Studia i Materiały Informatyki Stosowanej, Tom 17, Nr 2, 2025
str. 15-20

IMPLEMENTACJA GRY WIRTUALNEJ RZECZYWISTOŚCI Z WYKORZYSTANIEM SILNIKA UNITY

Julia Cieślicka[1], Bartosz Przybyłek[1], Katarzyna Kazimierska-Drobny[2], Piotr Prokopowicz[1]

Uniwersytet Kazimierza Wielkiego
[1] Wydział Informatyki, [2] Wydział Mechatroniki
ul. Kopernika 1, 85-074 Bydgoszcz
e-mail: julia.cieslicka@student.ukw.edu.pl

Streszczenie: *W niniejszym artykule szczegółowo omówiono proces projektowania i implementacji gry wirtualnej rzeczywistości z wykorzystaniem silnika Unity oraz dodatkowych narzędzi programistycznych. Skupiono się na zagadnieniach technicznych, takich jak wykorzystanie XR Interaction Toolkit, Blender oraz Visual Studio, jednocześnie zwracając uwagę na wyzwania związane z projektowaniem ergonomicznego interfejsu i optymalizacji środowiska VR. Głównym celem pracy było stworzenie intuicyjnej i immersyjnej gry skierowanej do szerokiej grupy odbiorców, w tym osób nieposiadających wcześniejszego doświadczenia z wirtualną rzeczywistością. Wyniki pracy wskazują na duży potencjał technologii VR w kontekście edukacyjnym, rozrywkowym i przemysłowym.*

Słowa kluczowe: *wirtualna rzeczywistość, VR, Unity, gra komputerowa, C#, grafika 3D, Blender*

IMPLEMENTATION OF A VIRTUAL REALITY GAME USING THE UNITY ENGINE

Abstarct: *This article provides a detailed discussion of the process of designing and implementing a virtual reality game using the Unity engine and additional programming tools. The focus is placed on technical aspects, such as the use of XR Interaction Toolkit, Blender, and Visual Studio, while also addressing challenges related to designing an ergonomic interface and optimizing the VR environment. The main goal of the project was to create an intuitive and immersive game targeted at a wide audience, including individuals with no prior experience in virtual reality. The results of the work highlight the significant potential of VR technology in educational, entertainment, and industrial contexts.*

Keywords: *virtual reality, VR, Unity, video game, C#, 3D graphics, Blender*

1. Wstęp

Technologia wirtualnej rzeczywistości (VR) rozwija się dynamicznie, znajdując zastosowanie w wielu dziedzinach życia, takich jak edukacja, medycyna, szkolenia czy rozrywka. VR umożliwia użytkownikom zanurzenie się w wirtualnym świecie, dostarczając unikalnych doznań, które w tradycyjnych mediach są trudne do osiągnięcia. Jednym z najważniejszych aspektów tej technologii jest jej potencjał do dostarczania angażujących i interaktywnych doświadczeń, co sprawia, że VR jest szczególnie atrakcyjna w branży gier komputerowych. Gry komputerowe, w tym także gry VR, są dla wielu osób formą rozrywki, relaksu oraz sposobem na chwilowe oderwanie się od codziennych obowiązków. Dla niektórych graczy stanowią one także środek do wyrażania kreatywności, uczenia się nowych umiejętności lub nawiązywania kontaktów społecznych [1-3]. Współczesne badania wskazują, że gry mogą pełnić funkcje terapeutyczne, wspierać rozwój poznawczy czy nawet poprawiać zdolności motoryczne [4]. Popularność gier wirtualnej rzeczywistości wynika z ich zdolności do oferowania wyjątkowego poziomu immersji, który sprawia, że gracz czuje się częścią przedstawionego świata. Kluczowym wyzwaniem przy projektowaniu aplikacji VR jest stworzenie środowiska, które zapewni wysoki poziom immersji oraz intuicyjną interakcję z wirtualnym światem. Intuicyjność jest szczególnie istotna, ponieważ wirtualna rzeczywistość przyciąga zarówno zaawansowanych graczy, jak i osoby

bez wcześniejszego doświadczenia z tego typu technologią. Dostosowanie interfejsu i mechaniki rozgrywki do potrzeb szerokiej grupy odbiorców wymaga zastosowania zaawansowanych narzędzi i technik projektowych.

W pracy skupiono się na stworzeniu gry VR, w której gracz realizuje zadania polegające na budowie obiektów w środowisku wirtualnym. Istotnym elementem projektu było opracowanie ergonomicznego systemu interakcji oraz wykorzystanie nowoczesnych narzędzi, takich jak silnik Unity i XR Interaction Toolkit, które wspierają rozwój aplikacji VR. Dodatkowo zwrócono uwagę na optymalizację środowiska gry, aby zapewnić komfort użytkowania oraz minimalizować potencjalne problemy, takie jak choroba symulatorowa.

2. Wirtualna rzeczywistość, możliwości i zastosowania

Wirtualna rzeczywistość (VR) to technologia umożliwiająca tworzenie symulacji środowisk rzeczywistych lub fikcyjnych, zwiększając immersję użytkownika, czyli jego „zanurzenie" w symulacji. Zbliżone technologie to AR (rzeczywistość rozszerzona) i MR (rzeczywistość mieszana), które łączą elementy świata realnego z cyfrowym. Wszystkie te pojęcia zbiorczo określa się jako XR (rozszerzona rzeczywistość) [1-3]. Sprzęt VR Podstawowym elementem zestawów VR są gogle (HMD), które dzielą się na:

- Zestawy mobilne, wykorzystujące urządzenia takie jak smartfony.
- Zestawy autonomiczne, które działają samodzielnie dzięki wbudowanemu procesorowi i pamięci.
- Zestawy PCVR, oferujące największe możliwości, wymagające połączenia z komputerem. Wybór odpowiedniego zestawu zależy od potrzeb użytkownika, np. autonomiczne zestawy są idealne dla początkujących, a PCVR dla zaawansowanych graczy.

Widok 3D

Gogle VR wyświetlają obraz stereoskopowy na dwóch ekranach (po jednym dla każdego oka), co wymaga podwójnego renderowania i zwiększonej mocy obliczeniowej. Technika ta tworzy iluzję głębi i widoku 3D, kluczową dla realistycznych doświadczeń VR.

Zastosowania VR

VR znajduje zastosowanie w edukacji, szkoleniach BHP, medycynie (np. terapia ekspozycyjna, redukcja bólu), architekturze czy turystyce [5, 6]. W rozrywce VR rozwija się dynamicznie, a liczba użytkowników stale rośnie, co potwierdzają dane ze Steama o wzroście sprzedaży gier VR o 71% w 2020 roku [9].

Wyzwania

Do najczęstszych problemów należą objawy choroby symulatorowej (nudności, zawroty głowy), które dotykają około połowy użytkowników [7]. Ryzyko można ograniczyć, tworząc komfortowe i bezpieczne aplikacje VR oraz dostosowując warunki użytkowania, np. zmniejszając ruchy głowy [8].

OpenXR

OpenXR to standard ujednolicający interfejsy aplikacji XR, umożliwiający ich działanie na różnych platformach. Silnik Unity, wykorzystywany w niniejszej pracy, wspiera OpenXR, co ułatwia konfigurację i integrację aplikacji VR.

3. Opis narzędzi

Unity: Silnik Unity był głównym narzędziem do budowy gry. Wybrano XR Interaction Toolkit jako kluczowy element umożliwiający obsługę interakcji użytkownika z wirtualnym światem [10].

Blender: Blender został wykorzystany do tworzenia modeli 3D, animacji oraz tekstur obiektów używanych w grze [11].

Visual Studio i C#: Visual Studio było podstawowym środowiskiem programistycznym. Skrypty w języku C# obsługiwały logikę gry, takie jak zarządzanie mechaniką rozgrywki czy integracja z interfejsem [12,13].

OpenXR: Standard OpenXR zapewnił kompatybilność gry z różnymi platformami VR, co pozwoliło na elastyczne dopasowanie aplikacji do urządzeń takich jak Oculus czy HTC Vive [14].

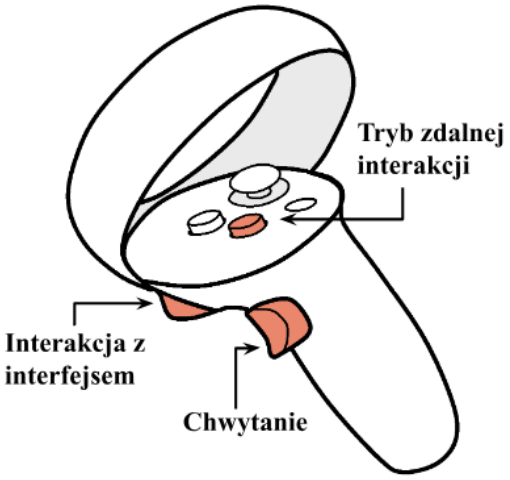
BoscaCeoil: Program ten wspierał tworzenie prostych efektów dźwiękowych i muzyki w grze, co wzbogaciło doświadczenia immersyjne [15].

4. Projekt i założenia projektowe

Gra polega na zdobywaniu punktów poprzez

składanie zabawek w fabryce w określonym czasie. Gracz może manipulować częściami zabawek zarówno bezpośrednio, jak i za pomocą zdalnej interakcji (promień pozwalający na sięganie po elementy spoza zasięgu rąk). Rozgrywka uwzględnia trzy typy zabawek: plastikowy robot, pluszowy miś oraz drewniany żołnierz, które składają się z sześciu części (tułów, głowa, ręce, nogi). Pełne punkty przyznawane są za poprawnie złożone zabawki z elementów jednej kategorii. Gra premiuje różnorodność składanych zabawek (np. za złożenie robota i misia po sobie).

Gracz korzysta z interfejsu z tablicą (menu, timer, licznik punktów) oraz pasa podającego, z którego losowo spadają części zabawek. Elementy przesuwają się w dół pasa i zostają zniszczone, jeśli nie zostaną użyte. Gracz musi planować strategię i szybko podejmować decyzje, aby zebrać potrzebne części. Po prawej stronie znajduje się pudełko prezentowe, do którego wrzuca się złożone zabawki, co podlicza punkty. Czas rozgrywki wynosi 5 minut, a po zakończeniu wynik jest porównywany z dotychczasowym rekordem. Na rysunku 1 przedstawiono schemat sterowania w grze na podstawie kontrolera Meta Quest 2.



Rys.1. Schemat sterowania w grze na podstawie kontrolera Meta Quest 2.

Sterowanie zostało uproszczone, aby było intuicyjne dla początkujących użytkowników VR. Projekt zakłada minimalizację ruchów gracza, co zwiększa komfort i zmniejsza ryzyko choroby symulatorowej. Gra wykorzystuje jedną scenę, aby uniknąć ekranów ładowania i zwiększyć immersję.

5. Implementacja gry

W pierwszej kolejności skonfigurowano środowisko pracy w Unity, dostosowując ustawienia projektu do specyfiki platform VR. Projekt został zoptymalizowany pod urządzenia obsługujące XR, takie jak Meta Quest 2. Do zarządzania wersjami wykorzystano system kontroli wersji PlasticSCM, rekomendowany przez Unity Technologies. Dodano pakiety XR Plugin Framework i XR Interaction Toolkit, kluczowe dla implementacji interakcji w środowisku wirtualnej rzeczywistości. Jako interfejs komunikacyjny z urządzeniem VR wybrano OpenXR, co zapewniło kompatybilność z szeroką gamą urządzeń. Ustawienia obejmowały również konfigurację platformy docelowej, jakości graficznej, oraz parametrów fizyki w scenie.

Silnik Unity bazuje na Obiektach Gry (GameObjects) i komponentach. Każdy obiekt zawiera komponent Transform, który definiuje jego położenie, rotację i skalę. Komponenty nadają obiektom cechy i zachowania. Wymieniono najczęściej używane komponenty oferowane przez Unity, takie jak MeshRenderer, Animator, Audio Source, Rigidbody, i Collider. Scena zawiera hierarchię Obiektów Gry, w tym GameManager, XR Origin, XR Interaction Manager, Object Spawner, UI Canvas, Global Volume, Reflection Probe, Gift Box, Toy Discard Bin, Environment, i Event System. Każdy z tych obiektów pełnił określoną rolę w grze.

W głównej scenie projektu wyróżniono następujące Obiekty Gry i ich hierarchie:

- GameManager - obiekt zawierający komponent o tej samej nazwie, odpowiedzialny za sterowanie przebiegiem gry
- XR Origin - hierarchia obiektów odpowiedzialnych za przetwarzanie danych wejściowych z urządzeń VR na odpowiednie funkcje gry
- XR Interaction Manager - komponent pakietu XR Interaction Toolkit, zarządza systemem interakcji
- Object Spawner - jest to obiekt wczytujący i umieszczający obiekty w scenie z danej puli obiektów
- UI Canvas - hierarchia obiektów służących jako interfejs gracza
- Global Volume - obiekt pozwalający na dodanie efektów wizualnych do wyświetlanego obrazu, tzw. przetwarzanie końcowe

- ReflectionProbe - obiekt odpowiedzialny za realistyczne symulowanie odbić światła na powierzchniach obiektów
- Gift Box - obiekt wyznaczający docelowe miejsca oddania złożonej przez gracza zabawki
- ToyDiscard Bin - obiekt wyznaczający obszar, który wchodząc w kolizję z obiektem części zabawki usuwa ją ze sceny
- Environment - hierarchia obiektów odpowiedzialnych za wyświetlanie modeli 3D otoczenia oraz oświetlenia
- Event System - obiekt umożliwiający obsługę zdarzeń Unity w scenie, niezbędny do obsługi interfejsu gracza

Obiekt XR Origin i jego hierarchia zawierały komponenty reprezentujące położenie gracza. W hierarchii umieszczono główną kamerę sceny, która definiuje punkt widzenia. Do obsługi danych wejściowych z urządzeń peryferyjnych wykorzystano Unity Input System, definiując akcje w pliku Input Action Asset. Pakiet XR Interaction Toolkit zawierał plik Input ActionsAsset ze zdefiniowanymi akcjami. Obiekt kamery zawierał komponent TrackedPose Driver. W hierarchii obiektu XR Origin znajdowały się obiekty reprezentujące kontrolery gracza, w tym modele dłoni, interaktory (Direct Interactor i Ray Interactor), oraz komponenty XR Controller. Zaimplementowano funkcję przełączania się między interaktorami za pomocą komponentu RayInteractorToggler.

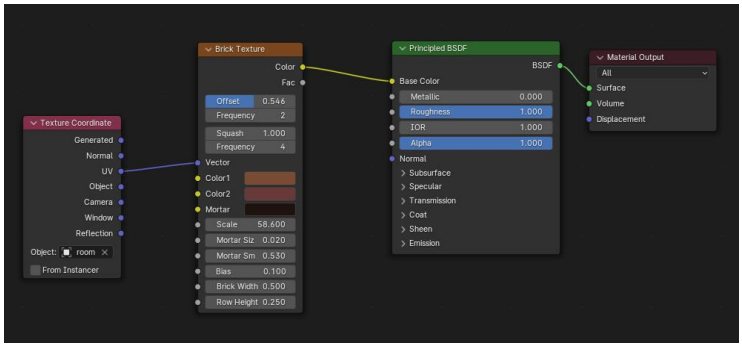
W komponencie GameManager zdefiniowano zmienne i metody odpowiedzialne za przebieg gry, manipulację punktacją, m.in. SubmitToy, SetScore, AddScore, BeginGame, UpdateTimer, i EndGame. Zaimplementowano zapisywanie i wyświetlanie rekordowego wyniku za pomocą klasy PlayerPrefs. Wykorzystano komponent Canvas do tworzenia interfejsu użytkownika. Umieszczono dwa obiekty UI Canvas jako kontenery dla elementów interfejsu użytkownika (interaktywne i nieinteraktywne). Elementy nieinteraktywne to obiekty wyświetlające tekst, a interaktywne elementy to obiekty zawierające komponent Button. Za pomocą przycisków gracz mógł rozpocząć grę, zamknąć aplikację, zmienić poziom głośności dźwięków, i zresetować wynik.

Wyróżniono komponenty XRSimpleInteractable oraz XRGrabInteractable. Na funkcję składania zabawek składały się trzy komponenty: ToyBase, ToyAttachPoint, i ToyPart. Klasa ToyPart dziedziczyła z klasy XRGrabInteractable. Każdy obiekt tułowia zabawki miał w hierarchii pięć obiektów z komponentem ToyAttachPoint. Utworzono klasę ConveyorBelt reprezentującą linię produkcyjną. Obiekt Gift Box reprezentował pudełko, do którego gracz wrzucał gotową zabawkę. Obiekt Gift Box zawierał komponent Animator odpowiedzialny za sterowanie animacjami w oparciu o drzewo decyzyjne.

Modele 3D były tworzone w programie Blender, gdzie każda zabawka składała się z oddzielnych siatek (mesh) dla różnych części, takich jak tułów, głowa, ramiona i nogi, co umożliwiało łatwą manipulację nimi w silniku gry. Modele zaprojektowano z zachowaniem dobrej topologii, czyli odpowiedniej struktury wielokątów, w celu zredukowania obciążenia procesora i zwiększenia płynności rozgrywki. Średnio jedna zabawka składa się z około 2400 trójkątów, a cała scena zawiera około 40 000 trójkątów, co jest stosunkowo niewielką liczbą, która zapewnia płynną rozgrywkę nawet na mniej wydajnych urządzeniach.

Każdy model posiada własną teksturę, która określa jego kolor powierzchni. Proces ten wymagał mapowania modelu 3D na powierzchnię 2D za pomocą mapy UV przed nałożeniem tekstur. Tekstury zostały stworzone przy użyciu wbudowanych narzędzi Blendera, co pozwoliło na bezpośrednie malowanie kolorów na modelach, które następnie zostały zapisane jako pliki PNG o rozdzielczości 1024x1024.

W zasięgu rąk gracza znajduje się taśmociąg, stół roboczy i tablica. Elementy otoczenia zostały zaprojektowane z uwzględnieniem rzeczywistych rozmiarów, aby zwiększyć immersję gracza. Tekstury dla otoczenia zostały stworzone za pomocą proceduralnych tekstur Blendera w Shader Editor. Shader Editor umożliwia manipulowanie tym, jak obiekty są renderowane. Na przykład, tekstury dla elementów takich jak ceglane ściany zostały stworzone przez połączenie węzłów, aby uzyskać efekty takie jak kolor i skala. Do ściany dodano teksturę szumu, aby uzyskać efekt 3D wystających cegieł. Tekstura podłogi została stworzona przez połączenie tekstur w kratkę, tekstury szumu i tekstury cegły.
Na rysunku 2 przedstawiono narzędzie Shader Editor tworzącego teksturę ceglanej ściany.



Rys. 2. Shader Editor.

Na rysunku 3 przedstawiono ścianę z dodanymi efektami 3D. Taki efekt uzyskano poprzez łączenie węzłów w narzędziu Shader Editor pokazanym na rysunku 2.

Rys. 3. Widok ściany z efektem 3D.

Taśmociąg używa powtarzającej się tekstury, która wydaje się przesuwać po jego powierzchni. Efekt ten osiągnięto za pomocą niestandardowego shadera, który manipuluje pozycją tekstury. Materiał dla taśmociągu został stworzony przy użyciu ShaderGraph. Tekstury stworzone w Blenderze zostały "upieczone" (ang. baked) do plików graficznych w celu ich użycia w silniku gry Unity. Proces ten tworzy tekstury kolorów oraz mapy normalnych (ang. normals). Mapy normalnych przechowują dane 3D jako wartości RGB, dodając szczegóły bez zwiększania złożoności modeli 3D. W Unity materiały są używane jako dane wejściowe dla shaderów, określając sposób, w jaki obiekty prezentowane są w świecie gry. Ostateczny pokój gry jest oświetlony ciepłym światłem i wzbogacony efektami post-processingu, takimi jak winietowanie, zmniejszona nasycenie kolorów i mgła. Finalny widok gry w edytorze Unity możemy zobaczyć na rysunku 4.

Rys. 4. Widok gry w edytorze Unity

Model ręki przygotowano poprzez dodanie armatury (szkieletu) w Blenderze. Przypisano ciężary (weight) mesha do kości, używając "weight painting". Stworzono 3 pozycje rąk. Po dodaniu modelu do Unity wybrano klatki pozycji i odpowiednio je nazwano. Dodano węzeł Blend Tree w drzewie decyzyjnym komponentu Animator. Dla modelu dłoni utworzono drugą warstwę obejmującą jedynie kości palca wskazującego. Utworzono komponent "HandAnimationHandler" do przekazywania wartości z kontrolerów do parametrów animatora.

Dźwięki do gry stworzono w programie BoscaCeoil. Niektóre odgłosy nagrano przy użyciu fizycznych przedmiotów. Do obiektu kamery dodano komponent Audio Listener, obiekty wydające dźwięk otrzymały komponent Audio Source. Dodano dźwięki interakcji z interfejsem, dołączania i odłączania elementów zabawki, poprawnego złożenia zabawki, oraz dźwięki kolizji zabawki z otoczeniem.

Aby zapewnić płynność działania gry, zastosowano następujące optymalizacje:

- Redukcja liczby wielokątów w modelach 3D poprzez stosowanie uproszczonych siatek dla obiektów tła.
- Dynamiczne wczytywanie zasobów w celu minimalizacji zużycia pamięci RAM.
- Optymalizacja oświetlenia sceny poprzez zastosowanie map światła (lightmaps).
- Kompresja tekstur, aby zmniejszyć ich rozmiar bez utraty jakości wizualnej.

Prototyp został przetestowany na urządzeniach docelowych, takich jak Meta Quest 2, w celu weryfikacji poprawności działania i identyfikacji potencjalnych problemów. Testy obejmowały sprawdzenie ergonomii interakcji, płynności animacji oraz komfortu użytkowania. Zebrane dane posłużyły do dalszej optymalizacji projektu i usunięcia błędów.

6. Analiza wyników i perspektywy rozwoju

Efektem pracy było stworzenie gry VR, która skutecznie łączy nowoczesne technologie z przyjaznym dla użytkownika projektem. Analiza wskazuje, że zastosowane narzędzia pozwalają na elastyczne rozwijanie gry w następujących obszarach:

- Rozszerzenie zawartości: Dodanie nowych rodzajów zadań oraz zabawek, co zwiększy różnorodność rozgrywki.

- Integracja sztucznej inteligencji: Wprowadzenie AI mogłoby umożliwiać dynamiczne dostosowywanie poziomu trudności gry, co przyczyniłoby się do dłuższego zaangażowania graczy.
- Tryb wieloosobowy: Implementacja rozgrywki sieciowej pozwoliłaby na współpracę lub rywalizację między graczami.
- Rozszerzona analiza danych: Użycie narzędzi analitycznych mogłoby wspierać lepsze zrozumienie preferencji graczy i optymalizację rozgrywki.

7. Podsumowanie

Celem niniejszej pracy było przeanalizowanie wyzwań związanych z technologią wirtualnej rzeczywistości oraz stworzenie gry przeznaczonej na tę platformę z wykorzystaniem silnika Unity. Projekt skupiał się na opracowaniu immersyjnego środowiska wirtualnego, które w pełni wykorzystuje możliwości VR do dostarczania interaktywnych doświadczeń. Realizacja pracy obejmowała zagadnienia techniczne, takie jak obsługa urządzeń VR, integracja z silnikiem Unity oraz proces tworzenia graficznych elementów dedykowanych grom komputerowym.

Literatura

[1] Linowes, J. *Unity Virtual Reality projects.*(2018). Packt Publishing.

[2] Etchells, P. *Lost in a good game: Why we play video games and what they can do for us.* (2019). Icon Books.

[3] Gerhard, D., & Norton, W. J. *Virtual Reality Usability Design*. (2022). CRC Press.

[4] The Board of Trustees of the University of Illinois, *National Center for Supercomputing Applications: History https://web.archive.org/web/20150821054144/http://archive.ncsa.illinois.edu/Cyberia/VETopLevels/VR.History.html*

[5] Boeldt D, McMahon E, McFaul M, Greenleaf W. *Using Virtual Reality Exposure Therapy to Enhance Treatment of Anxiety Disorders: Identifying Areas of Clinical Adoption and Potential Obstacles.*Front Psychiatry. 2019 Oct 25;10:773. doi: 10.3389/fpsyt.2019.00773. PMID: 31708821; PMCID: PMC6823515. *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6823515/*

[6] Li A, Montaño Z, Chen VJ, Gold JI.*Virtual reality and pain management: current trends and future directions.*PainManag. 2011 Mar;1(2):147-157. doi: 10.2217/pmt.10.15. PMID: 21779307; PMCID: PMC3138477. *https://pubmed.ncbi.nlm.nih.gov/21779307/*

[7] William Bruce Stone III *Psychometric evaluation of the Simulator Sickness Questionnaire as a measure of cybersickness* (2017) Iowa State University *https://dr.lib.iastate.edu/server/api/core/bitstreams/ee68bdd0-a92e-4bd2-9f1a-448476e5051d/content*

[8] Merhi, O., Faugloire, E., Flanagan, M., & Stoffregen, T. A. *Motion Sickness, Console Video Games, and Head-Mounted Displays.*(2007) Human Factors, 49(5), 920-934. *https://doi.org/10.1518/001872007X230262*

[9] Valve Corporation, *Steam - 2020 Year in Review https://store.steampowered.com/news/group/4145017/view/2961646623386540826*

[10] Unity Technologies, *Unity User Manual https://docs.unity3d.com/Manual/index.html*

[11] The Khronos® OpenXR Working Group, *The OpenXR™ Specification https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html*

[12] Blender Foundation, *Blender Manual https://docs.blender.org/manual/en/latest/?utm_medium=www-footer*

[13] Microsoft, *Visual Studio documentation https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022*

[14] Microsoft, *C# Guide https://learn.microsoft.com/en-us/dotnet/csharp/*

[15] Terry Cavanagh, *Bosca Ceoil https://terrycavanagh.itch.io/bosca-ceoil*