

**KOLEGIUM INFORMATYKI STOSOWANEJ**

**Kierunek: INFORMATYKA**

**Specjalność: Zaawansowane programowanie**

Mateusz Krok

Nr albumu studenta: 60103

***„Space Survivor” – gra 2D typu top-down shooter***

Promotor: Dr Marek Jaszuk

**PRACA DYPLOMOWA INŻYNIERSKA**

**Rzeszów 2023**

Spis treści

[Wstęp 4](#_Toc135065966)

[1. Wybór technologii oraz narzędzi 6](#_Toc135065967)

[1.1 Silniki gier komputerowych 6](#_Toc135065968)

[1.2 Dodatkowe narzędzia i zasoby do tworzenia gier komputerowych 8](#_Toc135065969)

[2. Projektowanie gry komputerowej 11](#_Toc135065970)

[2.1 Opis podobnych gier 12](#_Toc135065971)

[Vampire Survivor 12](#_Toc135065972)

[Survivor.io 13](#_Toc135065973)

[2.2 Założenia projektu gry 14](#_Toc135065974)

[2.3 Wymagania funkcjonalne 15](#_Toc135065975)

[2.4 Wymagania niefunkcjonalne 15](#_Toc135065976)

[2.5 Diagram klas 16](#_Toc135065977)

[2.6 Diagram przypadków użycia 17](#_Toc135065978)

[2.7 Diagram przepływu gry 17](#_Toc135065979)

[2.8 Opis algorytmu scrollującej nieskończonej mapy. 18](#_Toc135065980)

[3 Unity Game Engine 21](#_Toc135065981)

[3.1 Czym jest Unity 21](#_Toc135065982)

[3.2 Technologie Unity 22](#_Toc135065983)

[3.3 Przestrzeń gry 23](#_Toc135065984)

[4. Prezentacja gry 26](#_Toc135065985)

[4.1. Podsumowanie 30](#_Toc135065986)

[Literatura 30](#_Toc135065987)

[Streszczenie 31](#_Toc135065988)

# Wstęp

W dzisiejszych czasach gry 3D dominują na rynku, oferując trójwymiarowe środowiska, realistyczne efekty wizualne i pełną swobodę poruszania się dla graczy. Niemniej jednak, gry 2D wciąż znajdują swoje miejsce w świecie gier, zarówno wśród graczy, jak i twórców. Gry 2D charakteryzują się płaską, dwuwymiarową grafiką, która jest znacznie prostsza do stworzenia niż trójwymiarowe modele i tekstury gier 3D. Dlatego gry 2D często są tworzone przez mniejsze zespoły lub pojedynczych twórców, co sprawia, że są popularnym wyborem dla małych studiów i niezależnych (indie) twórców gier.

Istnieje kilka czynników, które przyczyniają się do popularności gier 2D wśród małych twórców gier typu indie. Po pierwsze, tworzenie gier 2D wymaga mniejszych zasobów graficznych i technologicznych, co przekłada się na niższe koszty produkcji. Twórcy indie często działają na ograniczonych budżetach, dlatego wybór gier 2D pozwala im skupić się na kreatywnym aspekcie projektu, zamiast inwestować znaczną część środków w tworzenie skomplikowanych modeli i animacji.

Ponadto, gry 2D posiadają specyficzny estetyczny urok, który przyciąga wielu graczy. Często mają one unikalny styl artystyczny, który pozwala twórcom wyrazić swoją kreatywność i wyróżnić się na rynku. Indywidualne podejście do projektowania grafiki i świata gry może przyciągnąć uwagę potencjalnych graczy, zwłaszcza jeśli jest to coś, co wyróżnia daną grę spośród konkurencji.

Gry 2D posiadają wiele zalet w porównaniu do gier 3D. Przede wszystkim, grafika gier 2D jest znacznie prostsza do stworzenia i może być tworzona przez mniejsze zespoły lub nawet pojedynczych twórców, co przekłada się na niższe koszty produkcji. Ponadto, gry 2D są często bardziej intuicyjne w obsłudze niż gry 3D, ponieważ gracze nie muszą poruszać się w trójwymiarowym środowisku, a zamiast tego mają do czynienia z płaskim, dwuwymiarowym światem.

Inną zaletą gier 2D jest to, że czasami łatwiej jest osiągnąć zamierzony efekt lub cel. W grach 3D, ze względu na swobodę ruchu i pełną trójwymiarowość, trudno jest zaplanować każdy aspekt rozgrywki  i uczynić go perfekcyjnym. W przypadku gier 2D, dzięki prostocie ich konstrukcji, jest to znacznie łatwiejsze. Ponadto, gry 2D umożliwiają skoncentrowanie się na samym gameplayu i wyzwaniach, które stawiają przed graczem, a nie tylko na efektach wizualnych.

Wybór gatunku gry 2D typu top-down shooter był podyktowany kilkoma czynnikami. Po pierwsze, top-down shooter to klasyczny gatunek, który cieszy się popularnością wśród graczy od wielu lat. Z drugiej strony, istnieje wiele różnych sposobów na interpretację tego gatunku, co pozwala twórcom wykazać się kreatywnością w projektowaniu gry.

Po drugie, top-down shooter oferuje wiele możliwości w zakresie rozgrywki. Gracz kontroluje postać z górnego punktu widzenia i musi poruszać się po planszy, unikać przeszkód i pokonywać przeciwników. Dzięki temu gra może oferować wiele różnych wyzwań i poziomów trudności, a także umożliwia twórcom projektowanie różnych rodzajów broni i wrogów.

Po trzecie, niedawny sukces gry "Vampire Survivors" ożywił ten gatunek gier, co skutkowało powstawaniem wielu gier o podobnym temacie. Gra opiera się na prostych założeniach, gdzie gracz znajduje się na niewielkiej mapie i stawia czoła hordom wrogów. Po pokonaniu wrogów, gracz zdobywa kryształy, które służą do zdobywania doświadczenia. Zdobycie doświadczenia pozwala na podniesienie poziomu postaci, co umożliwia wybór jednego z losowo wyświetlanych przedmiotów. Im więcej przetrwamy i uzbieramy doświadczenia, tym nasza postać staje się potężniejsza.

Zakres pracy będzie obejmował wiele różnych aspektów projektu. Przede wszystkim, konieczne będzie zaprojektowanie i stworzenie grafiki oraz modeli postaci i otoczenia. Następnie, należy zaprogramować logikę gry, w tym poruszanie się postaci, mechanikę walki, sterowanie bronią i sztuczną inteligencję przeciwników.

Gry 2D, mimo że nie oferują takiego wrażenia przestrzenności jak gry 3D, wciąż cieszą się dużą popularnością, szczególnie wśród małych twórców gier typu indie. Są one łatwiejsze i tańsze w produkcji, co jest istotne dla twórców działających na ograniczonych budżetach. Tworzenie gier 2D pozwala im skupić się na kreatywnym aspekcie projektu, zamiast inwestować znaczną część środków w tworzenie skomplikowanych modeli i animacji.

Ponadto, gry 2D oferują specyficzny estetyczny urok, który przyciąga wielu graczy. Ich unikalny styl artystyczny pozwala twórcom wyrazić swoją kreatywność i wyróżnić się na rynku. Indywidualne podejście do projektowania grafiki i świata gry może przyciągnąć uwagę potencjalnych graczy, zwłaszcza jeśli jest to coś, co wyróżnia daną grę na tle konkurencji.

# Wybór technologii oraz narzędzi

Wybór odpowiednich narzędzi jest kluczowy dla projektów gier. Muszą one być łatwe w użyciu oraz posiadać funkcje odpowiadające potrzebom projektu. Narzędzia programistyczne i edytory umożliwiają łatwiejsze tworzenie i modyfikowanie elementów gry. Ponadto, wybór odpowiednich narzędzi może znacznie wpłynąć na koszty produkcji oraz czas potrzebny do zakończenia projektu.

## Silniki gier komputerowych

Wybór odpowiedniego silnika gier komputerowych to kluczowy element projektowania gier, który wpływa na ostateczny wygląd i funkcjonalność gry. Silnik stanowi podstawę, na której buduje się całą grę i zapewnia jej stabilne działanie na różnych platformach. Wybór odpowiedniego silnika zależy od indywidualnych potrzeb projektu oraz umiejętności i preferencji twórców gry. Istnieje wiele dostępnych silników na rynku, każdy z nich oferujący różne funkcjonalności, narzędzia i koszty.

Wybór odpowiedniego silnika do tworzenia gier komputerowych to kluczowy element projektowania, który ma wpływ na ostateczny wygląd i funkcjonalność gry. Silnik stanowi fundament, na którym opiera się cała gra i zapewnia jej stabilne działanie na różnych platformach. Wybór odpowiedniego silnika zależy od indywidualnych potrzeb projektu, umiejętności oraz preferencji twórców. Na rynku dostępnych jest wiele silników, z różnymi funkcjonalnościami, narzędziami i kosztami.

W przypadku mojego projektu, wybór silnika gier komputerowych był jednym z pierwszych kroków w procesie projektowania gry. Musiałem dokładnie przeanalizować różne opcje i wybrać taki silnik, który spełniłby potrzeby mojego projektu. Musiał być łatwy w obsłudze, posiadać bogatą dokumentację i odpowiednie narzędzia do tworzenia gier 2D. Postanowiłem przyjrzeć się dwóm silnikom. Są nimi:

* Unity[[1]](#footnote-1) - jest jednym z najpopularniejszych narzędzi do tworzenia gier, umożliwiającym tworzenie gier na różne platformy. Oferuje wiele narzędzi i zasobów, takich jak gotowe modele postaci, tekstury i efekty dźwiękowe, które można wykorzystać w grze. Silnik ten oparty jest na języku C# i umożliwia tworzenie zarówno gier 2D, jak i 3D. Najważniejsze zalety Unity to wieloplatformowość, umożliwiająca tworzenie gier na komputery, konsole, smartfony i tablety, oraz bogata dokumentacja, poradniki, kursy i tutoriale, które pomagają początkującym łatwo wejść w świat tworzenia gier komputerowych.

Unity oferuje również wiele narzędzi i zasobów dostępnych w sklepie Unity Asset Store, a także możliwość integracji z innymi narzędziami i platformami, takimi jak Visual Studio, Adobe Photoshop i inne. Dzięki tym zaletom nawet początkujący twórcy gier mogą korzystać z najlepszych narzędzi i procesów, aby tworzyć gry wysokiej jakości.



Rys. 1 Logo Silnika Unity  
Źródło: https://unity.com/logo-unity-web.png

* Unreal Engine[[2]](#footnote-2) - to profesjonalne narzędzie do tworzenia gier komputerowych i wizualizacji w czasie rzeczywistym. Silnik ten został stworzony przez firmę Epic Games i jest oparty na języku C++. Oferuje zaawansowane narzędzia do projektowania i tworzenia gier w 3D. Może być również używany do tworzenia gier 2D, choć nie posiada natywnego wsparcia i wymaga pewnych obejść. Program ten ma jednak znacznie wyższy próg wejścia, szczególnie dla początkujących. Dla osób nieumiejących programować dostępny jest ciekawy system wizualnego programowania o nazwie Blueprints. Unreal Engine oferuje lepsze wsparcie graficzne, dość dobrą dokumentację oraz wiele dostępnych platform (choć nieco mniej niż Unity).



Rys. 2 Logo Silnika Unreal Engine  
Źródło:https://cdn2.unrealengine.com/ue-logo-1400x788-1400x788-8f185e1e3635.jpg

Ostatecznie zdecydowałem się na użycie silnika graficznego Unity, który jest jednym z najbardziej popularnych i wszechstronnych silników w branży gier komputerowych. Unity oferuje bogaty zestaw narzędzi, które umożliwiają projektowanie gier 2D. Posiada również dużą społeczność oraz mnóstwo dostępnych tutoriali, co ułatwia naukę i rozwiązywanie problemów.

Korzystając z Unity, będę mógł skupić się na projektowaniu gry i tworzeniu nowych funkcjonalności, zamiast tracić czas na tworzenie podstawowych mechanik od zera. Dzięki temu uda mi się skrócić czas potrzebny na rozwój projektu i naukę.

## Dodatkowe narzędzia i zasoby do tworzenia gier komputerowych

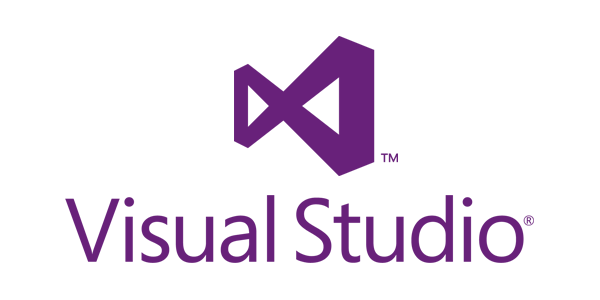
Do tworzenia gry komputerowej w Unity będziemy potrzebować środowiska programistycznego (IDE). Wybór edytora kodu zależy od indywidualnych preferencji programisty oraz potrzeb projektu. Dostępne na rynku edytory, takie jak Visual Studio, Visual Studio Code, JetBrains Rider czy Eclipse, oferują wiele funkcjonalności ułatwiających pracę nad kodem, takich jak debugowanie, podpowiadanie składni, system kontroli wersji i wiele innych. W zależności od umiejętności i preferencji programisty, można wybrać edytor, który najlepiej odpowiada jego potrzebom. Przy wyborze edytora kodu należy zwrócić uwagę na funkcjonalności, wydajność, integrację z silnikiem gier oraz dostępność rozszerzeń i wtyczek. Warto również zastanowić się nad wyborem edytora, który jest kompatybilny z systemem operacyjnym używanym w projekcie, aby uniknąć problemów z integracją.

Visual Studio Code[[3]](#footnote-3) - to darmowy edytor kodu źródłowego opracowany przez firmę Microsoft. Jest dostępny na platformy Windows, Linux i macOS. Oferuje wiele funkcji, takich jak debugowanie, podpowiadanie składni, system kontroli wersji, narzędzia do refaktoryzacji kodu oraz wsparcie dla wielu języków programowania. Posiada wiele rozszerzeń, które pozwalają na dostosowanie edytora do indywidualnych potrzeb programisty.



Rys. 3 Logo Visual Studio Code  
Źródło: https://www.vectorlogo.zone/logos/visualstudio\_code/visualstudio\_code-ar21.png

Visual Studio[[4]](#footnote-4) - to zintegrowane środowisko programistyczne (IDE) opracowane przez Microsoft, oferujące wiele funkcji, takich jak debugowanie, testowanie, refaktoryzację kodu, integrację z Azure i wiele innych. Posiada bogate narzędzia do pracy z kodem, w tym IntelliSense, automatyczne uzupełnianie kodu, sprawdzanie pisowni oraz narzędzia do zarządzania projektem i wersjonowania kodu. Visual Studio oferuje wsparcie dla wielu języków programowania i środowisk deweloperskich. W przeciwieństwie do darmowego Visual Studio Code, Visual Studio jest płatne, ale oferuje również darmową wersję Community z pewnymi ograniczeniami funkcjonalności.



Rys. 4 Strona Logo Visual Studio  
Źródło: https://e7.pngegg.com/pngimages/910/226/png-clipart-visual-studio-2010-developpez-pour-le-web-avec-c-4-framework-entity-4-asp-net-4-silverlight-4-et-wcf-ria-services-logo-visual-basic-microsoft-visual-studio-visual-programming.png

Rider [[5]](#footnote-5)- to zintegrowane środowisko programistyczne opracowane przez firmę JetBrains, oferujące wiele zaawansowanych funkcji, takich jak inteligentne podpowiedzi składni, refaktoryzację kodu, debugowanie, narzędzia do testowania oraz integrację z wieloma narzędziami deweloperskimi. Rider oferuje wsparcie dla wielu języków programowania, w tym C#, VB.NET, F# i innych. W przeciwieństwie do Visual Studio i Visual Studio Code, Rider jest głównie przeznaczony dla programistów pracujących w systemie operacyjnym Linux i macOS. Rider jest płatny, ale oferuje 30-dniową wersję próbną oraz wersję Community z pewnymi ograniczeniami w funkcjonalności.



Rys. 5 Logo Rider JetBrains  
Źródło: https://resources.jetbrains.com/storage/products/rider/img/meta/rider\_logo\_300x300.png

Warto wspomnieć, że wszystkie trzy wymienione środowiska programistyczne (IDE), czyli Visual Studio, Visual Studio Code i Rider, oferują wsparcie dla środowiska Unity. W moim przypadku, zdecydowałem się na Visual Studio, ponieważ znam go najlepiej i mam najwięcej doświadczenia z jego użytkowania. Jednak każde z wymienionych narzędzi może być dobrym wyborem dla programisty tworzącego gry w Unity, w zależności od indywidualnych preferencji i potrzeb.

Oprócz IDE, będziemy potrzebować również programu graficznego. Możemy użyć na przykład:

* Photoshop[[6]](#footnote-6) – profesjonalne oprogramowanie graficzne, które oferuje szeroki zakres narzędzi do edycji i tworzenia grafiki 2D. Pozwala na pracę zarówno z grafiką wektorową, jak i rastrową, umożliwiając tworzenie grafiki w różnych formatach. Photoshop posiada wiele narzędzi, takich jak filtry, pędzle, korektory, narzędzia selekcji, maski i warstwy, co umożliwia tworzenie skomplikowanej grafiki wraz z efektami specjalnymi. Photoshop jest szeroko stosowany w przemyśle gier komputerowych, zwłaszcza w grach AAA, ze względu na swoją zaawansowaną funkcjonalność oraz wsparcie dla wielu formatów plików.



Rys. 6 Logo Adobe Photoshop  
Źródło: https://www.adobe.com/pl/products/photoshop.html

* GIMP[[7]](#footnote-7) - bezpłatny program graficzny o otwartym kodzie źródłowym, który może pomóc w tworzeniu grafiki dla gier 2D. Oferuje wiele narzędzi, takich jak pędzle, narzędzia selekcji, filtry, warstwy, wypełnianie kolorami oraz wiele innych, co pozwala na tworzenie i edycję grafiki w różnych formatach. GIMP oferuje również wtyczki, które pozwalają na dodatkowe funkcje, takie jak dodawanie animacji czy tworzenie sprite'ów. Jest to popularny wybór dla twórców gier 2D, ze względu na jego wsparcie dla wielu formatów plików oraz funkcjonalność porównywalną z programami komercyjnymi.



Rys. 6 Logo Gimp  
Źródło: https://w7.pngwing.com/pngs/438/441/png-transparent-paintbrush-gimp-logo-design-logo-brush-paint.png

Ostatecznie zdecydowałem się na Photoshop, ponieważ jest to program, którego najczęściej używam i który jest dla mnie bardziej intuicyjny w obsłudze.

Myślę, że gdy mówimy o grafice, warto również wspomnieć o Unity Asset Store. Ten sklep oferuje szeroki wybór gotowych zasobów graficznych, dźwiękowych i skryptów, które można wykorzystać w tworzeniu gier w silniku Unity. Dostępne tam są między innymi modele 2D, tekstury, animacje, efekty specjalne, dźwięki i muzyka. Korzystanie z Asset Store umożliwia uzyskanie wysokiej jakości materiałów, które w przeciwnym wypadku wymagałyby długiego procesu tworzenia od zera. Wiele z nich można wykorzystać za darmo. Dlatego będzie on źródłem spritów i tekstur, które będę używał w moim projekcie.



Rys. 6 Przykład darmowej kolekcji 2D  
Źródło https://assetstore.unity.com/packages/2d/environments/2d-space-kit-27662

# Projektowanie gry komputerowej

W dzisiejszych czasach projektowanie gier komputerowych stało się o wiele łatwiejsze dzięki dostępności narzędzi oraz różnych platform. Dla wielu ludzi staje się to coraz bardziej interesującą dziedziną, a wyniki ich prac można łatwo opublikować i udostępnić na różnych portalach. Warto zainteresować się projektowaniem gier nie tylko ze względu na popularność i rozrywkę, jaką gry zapewniają, ale także ze względu na ich rosnące znaczenie w branży rozrywkowej. Projektowanie gier komputerowych może być fascynującym i kreatywnym zajęciem, które daje możliwość realizowania swoich pomysłów i wizji.

Projektowanie gier w dużej studiu różni się od samodzielnego projektowania, przede wszystkim ze względu na skalę pracy i liczne zespoły specjalistów. Tworząc grę samemu, jesteśmy odpowiedzialni za całość procesu i musimy samodzielnie radzić sobie z każdym etapem produkcji.

Nie ma jednak wątpliwości, że projektowanie gier może przynieść dużo satysfakcji i daje wiele możliwości wyrażenia swojej kreatywności. Co więcej, projektowanie gier łączy wiele dziedzin, takich jak grafika, programowanie, muzyka i animacja.

Nie ważne jednak, czy projektujemy grę sami, czy w dużej firmie, kluczowe znaczenie ma kreatywność i innowacyjność. Najważniejszym punktem łączącym projektowanie gier w dużej studiu i samodzielne projektowanie jest realizacja wizji. Dobra gra musi być interesująca, angażująca i satysfakcjonująca dla graczy, a jednocześnie spełniać swoje założenia projektowe i działać płynnie.

Etapami produkcji gier komputerowych można podzielić na następujące punkty:

* • Pomysł na grę - Określenie koncepcji gry i jej funkcjonalności.
* • Projektowanie gry - Przygotowanie planu gry, jej mechanik i poziomów.
* • Tworzenie grafiki i dźwięków - Przygotowanie grafiki, animacji i dźwięków do gry.
* • Programowanie - Kodowanie mechanik i funkcjonalności gry.
* • Testowanie - Testowanie gry w celu znalezienia i naprawienia błędów oraz usprawnienia gry.
* • Wydanie gry - Ostateczne przygotowanie gry do wydania oraz jej publikacja.
* • Wsparcie i aktualizacje - Utrzymywanie gry poprzez regularne aktualizacje oraz zapewnienie wsparcia technicznego dla użytkowników.

## Opis podobnych gier

### Vampire Survivor[[8]](#footnote-8)

Gra komputerowa pt. "Vampire Survivors" to czasowa gra survivalowa typu shoot'em up z minimalistycznym gameplayem i elementami roguelike. Została stworzona przez jedną osobę o imieniu Luca Galante. Gracz wybiera jedną z wielu postaci z różnymi startowymi bonusami i broniami oraz nimi steruje na nieskończonej, automatycznie generowanej mapie o powtarzającym się układzie. Bronie postaci atakują automatycznie, a celem gry jest przetrwanie jak najdłużej przeciwko ciągłym falom potworów, które zadają obrażenia, gdy zbliżą się do gracza. Po pokonaniu potwora zostaje po nim kryształ doświadczenia, który po zebraniu zwiększa poziom postaci i daje doświadczenie. Każdy poziom pozwala na wybór jednej z trzech opcji broni lub umiejętności pasywnych. Kolejne poziomy umożliwiają ulepszanie tych rzeczy. Na mapie możemy znaleźć także kurczaka, który przywraca część zdrowia, złoto, które pozwala na ulepszanie umiejętności pasywnych w menu głównym, oraz skrzynię, która po pokonaniu bossów losuje darmowe przedmioty, poprawiające nasze bronie. Bronie i umiejętności pasywne można ulepszać do 6 razy, a po spełnieniu odpowiednich wymagań można wylosować ich ostateczną wersję ze skrzyń.

Sesja gry ma limit czasowy od 15 do 30 minut, w zależności od wybranej mapy. Gdy nadejdzie koniec czasu, wszystkie potwory zostają pokonane, a pojawia się ostateczny, potężny boss o nazwie "Death". Następnie, co minutę pojawia się kolejny taki sam boss, aby upewnić się, że nasz bohater skończy swoje życie. Sesja, która osiągnęła limit czasowy, jest uznawana za udaną i nagradza gracza dodatkowymi sztukami złota. Dodatkowo, gra posiada system wyzwań, za które gracz jest nagradzany nowymi mapami, broniami i postaciami.



Rys.7 Gameplay Vampire Survivors  
Źródło https://cdn.vox-cdn.com/thumbor/VUfHTW2et7dCKOBK3dKQd3tEn2c=/1400x1400/filters:format(jpeg)/cdn.vox-cdn.com/uploads/chorus\_asset/file/23232679/Vampire\_Survivors\_3.jpg

Mimo że gra wydaje się być prosta, jej design rozgrywki i możliwość powtarzalności zostały docenione przez setki tysięcy graczy. Gra posiada 98% pozytywnych opinii spośród 186 000 na platformie Steam. Otrzymała wiele nagród, takich jak:

* Nagroda Golden Joystick - "Przełom" w 2022 roku.
* New York Game Awards - Off Broadway Award for Best Indie Game w 2023 roku.
* D.I.C.E. Awards - Action Game of the Year w 2023 roku.
* British Academy Games Awards - Best Game oraz Game Design w 2023 roku.

Ta gra udowodniła, że można stworzyć świetną grę o zasięgu globalnym w prostym wydaniu. Wystarczy mieć bardzo dobry pomysł oraz umiejętności.

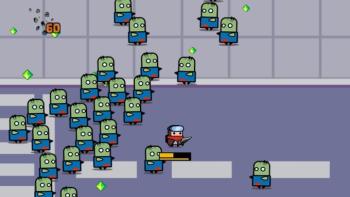
Sukces tej gry odsłonił lukę na rynku gier tego typu, co skutkowało powstawaniem wielu kopii, które odnosiły równie duże sukcesy na rynku mobilnym. Dzięki prostym 2D grafikom, gry tego typu nie zajmują dużo miejsca i nie obciążają zasobów systemowych. Wersja mobilna "Vampire Survivors" została wydana pod koniec 2022 roku jako darmowa wersja z reklamami. Wersje desktopowe są płatnymi grami, kosztującymi około 20 złotych.



Rys.8 Okładka Vampire Survivors  
Źródło https://cdn.cloudflare.steamstatic.com/steam/apps/1794680/capsule\_616x353.jpg?t=1681394120

### Survivor.io[[9]](#footnote-9)

Gra o podobnych założeniach do "Vampire Survivors" oscyluje wokół apokalipsy zombie, gdzie gracz stara się odeprzeć hordy zombie. Gra wyróżnia się kilkoma elementami w porównaniu do swojego pierwowzoru. Teraz gracz przechodzi mapy, zabijając końcowych bossów, przy czym na każdej mapie pojawia się trzech z nich. Gra jest skoncentrowana na linearnym postępie, więc zaczynamy od pierwszej mapy, a każda kolejna staje się trudniejsza. Wprowadzono również system poziomów postaci, dzięki któremu możemy odblokować dodatkowe umiejętności pasywne, które są ograniczone przez nasz aktualny poziom. Gra posiada również system tworzenia i ulepszania przedmiotów, podobny do tych stosowanych w grach typu RPG.



Rys.9 Gameplay Survivor.io  
Źródło https://www.commonsensemedia.org/sites/default/files/styles/ratio\_16\_9\_small/public/screenshots/csm-app/survivor.io\_.2.jpeg

Gra Survivor.io została stworzona głównie jako gra mobilna, tak więc ma o wiele bardziej rozwinięty model płatności i reklam. Nie możemy także przez to grać w nieskończoność ponieważ mamy pasek energii, który kurczy się wraz z wejściem na mapę w grze. Możemy oczywiście odnowić pasek za walutę Premium lub oglądanie reklam. Gra jest także ciągle wspierana dodatkową zawartością oraz naprawami.

Ogółem jest o wiele bardziej przejrzysta i przygotowana dla współczesnego użytkownika smartfona. System poruszania się mimo że jest taki sam jak w Vampire Survivors jest o wiele przyjemniejszy w użyciu na ekranie dotykowym.



Rys.10 Okładka gry Survivor.io  
Źródło: https://media.pocketgamer.com/artwork/ra-91594-1661501132/survivorio-image-2.jpg

## Założenia projektu gry

Projekt gry, który stworzyłem, to 2D survival top-down shooter inspirowany grą "Vampire Survivor", która jest podobnego typu, ale w mojej grze postać kieruje się tylko w dwóch kierunkach, a środowisko gry znajduje się na powierzchni ziemi. Jednak moja gra ma zupełnie inną koncepcję, ponieważ gracz kontroluje statek kosmiczny, który kieruje się w płaszczyźnie 2D w 8 kierunkach.

Głównym celem gry jest przetrwanie jak najdłużej w nieskończonej przestrzeni kosmicznej, atakując wrogów i unikając ich ataków. Gracz steruje statkiem kosmicznym, który automatycznie strzela. Wrogowie poruszają się prosto w kierunku gracza, a jeśli któryś z nich dotknie hitboxa gracza, zada mu obrażenia. Gracz może zdobywać punkty doświadczenia i ulepszać swoje statystyki, a także szukać skrzyń z apteczkami, które go uleczy.

W przeciwieństwie do "Vampire Survivors", moja gra to survival 2D top-down shooter, w której postać gracza może kierować się w 8 kierunkach, co zapewnia pełną swobodę poruszania się w przestrzeni kosmicznej i unikanie wrogów. Celem takiego rodzaju gry jest zapewnienie graczom ekscytującej rozgrywki, która wymaga szybkiego reagowania i umiejętności taktycznych.

Gra będzie zawierać elementy losowości, takie jak przypadkowe pojawianie się skrzyń z apteczkami, bronią i innymi przedmiotami, co dodaje dynamiki i zwiększa jej replayability. Planuję również wprowadzić system postępów, który umożliwi graczom ulepszanie swojej postaci i zdobywanie punktów doświadczenia, co zachęci ich do kontynuowania gry.

Podsumowując, moja gra zapewni wciągającą rozgrywkę, w której gracze będą mogli zanurzyć się w nieskończonej przestrzeni kosmicznej i przetrwać jak najdłużej. Elementy losowości i ulepszania postaci dodają dynamiki i motywują graczy do kontynuacji rozgrywki.

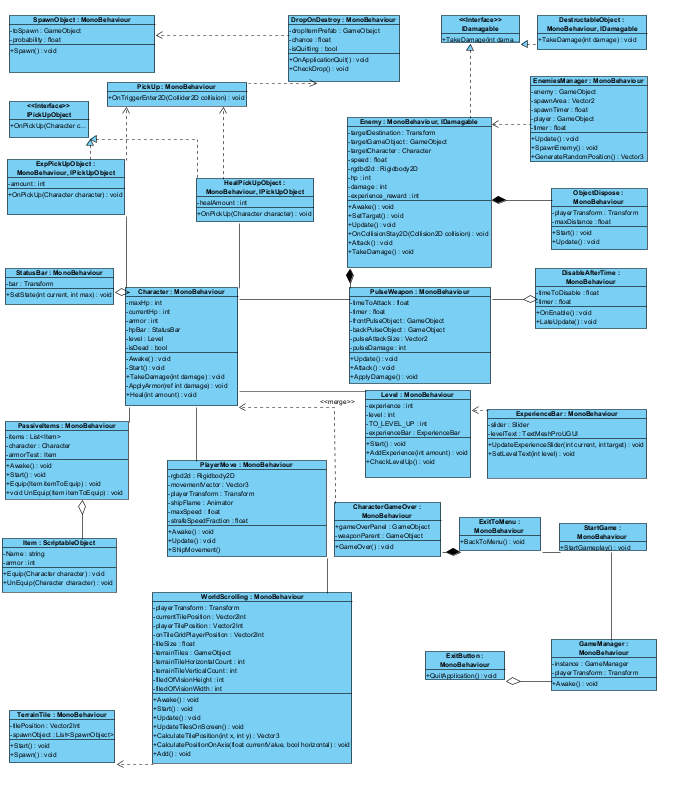
## Wymagania funkcjonalne

* Gracz może poruszać się statkiem kosmicznym na płaszczyźnie 2D w 8 kierunkach.
* Strzelanie z broni jest automatyczne.
* Wrogowie poruszają się prosto na gracza.
* Mapa jest nieskończona przez funkcję autoscrollingu 9 tile'ów.
* Wrogowie, którzy dotkną hitbox gracza, zadają obrażenia.
* Kiedy obrażenia zmniejszą pasek zdrowia do zera, gracz przegrywa, a na ekranie pojawia się opcja powrotu do menu.
* Na mapie znajdują się skrzynie, które można zniszczyć, a z których może pojawić się pick-up, który uleczy postać gracza.
* Każde zabicie wroga pojawia pick-up, który zwiększa ilość doświadczenia na pasku doświadczenia.
* Jeżeli pasek doświadczenia się wypełni, gracz dostaje kolejny poziom.

## Wymagania niefunkcjonalne

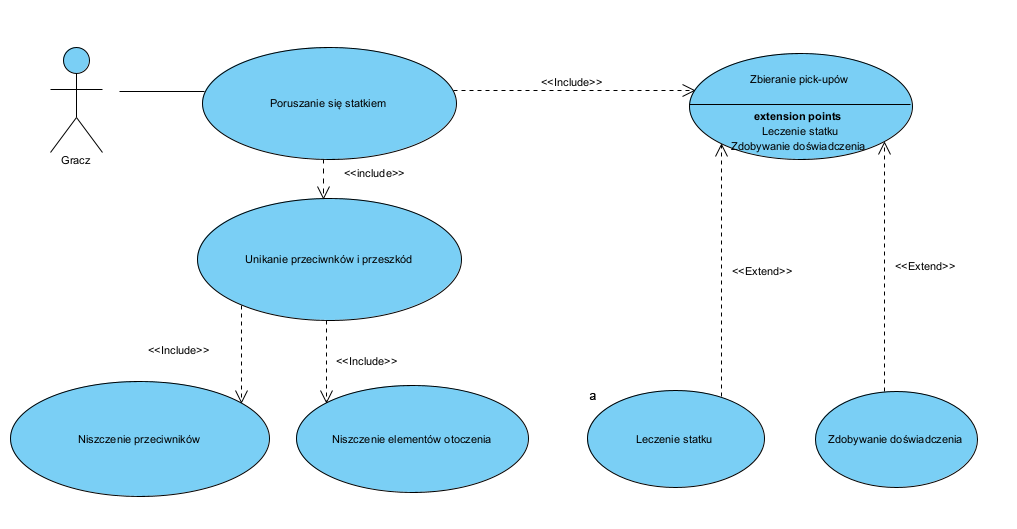
* Gra powinna działać płynnie i bez problemów na różnych systemach operacyjnych, takich jak Windows, Linux i macOS.
* Czas ładowania gry powinien być krótki, aby gracze nie musieli długo czekać na rozpoczęcie rozgrywki.
* Grafika i dźwięk powinny być w pełni dostosowane do tematu i stylu gry.
* Sterowanie grą powinno być intuicyjne i łatwe do opanowania dla nowych graczy.
* Gra powinna być w pełni przetestowana pod kątem błędów i problemów z wydajnością, aby zapewnić jak najlepszą jakość rozgrywki.
* Wszystkie elementy gry powinny być dobrze wyważone, aby gra była odpowiednio trudna, ale nie frustrująca dla graczy.
* Gra powinna być dostępna w języku angielskim, aby przyciągnąć większą liczbę graczy z całego świata.

## Diagram klas



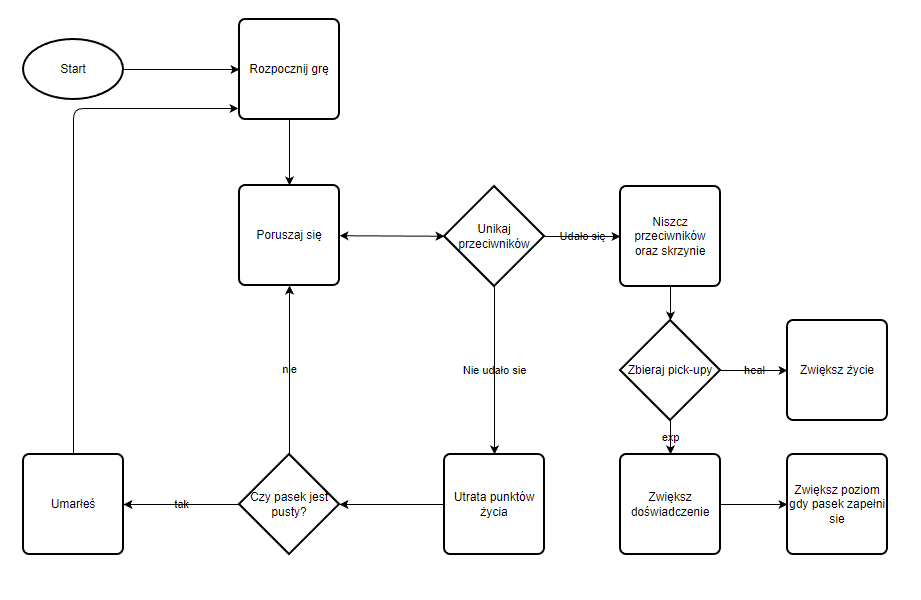
Rys.11 Diagram klas  
Źródło: Własne

## Diagram przypadków użycia



Rys.12 Diagram przypadków uzycia  
Źródło: Własne

## Diagram przepływu gry

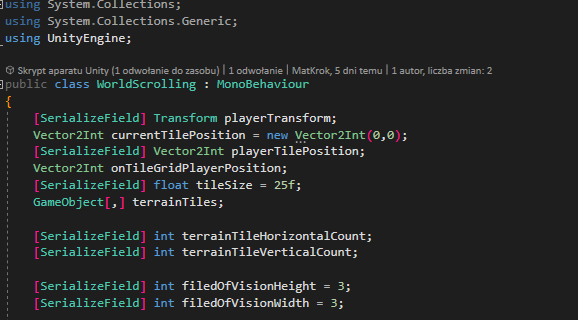


Rys.13 Diagram Flowchart  
Źródło: Własne

## Opis algorytmu scrollującej nieskończonej mapy.

Ten skrypt jest odpowiedzialny za przesuwanie świata gry (tzw. "world scrolling") wraz z graczem. Algorytm o nazwie WorldScrolling dziedziczy klasę MonoBehaviour. Wszystkie klasy, które chcą korzystać z rozwiązań z Unity, takie jak np. Start(), Update(), muszą ją dziedziczyć. Na początku algorytmu mamy inicjalizację atrybutów, takich jak:

* playerTransform: pole to przechowuje referencję do transformacji obiektu gracza.
* currentTilePosition: atrybut ten tworzy nową zmienną o nazwie currentTilePosition, która jest typu Vector2Int. Vector2Int to specjalny typ w Unity, który reprezentuje wektor 2D o wartościach całkowitych. Ta zmienna jest ustawiona początkowo na pozycję (0, 0).
* playerTilePosition: pole to przechowuje pozycję gracza na siatce kwadratów.
* onTileGridPlayerPosition: ta zmienna przechowuje pozycję gracza na siatce kwadratów.
* tileSize: pole to przechowuje rozmiar kwadratu terenu, który jest używany do wyliczania pozycji gracza na siatce.
* terrainTiles: atrybut ten tworzy dwuwymiarową tablicę obiektów typu GameObject, która przechowuje referencje do terenowych kafelków, które zostaną wyświetlone na ekranie.
* terrainTileHorizontalCount: liczba płytek terenu na poziomie horyzontalnym. Określa szerokość obszaru gry.
* terrainTileVerticalCount: liczba płytek terenu na poziomie pionowym. Określa wysokość obszaru gry.
* fieldOfVisionHeight: liczba płytek, które są widoczne w pionie na ekranie jednocześnie. Określa, jak duży jest obszar gry, który jest widoczny dla gracza w pionie.
* fieldOfVisionWidth: liczba płytek, które są widoczne w poziomie na ekranie jednocześnie. Określa, jak duży jest obszar gry, który jest widoczny dla gracza w poziomie.



Rys.14 Klasa WorldScrolling- zmienne  
Źródło: Własne

Metoda Awake() jest wywoływana zaraz po utworzeniu instancji obiektu, jeszcze przed wywołaniem metody Start(). W tym przypadku tworzy ona tablicę terrainTiles o wymiarach zdefiniowanych przez zmienne terrainTileHorizontalCount i terrainTileVerticalCount.

Metoda Start() jest wywoływana zaraz po utworzeniu instancji obiektu i po wykonaniu metody Awake(). W tym przypadku ustawia pozycję obiektu gracza na pozycji odpowiadającej położeniu gracza na siatce płytek oraz wywołuje metodę UpdateTilesOnScreen() w celu aktualizacji wyświetlanych płytek.

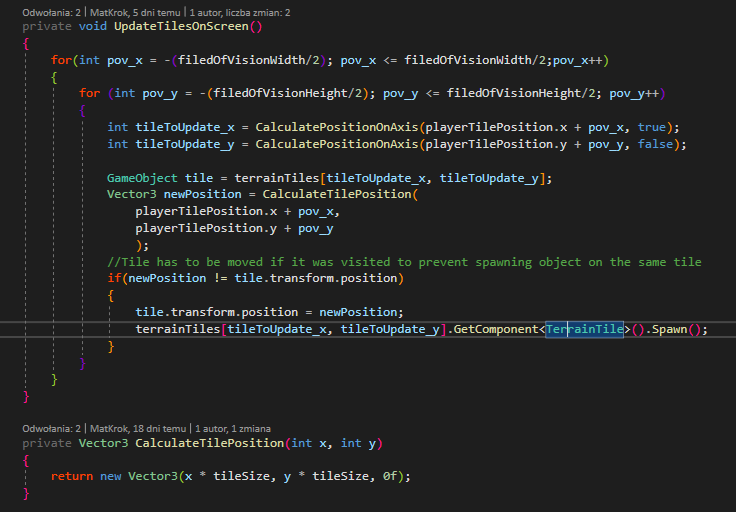
Metoda Update() jest wywoływana co klatkę animacji. W tym przypadku aktualizuje ona pozycję gracza na siatce płytek oraz wywołuje metodę UpdateTilesOnScreen(), jeśli gracz przesunął się na inną płytkę. Funkcja ta korzysta z metod CalculatePositionOnAxis() oraz CalculateTilePosition() w celu obliczenia pozycji gracza i płytek na siatce.



Rys.15 Klasa WorldScrolling- metody Awake(), Start(), Update()  
Źródło: Własne

Metoda UpdateTilesOnScreen() aktualizuje pozycję kafelków terenu na ekranie w zależności od położenia gracza. Dla każdego kafelka w polu widzenia gracza, wyznacza nową pozycję na podstawie położenia gracza i przemieszcza kafelek na tę nową pozycję, jeśli jeszcze go nie odwiedzono. Następnie wywołuje metodę Spawn() dla aktualnego kafelka, aby wygenerować na nim losowe obiekty.

Metoda CalculateTilePosition(int x, int y) oblicza pozycję kafelka na podstawie położenia na osi x i y oraz wielkości kafelka tileSize. Zwraca nowy wektor pozycji, który jest wynikiem pomnożenia każdej wartości osi przez wielkość kafelka, a wartość osi Z jest ustawiona na 0.



Rys.16 Klasa WorldScrolling- metody UpdateTilesOnScreen(), CalculateTilePosition()  
Źródło: Własne

Metoda CalculatePositionOnAxis() jest odpowiedzialna za obliczenie pozycji kafelka na jednej z osi (poziomej lub pionowej), gdy wartość współrzędnej przekracza rozmiar siatki. Metoda ta zwraca poprawioną wartość współrzędnej kafelka na podstawie rozmiaru siatki.

Metoda Add() jest odpowiedzialna za dodanie obiektu reprezentującego kafelek terenu do dwuwymiarowej tablicy terrainTiles na odpowiedniej pozycji w siatce. Metoda ta jest wywoływana przez obiekt reprezentujący kafelek terenu w momencie jego utworzenia.



Rys.17 Klasa WorldScrolling- metody CalculatePositionOnAxis  
Źródło: Własne

# Unity Game Engine

## Czym jest Unity

Unity Game Engine to kompleksowe środowisko do tworzenia gier komputerowych, które oferuje wiele różnych technologii, narzędzi i bibliotek. Oto krótki opis niektórych z nich:

* Silnik graficzny - Unity wykorzystuje zaawansowany silnik graficzny, który umożliwia tworzenie wspaniałych grafik i efektów specjalnych w grach. Silnik obsługuje wiele technik takich jak PBR (physically based rendering), cienie i oświetlenie w czasie rzeczywistym oraz zaawansowane efekty postprocesingu.
* Fizyka - Unity ma wbudowany silnik fizyczny, który umożliwia programistom dodawanie realistycznej symulacji fizycznej do swoich gier. Silnik obsługuje kolizje, detekcję zderzeń, grawitację, a także wiele innych zaawansowanych funkcji.
* Sieciowość - Unity oferuje wiele narzędzi do tworzenia gier sieciowych, w tym wsparcie dla różnych protokołów sieciowych, takich jak TCP/IP i UDP, a także biblioteki do synchronizacji stanu gry na wielu urządzeniach.
* Wirtualna rzeczywistość i rzeczywistość rozszerzona - Unity jest jednym z najpopularniejszych silników do tworzenia aplikacji VR i AR. Oferuje wbudowane wsparcie dla wielu urządzeń VR/AR, takich jak Oculus Rift, HTC Vive, HoloLens i wiele innych.Animacja - Unity oferuje wiele zaawansowanych narzędzi do tworzenia animacji postaci i obiektów w grze. Obejmuje to m.in. system animatorów, ikonografii, motywów, a także narzędzia do edycji i tworzenia animacji w czasie rzeczywistym.
* Audio - Unity oferuje zaawansowany system audio, który umożliwia programistom dodawanie dźwięków, muzyki i efektów dźwiękowych do swoich gier. System obsługuje wielo-kanałowe audio, dynamiczną zmianę ścieżek dźwiękowych w czasie rzeczywistym, a także wiele innych zaawansowanych funkcji.
* Skrypty i języki programowania - Unity umożliwia tworzenie gier za pomocą wielu języków programowania, takich jak C#, C++, JavaScript i Boo. Oferuje również wiele zaawansowanych narzędzi do pisania skryptów i zarządzania nimi.
* Asset Store - Unity posiada wirtualny sklep, w którym programiści mogą kupować i sprzedawać gotowe assety, takie jak modele 3D, tekstury, dźwięki i skrypty. To świetne narzędzie dla osób, które chcą szybko i łatwo rozpocząć tworzenie gier.

## Technologie Unity

W Unity Game Engine obiekty typu GameObject[[10]](#footnote-10) są podstawowymi elementami składowymi sceny. Każdy obiekt GameObject może mieć przypisaną do siebie różnorodną funkcjonalność w postaci komponentów, które dodaje się do niego w sposób modularny. O to niektóre z nich:

* Player[[11]](#footnote-11) - obiekt reprezentujący postać gracza w grze.
* Enemy - obiekt reprezentujący wroga w grze.
* Camera[[12]](#footnote-12) - obiekt reprezentujący kamerę w scenie. Służy do renderowania widoku z perspektywy gracza. Można nim manipulować, aby zmienić położenie, rotację, pole widzenia itp.
* UI Element[[13]](#footnote-13) - obiekt reprezentujący element interfejsu użytkownika. Można nim manipulować, aby stworzyć różne elementy interfejsu, takie jak przyciski, pola tekstowe, suwaki itp.
* Collider[[14]](#footnote-14) - obiekt który umożliwia obiektom interakcję z innymi obiektami w grze poprzez detekcję kolizji. Collider może zostać skonfigurowany tak, aby dokładnie określić jego kształt i wielkość. Kiedy Collider obiektu jest dotknięty lub zderza się z innym Colliderem, Unity informuje o tym w kodzie gry, co umożliwia programowanie reakcji na kolizje.
* Rigidbody[[15]](#footnote-15) - obiekt który pozwala na symulowanie fizyki obiektów w grze. Dzięki niemu obiekty mogą reagować na siły, grawitację, kolizje z innymi obiektami oraz na interakcje z użytkownikiem.
* Terrain[[16]](#footnote-16) - obiekt reprezentujący teren, który można tworzyć za pomocą narzędzi terenowych Unity.
* Light[[17]](#footnote-17) - obiekt reprezentujący źródło światła w scenie, który może być dodany do dowolnego GameObject.
* Transform[[18]](#footnote-18) – obiekt który zna dane o pozycji, rotacji oraz skali obiektu znajdującego się w przestrzeni trójwymiarowej.

Obiekty typu GameObject są reprezentacją dowolnego elementu w grze, takiego jak postać, drzewo, samochód itp. Mogą one zawierać różne rodzaje danych, takie jak grafika, dźwięk, modele, skrypty, kamery i wiele innych. Są również wykorzystywane do tworzenia hierarchii obiektów w scenie, co umożliwia łatwe zarządzanie nimi.

Przykładowo, jeśli chcemy stworzyć postać gracza, tworzymy obiekt typu GameObject, do którego przypisujemy komponenty, takie jak kontroler postaci, animacje, kolider, itd. Obiekt ten będzie również miał pozycję w przestrzeni 3D, dzięki której będzie mógł poruszać się po scenie.

Obiekty typu GameObject są bardzo elastyczne i umożliwiają tworzenie różnorodnych elementów gry. Ich hierarchiczna struktura pozwala na łatwe zarządzanie nimi i wprowadzanie zmian w ich funkcjonalności.

## Przestrzeń gry

Dwuwymiarowa przestrzeń w Unity to po prostu płaszczyzna, na której poruszają się obiekty typu GameObject[[19]](#footnote-19). W Unity istnieje wiele sposobów na stworzenie takiej przestrzeni, jednak najprostszy to utworzenie sceny i dodanie do niej obiektu typu "Plane". W ten sposób tworzymy płaską powierzchnię, po której mogą poruszać się obiekty.

W dwuwymiarowej przestrzeni w Unity możemy wykorzystać wiele różnych komponentów, takich jak Rigidbody, Collider, Sprite Renderer czy Script, aby stworzyć interaktywne środowisko i umożliwić interakcję z obiektami. Możemy też dodać kamery, światła czy efekty wizualne, aby poprawić wrażenia z gry.

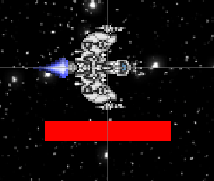
W przestrzeni dwuwymiarowej w Unity używamy również innych typów obiektów, takich jak Tilemap, Sprite, Animator czy Canvas, które pozwalają nam na tworzenie bardziej zaawansowanych i interaktywnych gier.

Na planszy mamy 9 kafelków, które są takiej samej wielkości oraz są równo poukładane, aby stworzyć wielką mapę. Podzielona mapa w ten sposób może później zostać przesuwana, aby tworzyć przewijający się świat gry. Na każdym kafelku znajduje się para dodatkowych tekstur oraz prefabów, aby zróżnicować świat gry.



Rys.18 Mapa  
Źródło: Własne

Na naszej mapie poruszamy się statkiem kosmicznym, który jest obiektem typu "Prefab". Sprite statku został pobrany ze sklepu Unity w paczce. Sprite został umieszczony w kontenerze o nazwie "Player", co pozwala nam odnosić się ogólnie do gracza, niezależnie od jego konkretnej postaci. Do statku został również dodany Collider, aby ustalić hitbox, który będzie reagować z różnymi obiektami. Statek posiada również Rigidbody, co umożliwia mu oddziaływanie fizyczne w grze. Do postaci został również dołączony pasek zdrowia, który pokazuje ilość pozostałych punktów życia.



Rys.19 Sprite statku  
Źródło: Własne

Za naszym statkiem podążają sprite'y kosmitów. Są to "Prefab-y" wrogów, których zadaniem jest kierowanie się w stronę postaci gracza. Mają collider oraz rigidbody, aby ustawić hitboxy i fizykę. Gdy hitbox wroga zetknie się z hitboxem statku, życie statku zacznie się zmniejszać. Gdy wróg zostanie zniszczony, pozostawia na swoim miejscu pick-up doświadczenia, który po wejściu w interakcję z hitboxem dodaje doświadczenie do paska doświadczenia. Wrogowie posiadają również skrypt, który usuwa ich, gdy gracz jest zbyt daleko.



Rys 20 Sprite wroga  
Źródło: Własne

Na ekranie gry mamy pasek doświadczenia oraz poziom. Znajdują się one na obszarze Canvas, który jest GameObjectem tworzonym w obszarze UI. Są one widoczne przez cały czas podczas rozgrywki na mapie. Pasek doświadczenia wizualnie pokazuje, ile obecnie brakuje nam do osiągnięcia następnego poziomu. Gdy pasek osiągnie koniec, poziom zostanie zwiększony.



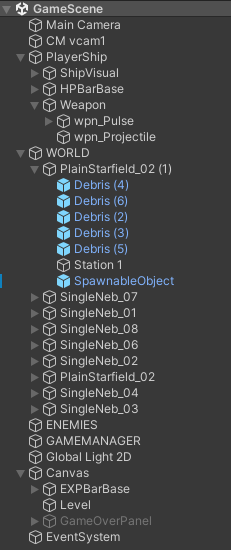
Rys.21 Pasek doświadczenia  
Źródło: Własne

Po mapie rozmieszczone są obiekty otoczenia, które można zniszczyć. Po zniszczeniu tych obiektów przez statek, istnieje szansa na pozostawienie po sobie pick-upu zdrowia. Pozwala nam to po interakcji z jego hitboxem uleczyć pasek zdrowia statku. Obiekt posiada przypisanego collidera oraz skrypt, który usunie go, gdy gracz oddali się zbyt daleko.



Rys.22 Sprite obiektu przechowującego zdrowie  
Źródło: Własne

Hierarchia obiektów prezentuję się w taki sposób:



Rys.23 Hierarchia obiektów  
Źródło: Własne

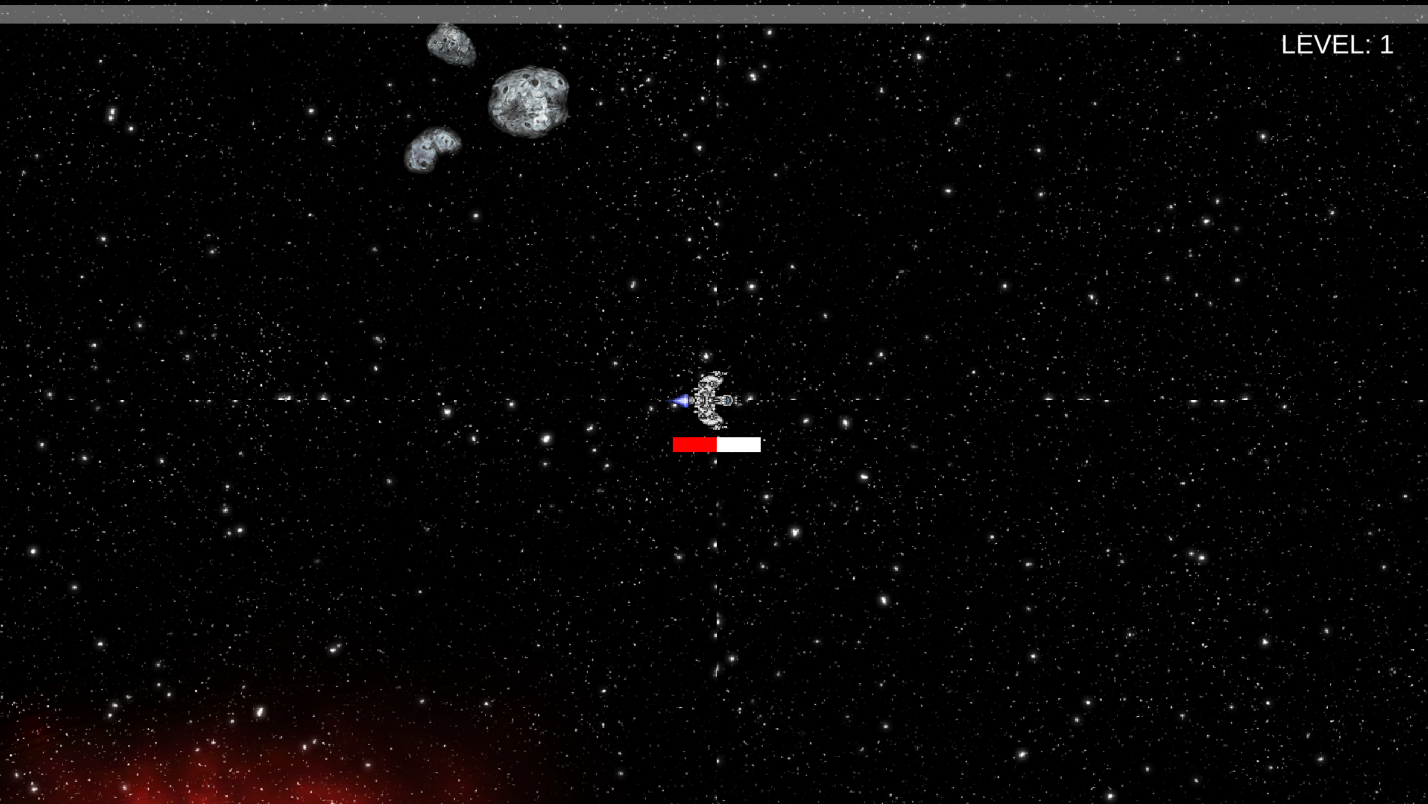
# Prezentacja gry

Po włączeniu gry, wita nas menu główne, z którego możemy rozpocząć grę i wyjść z aplikacji.



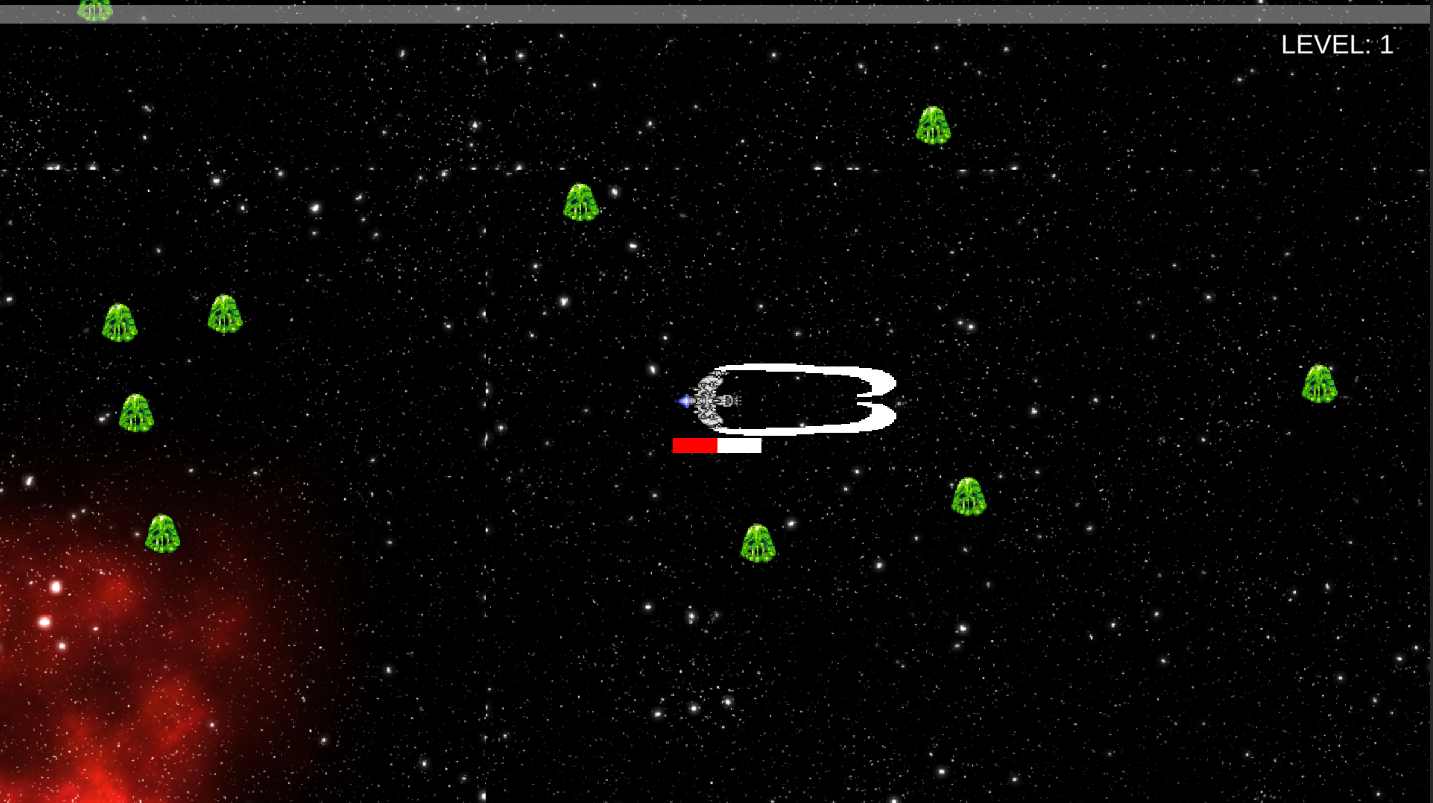
Rys.24 Menu Start gry  
Źródło: Własne

Po wystartowaniu gry pojawia się mapa z naszym statkiem. Poruszamy się nim za pomocą klawiszy WASD lub strzałkami. Na górnym ekranie widoczne są wskaźnik poziomu oraz pasek doświadczenia. Obok statku znajduje się pasek jego życia.



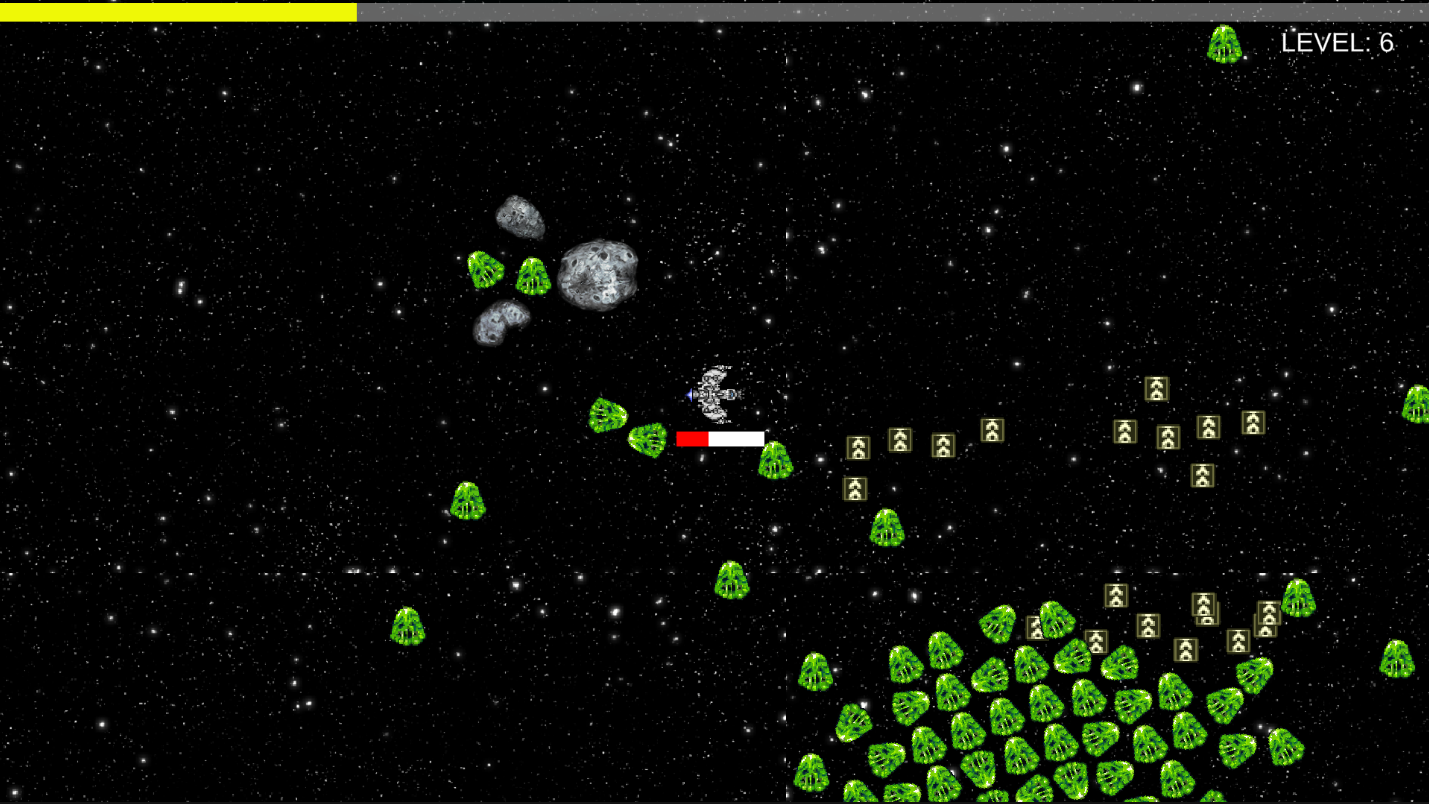
Rys.25 Rozpoczęcie gry  
Źródło: Własne

Wystrzeliwanie z broni wykonuje się automatycznie, a wrogowie zaczynają się pojawiać i kierować w naszym kierunku zaraz po starcie gry.



Rys.26 Atak statku  
Źródło: Własne

Po skierowaniu statku w kierunku wrogów, broń zabija kosmitów, którzy zostawiają po sobie pick-up'y doświadczenia. Po trafieniu w wrogów i podniesieniu pick-up'a doświadczenia, nasz pasek doświadczenia zaczyna się zapełniać. Po jego pełnym zapełnieniu, gra przyznaje graczowi poziom.



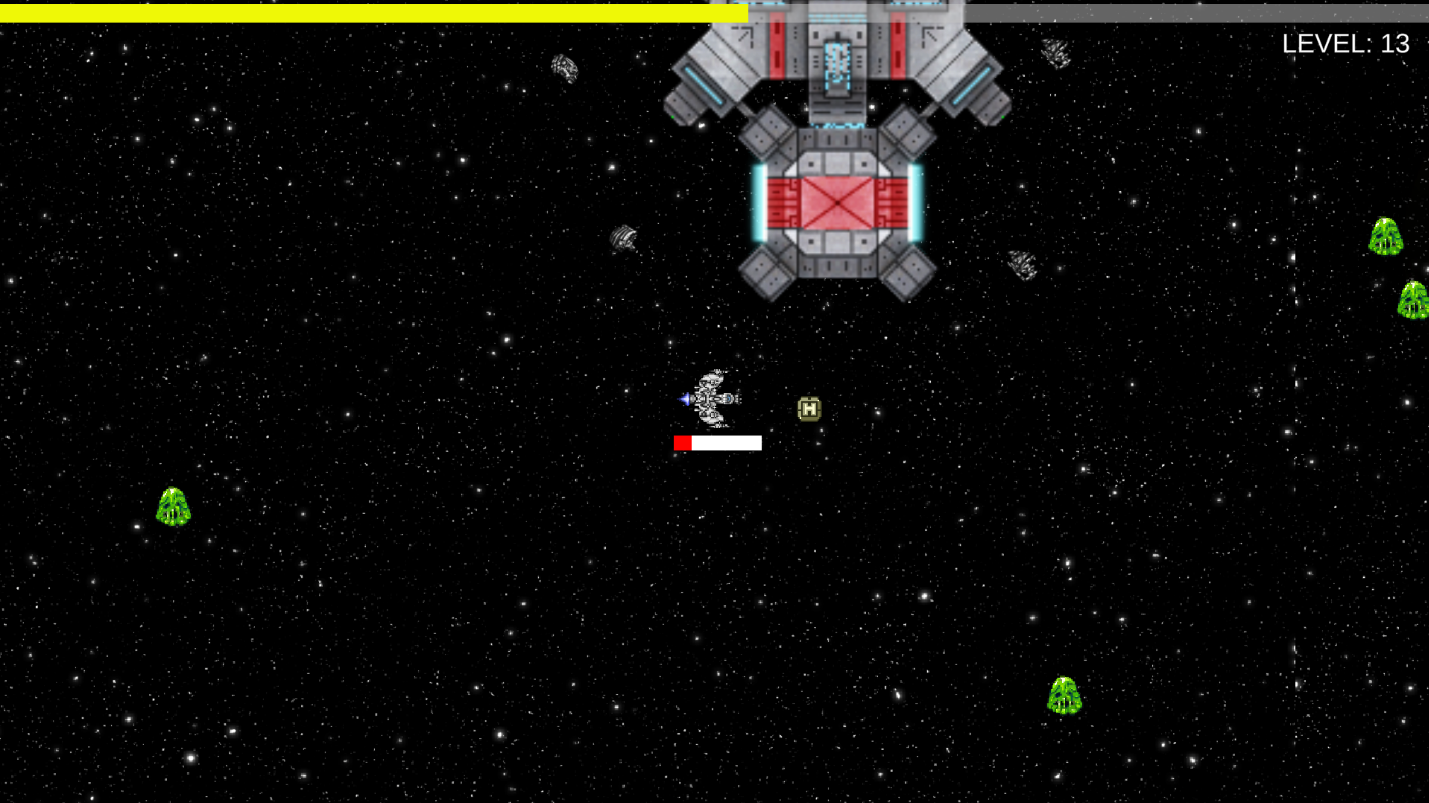
Rys.27 Pick-up’y doświadczenia oraz wrogowie  
Źródło: Własne

Gdy wrogowie zbliżą się do statku gracza, zaczną mu zadawać obrażenia. Aby uleczyć statek, należy znaleźć żółte obiekty ukryte na mapie. Jeśli je zaatakujemy, istnieje szansa na wyrzucenie pick-up'u zdrowia, który leczy naszą postać.



Rys.28 Żółty obiekt otoczenia do zniszczenia  
Źródło: Własne

Gdy żółty obiekt zostanie zaatakowany, ma szansę na wyrzucenie pick-up’u zdrowia.



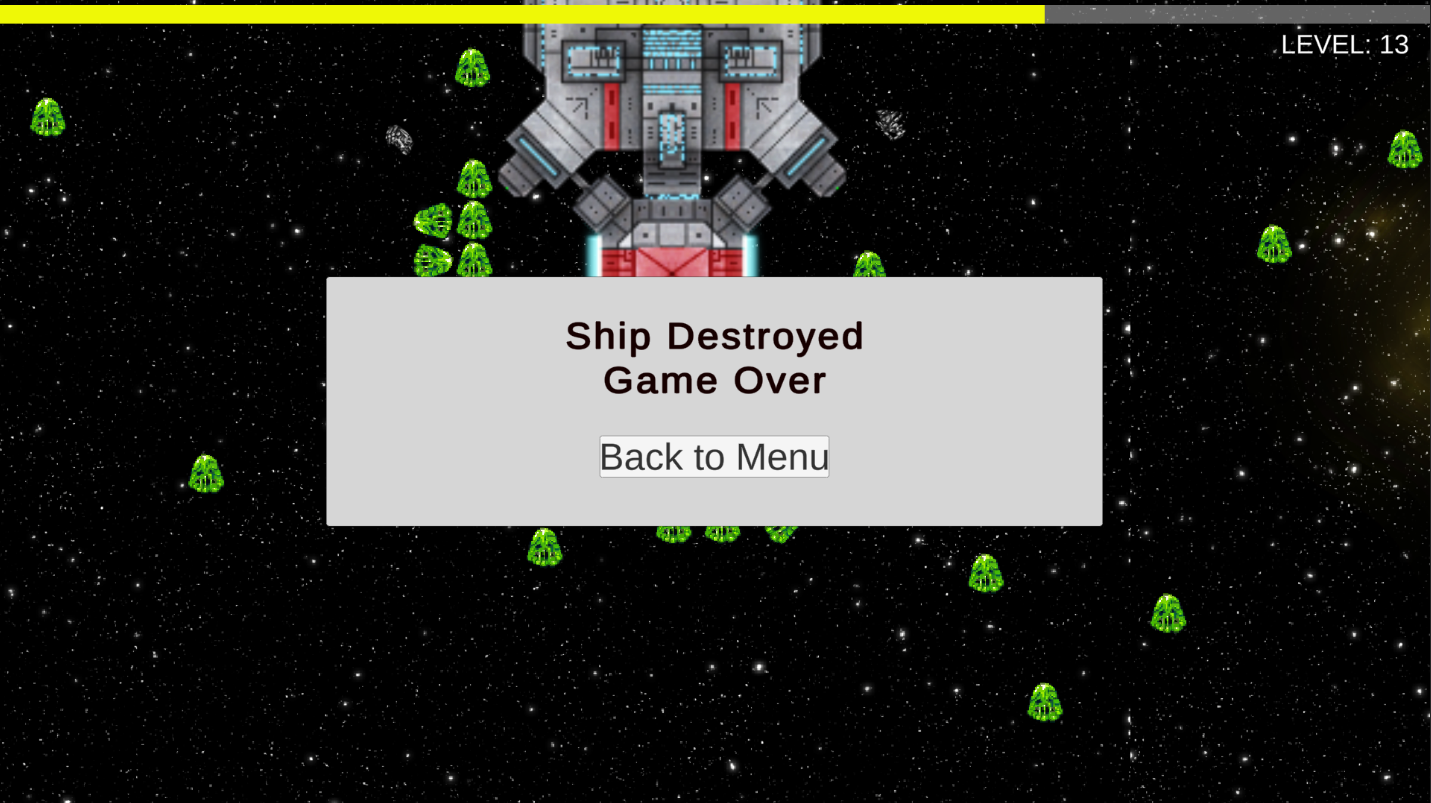
Rys.29 Pick-up zdrowia  
Źródło: Własne

Po zebraniu pick-up’u zwiększy się zdrowie statku, co za tym idzie pasek zdrowia także się zwiększy.



Rys.30 Zwiększenie paska zdrowia po podniesieniu pick-up’u  
Źródło: Własne

Gdy zdrowie statku spadnie do 0, pojawia się ekran, który wstrzymuje nasz ruch i broń. Wtedy możemy wcisnąć przycisk i wrócić do menu.



Rys.31 Ekran Game Over  
Źródło: Własne

## Podsumowanie

Praca inżynierska dotyczyła stworzenia gry typu survival 2D top-down shooter w Unity. Głównym celem projektu było zaprojektowanie i zaimplementowanie interaktywnej rozgrywki, w której gracz wciela się w statek kosmiczny i stawia czoło hordom wrogów.

W grze zastosowano mechanikę automatycznego strzelania, w której gracz porusza się na płaszczyźnie 2D w ośmiu kierunkach. Wrogowie poruszają się prosto na gracza, a kontakt z nimi powoduje zadawanie obrażeń. Gracz musi uważać na swoje zdrowie i unikać kolizji z przeciwnikami, ponieważ przekroczenie pewnego poziomu obrażeń prowadzi do przegranej.

Gra oferuje nieskończoną mapę dzięki funkcji autoscrollingu 9 tile'ów. Na mapie znajdują się również skrzynie, które można zniszczyć za pomocą broni. Zniszczenie skrzyni ma szansę wygenerowania pick-upa, który leczy postać gracza. Zabijanie wrogów pozwala zdobywać doświadczenie, a po jego pełnym wypełnieniu gracz zdobywa kolejny poziom.

W trakcie pracy inżynierskiej zostałem zainspirowany grą "Vampire Survivor", która ma podobny typ rozgrywki, jednak moim celem było przeniesienie akcji do kosmosu, z pełnym poruszaniem się w dziewięciu kierunkach.

W wyniku pracy udało mi się zaimplementować wszystkie założone funkcje i stworzyć grywalny prototyp gry. W trakcie tworzenia projektu wykorzystałem różne technologie i narzędzia oferowane przez Unity, takie jak skrypty, komponenty fizyki, animacje, kolizje, kamery i wiele innych.

Praca inżynierska była okazją do poszerzenia mojej wiedzy i umiejętności w dziedzinie tworzenia gier, a także do zastosowania zdobytej wcześniej wiedzy teoretycznej w praktyce. Projekt ten dostarczył mi cennego doświadczenia w projektowaniu i programowaniu gier, a także umożliwił mi rozwinięcie umiejętności pracy z narzędziami Unity.

# Literatura

1. <https://docs.unity.com> z dnia 15.05.2023
2. <https://www.unrealengine.com/en-US> z dnia 15.05.2023
3. <https://code.visualstudio.com> z dnia 15.05.2023
4. <https://visualstudio.microsoft.com/pl/https://code.visualstudio.com> z dnia 15.05.2023
5. <https://www.jetbrains.com/rider/> z dnia 15.05.2023
6. <https://www.adobe.com/pl/products/photoshop.html> z dnia 15.05.2023
7. <https://www.gimp.org/docs/> z dnia 15.05.2023
8. <https://en.wikipedia.org/wiki/Vampire_Survivors> z dnia 15.05.2023
9. <https://play.google.com/store/apps/details?id=com.dxx.firenow&hl=en_US> z dnia 15.05.2023
10. <https://docs.unity3d.com/ScriptReference/GameObject.html> z dnia 15.05.2023
11. <https://docs.unity3d.com/Manual/class-PlayerSettings.html> z dnia 15.05.2023
12. <https://docs.unity3d.com/ScriptReference/Camera.html> z dnia 15.05.2023
13. <https://docs.unity3d.com/Manual/UIElements.html> z dnia 15.05.2023
14. <https://docs.unity3d.com/ScriptReference/Collider.html> z dnia 15.05.2023
15. <https://docs.unity3d.com/Manual/class-Rigidbody.html> z dnia 15.05.2023
16. <https://docs.unity3d.com/Manual/terrain-UsingTerrains.html> z dnia 15.05.2023
17. <https://docs.unity3d.com/Manual/Lighting.html> z dnia 15.05.2023
18. <https://docs.unity3d.com/ScriptReference/Transform.html> z dnia 15.05.2023
19. <https://docs.unity3d.com/ScriptReference/GameObject.Find.html> z dnia 15.05.2023
20. <https://docs.unity3d.com/ScriptReference/MonoBehaviour.html> z dnia 15.05.2023

# Streszczenie

**Wyższa Szkoła Informatyki i Zarządzania z siedzibą w Rzeszowie  
Kolegium Informatyki Stosowanej**

**Streszczenie pracy dyplomowej**

„Space Survivor” – gra 2D typu top-down shooter

**Autor** Mateusz Krok  
**Promotor:** Dr Marek Jaszuk  
**Słowa kluczowe:** gra komputerowa, Unity Game Engine, gra 2D, silnik gier komputerowych

W celu wykonania pracy inżynierskiej stworzyłem grę 2D inspirowaną „Vampire Survivors”. Projekt obejmuje prosty zarys gry, z zaimplementowanymi głównymi jej mechanikami. Są to m.in.:

* Poruszanie się po mapie w 9 kierunkach
* Nieskończona przewijająca się mapa
* Hordy wrogów kierujące się na gracza
* Zdobywanie doświadczenia za pokonywanie wrogów
* Utrzymywanie zdrowia gracza

Całość została wykonana przy pomocy silnika gier komputerowych Unity. Język programowania użyty do skryptów to C#.

W początkowej części pracy przedstawiono cel i zakres projektu oraz wymagania funkcjonalne i niefunkcjonalne. Następnie omówiono wybór narzędzi. Kolejnym etapem było opisanie projektowania rozwiązań, a na zakończenie przedstawiono prezentację graficzną całego projektu.

1. <https://docs.unity.com> (15.03.2023) [↑](#footnote-ref-1)
2. <https://www.unrealengine.com/en-US> (15.03.2023) [↑](#footnote-ref-2)
3. <https://code.visualstudio.com> (15.03.2023) [↑](#footnote-ref-3)
4. <https://visualstudio.microsoft.com/pl/> (15.03.2023) [↑](#footnote-ref-4)
5. <https://www.jetbrains.com/rider/> (15.03.2023) [↑](#footnote-ref-5)
6. <https://www.adobe.com/pl/products/photoshop.html> (15.03.2023) [↑](#footnote-ref-6)
7. <https://www.gimp.org/docs/> [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Vampire_Survivors> (15.03.2023) [↑](#footnote-ref-8)
9. <https://play.google.com/store/apps/details?id=com.dxx.firenow&hl=en_US> (15.03.2023) [↑](#footnote-ref-9)
10. <https://docs.unity3d.com/ScriptReference/GameObject.html> (15.03.2023) [↑](#footnote-ref-10)
11. <https://docs.unity3d.com/Manual/class-PlayerSettings.html> (15.03.2023) [↑](#footnote-ref-11)
12. <https://docs.unity3d.com/ScriptReference/Camera.html> (15.03.2023) [↑](#footnote-ref-12)
13. <https://docs.unity3d.com/Manual/UIElements.html> (15.03.2023) [↑](#footnote-ref-13)
14. <https://docs.unity3d.com/ScriptReference/Collider.html> (15.03.2023) [↑](#footnote-ref-14)
15. <https://docs.unity3d.com/Manual/class-Rigidbody.html> (15.03.2023) [↑](#footnote-ref-15)
16. <https://docs.unity3d.com/Manual/terrain-UsingTerrains.html> (15.03.2023) [↑](#footnote-ref-16)
17. <https://docs.unity3d.com/Manual/Lighting.html> (15.03.2023) [↑](#footnote-ref-17)
18. <https://docs.unity3d.com/ScriptReference/Transform.html> (15.03.2023) [↑](#footnote-ref-18)
19. <https://docs.unity3d.com/ScriptReference/GameObject.Find.html> (15.03.2023) [↑](#footnote-ref-19)