Kierunek: ***Informatyka***

2024/2025

Konrad Wszołek

PRACA INŻYNIERSKA

Gra wyścigowa z przeszkodami

Promotor pracy:

mgr inż. Tomasz Gądek

Tarnów, *2025*

Spis treści

1. Wstęp

Gry komputerowe odgrywają coraz większą rolę w świecie współczesnej rozrywki i technologii. Wraz z dynamicznym rozwojem narzędzi do tworzenia gier, takich jak Unreal Engine 5, możliwości projektowania realistycznych oraz wciągających produkcji stają się coraz większe. Inspiracja do stworzenia gry wyścigowej z przeszkodami wynika z rosnącego zapotrzebowania na interaktywne i angażujące produkcje, które jednocześnie oferują rozrywkę oraz wymagają od graczy zręczności i strategicznego myślenia.

1.1 Cel projektu

Podstawowym celem projektu jest stworzenie zręcznościowej gry wyścigowej przeznaczonej dla jednego gracza. Głównym elementem rozgrywki jest umiejętne balansowanie pomiędzy unikaniem przeszkód a zarządzaniem czasem. Wyzwaniem dla gracza jest przejechanie wybranego toru w określonym limicie czasu. Na trasie znajdują się liczne przeszkody, które należy omijać, oraz bramki czasowe, które po ich przekroczeniu dodają określoną ilość sekund do pozostałego czasu. Celem gry jest ukończenie toru przed upłynięciem limitu czasu, co wymaga zręczności i poznania mapy.

1.2 Zakres pracy

Zakres pracy obejmuje wszystkie etapy tworzenia gry, począwszy od projektowania koncepcji, przez implementację mechanik, aż po testowanie oraz ocenę wyników. Kluczowe elementy pracy to:

1. **Projektowanie i modelowanie środowiska gry** – wykorzystanie technologii Megascans do tworzenia realistycznych map i obiektów.
2. **Implementacja mechaniki rozgrywki** – opracowanie logiki gry przy użyciu systemu blueprintów w Unreal Engine 5.
3. **Testowanie gry** – przeprowadzenie testów w celu oceny grywalności oraz wydajności.
4. **Analiza i podsumowanie wyników** - ocena osiągniętych rezultatów i wyciągnięcie wniosków na przyszłość.

2. Analiza rynku gier wyścigowych

Projekt gry wyścigowej skupia się na dostarczeniu emocjonującej rozgrywki dla pojedynczego gracza, której celem będzie osiągnięcie jak najlepszego wyniku czasowego na zróżnicowanych trasach pełnych przeszkód. Kluczową mechaniką gry będzie pokonywanie toru w wyznaczonym czasie, z możliwością zdobywania dodatkowych sekund poprzez przejazd przez specjalne bramki.

2.1 Analiza rynku i konkurencji

Gra należy do kategorii gier zręcznościowych, skupiających się na mechanice pokonywania torów w ograniczonym czasie. Na rynku można znaleźć tytuły o podobnych założeniach, jak np. *TrackMania*, lecz projektowana gra wyróżnia się uproszczoną mechaniką opartą na jednoosobowej rozgrywce oraz prostym systemie zarządzania czasem. Gra będzie atrakcyjna dla graczy którzy szukają krótkiej, intensywnej rozgrywki z możliwością szybkiego restartu poziomów w razie porażki.

2.2 Wymagania funkcjonalne

- **Interfejs użytkownika (UI)** - Gra musi posiadać przejrzysty interfejs użytkownika, który pozwala na łatwy wybór opcji startu, pauzy, ponownego uruchomienia poziomu oraz wyboru mapy. Podczas rozgrywki na ekranie powinna być wyświetlana liczba pozostałych sekund.
- **Mechanika gry** - Gracz będzie kierował jednym samochodem, którego zadaniem jest jak najszybsze ukończenie trasy pełnej przeszkód. Gracz musi ukończyć tor przed końcem czasu. Na trasie powstaną różnorodne przeszkody, które gracz musi omijać oraz bramki, które pozwalają na zdobycie dodatkowego czasu.
- **Zarządzanie czasem** - Rozgrywka opiera się na ograniczonym czasie. Gracz rozpoczyna z ustaloną liczbą sekund, które mogą zostać wydłużone przez przejazd przez bramki.
- **Trasy i poziomy** - Gra musi oferować różnorodne trasy, zróżnicowane pod względem długości, układu przeszkód i rozmieszczenia bramek czasowych. Każda trasa powinna posiadać inny poziom trudności, tak aby gracze stopniowo mogli zwiększać swoje umiejętności.

- **Fizyka jazdy -** Gra musi zapewniać prostą, ale responsywną mechanikę jazdy, która będzie sprzyjała szybkiej i płynnej rozgrywce. Sterowanie pojazdem powinno być intuicyjne.

2.3 Wymagania niefunkcjonalne

- **Wydajność -** Rozgrywka powinna być płynna, a czas ładowania krótki, aby zapewnić możliwość powtórzenia toru po nieudanym przejeździe.
- **Kompatybilność -** Gra powinna wspierać różne urządzenia wejściowe, takie jak klawiatura, mysz czy pady.
- **Niezawodność -** Gra nie może się zawieszać ani ulegać awariom podczas rozgrywki. Musi zapewniać bezbłędne działanie restartu poziomu.
- **Użyteczność -** Interfejs musi być przejrzysty i intuicyjny, aby gracz mógł łatwo nawigować po opcjach gry oraz szybko wznawiać rozgrywkę.
- **Skalowalność -** Architektura gry powinna pozwalać na łatwe dodawanie kolejnych tras lub wyzwań w przyszłych aktualizacjach, co może przyciągać graczy do dalszej zabawy.

3. Opis wykorzystanych technologii

3.1 Unreal Engine 5

Unreal Engine to zaawansowany silnik gier, stworzony przez firmę **Epic Games**, który służy do tworzenia gier komputerowych, symulacji, aplikacji VR (wirtualnej rzeczywistości). Jest szeroko stosowany zarówno przez duże studia, jak i niezależnych twórców. Silnik ten pozwala na projektowanie gier różnego typu. Unreal Engine 5 oferuje narzędzia i funkcje, które umożliwiają projektowanie realistycznych środowisk i mechanik gry [1].

Kluczowe technologie Unreal Engine 5:

- **Nanite** to technologia, która umożliwia silnikowi renderowanie ogromnych ilości szczegółów bez obciążania zasobów sprzętowych. Twórcy mogą używać wysokiej jakości modeli bez konieczności ich ręcznej optymalizacji. Nanite automatycznie dostosowuje poziom szczegółowości obiektów w zależności od odległości kamery [2].
- **Lumen** to system globalnego oświetlenia, który zapewnia realistyczne efekty świetlne. Umożliwia dynamiczne zmiany oświetlenia oraz symulację odbić światła w czasie rzeczywistym [3].
- **Chaos Physics** to zaawansowany system fizyki, który umożliwia realistyczne symulowanie ruchu, kolizji oraz zniszczeń obiektów [4].
- Wsparcie dla animacji i interfejsu użytkownika.

3.2 Blueprint Visual Scripting

System Blueprint Visual Scripting w Unreal Engine to wizualny język programowania, który wykorzystuje interfejs oparty na węzłach do tworzenia elementów rozgrywki. Oparty na węzłach przepływ pracy zapewnia projektantom szeroki zakres koncepcji i narzędzi skryptowych, które są zazwyczaj dostępne tylko dla programistów. Ponadto specyficzne dla Blueprint znaczniki dostępne w implementacji C++ Unreal Engine zapewniają programistom sposób tworzenia systemów bazowych, które projektanci mogą rozszerzać [5].

Kluczowe cechy Blueprint Visual Scripting:

- Integracja z C++ - w Unreal Engine stanowi jeden z kluczowych elementów, który pozwala na pełne wykorzystanie możliwości obu systemów – wizualnego skryptowania i tradycyjnego programowania.
- Wizualny interfejs skryptowania - Blueprints wykorzystuje graficzny interfejs, który pozwala użytkownikom na tworzenie logiki gry za pomocą "węzłów" (nodes) połączonych za pomocą "kabli" (wires). Każdy węzeł reprezentuje konkretną funkcję, akcję lub zmienną.
- Tworzenie prefabrykatów i wielokrotnego użytku elementów - Dzięki blueprintom można tworzyć tzw. prefabrykaty (prefabs), czyli gotowe do użycia obiekty, które można wielokrotnie wykorzystywać w różnych miejscach gry.
- Intuicyjne i szybkie prototypowanie - W przypadku gry wyścigowej można szybko implementować mechanikę kolizji, przyspieszenia pojazdów czy liczniki czasu bez konieczności pisania skomplikowanego kodu.
- Łatwość użycia i dostępność dla różnych użytkowników - Blueprints zostało zaprojektowane tak, aby było dostępne zarówno dla nowicjuszy, jak i doświadczonych twórców. Dzięki intuicyjnemu interfejsowi graficznemu, użytkownicy mogą łatwo nauczyć się korzystać z tego narzędzia.
- Rozbudowany system zdarzeń i komunikacji - Blueprints pozwala na łatwą obsługę zdarzeń, takich jak kliknięcia przycisków, kolizje, czy interakcje z obiektami w grze.
- Dostosowywanie interfejsu użytkownika - Blueprints może być używane do tworzenia interaktywnych elementów interfejsu użytkownika (UI), takich jak menu. Dzięki systemowi UMG (Unreal Motion Graphics), twórcy mogą łatwo projektować i animować elementy UI, które są w pełni interaktywne.

3.3 Quixel Bridge i Megascans

Quixel Bridge to platforma umożliwiająca bezpośredni dostęp do ogromnej biblioteki zasobów **Megascans** – jednych z najbardziej realistycznych modeli i tekstur dostępnych na rynku. Megascans oferuje szeroką gamę fotorealistycznych materiałów, które są wynikiem skanów rzeczywistych obiektów i powierzchni. Dzięki tej technologii mogę tworzyć otoczenie o wysokim poziomie szczegółowości i realizmu, co w przypadku gry wyścigowej pełnej różnorodnych tras i przeszkód znacznie podnosi jakość doświadczenia wizualnego [6].

Kluczowe cechy Quixel Bridge:

- Dostęp do Quixel Megascans – Obszernej biblioteki zasobów 3D.
- Bezproblemowa integracja z Unreal Engine - Quixel Bridge jest w pełni zintegrowany z Unreal Engine, co pozwala na łatwe importowanie zasobów bezpośrednio do projektu jednym kliknięciem. Proces ten automatycznie dostosowuje modele i tekstury do optymalnych ustawień Unreal Engine.
- Biblioteka gotowych materiałów PBR (Physically Based Rendering) - Quixel Bridge dostarcza materiały PBR, które realistycznie reagują na oświetlenie i inne efekty wizualne w czasie rzeczywistym.
- Szybkie przeglądanie, zarządzanie i organizacja zasobów.
- Synchronizacja z kontem Quixel i chmura zasobów - Dzięki integracji z kontem Quixel, Quixel Bridge umożliwia synchronizację pobranych zasobów między różnymi urządzeniami.
- Aktualizacje i rozszerzenia bazy zasobów - Quixel Bridge i Quixel Megascans są regularnie aktualizowane, co oznacza, że baza dostępnych zasobów stale się powiększa.

3.4 Epic Games Marketplace

Do zaprojektowania gry wykorzystane zostaną zasoby (assety) dostępne na **Epic Games Marketplace**, aby skrócić czas potrzebny na stworzenie elementów wizualnych, takich jak modele pojazdów i elementy dekoracyjne. Zasoby te służą jako baza, która zostanie dostosowana do specyfiki mojej gry, co pozwala mi skupić się na unikalnych aspektach rozgrywki, takich jak specyficzne przeszkody oraz układ toru [7].

Kluczowe cechy Epic Games Marketplace:

- Bogata biblioteka zasobów - Zasoby dostępne w **Epic Games Marketplace** charakteryzują się wysoką jakością. Można znaleźć modele 3D w fotorealistycznej jakości, szczegółowe tekstury oraz animacje.
- Dostęp do bezpłatnych zasobów - Epic Games Marketplace oferuje również bezpłatne zasoby w każdej kategorii, co pozwala twórcom gier na łatwe i szybkie rozpoczęcie pracy nad projektem, nie ponosząc dodatkowych kosztów.
- Zasoby dostosowane do Unreal Engine - Wszystkie zasoby dostępne w Epic Games Marketplace są zoptymalizowane pod kątem Unreal Engine, co zapewnia ich kompatybilność z systemami renderowania, fizyki i materiałów UE. Dzięki temu integracja zasobów jest prosta i wymaga minimalnych modyfikacji. Marketplace

oferuje także zasoby, które wspierają najnowsze technologie Unreal Engine, takie jak **Nanite**.

- Łatwa integracja i szybkie wdrożenie - Zasoby z Epic Games Marketplace mogą być szybko zintegrowane z projektem, co oszczędza czas twórców. Wystarczy pobrać zasób i zaimportować go do projektu w Unreal Engine. Wiele zasobów jest już przygotowanych do użycia, co umożliwia natychmiastowe wdrożenie w grze bez potrzeby dodatkowej konfiguracji.
- Wsparcie dla różnych typów zasobów - Epic Games Marketplace oferuje szeroki wybór zasobów w różnych kategoriach:
  - Modele 3D,
  - Tekstury i materiały PBR,
  - Animacje,
  - Dźwięki i efekty dźwiękowe,
  - Skrypty i blueprinty.
- Społeczność i wsparcie twórców - Epic Games Marketplace ma silną społeczność twórców, która dzieli się swoimi doświadczeniami, opiniami o zasobach oraz poradami. Twórcy mogą komentować zasoby, zadawać pytania oraz oceniać je.
- Licencjonowanie i prawa do zasobów - Zakupione zasoby w Epic Games Marketplace są objęte licencją, która umożliwia ich używanie w komercyjnych i niekomercyjnych projektach.

4. Implementacja

4.1 Architektura

W ramach implementacji gry wyścigowej z przeszkodami wykorzystano narzędzie Blueprints Visual Scripting dostępne w Unreal Engine. Zastosowanie tego podejścia pozwoliło na szybkie i efektywne tworzenie logiki gry w sposób wizualny, bez konieczności pisania tradycyjnego kodu. Wszystkie kluczowe aspekty mechaniki gry, takie jak działanie pojazdu, obsługa bramek czasowych, wyświetlanie interfejsów użytkownika oraz logika związana z warunkami wygranej i przegranej, zostały zaimplementowane w postaci blueprintów. Dzięki temu możliwe było zachowanie modularności projektu, co ułatwia zarówno debugowanie, jak i dalszy rozwój gry.

Mapy gry zostały zbudowana przy użyciu gotowych zasobów dostępnych na platformach **Epic Games Sore** oraz **Quixel Bridge**. Wykorzystanie zasobów 3D umożliwiło stworzenie estetycznego i realistycznego środowiska rozgrywki, jednocześnie pozwalając na skupienie się na implementacji mechanik. Dostosowanie mapy, w tym rozmieszczenie przeszkód i bramek czasowych, zostało zrealizowane za pomocą edytora poziomów Unreal Engine, co pozwoliło na precyzyjne ustawienie elementów na torze wyścigowym.

Gra została zaprojektowana w sposób modułowy, z podziałem na osobne komponenty odpowiedzialne za różne funkcje. Każdy blueprint realizuje konkretne zadania, takie jak:

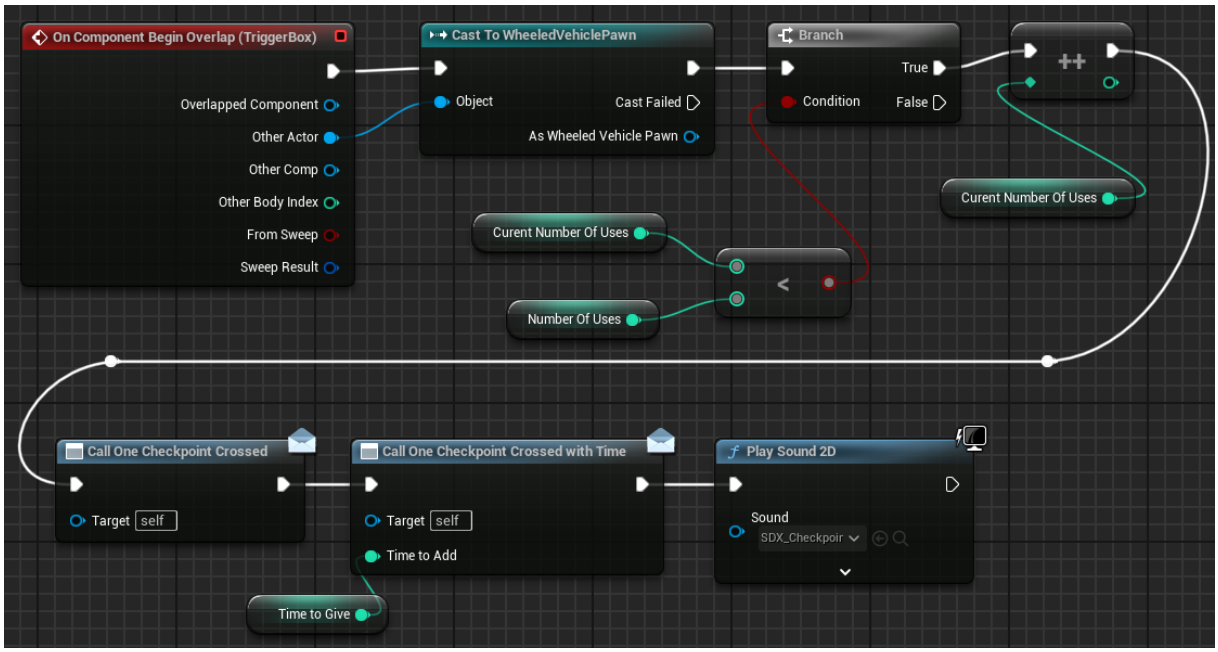
- Mechanika pojazdu: sterowanie, fizyka jazdy, interakcje z przeszkodami.
- Logika bramek czasowych: wykrywanie przejazdu przez bramki, dodawanie czasu do licznika.
- Interfejs użytkownika: główne menu, wyświetlanie licznika czasu, informacji o stanie gry, komunikatów o wygranej/przegranej.
- Zarządzanie grą: kontrolowanie przebiegu rozgrywki, weryfikacja warunków zakończenia gry.

Zastosowanie blueprintów pozwoliło na intuicyjną implementację powyższych mechanik i ich łatwe integrowanie w jedną, spójną całość. W ramach architektury gry zadbano o to, aby każdy element działał niezależnie, ale jednocześnie mógł komunikować się z innymi komponentami za pomocą zdarzeń i interfejsów. W efekcie powstała skalowalna i elastyczna struktura, która

umożliwia łatwe dodawanie nowych funkcjonalności, takich jak dodatkowe przeszkody, czy nowe rodzaje bramek czasowych. Taka architektura sprzyja również szybkiemu prototypowaniu i testowaniu poszczególnych elementów gry.

4.2 Systemu punktów kontrolnych

Rysunek 4.1. przedstawia blueprint w silniku Unreal Engine, który jest odpowiedzialny za dodawanie dodatkowego czasu po przekroczeniu punktu kontrolnego umieszczonego na torze. Mechanizm ten pozwala graczowi przedłużyć czas dostępny na ukończenie toru.



Rys. 4.1. Skrypt odpowiedzialny za działanie bramek czasowych.

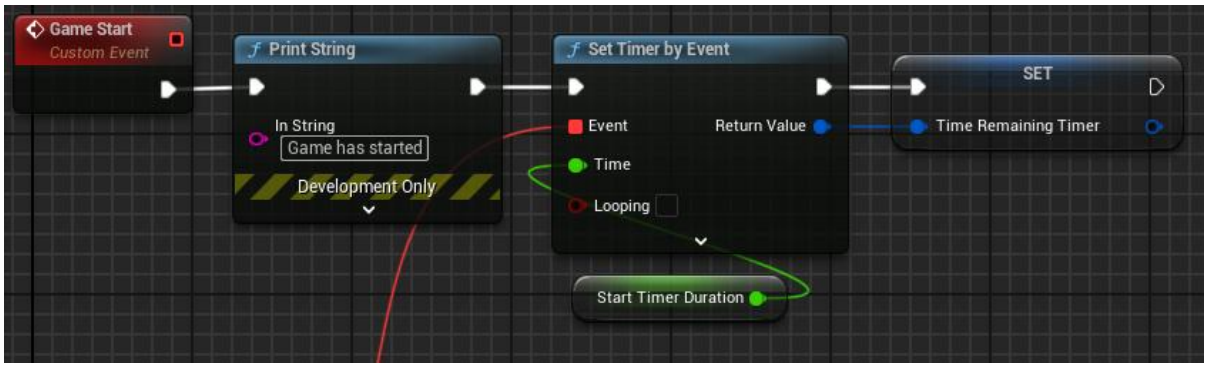
Opis poszczególnych elementów:

- *On Component Begin Overlap (TriggerBox)* odpowiada za wyzwalanie całej logiki punktu kontrolnego. Zdarzenie *On Component Begin Overlap* jest aktywowane, gdy dowolny obiekt przekroczy wyznaczony obszar kolizyjny.
- *Cast to WheeledVehiclePawn* sprawdza czy obiekt przekraczający strefę kolizyjną jest częścią pojazdu.
- *Branch (True/False)* wprowadza warunek sprawdzający, czy punkt kontrolny został już wykorzystany. Warunek ten zapobiega wielokrotnemu aktywowaniu tej samej bramki.

- Inkrementacja zmiennej *Current Number of Uses* w przypadku pozytywnego spełnienia warunku (True), licznik *Current Number of Uses* jest zwiększany o 1. Ta operacja aktualizuje liczbę użyć dla danego punktu, co zapobiega wielokrotnej aktywacji tej samej bramki.
- *Call One Checkpoint Crossed with Time* odpowiada za dodanie określonej ilości czasu, zapisanej w zmiennej *Time to Give*, do aktualnego limitu czasowego gracza.
- *Play Sound 2D* odpowiada za odtwarzanie dźwięku w chwili przekroczenia bramki przez gracza.

4.3 Start Gry

Rysunek 4.2. przedstawia fragment blueprintu odpowiedzialny za rozpoczęcie gry w momencie, gdy gracz uruchamia rozgrywkę. Skrypt ten jest aktywowany przez zdarzenie niestandardowe, które inicjuje sekwencję ustawiającą licznik czasu oraz sygnalizującą rozpoczęcie rozgrywki.



Rys. 4.2. Skrypt rozpoczynający grę i odliczanie czasu.

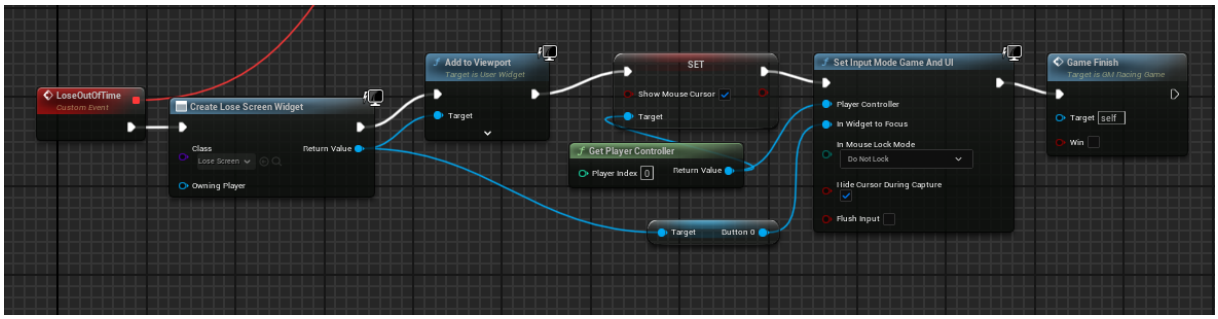
Opis poszczególnych elementów:

- *Game Start (Custom Event)* to niestandardowe zdarzenie, które uruchamia całą logikę startową gry.
- *Print String* wyświetla komunikat tekstowy "Game has started" w konsoli wyłącznie na potrzeby programowania.
- *Set Timer by Event* to funkcja ustawiająca timer, który rozpoczyna odliczanie czasu gry. Czas odliczania jest ustawiony za pomocą wartości z *Start Timer Duration*, a timer jest zaplanowany tak, aby zakończyć działanie po osiągnięciu zera.
- *Start Timer Duration* to wartość określająca ilość czasu, który ma zostać ustawiony na starcie.

- *SET Time Remaining Timer* - dzięki tej zmiennej timer może być później łatwo wyłączony w momencie, gdy gracz osiągnie metę lub czas osiągnie limit.

4.4 Ekran Przegranej

Rysunek 4.3. przedstawia fragment blueprintu odpowiedzialny za wyświetlenie ekranu przegranej w momencie, gdy graczowi skończy się czas na ukończenie toru. Skrypt ten jest aktywowany poprzez zdarzenie niestandardowe i zapewnia przejście do ekranu końca gry, wyłączając graczowi możliwość dalszego sterowania pojazdem.



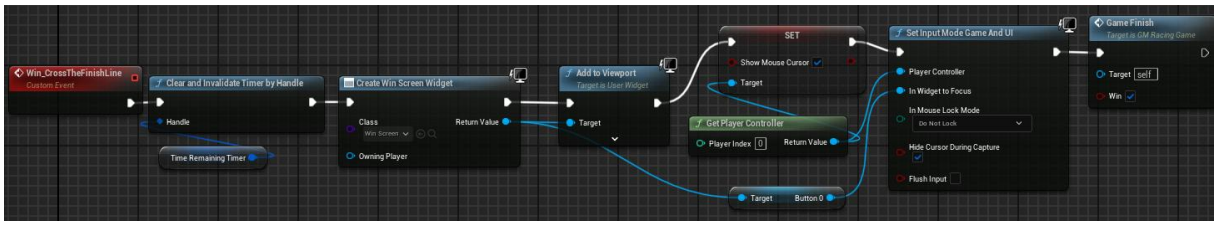
Rys. 4.3. Skrypt wyświetlający ekran przegranej.

Opis poszczególnych elementów:

- *Lose Out Of Time* to zdarzenie niestandardowe które jest wywoływane w momencie, gdy graczowi zabraknie sekund. Uruchamia ono całą logikę odpowiedzialną za ekran przegranej.
- *Create Lose Screen Widget* po wywołaniu zdarzenia tworzony jest widget odpowiedzialny za wyświetlenie ekranu przegranej.
- *Add to Viewport* - Dzięki temu ekran przegranej jest widoczny dla gracza i stanowi informację o zakończeniu rozgrywki.
- *SET Show Mouse Cursor* ustawia widoczność kursora myszy, aby umożliwić graczowi interakcję z ekranem przegranej, np. ponowne rozpoczęcie gry.
- *Get Player Controller* pozwala na chwilowe przełączenie się z trybu rozgrywki na tryb obsługi ekranu końca gry.
- *Set Input Mode Game and UI* pozwala na przełączenie trybu sterowania na interfejs gry oraz UI.
- *Game Finish* sygnalizuje zakończenie gry w wyniku przegranej. Funkcja ta wykonuje niezbędne operacje końcowe, jak zakończenie odliczania czasu i dezaktywacja elementów rozgrywki.

4.5 Ekran Wygranej

Rysunek 4.4. przedstawia fragment blueprintu odpowiedzialny za wyświetlenie ekranu wygranej w momencie, gdy gracz pomyślnie ukończy tor przed upływem czasu. Skrypt ten jest aktywowany przez niestandardowe zdarzenie, które uruchamia logikę zakończenia rozgrywki, przełączając gracza na ekran końca gry i dając mu możliwość interakcji z opcjami interfejsu.



Rys. 4.4. Skrypt wyświetlający ekran wygranej.

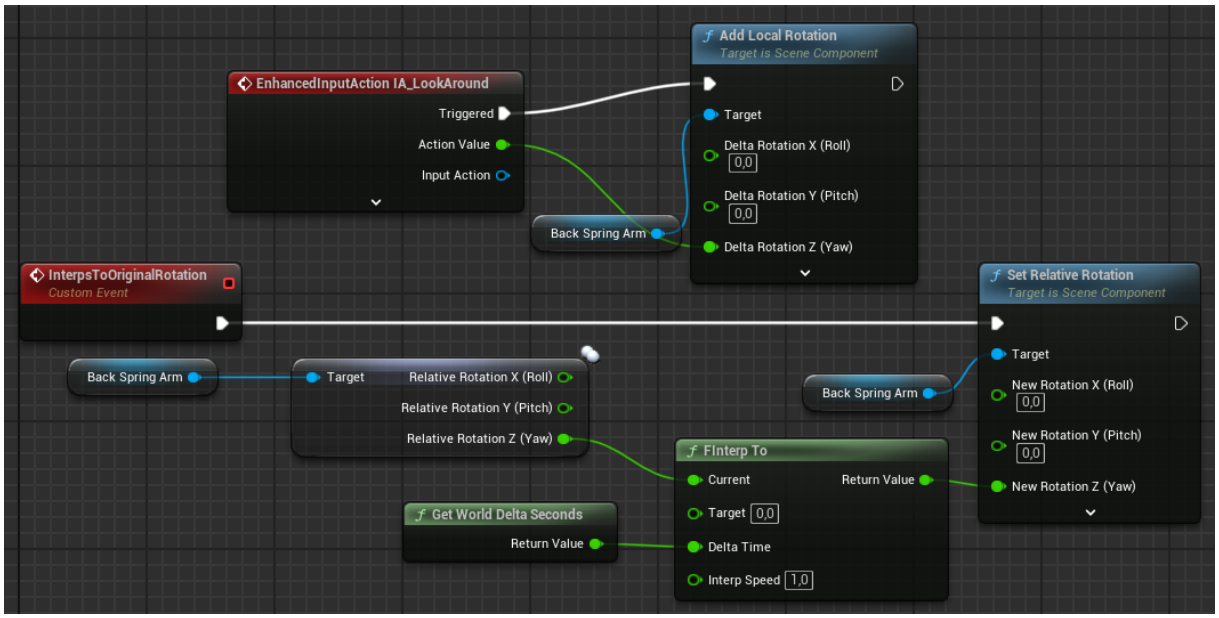
Opis poszczególnych elementów:

- *Win_CrossTheFinishLine* to zdarzenie niestandardowe, które wyzwala całą logikę związaną z ekranem wygranej. Zdarzenie jest wywoływane w momencie, gdy gracz dotrze do mety przed końcem czasu.
- *Clear and Invalidate Timer by Handle* - funkcja ta zatrzymuje i unieważnia timer odliczający pozostały czas na ukończenie toru.

Pozostałe elementy działają na tej samej zasadzie co w fragmencie odpowiedzialnym za przegraną.

4.6 Kontrola kamery

Rysunek 4.5. przedstawia blueprint odpowiedzialny za obsługę kamery w grze. Kamera automatycznie wyrównuje się do pozycji wyjściowej z biegiem czasu, zapewniając graczowi płynne i intuicyjne sterowanie. Skrypt obsługuje zarówno ręczne poruszanie kamerą za pomocą wejścia gracza, jak i powrót do oryginalnej orientacji, gdy wejście zostanie zwolnione.



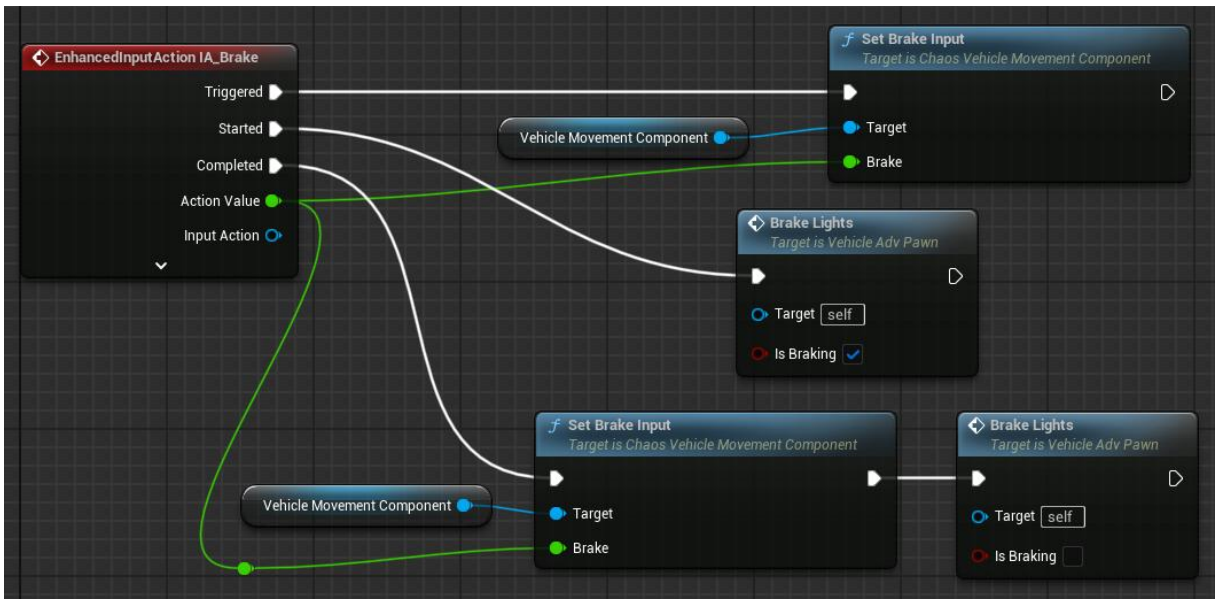
Rys. 4.5. Skrypt kontrolujący ruch kamery.

Opis poszczególnych elementów:

- *EnhancedInputAction IA_LookAround* to akcja wejściowa przypisana do ruchu kamery w grze (np. poruszanie kamerą za pomocą myszy).
- *Add Local Rotation* dodaje zmianę rotacji w osi Y (obrót w poziomie) do komponentu *Back Spring Arm* w odpowiedzi na dane wejściowe z akcji *IA_LookAround*. Pozwala to graczowi obracać kamerę wokół postaci.
- *InterpsToOriginalRotation* to niestandardowe zdarzenie odpowiedzialne za stopniowy powrót kamery do oryginalnej pozycji, gdy gracz przestaje nią poruszać.
- *Back Spring Arm* pobiera aktualną rotację w osiach X, Y i Z, która następnie jest używana do interpolacji w celu płynnego wyrównania.
- *FInterp To* stopniowo wyrównuje rotację kamery, zapewniając płynny i naturalny powrót do pozycji wyjściowej.
- *Get World Delta Seconds* jest używany w obliczeniach interpolacji w celu zapewnienia, że ruch kamery jest płynny i niezależny od liczby klatek na sekundę.
- *Set Relative Rotation* przypisuje wynik funkcji interpolacji do rotacji kamery, zapewniając jej wyrównanie do pozycji początkowej w czasie rzeczywistym.

4.7 Sterowanie hamulcami

Rysunek 4.6. przedstawia blueprint odpowiedzialny za obsługę hamulców w pojeździe. W momencie aktywacji akcji hamowania, skrypt zapewnia zastosowanie siły hamowania oraz aktywację świateł stopu, co zwiększa realizm i immersję rozgrywki.



Rys. 4.6. Skrypt odpowiedzialny za działanie hamulców.

Opis poszczególnych elementów:

- *EnhancedInputAction IA_Brake* odpowiada za rozpoczęcie i zakończenie procesu hamowania w odpowiedzi na dane wejściowe od gracza:
  - Started – Rozpoczyna działanie hamulców.
  - Completed – Zatrzymuje działanie hamulców po zwolnieniu przycisku.
- *Set Brake Input* ustawia wartość siły hamowania dla pojazdu. Zapewnia natychmiastowe spowolnienie pojazdu po aktywacji akcji hamowania.
- *Vehicle Movement Component* to komponent odpowiadający za kontrolę ruchu pojazdu. Przekazuje informacje o działaniu hamulców do funkcji *Set Brake Input*, umożliwiając zastosowanie siły hamowania w silniku Chaos.
- *Brake Lights* odpowiada za aktywację lub dezaktywację świateł stopu w zależności od stanu hamowania.

  Parametry:
  - *Target* odnosi się do pojazdu, w którym światła stopu mają zostać aktywowane.
  - *Is Braking* decyduje o włączeniu lub wyłączeniu świateł stopu.

5. Interfejsy

5.1 Menu główne

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

Rys. 5.1. Menu główne

- **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 „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.

5.2 Interfejs HUD - wyświetlanie prędkości i biegu

Kluczowym elementem interfejsu heads-up display (HUD) jest sekcja informująca o aktualnej prędkości pojazdu oraz zapiętym biegu przedstawiona na rysunku 5.2. Wyświetlacz ten znajduje się w dolnej części ekranu, co pozwala graczowi na szybkie zorientowanie się w sytuacji bez odrywania uwagi od trasy:

Rys. 5.2. Wyświetlacz prędkości i biegu

- **Prędkość pojazdu** – prezentowana jest w centralnej części wyświetlacza w formie dużych, wyraźnych cyfr. Wartość podawana jest w kilometrach na godzinę (km/h), co jest standardem w większości gier wyścigowych. Rozmiar czcionki zapewnia jej czytelność nawet podczas dynamicznej jazdy.
- **Aktualny bieg** – znajduje się po prawej stronie, nieco mniejszą czcionką niż prędkość, co jednak nadal pozwala na łatwą identyfikację. Wartość ta informuje gracza o bieżącym ustawieniu skrzyni biegów, co ma kluczowe znaczenie dla bardziej zaawansowanej mechaniki jazdy, np. podczas przyspieszania, redukcji biegów czy manewrów na zakrętach.

Układ tych dwóch elementów jest prosty i funkcjonalny, co minimalizuje rozpraszanie uwagi gracza. Dzięki temu może on skupić się na omijaniu przeszkód i pokonywaniu trasy. Wprowadzenie takiego interfejsu podnosi poziom immersji i pozwala lepiej kontrolować pojazd w trakcie rozgrywki.

5.3 Interfejs HUD - licznik czasu

W górnej centralnej części ekranu znajduje się wyświetlacz czasu przedstawiony na rysunku 5.3. Pełni on kluczową rolę w rozgrywce. Jest to element interfejsu *heads-up display* (HUD) informujący gracza o pozostałym czasie na ukończenie trasy. Jego obecność podkreśla dynamikę rozgrywki i wprowadza dodatkowy element wyzwania.

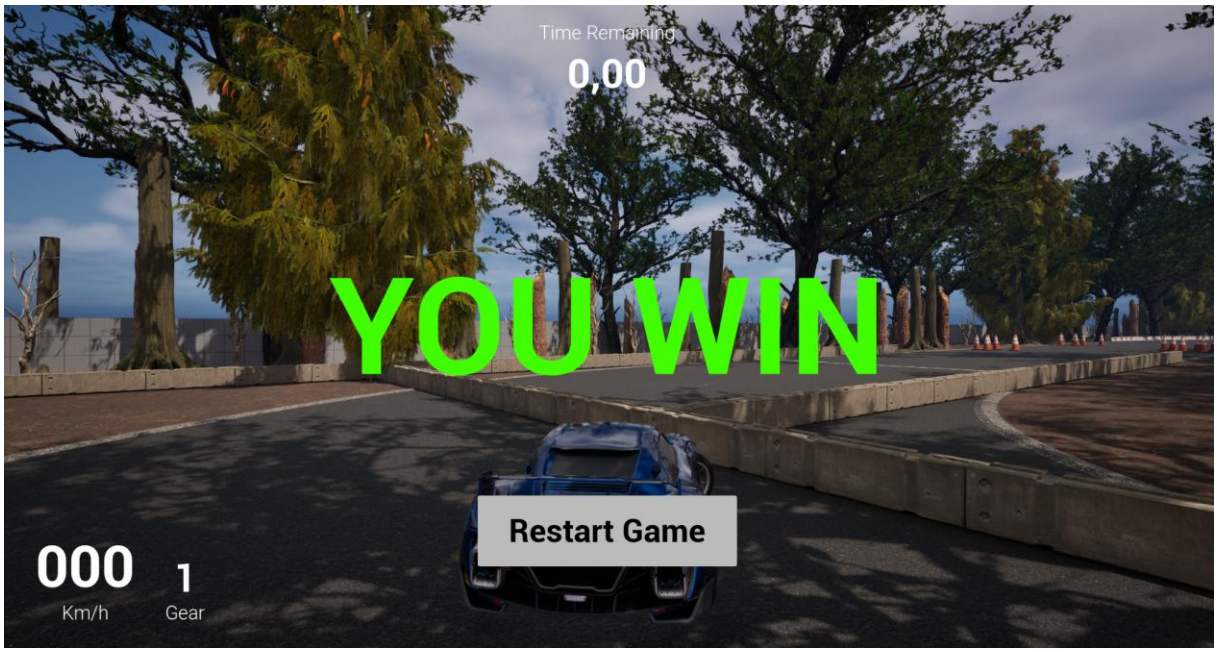
Rys. 5.3. Licznik czasu

- **Napis „Time Remaining”** – w górnej części wyświetlacza znajduje się czytelny opis funkcji, dzięki czemu gracz od razu wie, że prezentowana wartość dotyczy czasu pozostałego na ukończenie toru.
- **Pozostały czas** – umieszczony jest centralnie i prezentowany dużymi, wyraźnymi cyframi z dokładnością do setnych części sekundy (format „0,00”). Rozmiar umożliwia odczytanie wartości nawet podczas intensywnej rozgrywki.

Ten element HUD jest szczególnie istotny w kontekście wyzwań czasowych, które zmuszają gracza do efektywnego zarządzania tempem jazdy i podejmowania szybkich decyzji. Po wyzerowaniu licznika gra zostaje przerwana. Centralne umieszczenie wyświetlacza sprawia, że jest on stale widoczny, ale nie zasłania pola widzenia gracza.

5.4 Ekran Wygranej

Ekran wygranej widoczny na rysunku 5.4. jest prosty, ale wyrazisty, mający na celu poinformowanie gracza o pomyślnym ukończeniu trasy. Jest to element podsumowujący sukces gracza i oferujący możliwość kontynuowania rozgrywki:



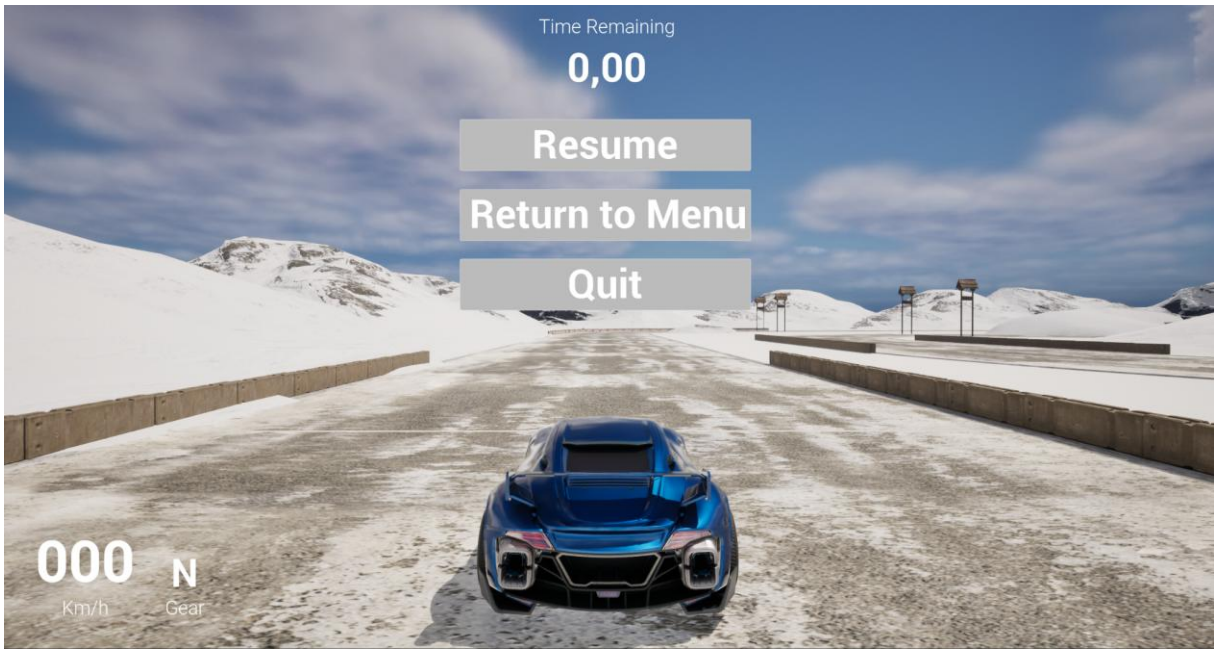
Rys. 5.4. Ekran Wygranej

- **Napis „You Win”** – zajmuje centralną część ekranu i jest kluczowym elementem wizualnym. Wykonany dużą, czytelną czcionką, ma na celu przyciągnięcie uwagi i podkreślenie triumfu gracza.
- **Przycisk „Restart Game”** – znajduje się poniżej napisu i umożliwia graczowi natychmiastowe rozpoczęcie rozgrywki od nowa. Jest to przydatna funkcja, szczególnie dla graczy, którzy chcą poprawić swój wynik lub po prostu powtórzyć zabawę.

Ekran wygranej został zaprojektowany w minimalistycznym stylu, dzięki czemu nie rozprasza użytkownika. Pełni dwie funkcje: nagradza gracza za osiągnięcie celu oraz pozwala mu szybko powrócić do rozgrywki. Jego prostota i przejrzystość sprawiają, że jest zrozumiały i atrakcyjny wizualnie.

5.5 Ekran pauzy

Ekran pauzy ukazany na rysunku 5.5. został zaprojektowany z myślą o wygodzie i szybkości działania, aby umożliwić graczowi chwilowe przerwanie rozgrywki bez wychodzenia z gry. Dzięki minimalistycznemu układowi interfejsu gracz może łatwo i intuicyjnie zarządzać swoimi opcjami w trakcie gry. Interfejs ekranu pauzy zawiera dwa kluczowe elementy:



Rys. 5.5. Ekran pauzy

- **Przycisk „Resume”** – Po naciśnięciu tego przycisku gracz wraca do rozgrywki dokładnie w miejscu, w którym została zatrzymana. Pozwala to na płynne kontynuowanie gry bez zbędnych opóźnień.
- **Przycisk „Return to Menu”** – Umożliwia graczowi powrót do głównego menu gry. Jest to użyteczne rozwiązanie, jeśli gracz zdecyduje się zakończyć bieżącą rozgrywkę lub zmienić mapę.
- **Przycisk „Quit”** – Pozwala graczowi wyłączyć grę bez konieczności powrotu do menu.

Ekran pauzy jest przezroczysty, co pozwala zachować widok gry w tle. Taki projekt sprawia, że użytkownik jest świadomy kontekstu gry, jednocześnie mając możliwość łatwego zarządzania dostępnymi funkcjami.

5.6 Menu wyboru mapy

Rysunek 5.6. przedstawia menu wyboru mapy, które zostało zaprojektowane tak, aby umożliwić graczowi szybki i intuicyjny wybór jednego z dostępnych poziomów gry. Dzięki przejrzystemu układowi i spójnej estetyce, menu to wpisuje się w minimalistyczny charakter interfejsu aplikacji, jednocześnie zapewniając pełną funkcjonalność. Główne elementy interfejsu to:

Rys. 5.6. Menu wyboru mapy

- **Przyciski wyboru mapy** – Menu zawiera trzy przyciski, z których każdy odpowiada jednej mapie do wyboru. Każda mapa reprezentuje inny poziom trudności lub styl rozgrywki, pozwalając graczowi dostosować doświadczenie do swoich preferencji. Przyciski są odpowiednio opisane, aby gracz od razu wiedział, jaką mapę wybiera.
- **Przycisk „Back to Menu”** – Umieszczony na dole ekranu, przycisk umożliwia powrót do głównego menu gry, co pozwala graczowi zrezygnować z wyboru mapy i zdecydować się na inną opcję dostępną w głównym menu.

W tle menu wyboru mapy wyświetlany jest ten sam czerwony samochód, który pojawia się w głównym menu. Dzięki temu interfejs zachowuje spójność wizualną, a estetyka pozostaje atrakcyjna i tematyczna. Samochód jest wyeksponowany na ciemnym tle, co podkreśla jego dynamiczny charakter i nawiązuje do motywu gry.

6. Testowanie gry

W niniejszym rozdziale przedstawione zostały przypadki testowe opracowane dla poszczególnych elementów gry. Testy obejmują takie aspekty, jak: poprawne działanie menu głównego, mechanizmy wyboru map, płynność rozgrywki, reakcje gry na interakcje gracza z bramkami czasowymi i przeszkodami, a także prawidłowe odliczanie czasu oraz system rozstrzygania wygranej i przegranej. Każdy przypadek testowy zawiera kroki do wykonania oraz oczekiwane rezultaty. Testy będą wykonane manualnie, a każdy rozdział prezentuje scenariusze testowe dla testera manualnego.

6.1 Testy menu głównego

Tabela 6.1. Scenariusz testowy poruszania się po menu głównym.

ID	001
Tytuł	Nawigacja po menu głównym.
Warunki początkowe	Gra uruchomiona, menu główne aktywne.
Kroki testowe	1. Użyj myszy do poruszania się po menu.
Oczekiwany rezultat	Gracz może swobodnie poruszać się po menu, a wybrane opcje są wyróżnione wizualnie.

Tabela 6.2. Scenariusz testowy rozpoczęcia rozgrywki na wybranej mapie.

ID	002
Tytuł	Rozpoczęcie gry.
Warunki początkowe	Gra uruchomiona, menu główne aktywne.
Kroki testowe	1. Wybierz opcję "Start". 2. Wybierz mapę z listy.
Oczekiwany rezultat	Gra przechodzi do rozgrywki na wybranej mapie.

Rys. 6.1. Testy menu głównego

6.2 Testy menu wyboru mapy

Tabela 6.3 Scenariusz testowy wyświetlania listy dostępnych map po wybraniu opcji "Start".

ID	003
Tytuł	Lista map.
Warunki początkowe	Gra uruchomiona, menu główne aktywne.
Kroki testowe	1. Wybierz opcję "Start".
Oczekiwany rezultat	Wyświetlana jest lista dostępnych map.

Tabela 6.4 Scenariusz testowy uruchamiania wybranej mapy.

ID	004
Tytuł	Wybór mapy.
Warunki początkowe	Gra uruchomiona, lista map aktywna.
Kroki testowe	1. Wybierz dowolną mapę z listy.
Oczekiwany rezultat	Wybrana mapa zostaje załadowana poprawnie.

Tabela 6.5 Scenariusz testowy dla opcji Back to Menu.

ID	005
Tytuł	Powrót do menu głównego.
Warunki początkowe	Gra uruchomiona, lista map aktywna.
Kroki testowe	1. Wybierz opcję "Back to Menu".
Oczekiwany rezultat	Gra wraca do menu głównego bez błędów.

Rys. 6.2. Testy menu wyboru mapy

6.3 Testy rozgrywki

Tabela 6.6 Scenariusz testowy dla odliczania czasu oraz kontroli nad pojazdem.

ID	006
Tytuł	Start rozgrywki.
Warunki początkowe	Mapa załadowana, licznik czasu widoczny.
Kroki testowe	1. Rozpocznij grę na wybranej mapie.
Oczekiwany rezultat	Gra rozpoczyna odliczanie czasu i umożliwia kontrolę pojazdu.

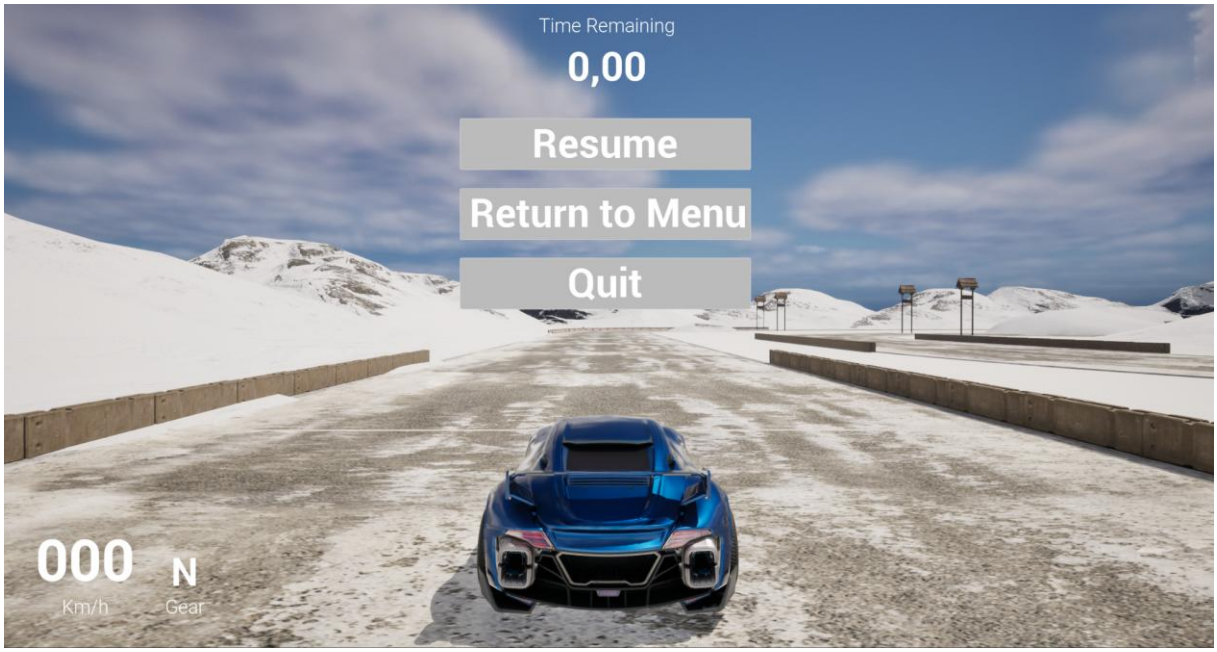
Tabela 6.7. Scenariusz testowy reakcji pojazdu na wciskanie klawiszy odpowiedzialnych za sterowanie.

ID	007
Tytuł	Ruch pojazdu.
Warunki początkowe	Gra uruchomiona, sterowanie aktywne.
Kroki testowe	1. Porusz pojazdem w przód, w tył, w lewo i w prawo za pomocą klawiszy WASD.
Oczekiwany rezultat	Pojazd reaguje poprawnie na polecenia sterowania.

Rys. 6.3. Test rozgrywki

Tabela 6.8. Scenariusz testowy dla menu pauzy.

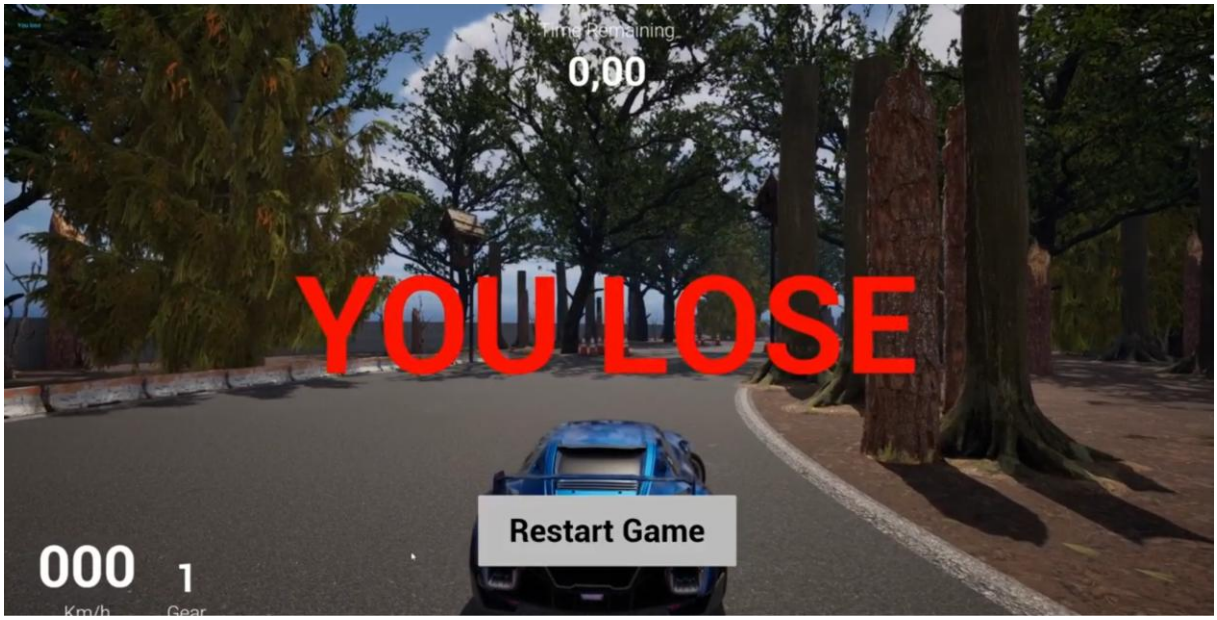
ID	008
Tytuł	Zatrzymanie gry (pauza).
Warunki początkowe	Gra uruchomiona, rozgrywka w toku.
Kroki testowe	1. Naciśnij przycisk pauzy "Escape".
Oczekiwany rezultat	Gra zatrzymuje się, a na ekranie pojawia się menu pauzy.



Rys 6.4. Test menu pauzy

Tabela 6.9. Scenariusz testowy wyświetlania ekranu przegranej.

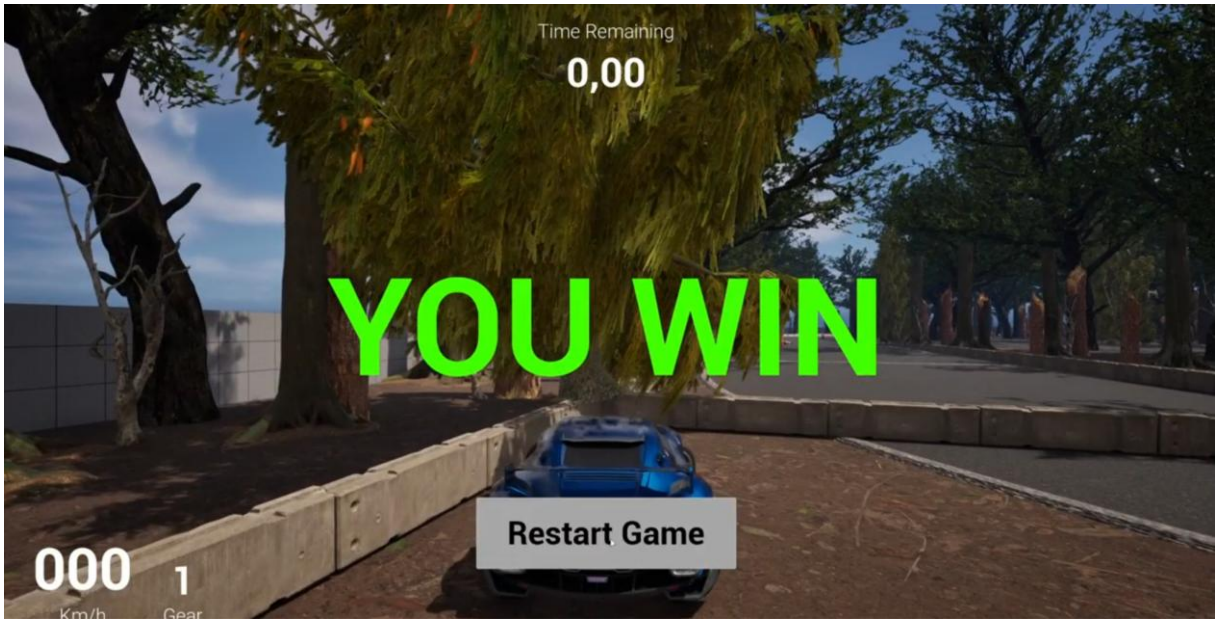
ID	009
Tytuł	Przegrana przez upływ czasu.
Warunki początkowe	Gra uruchomiona, licznik czasu aktywny.
Kroki testowe	1. Nie przejeżdżaj przez bramki czasowe . 2. Poczekaj, aż licznik osiągnie zero.
Oczekiwany rezultat	Gra wyświetla ekran przegranej, gdy czas dobiega końca.



Rys. 6.5. Test ekranu przegranej

Tabela 6.10. Scenariusz testowy wyświetlania ekranu wygranej.

ID	010
Tytuł	Pomyślne ukończenie toru.
Warunki początkowe	Gra uruchomiona, licznik czasu aktywny.
Kroki testowe	1. Rozpocznij jazdę samochodem. 2. Przejeżdżaj przez bramki czasowe. 3. Omijaj przeszkody. 4. Dotrzyj do mety przed końcem czasu.
Oczekiwany rezultat	Gra wyświetla ekran wygranej, licznik czasu się zatrzymuje.



Rys. 6.6. Test ekranu wygranej

6.4 Testy bramki czasowej

Tabela 6.11 Scenariusz testowy przejazdu przez bramkę czasową

ID	011
Tytuł	Przejazd przez bramkę.
Warunki początkowe	Gra uruchomiona, bramka czasowa widoczna na torze.
Kroki testowe	1. Przejedź przez bramkę czasową.
Oczekiwany rezultat	Licznik czasu zwiększa się o odpowiednią wartość.

Tabela 6.12 Scenariusz testowy przypadku kiedy bramka zostanie pominięta.

ID	012
Tytuł	Brak reakcji na bramkę.
Warunki początkowe	Gra uruchomiona, bramka czasowa widoczna na torze.
Kroki testowe	1. Omiń bramkę czasową.
Oczekiwany rezultat	Licznik czasu nie zmienia się.

Tabela 6.13 Scenariusz testowy dla wielokrotnego przejechania przez tę samą bramkę.

ID	013
Tytuł	Wielokrotny przejazd przez bramkę.
Warunki początkowe	Gra uruchomiona, bramka czasowa widoczna na torze.
Kroki testowe	1. Przejedź przez tę samą bramkę wielokrotnie.
Oczekiwany rezultat	Bramki czasowe zwiększają czas tylko raz.

Rys. 6.7. Test bramki czasowej

7. Podsumowanie i wnioski

W ramach pracy inżynierskiej opracowano grę wyścigową z przeszkodami w środowisku Unreal Engine 5, wykorzystując technologię Blueprintów. Gra pozwala graczowi na wybór jednej z dostępnych map, na której celem jest ukończenie toru przeszkód w określonym czasie. Elementem wspomagającym rozgrywkę są bramki czasowe, które dodają dodatkowe sekundy na ukończenie toru.

Projekt wykorzystuje nowoczesne technologie do tworzenia gier, ze szczególnym uwzględnieniem Blueprintów, które umożliwiły szybkie prototypowanie oraz intuicyjne zarządzanie logiką gry. Unreal Engine 5 zapewnił wysoki poziom graficzny oraz możliwość implementacji zaawansowanych mechanik rozgrywki, takich jak dynamiczne obiekty przeszkód czy interakcja pojazdu z otoczeniem [1;5].

Podczas realizacji projektu uwzględniono następujące aspekty techniczne i projektowe:

- **Projektowanie mechaniki rozgrywki**: Stworzono system bramek czasowych, licznik czasu.
- **Projektowanie tras i przeszkód**: Opracowano kilka unikalnych map, które różnią się poziomem trudności oraz układem przeszkód, co zapewnia graczowi różnorodność rozgrywki.
- **Optymalizacja gry**: Przeprowadzono testy wydajnościowe, aby zapewnić płynną rozgrywkę.

Realizacja projektu pozwoliła na zdobycie cennych umiejętności w zakresie tworzenia gier wideo, takich jak projektowanie mechanik rozgrywki, wykorzystanie narzędzi graficznych i implementacja logiki w technologii Blueprintów. Kluczowe wnioski z pracy:

- **Zastosowanie Blueprintów**: Blueprinty okazały się skutecznym narzędziem, szczególnie w zakresie szybkiego wprowadzania zmian w logice gry. Pozwoliły na znaczną redukcję czasu potrzebnego na rozwój gry, przy jednoczesnym zachowaniu wysokiej elastyczności.
- **Projektowanie poziomów**: Tworzenie tras o różnych poziomach trudności wymaga dokładnego przemyślenia balansu między wyzwaniem a satysfakcją z gry. Mapy powinny być zarówno wymagające, jak i motywujące do poprawiania wyników.

- **Testowanie i optymalizacja**: Proces testowania okazał się kluczowy dla zapewnienia płynności rozgrywki oraz eliminacji potencjalnych błędów. Równowaga między jakością grafiki a wydajnością gry była jednym z kluczowych wyzwań.
- **Znaczenie użyteczności i interfejsu**: Intuicyjny interfejs i jasne zasady gry są kluczowe dla pozytywnego doświadczenia gracza. Testy z udziałem użytkowników pozwoliły na dostosowanie rozgrywki do oczekiwań graczy.

Projekt może stanowić podstawę do dalszego rozwoju, na przykład poprzez dodanie trybu wieloosobowego, wprowadzenie dodatkowych pojazdów czy systemu personalizacji. Dzięki elastyczności Unreal Engine 5 oraz potencjałowi technologii Blueprintów, gra ma możliwość rozwinąć się w przyszłości.

8. Bibliografia

[1] Unreal Engine [Online]
https://www.unrealengine.com/en-US
[2] Nanite Virtualized Geometry [Online] https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine?application_version=5.4
[3] Lumen Global Illumination and Reflections [Online]
https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine
[4] Chaos Physic [Online] https://dev.epicgames.com/documentation/en-us/unreal-engine/chaos-physics-overview?application_version=4.27
[5] Marcos Romero, Brenden Sewell, Packt Publishing, Blueprints Visual Scripting for Unreal Engine 5, 2022
[6] Quixel [Online]
https://quixel.com
[7] Epic Games Marketplace [Online]
https://www.unrealengine.com/marketplace/en-US/

9. Spis tabel

10.Spis rysunków