline.
Optymalizacja wydajności	Gra powinna być zoptymalizowana pod kątem płynności na urządzeniach o przeciętnych parametrach.
Interfejs użytkownika	Interfejs użytkownika powinien być intuicyjny, zapewniając szybki dostęp do paska zdrowia, ekwipunku.

2.2. Wymagania funkcjonalne

Założenia funkcjonalne definiują konkretne mechaniki i interakcje, które wspierają rozgrywkę. W survivalowych grach komputerowych kluczowe wymagania funkcjonalne koncentrują się wokół zarządzania zdrowiem, przetrwania, eksploracji otwartego świata oraz zdobywania zasobów. W projektowanej grze survivalowej wymagania funkcjonalne obejmują:

Tabela 2.2. Wymagania funkcjonalne

Wymagania funkcjonalne	**Opis**
Świat gry	Gracz może swobodnie eksplorować duży, otwarty teren w trójwymiarowej przestrzeni.
Postać gracza	Postać gracza posiada animacje ruchu, możliwość interakcji z obiektami oraz statystyki (zdrowie, głód, energia).
System zdrowia	Postać posiada pasek zdrowia, który ulega zmniejszeniu w wyniku obrażeń lub głodu.
Inwentarz	Postać może zbierać przedmioty, które są przechowywane w ograniczonym inwentarzu.
Zbieranie zasobów	Gracz może zbierać zasoby naturalne (drewno, kamienie).
Tworzenie przedmiotów i struktur	System umożliwia tworzenie narzędzi, broni z zebranych materiałów przy wykorzystaniu prostego interfejsu.
Fauna	W grze pojawiają się dzikie zwierzęta, które mogą być zarówno źródłem pożywienia, jak i zagrożeniem.
Interakcja z otoczeniem	Gracz może używać narzędzi, budować struktury oraz wchodzić w interakcję z elementami otoczenia.
Eksploracja	Gracz może badać obszar świata.
System walki	Mechaniki przetrwania, które pozwalają graczowi na przetrwanie w obliczu zagrożeń.

3. Stos technologiczny

Wybór odpowiedniego stosu technologicznego jest kluczowy dla sprawnej realizacji projektu gry survivalowej, która wymaga wysokiej wydajności, realistycznej grafiki oraz zaawansowanych mechanik przetrwania. W tej części omówione zostaną narzędzia, frameworki i biblioteki zastosowane przy produkcji gry, wraz z uzasadnieniem ich wyboru.

3.1. Silnik gry

Na rynku istnieje wiele silników gier, które oferują szeroki wachlarz możliwości. Przykładowo, Unity jest popularnym wyborem dla twórców niezależnych ze względu na swoją wszechstronność i łatwość użycia [6]. Z kolei Godot wyróżnia się tym, że działa na mniej wydajnym sprzęcie dzięki integracji z wieloma bibliotekami zewnętrznymi i optymalizacją wydajności, co czyni go idealnym dla mniejszych projektów [7]. Do realizacji niniejszego projektu wybrano Unreal Engine jest jednym z najpopularniejszych i najbardziej zaawansowanych silników gier dostępnych na rynku. Został zaprojektowany przez firmę Epic Games i po raz pierwszy zaprezentowany w 1998 roku wraz z premierą gry „Unreal". Początkowo silnik był dedykowany wyłącznie do gier FPS (ang. *first-person shooter*), jednak z czasem został rozwinięty i wzbogacony o nowe możliwości, co pozwoliło na jego wykorzystanie w szerokim spektrum gatunków gier, w tym RPG (ang. *role-playing game*), strategiach, symulacjach oraz grach survivalowych. Od czasu wydania Unreal Engine 4 w 2014 roku, silnik stał się bardziej dostępny dla niezależnych deweloperów dzięki modelowi licencyjnemu opartemu na tantiemach [3]. Obecnie, Unreal Engine 5, wydany w 2021 roku, wnosi dodatkowe innowacje, takie jak Nanite i Lumen, które umożliwiają jeszcze bardziej realistyczną grafikę i dynamiczne oświetlenie [3].

Zalety Unreal Engine:

- **Wysokiej jakości grafika**: Unreal Engine umożliwia tworzenie fotorealistycznych scen oraz modeli 3D. Silnik ten oferuje nowoczesne efekty oświetleniowe, dynamiczne cieniowanie oraz wsparcie dla wysokiej jakości tekstur, co pozwala na pełne wykorzystanie potencjału grafiki 3D w grze survivalowej [3].
- **Blueprints – wizualne skrypty**: Unreal Engine udostępnia system Blueprint, który umożliwia tworzenie logiki gry za pomocą wizualnego programowania. Dzięki temu można szybko prototypować nowe funkcje bez potrzeby pisania kodu, co jest idealne przy pracy nad zaawansowanymi mechanikami przetrwania [3].

- **Zaawansowany system fizyki**: Unreal Engine obsługuje realistyczną fizykę, która pozwala na naturalne reakcje obiektów i postaci, co jest kluczowe dla immersji w grze survivalowej [3].
- **Wsparcie dla platformy Windows**: Unreal Engine pozwala na budowanie gier dla systemu Windows, co jest zgodne z wymaganiami projektu [3].

3.2. Rdzeń tworzenia gier w Unreal Engine: C++ i Blueprint

Podstawowym językiem programowania w Unreal Engine jest C++, który pozwala na tworzenie złożonego kodu o wysokiej wydajności. Język ten został opracowany w 1983 roku przez Bjarne Stroustrupa i stanowi jedno z najszybszych i najbardziej wydajnych narzędzi do programowania w branży gier [1]. Unreal Engine wspiera pełną integrację C++, umożliwiając dostęp do każdej funkcji silnika i maksymalną elastyczność w tworzeniu oraz optymalizacji kodu [4].

Zalety C++:

- **Wysoka wydajność**: C++ jest wynikiem kilku istotnych cech języka, które łączą niskopoziomowe zarządzanie zasobami z możliwością precyzyjnej optymalizacji kodu [1].
- **Elastyczność i kontrola**: Programowanie w C++ daje twórcom pełną kontrolę nad kodem źródłowym i pozwala na precyzyjną optymalizację poszczególnych elementów gry [1].
- **Wsparcie Unreal Engine**: C++ jest w pełni wspierany przez Unreal Engine, co umożliwia dostęp do zaawansowanych funkcji i optymalizacji bezpośrednio w silniku [1].

Unreal Engine zapewnia także także alternatywną metodę tworzenia elementów rozgrywki jakim jest wizualny system skryptowania Blueprint Visual Scripting. Umożliwia on definiowanie klas obiektowych i obiektów w silniku. System, wraz z obiektami, które zdefiniujesz, często nazywany jest po prostu "Blueprintami".

W projekcie zastosowano Blueprint Visual Scripting jako metodę tworzenia skryptów odpowiedzialnych za funkcjonowanie mechanik ze względu na prostotę implementacji.

3.3. Modelowanie 3D

Modelowanie 3D jest kluczowym elementem w procesie tworzenia gry komputerowej, zapewniając realizm i atrakcyjność wizualną. Na rynku dostępne są różne narzędzia wspierające proces modelowania i animacji, w tym zarówno darmowe, jak i komercyjne oprogramowanie. W ramach realizacji niniejszego projektu większość modeli 3D oraz tekstur została zaimportowana z platform Quixel Bridge oraz Quixel Megascans, które oferują wysokiej jakości fotorealistyczne zasoby. Dzięki pełnej integracji tych narzędzi z Unreal Engine możliwe było szybkie importowanie i wdrażanie gotowych elementów, co znacznie usprawniło proces tworzenia świata gry i zapewniło jego wysoki poziom wizualny. Pozostała część modeli 3D, wymagających indywidualnego podejścia lub modyfikacji, została zaprojektowana przy użyciu Blendera. To darmowe narzędzie open-source okazało się niezastąpione w realizacji unikalnych elementów projektu, takich jak specjalnie zaprojektowane obiekty czy dostosowanie materiałów. Blender pozwolił również na edycję animacji i zapewnił pełną kontrolę nad wyglądem niestandardowych modeli.
Dodatkowo projekt korzystał z zasobów dostępnych na platformie Fab od Epic Games, co pozwoliło na dalsze uzupełnienie detali otoczenia oraz zwiększenie różnorodności dostępnych modeli.

Blender to darmowe oprogramowanie do modelowania, animacji i renderowania 3D, które jest kompatybilne z Unreal Engine i umożliwia tworzenie realistycznych modeli [5]. Blender powstał w 1994 roku jako narzędzie do użytku wewnętrznego w firmie NeoGeo, a jego twórcą był Ton Roosendaal [2]. W 2002 roku Blender stał się oprogramowaniem open-source, co doprowadziło do dynamicznego rozwoju i rozbudowy funkcji dzięki społeczności użytkowników [5].

Zalety Blendera:

- **Darmowy dostęp**: Blender jest oprogramowaniem typu open-source, co czyni go dostępnym i opłacalnym rozwiązaniem [5].
- **Wsparcie dla Unreal Engine**: Modele zaprojektowane w Blenderze można łatwo eksportować do Unreal Engine, co pozwala na bezproblemowe dodanie ich do projektu [5].
- **Bogate narzędzia do teksturowania i animacji**: Blender umożliwia tworzenie realistycznych tekstur, materiałów i animacji, co zwiększa jakość wizualną gry [5].

Warto wspomnieć o komercyjnej alternatywie dla Blendera – oprogramowaniu Autodesk Maya. To zaawansowane narzędzie do modelowania i animacji 3D jest szeroko stosowane

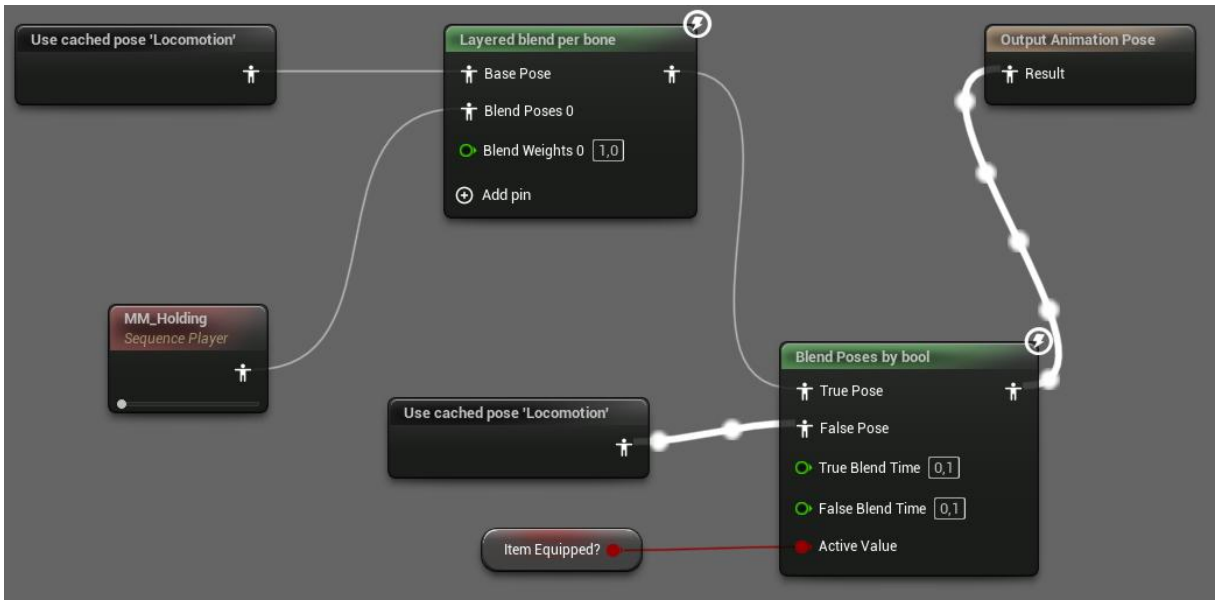
w przemyśle gier i filmów. Maya oferuje bogaty zestaw funkcji do tworzenia szczegółowych modeli oraz realistycznych animacji, co czyni ją standardem w dużych produkcjach.

4. Implementacja

Implementacja mechanik w grze survivalowej opiera się na wykorzystaniu silnika Unreal Engine i systemu Blueprint, który pozwala na tworzenie zaawansowanych funkcjonalności w sposób wizualny. W tej części opisano kluczowe mechaniki i systemy zastosowane w projekcie.

4.1. System trzymania przedmiotów

Na rysunku 4.1. przedstawiono fragment wizualnego skryptu w silniku Unreal Engine, który jest odpowiedzialny za zmianę stanu postaci na stan trzymania przedmiotu. Jest to element który jest kluczowy do późniejszych interakcji z narzędziami dostępnymi dla postaci.



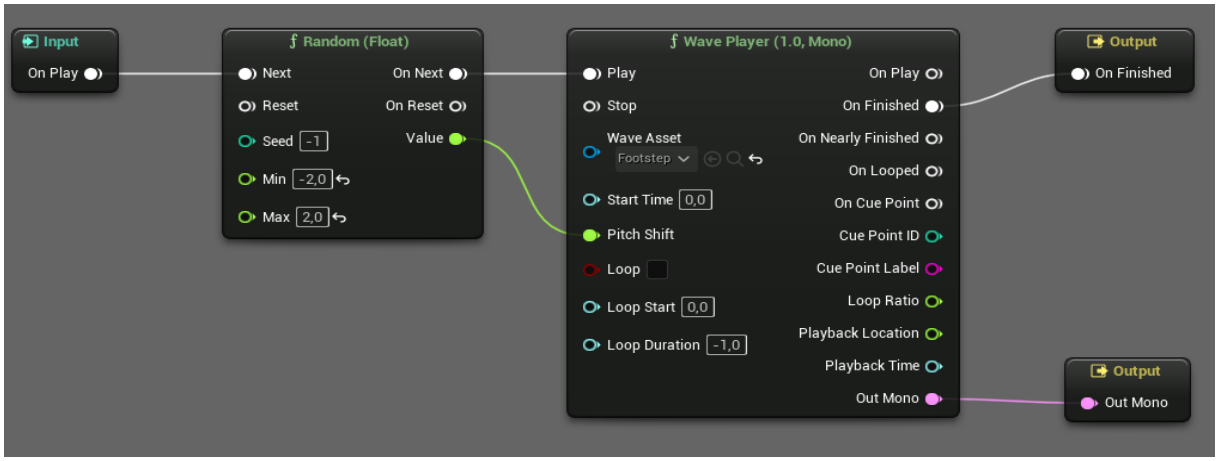
Rys 4.1. Blueprint trzymanie przedmiotu

System odpowiedzialny za trzymanie przedmiotu składa się z następujących elementów:

- *MM_Holding* - blok ten odpowiada za zapętlenie animacji trzymania przedmiotu.
- *Layered blend per bone* - blok, który odpowiada za miksowanie animacji w różnych częściach postaci. W tym przypadku szkielet postaci jest podzielony na dwie części jedna na którą przypada animacja trzymania przedmiotu druga wykorzystująca animację stanu ruchu.
- *Blend Poses by bool* - blok który pozwala na zmianę stanu w zależności od zmiennej boolean.

4.2. Dźwięk ruchu postaci

Poniższy rysunek (rysunek 4.2.) przedstawia mechanizm wywołania i głośności dźwięku ruchu postaci. Mechanizm ten jest analogicznie wykorzystywany w projekcie w momencie wywołania innych dźwięków np. dźwięków przeciwników.



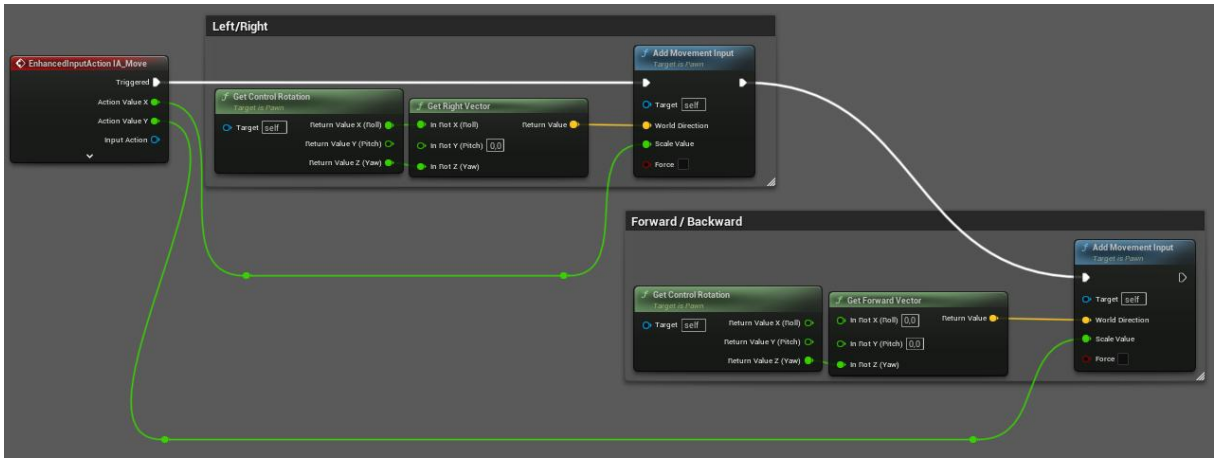
Rys 4.2. Blueprint dźwięku ruchu postaci

Skrypt wizualny odpowiedzialny za wywołanie dźwięku postaci składa się z następujących elementów:

- *Wave Player* - blok ten odpowiada za uruchomienie dźwięku.
- *Random (Float)* - blok, który odpowiada za przypisanie losowej wartości liczbie która jest później przypisana jako głośność odtwarzanego dźwięku.

4.3. Ruch postaci

Na rysunku 4.3. widać skrypt Blueprint, który jest odpowiedzialny za poruszanie się postaci.



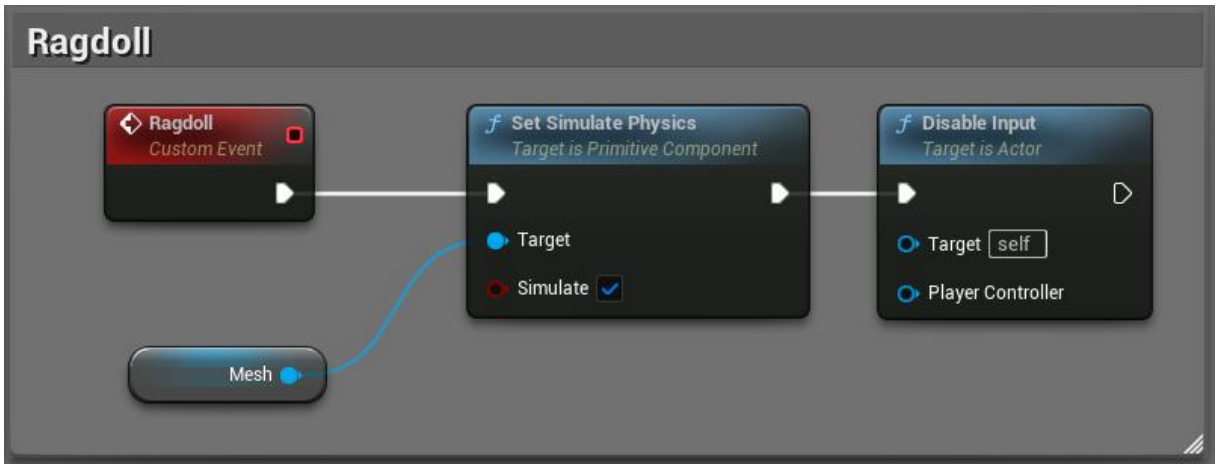
Rys 4.3. Blueprint ruchu postaci

Skrypt odpowiedzialny za poruszanie się postaci składa się z następujących elementów:

- *EnhancedInputAction IA_Move* - blok ten definiuje akcję ruchu i wiąże ją z odpowiednimi klawiszami.
- *Get Control Rotation* - blok, który zwraca rotację kontrolera gracza, pozwala na uzyskanie dokładnej orientacji w przestrzeni 3D, w którą stronę patrzy gracz.
- *Get (Right/Forward) Vector* - blok, który zwraca wektor wskazujący (w prawo/ w przód) względem rotacji kontrolera (kamery). Wektor ten wskazuje kierunek, w którym postać patrzy.
- *Add Movement Input* - blok, który jest odpowiedzialny za faktyczne poruszanie postaci w grze, bazując na wektorach określających kierunek ruchu. Używa on kierunków (np. wektorów uzyskanych z *Get Forward Vector* i *Get Right Vector*) oraz wartości wejściowych (np. naciśniętych klawiszy).

4.4. Mechanika ragdoll

Na poniższym rysunku (rysunek 4.4.) widnieje fragment skryptu, który jest odpowiedzialny za funkcjonowanie mechaniki fizyki ragdoll. Jest to popularna mechanika stosowana jako zamiennik statycznej animacji śmierci postaci.



Rys 4.4. Blueprint mechaniki fizyki ragdoll

Skrypt odpowiedzialny za mechanikę ragdoll składa się z następujących elementów:

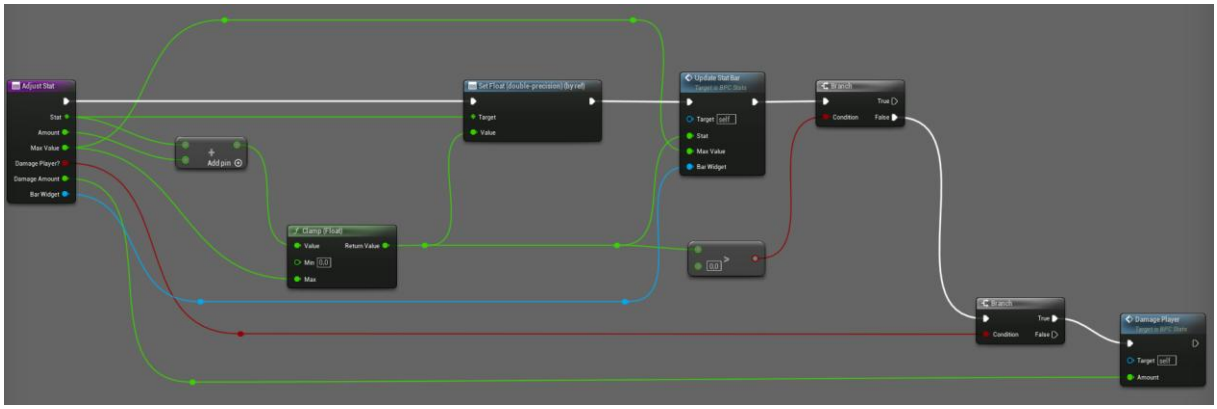
- *Mesh* - blok odpowiedzialny za przekazywanie wartości zmiennej Mesh postaci.
- *Set Simulate Physics* - blok odpowiedzialny za aktywację symulacji fizyki dla przekazanej wartości w tym przypadku postaci.
- *Disable Input* - blok odpowiedzialny za zablokowanie danych wejściowych przychodzących od gracza.

4.5. System zarządzania statystykami postaci

System zarządzania statystykami postaci jest kluczowym elementem mechaniki gry, umożliwiającym dynamiczne dostosowanie parametrów takich jak zdrowie, energia czy wytrzymałość. Statystyki te wpływają na trudność rozgrywki, a ich zmiany są wizualizowane w interfejsie użytkownika (UI), co pozwala graczowi monitorować stan postaci i podejmować odpowiednie decyzje strategiczne.

4.5.1. Funkcja dostosowania statystyk postaci

Na rysunku 4.5. widnieje fragment wizualnego skryptu w silniku Unreal Engine, który jest odpowiedzialny za funkcjonowanie funkcji *adjuststat* odpowiedzialnej za zwiększanie i zmniejszanie statystyk postaci. Jest to jeden z najważniejszych elementów który wpływa na poziom trudności gry



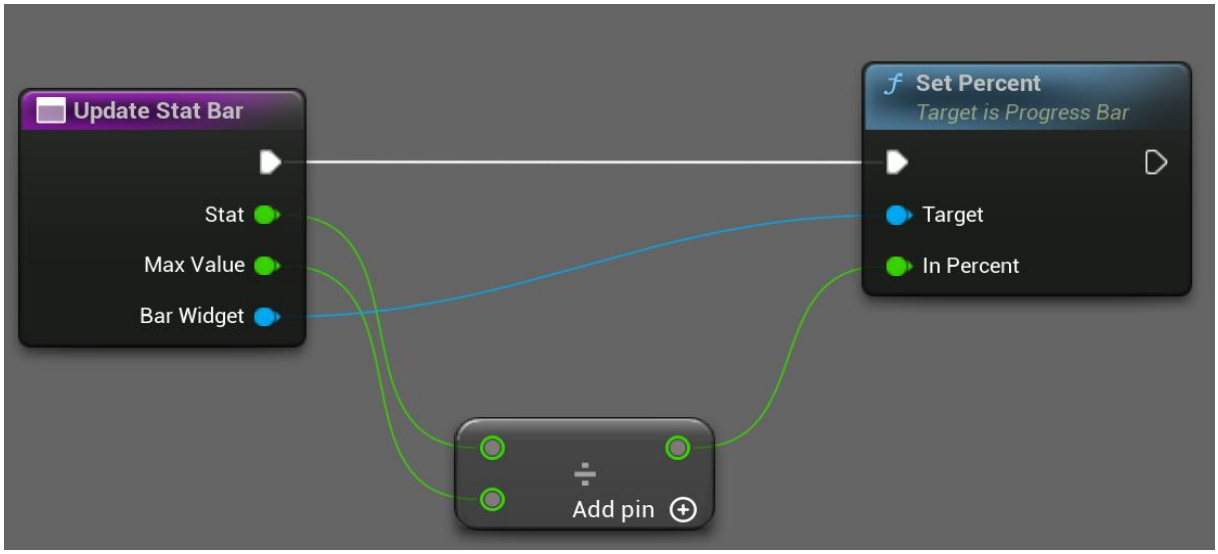
Rys 4.5. Blueprint dostosowania statystyk postaci

Skrypt odpowiedzialny za modyfikacje statystyk postaci składa się z następujących elementów:

- "+" - blok odpowiedzialny za inkrementację lub dekrementację którego działanie jest uzależnione od podanej wcześniej wartości amount.
- *Clamp(Float)* - blok odpowiedzialny za zwrócenie wartości znajdującej się między parametrami.
- ">" - blok odpowiedzialny za zwrócenie wartości w momencie gdy zmienna jest większa niż oznaczona w bloku.
- *Set Float(double precision)(by ref)* - blok odpowiedzialny za aktualizowanie wartości zmiennej bez potrzeby tworzenia kopii zmiennej.
- *Update Stat Bar* - blok odpowiedzialny za aktualizowanie paska statystyk.
- *Damage Player* - blok odpowiedzialny za wywołanie eventu zadającego obrażenia graczowi

4.5.2. Funkcja aktualizująca pasek statystyk w ui

Na rysunku 4.6. widać skrypt, który jest odpowiedzialny za funkcjonowanie metody *Update Stat Bar* odpowiedzialnej za zaktualizowanie statystyk w ui. Element ten znacząco pomaga graczowi ustalić obecny stan postaci i determinuje styl rozgrywki.



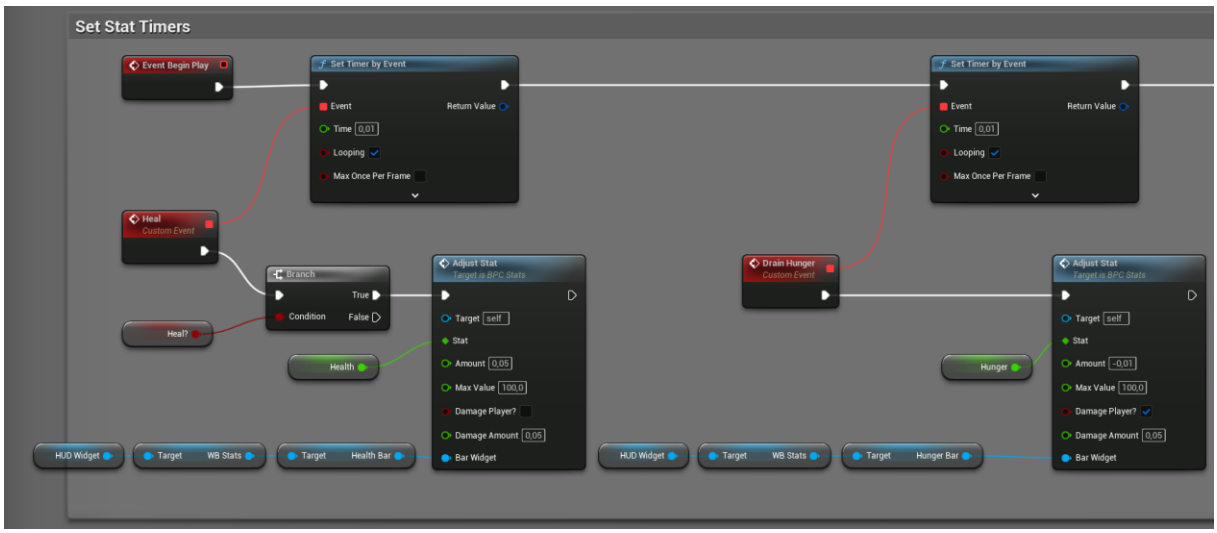
Rys 4.6. Blueprint aktualizowania statystyk w ui

System odpowiedzialny za modyfikacje statystyk postaci w ui składa się z następujących elementów:

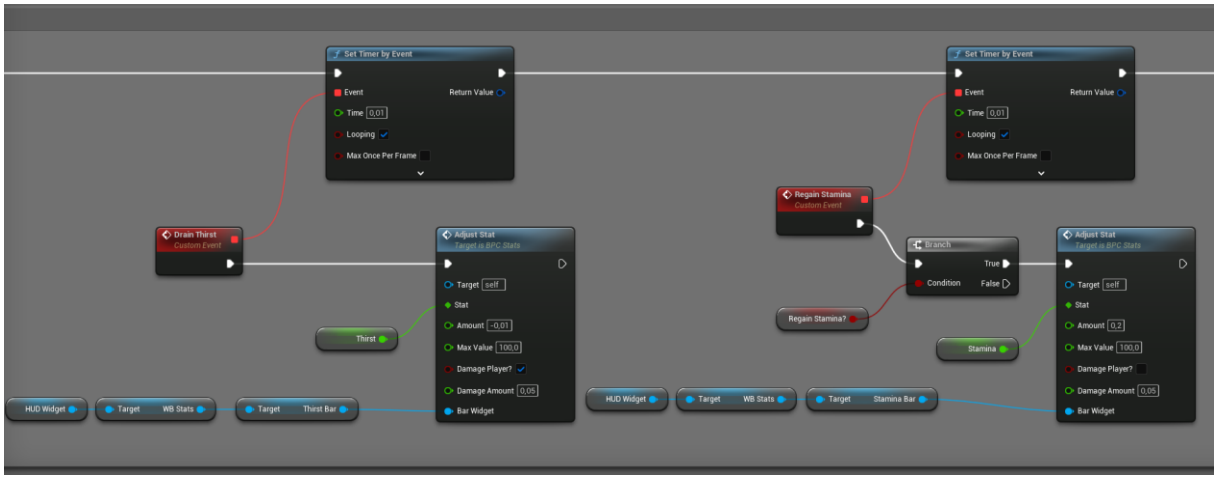
- *Set Percent* - blok odpowiedzialny za aktualizowanie procentu paska statystyk przekazywanego referencją z bloku *Update Stat Bar*.
- "÷" - blok który ma za zadanie podzielić obecnie przekazywaną wartość statystyki przez wartość maksymalną i przekazać ją dalej.

4.6. Set Stat Timer

Rysunek 4.7. i rysunek 4.8. przedstawiają wizualny skrypt w silniku Unreal Engine, który jest odpowiedzialny za zwiększanie i zmniejszanie statystyk postaci w czasie. Mechanizm ten pozwala na dynamiczną zmianę statystyk postaci w krótkich interwałach czasu dzięki czemu osiągany jest efekt płynnej zmiany wartości statystyki w ui.



Rys 4.7. Blueprint zwiększania i zmniejszania statystyk postaci w czasie, część 1



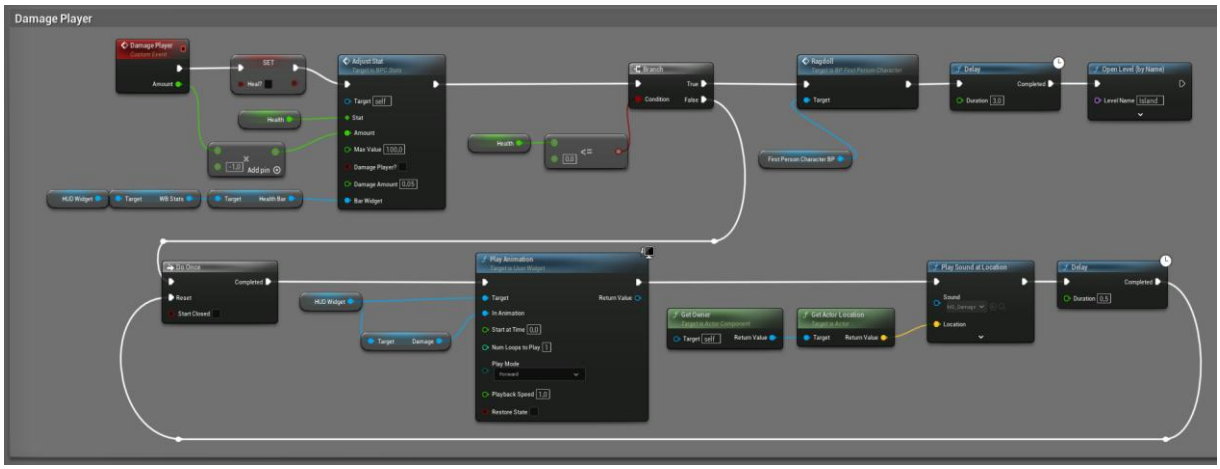
Rys 4.8. Blueprint zwiększania i zmniejszania statystyk postaci w czasie, część 2

Skrypt odpowiedzialny za modyfikacje statystyk postaci w czasie składa się z następujących elementów:

- *Set Timer By Event* - blok który umożliwia wykonanie kodu w określonych interwałach czasu w pętli.
- *Heal* - blok odpowiedzialny za zwiększenie poziomu życia postaci.
- *Drain Hunger* - blok odpowiedzialny za zmniejszenie poziomu głodu.
- *Drain Thirst* - blok odpowiedzialny za zmniejszenie poziomu nawodnienia.
- *Regain Stamina* - blok odpowiedzialny za zwiększenie poziomu wytrzymałości postaci.
- *Adjust Stat* - blok odpowiedzialny za wywołanie funkcji *Adjust Stat*.

4.7. Damage Player

Na poniższym rysunku (Rysunek 4.9.) widać skrypt, który jest odpowiedzialny za funkcjonowanie mechanizmu zadawania obrażeń postaci. Mechanizm ten determinuje obecny poziom punktów życia analizuje czy postać się leczy i wywołuje mechanikę ragdoll w momencie spadku zdrowia do zera.



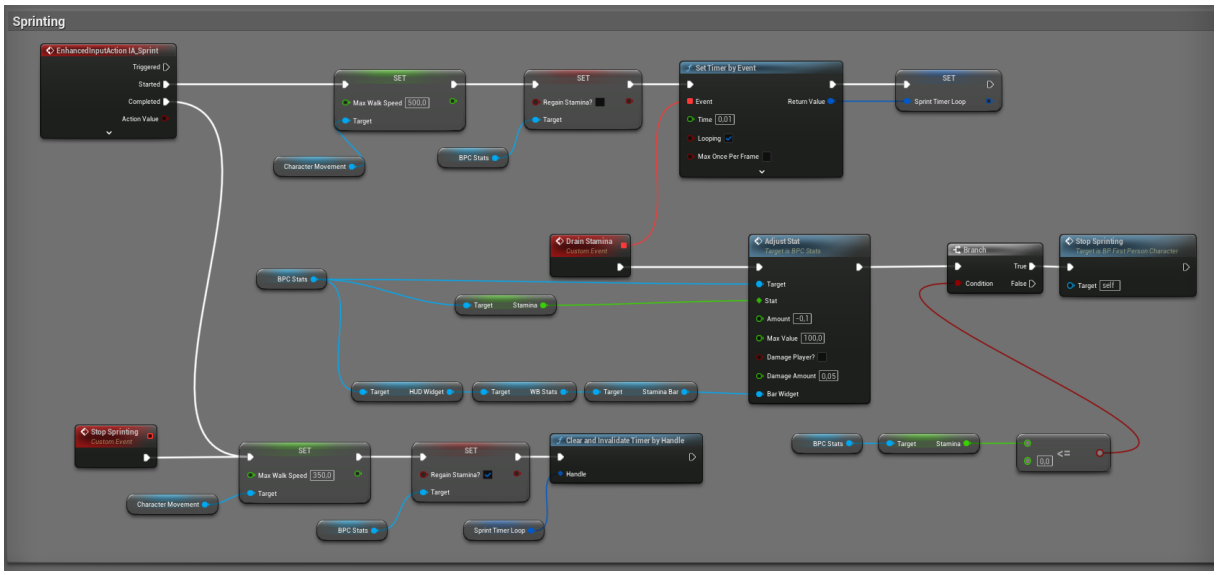
Rys 4.9. Blueprint mechanizmu zadawania obrażeń postaci

Blueprint odpowiedzialny za mechanizm zadawania obrażeń postaci składa się z następujących elementów:

- *SET* - ustawia wartość heal na fałsz(bazowo) postać nie jest leczona w tym momencie.
- *Adjust Stat* - blok odpowiedzialny za wywołanie funkcji Adjust Stat w tym przypadku odpowiedzialnej za zmniejszenie poziomu życia postaci.
- "x" - blok który ma za zadanie pomnożyć obecnie przekazywaną wartość przez wartość liczbę -1 w celu otrzymania wartości którą przekażemy funkcji adjust stat jako otrzymane obrażenia.
- *Delay* - blok odpowiedzialny za ustawienie wartości opóźnienia wykonania następnej akcji.
- *Open Level (by Name)* - blok który zresetuje level świata po odczekaniu czasu opóźnienia (delay).
- "<=" - blok który ma za zadanie sprawdzić czy obecny poziom życia jest większy niż 0.
- *Ragdoll* - blok który wywołuje działanie funkcji ragdoll.
- *Do Once* - blok który ma za zadanie odpalić następujące bloki po nim jednokrotnie niezależnie od innych bloków wpływających na częstotliwość odświeżania funkcji
- *Play Animation* - blok wywołujący przekazaną do niego animację w tym przypadku otrzymania obrażeń
- *Play Sound at Location* - blok wywołujący dźwięk w danym miejscu na mapie. W tym przypadku przypisany do postaci

4.8 Sprint

Na rysunku 4.10. widnieje skrypt odpowiedzialny za funkcjonowanie mechanizmu sprintu postaci.



Rys 4.10. Blueprint mechanizmu sprintowania postaci

System odpowiedzialny za mechanizm sprintu postaci składa się z następujących elementów:

- *SET* - ustawia wartość zmiennych Max Walk Speed i RegainStamina?
- *Set Timer by Event* - blok który umożliwia wykonanie kodu w określonych interwałach czasu w pętli.
- *Adjust Stat* - blok odpowiedzialny za wywołanie funkcji Adjust Stat w tym przypadku odpowiedzialnej za zmniejszenie poziomu wytrzymałości postaci.
- "<=" - blok który ma za zadanie sprawdzić czy obecny poziom wytrzymałości jest większy niż 0.
- *Clear and Invalidate Timer by Event* - kończy działanie bloku set timer by event.

5. Interfejs użytkownika

Rys 5.1. Menu główne

Na rysunku 5.1. widnieje główne menu zaprojektowane z myślą o prostocie i czytelności, co zapewnia użytkownikowi intuicyjne korzystanie z aplikacji już od pierwszego uruchomienia. Interfejs zawiera dwa kluczowe elementy:

- **Przycisk „Start”** – po naciśnięciu tego przycisku gracz zostaje przeniesiony do menu wyboru mapy. W tym miejscu użytkownik może wybrać poziom, na którym chce rozpocząć rozgrywkę, co pozwala na dostosowanie doświadczenia do własnych preferencji lub poziomu umiejętności
- **Przycisk „Options”** – służy do otwierania menu opcji, w którym gracz może dostosować ustawienia graficzne gry.
- **Przycisk „Exit"** - służy do wyjścia z gry. Po jego wybraniu aplikacja zostaje zamknięta, co jest wygodnym rozwiązaniem, jeśli gracz chce szybko zakończyć sesję.

Estetyka interfejsu jest minimalistyczna, z naciskiem na kontrastowe elementy: jasne przyciski na ciemnym tle. Prosty układ i brak zbędnych elementów sprawiają, że interfejs głównego menu jest czytelny i funkcjonalny, pozwalając graczowi szybko rozpocząć rozgrywkę lub wyjść z gry bez konieczności przeszukiwania wielu opcji.

Rys 5.2. HUD ze statystykami

Interfejs wyświetlany podczas rozgrywki został zaprojektowany z myślą o czytelności i intuicyjnym dostarczaniu graczowi kluczowych informacji o stanie postaci. Rysunek 5.2 obejmuje cztery główne wskaźniki statystyk: życie, głód, nawodnienie oraz wytrzymałość, które umożliwiają graczowi szybkie reagowanie na zmieniające się warunki przetrwania. Każdy wskaźnik jest wyświetlany w formie pionowego paska w lewej dolnej części ekranu gry. Kolory wskaźników ułatwiają ich rozróżnienie, a animacje ostrzegają gracza, gdy poziom statystyk zbliża się do krytycznego.

1. **Wskaźnik życia (Health)**
   Pasek życia, oznaczony czerwonym kolorem, pokazuje aktualny stan zdrowia postaci. Zmniejsza się w wyniku otrzymanych obrażeń, takich jak ataki przeciwników, utrata w momencie zerowego poziomu głodu i nawodnienia.
2. **Wskaźnik głodu (Hunger)**
   Pomarańczowy pasek reprezentuje poziom sytości postaci. Obniża się wraz z upływem czasu, co wymusza na graczu poszukiwanie pożywienia. Całkowite wyczerpanie tego wskaźnika może prowadzić do powolnej utraty życia.
3. **Wskaźnik nawodnienia (Hydration)**
   Niebieski pasek wskazuje poziom nawodnienia organizmu. Gracz musi regularnie uzupełniać nawodnienie, korzystając z dostępnych zasobów wody.
4. **Wskaźnik wytrzymałości (Stamina)**
   Zielony pasek obrazuje wytrzymałość postaci. Zużywana jest podczas biegania. Regeneruje

się automatycznie po chwilowym odpoczynku. Zarządzanie wytrzymałością jest kluczowe w sytuacjach wymagających intensywnego wysiłku, np. podczas ucieczki przed przeciwnikami.

Rys 5.3. Ekran ekwipunku

Na rysunku 5.3. widać ekran ekwipunku, który został zaprojektowany w oparciu o siatkę (grid) i sloty na przedmioty, co zapewnia intuicyjne zarządzanie zasobami i narzędziami przez gracza. Dzięki temu rozwiązaniu gracz może w łatwy sposób organizować swoje przedmioty oraz dostosowywać wyposażenie do bieżących potrzeb.

1. **Siatka ekwipunku (Grid)**
   Ekran ekwipunku składa się z prostokątnej siatki podzielonej na sloty, z których każdy może pomieścić jeden przedmiot.
2. **Sloty na przedmioty (Item Slots)**
   Każdy slot w siatce może pomieścić jeden przedmiot lub jego określoną ilość, jeśli przedmiot można układać w stosy (np. kamienie drewno). Sloty zawierają także wizualne oznaczenia ilości przedmiotów, co ułatwia szybkie przeglądanie zasobów.

Rys 5.4. Powiadomienie zdobytego surowca

Na rysunku 5.4. widać system powiadomień o zdobytych surowcach. System został zaprojektowany w sposób czytelny i minimalistyczny, aby szybko i skutecznie dostarczać graczowi informacji o zebranych zasobach, nie przerywając immersji ani nie odciągają uwagi od rozgrywki.

1. **Elementy powiadomienia**

   Powiadomienie składa się z trzech kluczowych elementów:

- Ikona surowca: Reprezentuje rodzaj zdobytego surowca (np. drewno, kamień, jagody). Ikona jest graficznie dopracowana, co umożliwia szybkie rozpoznanie zasobu bez potrzeby czytania jego nazwy.
- Nazwa surowca: Tekstowa nazwa zdobytego surowca wyświetlana obok ikony, np. „Drewno”, „Kamień”.
- Ilość zdobytego surowca: Wyświetlana liczba np. „3”, co jasno wskazuje, ile sztuk danego surowca zostało dodane do ekwipunku.

2. **Wyświetlanie powiadomienia**

- Powiadomienie pojawia się w dolnej lewej części ekranu nad słupkami ze statystykami postaci, tak aby nie przeszkadzać w rozgrywce.

Rys 5.5. Wyposażony przedmiot

Rysunek 5.5. ilustruje element interfejsu odpowiedzialny za prezentację aktualnie używanego przedmiotu został zaprojektowany z myślą o klarowności i szybkim dostępie do informacji, aby gracz zawsze wiedział, z jakim narzędziem lub bronią wchodzi w interakcję.

Wyświetlacz składa się z dwóch kluczowych komponentów:

- **Ikona przedmiotu**: Reprezentuje aktualnie używany przedmiot (np. młotek, siekiera, łuk). Ikona jest graficznie szczegółowa i intuicyjnie kojarzona z danym narzędziem lub bronią.
- **Nazwa przedmiotu**: Wyświetlana obok lub pod ikoną. W czytelny sposób informuje o nazwie przedmiotu, np. ,,Pickaxe".

6. Testowanie gry

Poniższy rozdział poświęcony został testom manualnym, które mają na celu weryfikację stanu gry, interakcji gracza z otoczeniem oraz poprawności animacji. Weryfikacji poddano kluczowe mechaniki rozgrywki, takie jak zbieranie zasobów czy interakcje z przedmiotami na mapie. Celem testów jest sprawdzenie, czy wszystkie aspekty rozgrywki działają zgodnie z założeniami projektowymi.

6.1. Scenariusze testowe

W tej części opisano scenariusze testowe przygotowane w celu weryfikacji poprawności działania kluczowych mechanik gry. Każdy scenariusz zawiera kroki testowe, warunki początkowe oraz oczekiwane rezultaty, umożliwiając systematyczną ocenę poprawności działania gry. Dzięki testom manualnym możliwe jest szybkie wykrycie potencjalnych problemów i weryfikacja zgodności rozgrywki z założeniami projektowymi.

Tabela 6.1. Scenariusz testowy TC001: Test ruchu postaci - chodzenie, bieganie, skakanie

ID	TC001
Tytuł	Test ruchu postaci: chodzenie, bieganie, skakanie.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Sprawdź animacje gracza dla każdego z kierunków obracając postać przyciskami WASD. 2. Sprawdź animacje gracza dla skoku używając klawisza Spacja samodzielnie lub w połączeniu z klawiszami kierunkowymi WASD. 3. Sprawdź animacje gracza dla biegu używając klawisza Lewy Shift samodzielnie lub w połączeniu z klawiszami kierunkowymi WASD.
Oczekiwany rezultat	Animacja wykonuje się dla każdego ruchu postaci: chodzenie, bieganie, skakanie.

Scenariusz TC001 weryfikuje poprawność animacji ruchu postaci w grze, testując różne kombinacje ruchów, w tym chodzenie, bieganie oraz skakanie. Dzięki temu możliwe jest wykrycie ewentualnych problemów związanych z animacjami.

Tabela 6.2. Scenariusz testowy TC002: Test kolizji postaci z obiektami świata

ID	TC002
Tytuł	Test kolizji postaci z obiektami świata.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Podejdź do danego obiektu np. drzewa,skały, koralu. 2. Spróbuj przejść przez obiekt.
Oczekiwany rezultat	Postać gracza poprawnie wchodzi w interakcję z obiektami świata i nie jest w stanie przez nie przechodzić.

Scenariusz TC002 weryfikuje poprawność systemu kolizji w grze. Test sprawdza, czy gracz nie może przechodzić przez obiekty środowiska, takie jak drzewa, skały czy korale, co pozwala upewnić się, że fizyka gry działa zgodnie z założeniami projektowymi.

Rys 6.1. Kolizja postaci z drzewem

Tabela 6.3. Scenariusz testowy TC003: Test interakcji z przedmiotami możliwymi do zebrania z otoczenia

ID	TC003
Tytuł	Test interakcji z przedmiotami możliwymi do zebrania z otoczenia.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Podejdź do danego obiektu, który można zebrać np. gałęzie, kamienie. 2. Naciśnij przycisk E w celu zebrania przedmiotu. 3. Sprawdź czy przedmiot został dodany do twojego ekwipunku naciskając przycisk I i porównując obecny stan ekwipunku ze stanem poprzednim.
Oczekiwany rezultat	Postać gracza jest w stanie zbierać przedmioty z otoczenia.

Scenariusz TC003 służy do weryfikacji mechaniki zbierania przedmiotów przez postać gracza. Test sprawdza, czy przedmioty znajdujące się w otoczeniu, takie jak gałęzie czy kamienie, można poprawnie dodać do ekwipunku. Działanie tej funkcji jest kluczowe dla systemów rozgrywki związanych z zarządzaniem zasobami.

Rys 6.2. Drewno przedmiot możliwy do zebrania z otoczenia

Rys 6.3. Stan ekwipunku przed zebraniem przedmiotu

Rys 6.4. Stan ekwipunku po zebraniu przedmiotu

Tabela 6.4. Scenariusz testowy TC004: Test działania ekwipunku postaci i wybierania przedmiotów w nim dostępnych

ID	TC004
Tytuł	Test działania ekwipunku postaci i wybierania przedmiotów w nim dostępnych.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Sprawdź możliwość otworzenia panelu ekwipunku przyciskając I. 2. Najedź kursorem na przedmiot, który twoja postać może wyposażyć np. Kilof. 3. Przyciśnij lewym przyciskiem myszy aby wyposażyć przedmiot. 4. Otwórz ponownie ekwipunek przyciskając I. 5. Zamknij ekwipunek przyciskając I.
Oczekiwany rezultat	Postać gracza jest w stanie posługiwać się ekranem ekwipunku i wybierać przedmioty, a także je trzymać w ręku.

Scenariusz TC004 weryfikuje poprawność działania systemu ekwipunku. Test sprawdza, czy gracz może otwierać panel ekwipunku, wybierać przedmioty i wyposażać postać w wybrane narzędzia. Jest to kluczowy element mechaniki gry, umożliwiający zarządzanie przedmiotami i przygotowanie postaci do eksploracji otoczenia.

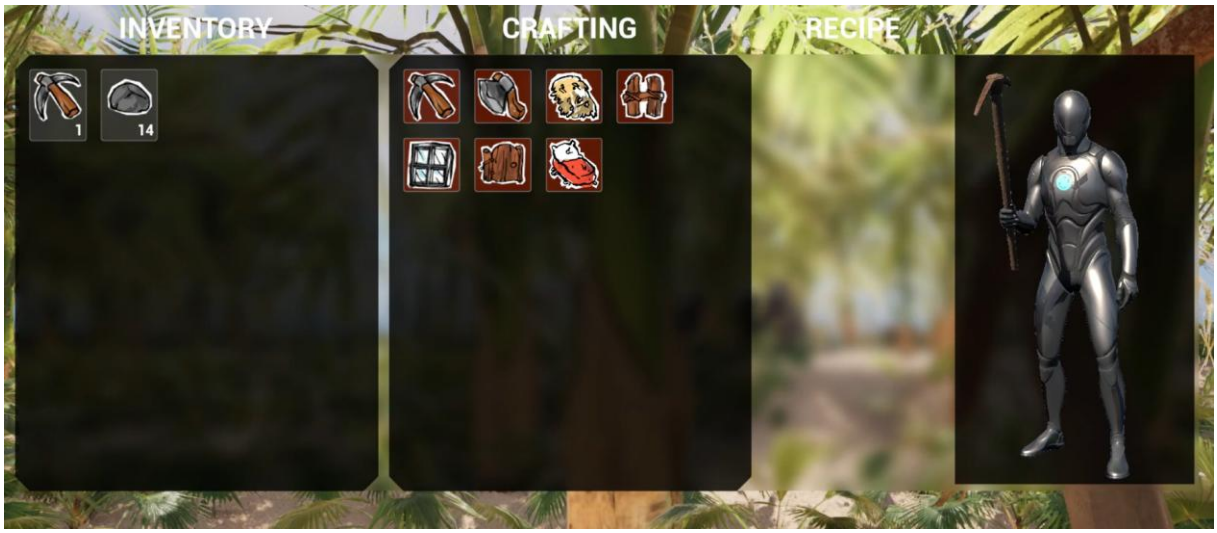
Rys 6.5. Wybór przedmiotu do wyposażenia w ekwipunku

Rys 6.6. Wyposażony przedmiot w ekwipunku

Tabela 6.5. Scenariusz testowy TC005: Test wydobywania zasobów z otoczenia

ID	TC005
Tytuł	Test wydobywania zasobów z otoczenia.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Otwórz ekwipunek naciskając przycisk I. 2. Wybierz narzędzie np. pickaxe. 3. Podejdź do zasobu wydobywanego z otoczenia np. drzewa. 4. Przyciskaj lewy przycisk myszy aż do zniknięcia drzewa. 5. Otwórz ponownie ekwipunek i porównaj ilość drewna obecnie do wartości sprzed ścięcia drzewa.
Oczekiwany rezultat	Postać gracza jest w stanie wydobywać zasoby z otoczenia posługując się narzędziami.

Scenariusz TC005 weryfikuje mechanikę wydobywania zasobów z otoczenia za pomocą narzędzi. Test sprawdza, czy gracz może skutecznie korzystać z narzędzi, takich jak kilof, oraz czy zasoby, takie jak drewno, są poprawnie dodawane do ekwipunku po ich wydobyciu. Mechanika ta jest kluczowym elementem gry, wspierającym system craftingu i progresję rozgrywki.



Rys 6.7. Stan ekwipunku przed zebraniem przedmiotu

Rys 6.8. Proces wydobywania zasobu (drewna)

Rys 6.9. Wydobyty zasób - usunięcie zasobu wydobywanego drzewa z mapy

Tabela 6.6. Scenariusz testowy TC006: Test wykrywania postaci przez przeciwnika i walki

ID	TC006
Tytuł	Test wykrywania postaci przez przeciwnika i walki.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Otwórz ekwipunek przyciskiem I i wyposaż narzędzie np. pickaxe. 2. Znajdź na mapie przeciwnika. 3. Podejdź do niego. 4. Przeciwnika powinien zacząć cię atakować. 5. Przyciskaj lewy przycisk myszy aż przeciwnik zginie.
Oczekiwany rezultat	Postać gracza jest wykrywana przez przeciwników, a gracz jest w stanie z nimi walczyć.

Scenariusz TC006 testuje interakcję gracza z przeciwnikami w grze, w tym mechanizmy wykrywania gracza oraz możliwość walki. Test sprawdza, czy przeciwnicy poprawnie reagują na obecność gracza i inicjują atak, oraz czy gracz może pokonać przeciwników za pomocą dostępnych narzędzi. Mechanizm ten jest kluczowy dla elementów walki i wyzwań w rozgrywce.

Rys 6.10. Zmiana wyglądu ekranu przy odniesieniu obrażeń

Rys 6.11. Efekt ataku na przeciwniku

Tabela 6.7. Scenariusz testowy TC007: Test działania mechanizmu śmierci postaci

ID	TC007
Tytuł	Test działania mechanizmu śmierci postaci.
Warunki początkowe	Gra jest uruchomiona.
Kroki testowe	1. Nie rób nic i poczekaj aż paski statystyk głodu i nawodnienia postaci osiągną 0 i postać zacznie tracić życie. 2. Gdy poziom życia osiągnie 0 gra się skończy i zrestartuje. 3. Podejdź do przeciwnika. 4. Czekaj pasywnie aż jego atak doprowadzi twój pasek życia do poziomu 0.
Oczekiwany rezultat	Postać gracza może zginąć po utracie życia w wyniku braku nawodnienia i głodu, a także w wyniku obrażeń odniesionych przez przeciwników.

Scenariusz TC007 weryfikuje poprawność działania mechanizmu śmierci postaci. Test obejmuje różne warunki prowadzące do śmierci, takie jak brak nawodnienia i głodu oraz obrażenia odniesione w walce. Weryfikacja poprawności zakończenia gry i restartu po śmierci postaci jest istotna dla zapewnienia spójności i wyzwań w rozgrywce.

Rys 6.12. Zmiana wyglądu ekranu przy utracie życia z braku jedzenia

7. Podsumowanie i wnioski

Celem niniejszej pracy dyplomowej było zaprojektowanie i zaimplementowanie gry survivalowej zbudowanej na silniku Unreal Engine 5, która łączy kluczowe mechaniki przetrwania, eksploracji, tworzenia przedmiotów i struktur oraz spójny i intuicyjny interfejs użytkownika. Projekt obejmował pełny cykl tworzenia gry, począwszy od koncepcyjnych założeń projektowych, przez implementację funkcjonalności, aż po testowanie. Finalny produkt spełnia wszystkie założenia określone na początku pracy, co czyni go w pełni funkcjonalnym i spełniającym wymagania.

W ramach realizacji projektu:

- **Zaprojektowano interfejs użytkownika**, w tym HUD (Heads-Up Display) wyświetlający statystyki postaci, takie jak życie, głód, nawodnienie i wytrzymałość, system ekwipunku oparty na siatce (grid) ułatwiający zarządzanie zasobami, a także powiadomienia o zdobytych zasobach i aktualnie używanych przedmiotach.
- **Zaimplementowano mechaniki gry**, takie jak zbieranie zasobów, zarządzanie zasobami, tworzenie przedmiotów i struktur oraz interakcje z przeciwnikami.
- **Wykonano fotorealistyczny świat gry** z unikalną estetyką, wykorzystując zaawansowane funkcje Unreal Engine 5, w tym Lumen (oświetlenie globalne) i Nanite (renderowanie geometrii wysokiej jakości), oraz modele dostępne do użycia z Quixel Bridge przygotowane w Blenderze.
- **Przeprowadzono testy** mające na celu zapewnienie stabilności rozgrywki i zoptymalizowanie jej działania na różnych urządzeniach.

Projekt dostarczył wartościowe doświadczenia w zakresie programowania w Blueprintach i C++, projektowania gier, modelowania 3D, a także optymalizacji wydajności. Realizacja projektu pozwoliła na praktyczne wykorzystanie zdobytej wiedzy oraz rozwój umiejętności związanych z tworzeniem gier. Praca dostarczyła również inspiracji do dalszego eksplorowania możliwości Unreal Engine 5 oraz technologii związanych z tworzeniem interaktywnych aplikacji 3D.

Bibliografia

Źródła książkowe:

[1] Stroustrup, B. (2013). The C++ Programming Language (4th Edition). Addison-Wesley Professional

[2] Roosendaal, T. (2002). Blender: A Comprehensive Guide to 3D Modeling, Animation, and Rendering. NeoGeo Press.

Źródła internetowe:

[3] Unreal Engine 5 Overview - Unreal Engine

The most powerful real-time 3D creation tool - Unreal Engine

[4] C++ Programming Language - W3Schools

C++ Tutorial

[5] Blender 3D Software - Blender

blender.org - Home of the Blender project - Free and Open 3D Creation Software

[6] Unity Game Development Engine: A Technical Survey - Afzal Hussain, Haad Shakeel, Faizan Hussain, Nasir Uddin, Turab Latif Ghouri.

Unity-Game-Development-Engine-A-Technical-Survey.pdf

[7] Features - Godot Engine

https://godotengine.org/features

Spis tabel

Spis rysunków