Studia i Materiały Informatyki Stosowanej, Tom 17, Nr 2, 2025
str. 27-32

UKW HALLOWEEN: GRA KOMPUTEROWA 3D OPARTA NA UNITY

Maksym Beliavskyi[1], Daniil Skrendo[1], Ekaterina Smirnova[1], Michał Pakuła[2]

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

Streszczenie: *Artykuł omawia projekt i implementację gry komputerowej 3D „UKW Halloween" opartej na silniku Unity, która powstała jako praca inżynierska[51]. Opracowana gra, z gatunku horror przenosi gracza do jednego z budynków Uniwersytetu Kazimierza Wielkiego w Bydgoszczy - Copernicanum, którego bryła oraz rozkład wewnętrznych pomieszczeń został odwzorowany na podstawie rzeczywistych danych architektonicznych. W pracy omówiono założenia projektowe, zastosowane technologie informatyczne, mechaniki gry oraz proces testowania. Wyniki przeprowadzonych testów potwierdzają skuteczność zastosowanego narzędzia (środowiska programistycznego Unity) do opracowywania gier komputerowych oraz pokazują możliwości dalszego rozwoju projektu.*

Słowa kluczowe: *Słowa kluczowe: Unity, horror, modelowanie 3D, mechanika gier, Copernicanum*

UKW HALLOWEEN: 3D COMPUTER GAME BASED ON UNITY

Abstarct: *The article describes the design and implementation of a 3D computer game, "UKW Halloween", created using the Unity engine, which was developed within the BSc thesis. The horror computer game takes the player to one of the Casimir the Great University buildings in Bydgoszcz - Copernicanum, whose shape and layout of internal rooms were recreated based on actual architectural data. The work discusses the design assumptions, IT technologies, game mechanics and the testing process. The test results confirm the effectiveness of the tool used (Unity programming environment) for creating computer games and show the possibilities for further project development.*
Keywords: *Unity, horror, 3D modeling, game mechanics, Copernicanum*

1. Wprowadzenie

Gry komputerowe od lat stanowią istotny element przemysłu rozrywkowego, a ich rozwój technologiczny umożliwia tworzenie coraz bardziej zaawansowanych i realistycznych produkcji [1]. Szczególną popularnością cieszą się gry z gatunku horror, które wykorzystują zarówno psychologiczne, jak i techniczne mechanizmy budowania napięcia [2]. Inspiracją do stworzenia gry „UKW Halloween" były klasyczne horrory z lat 90., w tym seria Silent Hill oraz ResidentEvil, które charakteryzowały się unikalnym wykorzystaniem ograniczonej widoczności oraz sterowania statycznymi kamerami w celu intensyfikacji doznań gracza [3]. Głównym celem projektu było stworzenie realistycznej gry grozy, której akcja rozgrywa się w autentycznym budynku Copernicanum. Wykorzystano nowoczesne narzędzia do modelowania oraz renderowaniaw wysokiej rozdzielczości, dzięki czemu możliwe było uzyskanie wciągającego klimatu i immersji [4]. Dodatkowo, silnik Unity pozwolił na implementację dynamicznego oświetlenia oraz efektów cząsteczkowych, co znacząco wpłynęło na jakość wizualną produkcji [5].

2. Projekt i implementacja gry

2.1 Założenia projektowe

Gra UKW Halloween została zaprojektowana jako survival horror osadzony w otoczeniu akademickim, mający na celu dostarczenie graczowi immersyjnego i angażującego doświadczenia. Główny bohater, inżynier informatyki, zostaje niespodziewanie uwięziony w budynku uczelni i zmuszony do rozwiązania szeregu zagadek logicznych oraz zręcznościowych, aby odnaleźć drogę ucieczki. Środowisko gry, inspirowane rzeczywistą architekturą budynku

Copernicanum, zostało wiernie odwzorowane na podstawie planów oraz dokumentacji fotograficznej, co pozwala graczowi poruszać się po realistycznie odtworzonych korytarzach i pomieszczeniach uczelni[6]. Kluczowym elementem rozgrywki jest mechanika zarządzania zasobami. Gracz dysponuje ograniczoną ilością energii latarki, co wymusza strategiczne podejście do eksploracji mrocznych zakamarków budynku. Ponadto, ekwipunek postaci został zaprojektowany tak, aby wymagał świadomego planowania – ilość przedmiotów, które można przenosić, jest ograniczona, a niektóre z nich mogą być zużywane lub wymagać łączenia w celu rozwiązania zagadek[7].

W grze zaimplementowano również dynamiczny system przeciwników oparty na sztucznej inteligencji. Wrogowie nie tylko patrolują określone obszary, ale także reagują na dźwięki i ślady pozostawione przez gracza, co sprawia, że unikanie konfrontacji staje się istotnym elementem rozgrywki. Ich zachowanie dostosowuje się do akcji gracza, co oznacza, że długotrwałe ukrywanie się w jednym miejscu może skutkować wzrostem ich czujności i intensywniejszymi poszukiwaniami[8].

Aby zwiększyć immersję, gra zawiera zaawansowany system fizyki pozwalający na eksplorację otoczenia oraz interakcję z przedmiotami. Gracze mogą otwierać szafki, przesuwać meble, zbierać i używać przedmiotów w kreatywny sposób. Interaktywność ta pozwala na nieliniowe podejście do rozwiązywania zagadek oraz odnajdywania ukrytych ścieżek. System ten został zintegrowany z mechaniką gry w taki sposób, aby gracz mógł wykorzystywać otoczenie zarówno do unikania przeciwników, jak i do rozwiązywania łamigłówek[9].

Dodatkowym aspektem projektu było stworzenie atmosfery grozy poprzez odpowiednie wykorzystanie dźwięku oraz oświetlenia. W grze zastosowano dynamiczny system oświetlenia, który zmienia się w zależności od postępów gracza. Cienie, migoczące światła i efekty świetlne budują napięcie oraz wpływają na poziom widoczności, co dodatkowo utrudnia eksplorację. Oprawa dźwiękowa składa się z ambientowych efektów dźwiękowych, szeptów, skrzypienia podłóg oraz nagłych dźwięków mających na celu wywołanie uczucia niepokoju i nieprzewidywalności.

Wszystkie te elementy zostały zaprojektowane w taki sposób, aby zapewnić spójne i angażujące doświadczenie dla gracza, jednocześnie podkreślając akademicki klimat miejsca akcji.

2.2 Technologie i narzędzia

W celu realizacji projektu wykorzystano szereg nowoczesnych technologii, które pozwoliły na efektywne opracowanie *UKW Halloween*. Wybrane narzędzia miały kluczowe znaczenie dla jakości, wydajności oraz immersji gry.

Unity, jako jeden z najpopularniejszych silników gier, stanowiło fundament projektu. Wybór tego narzędzia podyktowany był jego wszechstronnością, dostępnością oraz bogatą dokumentacją [12]. Gra wykorzystuje High Definition RenderPipeline (HDRP), co pozwoliło na uzyskanie wysokiej jakości efektów wizualnych [13]. HDRP umożliwiło implementację zaawansowanego oświetlenia, efektów wolumetrycznych oraz realistycznego cieniowania, co przyczyniło się do budowania atmosfery grozy [14].

W projekcie zaimplementowano niestandardowe shadery, które zostały stworzone przy użyciu ShaderGraph i HLSL [15]. Użyto także systemu post-processingu, który odpowiadał za efekty takie jak aberracja chromatyczna, efekt ziarnistości oraz winietowanie [16]. Dzięki wbudowanemu systemowi prefabów w Unity, możliwe było szybkie iterowanie na poziomach oraz efektywne zarządzanie obiektami [17]. Do zarządzania scenami wykorzystano SceneManager, co pozwoliło na dynamiczne ładowanie i przełączanie się między lokacjami w grze [18].

Wszystkie modele w grze zostały wykonane od podstaw przy użyciu Blender 4.0.1 [19]. Proces modelowania obejmował:

- Tworzenie niskopoligonowych obiektów w celu optymalizacji wydajności [20],
- Wykorzystanie modyfikatora boolean do tworzenia szczegółowych otworów drzwiowych i okiennych [21],
- Retopologię i optymalizację siatek dla modeli przeciwników oraz budynku *Copernicanum* [22],
- Generowanie map normalnych, map AO oraz map wysokości, aby dodać szczegóły przy jednoczesnym ograniczeniu liczby wielokątów [23].

Aby wiernie odwzorować budynek *Copernicanum*, wykorzystano narzędzie fSpy, które umożliwiło dokładne odwzorowanie proporcji budynku na podstawie zdjęć referencyjnych [24].

Wszystkie tekstury w grze były ręcznie edytowane i optymalizowane w Adobe Photoshop [25]. Proces ten obejmował:

- Korekcję kolorów oraz redukcję szumów na teksturach [26],
- Tworzenie masek dla efektów PBR [27],
- Przygotowanie niestandardowych tekstur dla interfejsu użytkownika (UI) [28].

W celu zachowania stylu gier retro, część tekstur została obniżona do niższej rozdzielczości oraz zastosowano dithering, co pozwoliło na uzyskanie stylizowanego efektu wizualnego inspirowanego horrorami z lat 90. [29].
Kod gry został napisany w C# przy użyciu JetBrains Rider oraz Visual Studio 2019 [30].

- Rider został wybrany ze względu na jego zaawansowany system podpowiedzi kodu, analizę błędów w czasie rzeczywistym oraz lepszą integrację z Unity [31],
- Visual Studio 2019 było używane głównie do debugowania oraz implementacji mechanik skryptowych [32].

Kod gry oparty jest na wzorcu projektowym Singleton, co pozwoliło na efektywne zarządzanie obiektami takimi jak menedżer stanów gry oraz system zapisu [33].
Do zarządzania kodem i wersjonowaniem projektu wykorzystano Unity Plastic SCM [34]. Dzięki temu możliwe było:

- Przechowywanie historii zmian [35],
- Praca zespołowa nad kodem w sposób rozproszony [36],
- Unikanie konfliktów oraz szybkie przywracanie starszych wersji plików [37].

Gra wykorzystuje silnik fizyki NvidiaPhysX, który odpowiada za:

- Interakcję gracza z otoczeniem [38],
- Fizykę obiektów dynamicznych, takich jak przewracane krzesła [39],
- Symulację ruchu przeciwnika oraz jego kolizji ze środowiskiem [40].

W celu zaawansowanego zarządzania dźwiękiem wykorzystano FMOD Studio [41]. System ten pozwala na dynamiczną zmianę parametrów dźwięku w zależności od otoczenia, np.:

- Pogłos i echo w zależności od wielkości pomieszczenia [42],
- Efekty tłumienia dźwięków podczas chowania się w szafce [43],
- Dźwięki przeciwnika zależne od jego pozycji względem gracza [44].

2.3 Mechaniki gry

W ramach implementacji mechanik gry stworzono szereg kluczowych systemów, które wpływają na płynność rozgrywki i poziom immersji gracza. Każdy z tych systemów został zaprojektowany z myślą o interaktywności oraz dynamicznym dostosowywaniu się do działań użytkownika.
Jednym z najważniejszych elementów jest system zapisu i wczytywania gry, który wykorzystuje pliki JSON do przechowywania stanów rozgrywki [45]. Implementacja bazuje na interfejsie ISaveable, co umożliwia każdemu obiektowi, który go implementuje, zapis swojego aktualnego stanu do pliku. System ten obsługuje pozycję gracza, stan drzwi, ekwipunek oraz stan przeciwników, co pozwala na kontynuowanie rozgrywki w dokładnie tym samym miejscu po ponownym uruchomieniu gry [46].
Interaktywność otoczenia realizowana jest poprzez system interaktywnych obiektów, który umożliwia manipulowanie elementami środowiska. Obiekty takie jak drzwi, szafki, przełączniki czy przedmioty kolekcjonerskie posiadają interfejs IInteractable, który pozwala graczowi na wykonywanie określonych działań. Wykorzystano tu system raycastingu, który identyfikuje najbliższy obiekt w polu widzenia gracza i umożliwia jego aktywację po naciśnięciu odpowiedniego przycisku [47].
Sterowanie postacią gracza opiera się na płynnym systemie ruchu z wykorzystaniem komponentu CharacterController w Unity [48]. Implementacja obejmuje realistyczny model poruszania się, uwzględniający grawitację, przyspieszenie oraz spowalnianie postaci w zależności od podłoża. Dodatkowo, postać posiada zdolność skradania się, co zmniejsza poziom hałasu, wpływając na sposób, w jaki przeciwnik wykrywa gracza.
Model przeciwnika sterowany jest przez system sztucznej inteligencji (AI), który dynamicznie reaguje na działania gracza [49]. Przeciwnik posiada trzy poziomy świadomości:
Patrolowanie – standardowe przemieszczanie się po wyznaczonych trasach,
Stan czujności – reakcja na nietypowe dźwięki i ruchy gracza,
Pościg – agresywna próba schwytania gracza.
Do wyznaczania ścieżek w środowisku zastosowano algorytmy A Pathfinding*, co pozwala przeciwnikom na inteligentne omijanie przeszkód i dynamiczne dostosowanie trasy do ruchów gracza [50]. Wrogowie mogą także przeszukiwać obszar, jeśli podejrzewają obecność gracza, co zwiększa poziom napięcia i wymaga strategicznego planowania działań.

Rys. 1. Wizualizacja jednej z lokacji gry

Rys. 2. Wizualizacja budynku Copernicanum

3. Testowanie i analiza wyników

Testy gry przeprowadzono w warunkach rzeczywistych, angażując grupę testerów składającą się zarówno z doświadczonych graczy, jak i osób mających mniejsze doświadczenie w grach survival horror. W celu uzyskania kompleksowej analizy gra została testowana na różnych konfiguracjach sprzętowych oraz w kilku iteracjach rozwoju projektu, co pozwoliło na wyeliminowanie krytycznych błędów oraz poprawę ogólnej grywalności.
Wydajność i optymalizacja zostały ocenione na podstawie stabilności klatek na sekundę (FPS) oraz zużycia pamięci RAM i VRAM. Testy wykazały, że gra działa płynnie na sprzęcie średniej klasy, osiągając średnio 60 FPS w rozdzielczości 1080p, przy dynamicznym oświetleniu oraz efektach post-processingu.
Sprzęt wykorzystany do prowadzenia testów:

- RAM: 8GB
- Procesor: Intel Core i5
- Karta graficzna: NVidiaGeForce GTX 1070

Optymalizacja obejmowała redukcję liczby aktywnych obiektów, implementację culling system oraz baking oświetlenia, co znacząco poprawiło wydajność.

Realizm odwzorowania budynku oceniono poprzez porównanie modelu 3D z rzeczywistym budynkiem Copernicanum. Studenci oraz wykładowcy Wydziału Informatyki wskazali na wysoki poziom zgodności architektonicznej, podkreślając szczegółowość wnętrz oraz rozmieszczenie pomieszczeń. Nieliczne poprawki dotyczyły skalowania niektórych korytarzy oraz drobnych różnic w rozmieszczeniu mebli.
Balans rozgrywki testowano pod kątem poziomu trudności i satysfakcji graczy. Testerzy wskazali, że mechaniki gry, takie jak zarządzanie zasobami, system skradania się oraz adaptacyjna AI, skutecznie budują napięcie, ale wymagają dodatkowego skalowania poziomu trudności. Wprowadzono poprawki do algorytmu wykrywania gracza przez przeciwnika oraz dodano stopniową progresję trudności w miarę postępu w grze.
Skuteczność systemu przeciwników analizowano na podstawie ich inteligencji i reakcji na działania gracza. AI przeciwnika zostało pozytywnie ocenione za dynamiczne patrolowanie, poszukiwanie dźwięków oraz reakcję na nagłe zmiany w otoczeniu. Testerzy zwrócili uwagę na konieczność poprawy kolizji wąskich przejść oraz bardziej zróżnicowane zachowania w zależności od lokalizacji gracza.

4. Wnioski

Projekt UKW Halloween wykazał, że zastosowanie Unity w produkcji realistycznych horrorów pozwala osiągnąć wysoką jakość audiowizualną oraz głęboką immersję gracza [12].
Dzięki wykorzystaniu High Definition RenderPipeline (HDRP) uzyskano realistyczne oświetlenie i efekty wizualne, które znacząco wpływają na klimat gry [13]. Implementacja zaawansowanego systemu sztucznej inteligencji (AI) przeciwników pozwoliła na stworzenie adaptacyjnych zachowań, zwiększających poziom napięcia i wyzwania dla gracza [49].
Testy wykazały, że projekt z powodzeniem odwzorował budynek Copernicanum, co docenili zarówno studenci, jak i wykładowcy Uniwersytetu Kazimierza Wielkiego [11]. Balans rozgrywki oraz poziom trudności wymagały kilku iteracji poprawek, co finalnie doprowadziło do uzyskania optymalnego poziomu wyzwania, szczególnie w mechanikach skradania się i zarządzania zasobami [48].
W przyszłości możliwe jest rozwinięcie gry o nowe lokacje, które wzbogacą eksplorację oraz różnorodność zagadek [6]. Rozważana jest również implementacja zaawansowanego AI przeciwników, wykorzystującego uczenie maszynowe,

co pozwoliłoby na bardziej realistyczne reakcje wrogów na działania gracza [8]. Kolejnym krokiem w rozwoju projektu może być tryb multiplayer, który umożliwi wspólną rozgrywkę, np. w trybie kooperacyjnym lub asymetrycznym, gdzie jeden gracz wciela się w antagonistę [7].

Podsumowując, UKW Halloween jest solidnym fundamentem pod dalszy rozwój, a jego koncepcja oraz techniczne rozwiązania mogą posłużyć jako baza do kolejnych, bardziej zaawansowanych projektów w gatunku survival horror.

Bibliografia

1. Unity Technologies, „Unity Documentation", dostęp: https://docs.unity3d.com
2. Perron B., „Horror Video Games: Essays on the Fusion of Fear and Play", McFarland, 2009.
3. McGowan J., „The Aesthetics of Survival Horror: Silent Hill and Resident Evil", Game Studies Journal, 2015.
4. Blender Foundation, „Blender Manual", dostęp: https://docs.blender.org
5. Engel W., „Advanced Lighting and Materials with Shaders", CRC Press, 2012.
6. Totten C., „An Architectural Approach to Level Design", CRC Press, 2014.
7. Smith J., „Game Mechanics: Advanced Game Design", Addison-Wesley, 2013.
8. Millington I., „Artificial Intelligence for Games", CRC Press, 2019.
9. Eberly D., „3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics", CRC Press, 2007.
10. Nystrom R., „Game Programming Patterns", Genever Benning, 2014.
11. Wikipedia, Bydunek Copernicanum https://pl.wikipedia.org/wiki/Copernicanum_w_Bydgoszczy [dostęp 29.01.25]
12. Unity Technologies. *Unity Documentation - HDRP*, 2024.
13. Morgan, A. *Real-Time Rendering with HDRP in Unity*, Packt Publishing, 2023.
14. Engel, W. *Programming Graphics Shaders in Unity*, CRC Press, 2021.
15. Unity Technologies. *Shader Graph Overview*, 2024.
16. Mitchell, M. *Post-Processing in Unity: A Complete Guide*, Apress, 2022.
17. Unity Technologies. *Prefab System in Unity Documentation*, 2023.
18. Unity Technologies. *Scene Management in Unity*, 2024.
19. Blender Foundation. *Blender 4.0.1 User Manual*, 2023.
20. Zeigler, J. *Optimizing 3D Models for Games*, Packt Publishing, 2021.
21. Blender Foundation. *Boolean Modifier in Blender*, 2024.
22. Foster, C. *Retopology Techniques in 3D Modeling*, CRC Press, 2020.
23. Baker, T. *Introduction to Normal, AO, and Height Maps*, Apress, 2022.
24. fSpy Documentation. *Using fSpy for Architectural Modeling*, 2023.
25. Adobe Inc. *Photoshop User Guide*, 2024.
26. Reinhard, E. *High Dynamic Range Imaging*, Elsevier, 2022.
27. GameTextures.com. *Physically Based Rendering (PBR) Workflow*, 2023.
28. Adobe Inc. *Creating UI Textures in Photoshop*, 2023.
29. McDermott, T. *Pixel Art & Retro Graphics*, 2021.
30. JetBrains. *JetBrains Rider - C# Development Guide*, 2024.
31. JetBrains. *Rider for Unity Development*, 2023.
32. Microsoft. *Visual Studio 2019 Documentation*, 2024.
33. Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
34. Unity Technologies. *Unity Plastic SCM Version Control*, 2023.
35. GitHub Inc. *Version Control Best Practices*, 2022.
36. Llopis, J. *Collaborative Game Development with Unity*, 2021.
37. Unity Technologies. *Unity Collaborate vs Plastic SCM*, 2023.
38. NVIDIA. *PhysX SDK Documentation*, 2024.
39. Millington, I. *Game Physics Engine Development*, Morgan Kaufmann, 2019.
40. NVIDIA. *Advanced Rigid Body Physics in PhysX*, 2023.
41. Firelight Technologies. *FMOD Studio API Reference*, 2024.
42. Farnell, A. *Designing Sound*, MIT Press, 2018.

43. FMOD. *Reverb and Occlusion Effects*, 2023.
44. Wwise vs FMOD. *Comparison of Audio Middleware for Games*, 2022.
45. Unity Technologies. *Unity JSON Utility Documentation*, 2024.
46. Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994 [33].
47. Unity Technologies. *Raycasting in Unity: Physics.Raycast() Guide*, 2023.
48. Unity Technologies. *CharacterController Component Documentation*, 2024.
49. Unity Technologies. *AI and Navigation in Unity*, 2023.
50. Hart, P., Nilsson, N., Raphael, B. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on Systems Science and Cybernetics, 1968.
51. Skrendo D., Beliavskyi M., Smirnova E., Pakuła M. *Gra komputerowa "UKW Halloween" oparta na silniku Unity,Uniwersytet Kazimierza Wielkiego, Wydział Informatyki,* 2024