

STRESZCZENIE (JAN WALCZAK)

Zaangażowanie uczniów w szkołach średnich spada, a lekcje w standardowej formule mogą wydawać się nieatrakcyjne. Matematyka jest jednym z trudniejszych przedmiotów w szkole, którego uczniowie uczą się na każdym etapie edukacji. Rośnie więc potrzeba dywersyfikacji sposobów nauczania i urozmaicania zajęć, tak aby stawały się one ciekawsze.

W ramach pracy zaprojektowano i wykonano aplikację matematycznego escape roomu dla uczniów szkół średnich, która miałaby przedstawić im naukę matematyki jako coś interesującego i wartego ich zaangażowania. Projekt obejmuje opracowanie 13 zagadek, z których każda dotyczy innego działu matematyki i porusza zagadnienia obowiązujące w podstawie programowej szkół średnich. Implementacja została wykonana w Laboratorium Zanurzonej Wizualizacji Przestrzennej z wykorzystaniem technologii VR. Głęboka immersja i interaktywność przedstawionego rozwiązania mają pozwolić uczniom na efektywną i przyjemną naukę.

Słowa kluczowe: matematyczny escape room, Laboratorium Zanurzonej Wizualizacji Przestrzennej, CAVE, Unreal Engine, VR, szkoły średnie.

ABSTRACT (JAN WALCZAK)

Fewer and fewer students are engaged in school, and lessons in the standard format may seem unattractive to them. Mathematics is one of the most difficult subjects that students study at every stage of their education. Because of this, the demand for diversifying teaching methods is increasing so that learning becomes more interesting.

As part of an engineering project, a mathematical escape room application for high school students was designed and developed to present mathematics as something fresh and worth their effort. The project consists of 13 different puzzles, each of which introduces a different branch of mathematics and addresses topics covered in the curriculum. The project was implemented in the Immersive 3D Visualization Lab using VR technology. The deep immersion and interactivity of the solution are intended to enable students to study effectively and enjoyably.

Keywords: mathematical escape room, Immersive 3D Visualization Lab, CAVE, Unreal Engine, VR, middle school, high school.

SPIS TREŚCI

1.	Wstęp i cel pracy	8
2.	Wprowadzenie do dziedziny	10
2.1.	Rola technologii w edukacji (Konrad Czarnecki)	10
2.1.1.	Nowoczesne metody nauczania matematyki	10
2.1.2.	Gamifinacka w edukacji	11
2.2.	Edukacyjne zastosowania escape roomów (Konrad Czarnecki)	11
2.2.1.	Historia i definicja escape roomu	12
2.2.2.	Escape roomy jako metoda aktywizacji uczniów	12
2.3.	Wirtualna rzeczywistość (Konrad Czarnecki)	13
2.3.1.	Wirtualna rzeczywistość i jej zastosowanie w nauce	13
2.3.2.	Wpływ interaktywności na zaangażowanie gracza	13
3.	Analiza istniejących rozwiązań	15
3.1.	Wirtualne escape roomy	15
3.2.	Przegląd gier edukacyjnych z elementami VR	15
3.2.1.	Blockerzz	15
3.2.2.	Mage Math	15
3.2.3.	EdScape XR	15
3.3.	Wnioski z analizy dostępnych rozwiązań	15
4.	Projekt systemu	16
4.1.	Wymagania funkcjonalne i niefunkcjonalne (Konrad Czarnecki)	16
4.1.1.	Wymagania funkcjonalne	16
4.1.2.	Wymagania niefunkcjonalne	17
4.2.	Scenariusz gry (Konrad Czarnecki)	17
4.3.	Model przypadków użycia	18
4.4.	Projekt architektury systemu	18
4.5.	Projekt interfejsu użytkownika i środowiska gry	18
5.	Technologie i narzędzia	19
5.1.	Silnik gry (Konrad Czarnecki)	19
5.2.	Laboratorium Zanurzonej Wizualizacji Przestrzennej (Konrad Czarnecki)	19
5.3.	Środowisko 3D (Konrad Czarnecki)	20
5.4.	System kontroli wersji (Konrad Czarnecki)	20
6.	Organizacja pracy (Jan Walczak)	21
6.1.	System kontroli wersji (Jan Walczak)	21
6.1.1.	Zastosowane rozwiązanie	21
6.1.2.	Kontrola przepływu	21
6.1.3.	Łączenie gałęzi	22
7.	Analiza dydaktyczna i potencjał edukacyjny	24
7.1.	Motywacja uczniów do nauki matematyki	24
7.2.	Możliwość personalizacji i różnicowania poziomu trudności	24
7.3.	Wpływ imersji na naukę	24

8. Opis teoretyczny zagadek	25
8.1. Zadanie 1 – Wzory skróconego mnożenia (Andrii Demyshyn)	25
8.1.1. Cel zadania	25
8.1.2. Zasady działania	25
8.1.3. Zakończenie zadania	26
8.2. Zadanie 2 – Planimetria (Jan Walczak)	26
8.2.1. Cel zadania	26
8.2.2. Zasady działania	26
8.2.3. Zakończenie zadania	26
8.3. Zadanie 3 – Nierówności (Konrad Czarnecki)	27
8.3.1. Cel zadania	27
8.3.2. Zasady działania	27
8.3.3. Zakończenie zadania	27
8.4. Zadanie 4 – Funkcje (Konrad Czarnecki)	27
8.4.1. Cel zadania	28
8.4.2. Zasady działania	28
8.4.3. Zakończenie zadania	28
8.5. Zadanie 5 – Geometria analityczna (Konrad Czarnecki)	28
8.5.1. Cel zadania	28
8.5.2. Zasady działania	28
8.5.3. Zakończenie zadania	28
8.6. Zadanie 6 – Kombinatoryka (Andrii Demyshyn)	29
8.6.1. Cel zadania	29
8.6.2. Zasady działania	29
8.6.3. Zakończenie zadania	30
8.7. Zadanie 7 – Liczby rzeczywiste i działania na zbiorach liczbowych (Jan Walczak)	30
8.7.1. Cel zadania	30
8.7.2. Zasady działania	31
8.7.3. Zakończenie zadania	31
8.8. Zadanie 8 – Znaki funkcji trygonometrycznych (Konrad Czarnecki)	31
8.8.1. Cel zadania	32
8.8.2. Zasady działania	32
8.8.3. Zakończenie zadania	32
8.9. Zadanie 9 – Ciągi liczbowe (Jan Walczak)	32
8.9.1. Cel zadania	32
8.9.2. Zasady działania	32
8.9.3. Zakończenie zadania	33
8.10. Zadanie 10 – Prawdopodobieństwo (Andrii Demyshyn)	33
8.10.1. Cel zadania	33
8.10.2. Zasady działania	33
8.10.3. Zakończenie zadania	34

8.11 Zadanie 11 – Optymalizacja i rachunek różniczkowy (Autor).....	34
8.11.1.Cel zadania.....	34
8.11.2.Zasady działania.....	34
8.11.3.Zakończenie zadania	35
8.12 Zadanie 12 – Układy równań (Andrii Demyshyn)	35
8.12.1.Cel zadania.....	35
8.12.2.Zasady działania.....	35
8.12.3.Zakończenie zadania	36
8.13 Zadanie 13 – Stereometria(Jan Walczak)	36
8.13.1.Cel zadania.....	36
8.13.2.Zasady działania.....	36
8.13.3.Zakończenie zadania	37
8.14 Zadanie 13 – Stereometria (Jan Walczak).....	37
8.14.1.Cel zadania.....	37
8.14.2.Zasady działania.....	37
8.14.3.Zakończenie zadania	37
9. Implementacja zagadek	38
9.1. Zadanie 1 – wzory skróconego mnożenia (Andrii Demyshyn).....	38
9.1.1. Opis obiektów pokoju	38
9.1.2. Zakończenie zadania	39
9.1.3. Zakończenie zadania	39
9.1.4. Przebieg zadania	40
9.2. Zadanie 2 – Planimetria (Jan Walczak)	41
9.2.1. Problemy i różnice w realizacji zadania w praktyce	41
9.2.2. Implementacja struktury danych przechowującej relację	41
9.2.3. Implementacja kontrolerów	42
9.2.4. Przebieg zadania	42
9.3. Zadanie 3 – Nierówności (Konrad Czarnecki)	45
9.3.1. Problemy i różnice w realizacji zadania w praktyce	45
9.3.2. Główny kontroler.....	45
9.3.3. Wybór kafelków i układ nierówności	45
9.3.4. Przebieg zadania	46
9.4. Zadanie 4 – Funkcje (Konrad Czarnecki).....	48
9.4.1. Problemy i różnice w realizacji zadania w praktyce	48
9.4.2. Wybór dostępnych punktów	48
9.4.3. Przebieg zadania	48
9.5. Zadanie 5 – Geometria analityczna (Konrad Czarnecki)	50
9.5.1. Problemy i różnice w realizacji zadania w praktyce	50
9.5.2. Kontroler przycisków.....	50
9.5.3. Wybór dostępnych wzorów funkcji	50
9.5.4. Przebieg zadania	51
9.6. Zadanie 6 – Kombinatoryka (Amdrii Demyshyn)	52
9.6.1. Różnice realizacji zadania w praktyce	52
9.6.2. Zaimplementowane zagadki	52
9.6.3. Sejf	53

9.6.4. Zakończenie zadania	54
9.6.5. Podsumowanie	54
9.7. Zadanie 7 – Liczby rzeczywiste i działania na zbiorach liczbowych	
(Jan Walczak)	54
9.7.1. Problemy i różnice w realizacji zadania w praktyce	54
9.7.2. Implementacja zadania	56
9.7.3. Przebieg zadania	56
9.7.4. Implementacja mechanizmu uzupełniania ścianki	59
9.8. Zadanie 8 – Znaki funkcji trygonometrycznych (Konrad Czarnecki)	59
9.8.1. Problemy i różnice w realizacji zadania w praktyce	59
9.8.2. Implementacja sześcianów	59
9.8.3. Przebieg zadania	61
9.9. Zadanie 9 – Ciągi liczbowe (Jan Walczak)	61
9.9.1. Problemy i różnice w realizacji zadania w praktyce	61
9.9.2. Implementacja pistoletu laserowego	61
9.9.3. Implementacja zadania	62
9.9.4. Przebieg zadania	63
9.10 Zadanie 10 – Prawdopodobieństwo (Andrii Demyshyn)	66
9.10.1. Praktyczna realizacja zadania	66
9.11 Zadanie 11 – Optymalizacja i rachunek różniczkowy (Autor)	69
9.12 Zadanie 12 – Układy równań (Andrii Demyshyn)	69
9.12.1. Różnice realizacji zadania w praktyce	69
9.12.2. Opis obiektów sceny	69
9.12.3. Zakończenie zadania	70
9.13 Zadanie 13 – Stereometria (Jan Walczak)	71
9.13.1. Problemy i różnice w realizacji zadania w praktyce	71
9.13.2. Implementacja zadania	71
9.13.3. Przebieg zadania	72
10. Podsumowanie	74
10.1. Ocena realizacji celów	74
10.2. Propozycje rozwoju projektu	74
10.3. Wnioski końcowe	74
Wykaz literatury	75
Spis rysunków	77
Spis tabel	78

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

Escape room – interaktywna gra, w której gracze rozwiązuje zagadki, by „uciec” z zamkniętego pokoju lub przejść do kolejnego etapu.

VR – wirtualna rzeczywistość, technologia pozwalająca na zanurzenie się w cyfrowym świecie za pomocą gogli i kontrolerów.

Laboratorium Zanurzonej Wizualizacji Przestrzennej – specjalne środowisko lub przestrzeń, gdzie łączy się elementy rzeczywistości wirtualnej i rzeczywistości rzeczywistej, często wykorzystywane do edukacji i interaktywnych doświadczeń.

1. WSTĘP I CEL PRACY

Współczesny rozwój technologii informatycznych i multimedialnych umożliwia tworzenie nowoczesnych, interaktywnych narzędzi wspomagających proces nauczania, które pozwalają uczniom nie tylko na bierne przyswajanie wiedzy, ale również na aktywne uczestnictwo w zajęciach oraz samodzielne rozwiązywanie problemów w angażującym środowisku. W szczególności dynamicznie rozwijające się technologie wirtualnej i rozszerzonej rzeczywistości znajdują coraz szersze zastosowanie nie tylko w przemyśle i rozrywce, lecz również w edukacji na różnych poziomach nauczania. Zastosowanie immersyjnych środowisk w nauczaniu matematyki może znacząco wpływać na wzrost zaangażowania uczniów oraz efektywność przyswajania materiału dydaktycznego.

Matematyka, jako jedna z podstawowych dziedzin nauki, często bywa postrzegana przez uczniów szkół średnich jako przedmiot trudny i mało atrakcyjny. Jednym ze sposobów przełamywania tego stereotypu jest wprowadzenie do procesu dydaktycznego elementów gier edukacyjnych, które poprzez zabawę i rywalizację angażują uczniów do aktywnego rozwiązywania problemów. W ostatnich latach coraz większym zainteresowaniem cieszą się tzw. escape roomy — gry logiczne, w których uczestnicy muszą w określonym czasie rozwiązać serię zagadek i łamigłówek, aby „wydostać się” z wirtualnego lub rzeczywistego pomieszczenia. Włączenie tego typu rozwiązań do nauczania matematyki stwarza szansę na połączenie nauki z zabawą oraz zastosowanie wiedzy teoretycznej w praktycznych sytuacjach problemowych.

Przedmiotem niniejszej pracy jest opracowanie i wykonanie aplikacji stanowiącej wirtualny pokój zagadek matematycznych (ang. escape room), przeznaczonej do działania w systemie jaskiń rzeczywistości wirtualnej dostępnych w Laboratorium Zanurzonej Wizualizacji Przestrzennej (LZWP). Laboratorium to wyposażone jest w nowoczesne rozwiązania umożliwiające projekcję obrazów w technologii VR na ścianach specjalnie przystosowanych pomieszczeń, dając użytkownikowi wrażenie całkowitego zanurzenia w trójwymiarowym środowisku. Aplikacja powinna umożliwiać uczestnikom interakcję z otoczeniem za pomocą kontrolerów ruchu wykorzystywanych w LZWP.

Celem głównym projektu jest stworzenie interaktywnej aplikacji edukacyjnej zawierającej 13 zagadek matematycznych, odpowiadających poszczególnym działom matematyki nauczany w szkołach średnich. Działy te obejmują: liczby rzeczywiste, wyrażenia algebraiczne, równania i nierówności, układy równań, funkcje, ciągi, trygonometrię, planimetrię, geometrię analityczną na płaszczyźnie kartezjańskiej, stereometrię, kombinatorykę, rachunek prawdopodobieństwa i statystykę oraz optymalizację i rachunek różniczkowy.

W ramach realizacji projektu przewidziano następujące etapy prac:

1. Zapoznanie się z funkcjonowaniem systemu jaskiń rzeczywistości wirtualnej dostępnych w LZWP oraz metodami programowania aplikacji dla tego typu środowisk.
2. Opracowanie koncepcji i projektów zagadek matematycznych reprezentujących wymienione działy matematyki, z uwzględnieniem ich formy, poziomu trudności oraz możliwych metod interakcji użytkownika z aplikacją.
3. Konsultacja opracowanych projektów zagadek z dydaktykami matematyki w celu dostosowania ich treści oraz sposobu prezentacji do wymagań programowych i poziomu uczniów szkół średnich.

4. Implementacja aplikacji integrującej wszystkie zaprojektowane i zatwierdzone zagadki w ramach jednego spójnego środowiska wirtualnego escape roomu.

5. Przeprowadzenie testów funkcjonalnych, wydajnościowych oraz badań pilotażowych z udziałem grupy uczniów i dydaktyków w celu weryfikacji poprawności działania aplikacji oraz jej efektywności dydaktycznej.

Ostatecznym rezultatem pracy będzie kompletna, działająca aplikacja edukacyjna przeznaczona do uruchomienia w jaskiniach rzeczywistości wirtualnej, umożliwiająca użytkownikom rozwijanie kompetencji matematycznych w angażującej i nowoczesnej formie. Wnioski płynące z przeprowadzonych badań pilotażowych pozwolą na ocenę przydatności tego typu rozwiązań w praktyce dydaktycznej oraz wskazanie możliwości ich dalszego rozwoju i wykorzystania w nauczaniu przedmiotów ścisłych.

2. WPROWADZENIE DO DZIEDZINY

2.1. Rola technologii w edukacji (Konrad Czarnecki)

Rozwój technologii cyfrowych wywarł ogromny wpływ na niemal wszystkie obszary współczesnego życia, w tym również na edukację. Tradycyjne metody nauczania, oparte w dużej mierze na wykładzie i pracy z podręcznikiem, są coraz częściej wzbogacane lub nawet zastępowane przez nowoczesne narzędzia dydaktyczne, które wspomagają i uatrakcyjnają proces kształcenia. W szczególności technologie informacyjno-komunikacyjne oraz środowiska immersywne, takie jak rzeczywistość wirtualna, rzeczywistość rozszerzona czy rzeczywistość mieszana, stają się istotnym elementem nowoczesnej edukacji.

Współczesne pokolenia uczniów i studentów od najmłodszych lat funkcjonują w środowisku przesyconym technologią i są przyzwyczajeni do interaktywnego oraz multimedialnego przekazu treści. Tradycyjny model nauczania nie zawsze jest w stanie zaspokoić potrzeby poznawcze młodych ludzi, co może prowadzić do spadku motywacji oraz trudności w przyswajaniu wiedzy. Odpowiedzią na to wyzwanie jest integracja narzędzi technologicznych z procesem dydaktycznym, co może przyczynić się do zwiększenia efektywności nauczania, ułatwienia zrozumienia trudnych zagadnień oraz podniesienia ogólnego poziomu zaangażowania uczniów.

Zastosowanie nowoczesnych technologii w edukacji nie tylko wpływa na sposób przekazywania wiedzy, ale również umożliwia tworzenie zupełnie nowych, interaktywnych form kształcenia. Szczególnie istotne znaczenie mają tutaj rozwiązania wykorzystujące elementy gamifikacji oraz immersyjnych doświadczeń edukacyjnych, które pozwalają uczniom wchodzić w bezpośrednią interakcję z materiałem dydaktycznym. W kontekście nauczania matematyki, będącej często postrzeganą jako przedmiot trudny i abstrakcyjny, technologie te mogą odegrać kluczową rolę w budowaniu pozytywnego nastawienia do nauki oraz lepszego zrozumienia prezentowanych treści.

2.1.1. Nowoczesne metody nauczania matematyki

Matematyka, jako dziedzina wymagająca myślenia analitycznego, logicznego rozumowania oraz umiejętności abstrakcyjnego operowania symbolami, od zawsze stawała przed uczniami szczególnie wyzwania. Z tego powodu nauczanie matematyki wymaga nieustannego poszukiwania skutecznych metod dydaktycznych, które nie tylko umożliwiają efektywne przekazanie wiedzy, ale również wzbudzą w uczniach zainteresowanie i motywację do nauki.

Współczesne podejścia do nauczania matematyki coraz częściej odchodzą od modelu opartego wyłącznie na wykładzie i ćwiczeniach przy tablicy, na rzecz metod aktywizujących, w których uczniowie samodzielnie odkrywają zależności matematyczne, rozwiązują problemy oraz współpracują w grupie. Szczególne miejsce wśród nowoczesnych metod zajmują rozwiązania oparte na technologiach komputerowych — aplikacje i platformy edukacyjne, programy do wizualizacji danych matematycznych, a także symulacje i środowiska interaktywne.

Zastosowanie narzędzi interaktywnych pozwala uczniom na dynamiczne eksplorowanie pojęć matematycznych, eksperymentowanie z danymi i obserwowanie skutków wprowadzanych zmian w czasie rzeczywistym. Programy takie jak GeoGebra, Desmos, czy MATLAB wspierają nauczanie

geometrii, analizy matematycznej i algebry w sposób znacznie bardziej przystępny i angażujący niż tradycyjne metody.

Również technologie immersywne, takie jak VR, zyskują coraz większe znaczenie w edukacji matematycznej. Umożliwiają one prezentację skomplikowanych struktur geometrycznych w przestrzeni trójwymiarowej, co szczególnie sprzyja nauczaniu stereometrii czy geometrii analitycznej. Dzięki temu uczniowie mogą lepiej zrozumieć zależności przestrzenne oraz intuicyjnie postrzegać abstrakcyjne pojęcia matematyczne.

2.1.2. Gamifinacka w edukacji

Gamifikacja (*ang. gamification*) to zastosowanie mechanizmów znanych z gier komputerowych i planszowych w kontekście niezwiązanym bezpośrednio z grami, takim jak edukacja, zarządzanie czy marketing. W praktyce oznacza to wprowadzanie elementów takich jak punkty, poziomy trudności, nagrody, rankingi czy wyzwania do tradycyjnych zadań edukacyjnych, co ma na celu zwiększenie motywacji, zaangażowania i satysfakcji uczestników procesu nauczania.

W edukacji gamifikacja znajduje szerokie zastosowanie, zwłaszcza w pracy z uczniami szkół podstawowych i średnich. Wprowadzenie grywalizacji do lekcji pozwala uczniom uczestniczyć w nauce w sposób bardziej aktywny i emocjonalnie zaangażowany. Zamiast biernego słuchania wykładu czy rozwiązywania zadań z podręcznika, uczniowie mogą wcielać się w bohaterów gier, zdobywać osiągnięcia i rywalizować z rówieśnikami w przyjazny sposób.

W kontekście nauczania matematyki, gamifikacja może znaczco ułatwić przyswajanie skomplikowanych treści. Zagadki logiczne, łamigłówki, quizy punktowane czy interaktywne escape roomy są przykładami narzędzi, które pozwalają uczniom na wykorzystanie wiedzy matematycznej w kontekście gry. Dzięki temu uczniowie nie tylko uczą się rozwiązywać konkretne typy zadań, ale także rozwijają umiejętności analitycznego myślenia, pracy zespołowej oraz podejmowania decyzji.

W szczególności w środowiskach immersyjnych, takich jak wirtualna rzeczywistość, gamifikacja osiąga nowy wymiar. Połączenie mechanizmów gry z wciągającym, interaktywnym środowiskiem pozwala użytkownikom na budowanie trwałych, pozytywnych skojarzeń z procesem nauki. Tego typu rozwiązania nie tylko zwiększają atrakcyjność zajęć, ale również mogą pozytywnie wpływać na wyniki edukacyjne i długofalowe postawy wobec nauki matematyki.

2.2. Edukacyjne zastosowania escape roomów (Konrad Czarnecki)

W ostatnich latach obserwuje się dynamiczny rozwój innowacyjnych metod dydaktycznych, które mają na celu zwiększenie zaangażowania uczniów oraz poprawę efektywności procesu nauczania. Jedną z takich metod są escape roomy, które pierwotnie funkcjonowały jako forma rozrywki, a z czasem zyskały również uznanie w środowisku edukacyjnym. Dzięki swojej interaktywnej i zespołowej formie, pokoje zagadek umożliwiają uczniom zdobywanie wiedzy oraz rozwijanie kompetencji miękkich w sposób atrakcyjny i emocjonujący.

Escape roomy w edukacji mogą przybierać różnorodne formy — od tradycyjnych wersji stacjonarnych, przez mobilne zestawy dydaktyczne, aż po aplikacje komputerowe i środowiska wirtualnej rzeczywistości. Ich celem jest nie tylko przekazywanie wiedzy merytorycznej, lecz także kształcenie umiejętności pracy zespołowej, logicznego myślenia, zarządzania czasem oraz podejmowania decyzji pod presją. Dlatego coraz częściej wykorzystywane są one w szkołach, na uczelniach wyższych oraz podczas szkoleń i warsztatów.

W kontekście nauczania matematyki escape roomy stanowią wyjątkowo wartościowe narzędzie. Zagadki oparte na treściach matematycznych wymagają od uczestników nie tylko opanowania materiału, ale również zastosowania wiedzy w praktyce, co sprzyja utrwalaniu wiadomości oraz rozwijaniu umiejętności rozwiązywania problemów. Połączenie elementów gry z edukacją umożliwia uczniom naukę w przyjaznej atmosferze i sprzyja budowaniu pozytywnego nastawienia do przedmiotów ścisłych.

2.2.1. Historia i definicja escape roomu

Escape room, znany również jako pokój zagadek lub gra typu „ucieczka z pokoju”, to interaktywna forma rozrywki polegająca na rozwiązywaniu serii łamigłówek i zadań logicznych w określonym czasie, aby wydostać się z zamkniętego pomieszczenia lub osiągnąć inny wyznaczony cel fabularny.

Pierwszy fizyczny escape room powstał w 2007 roku w Kioto w Japonii, a jego twórcą był Takao Kato, który postanowił przenieść ideę wirtualnej gry do świata rzeczywistego. Projekt szybko zyskał popularność, a w kolejnych latach podobne pokoje zaczęły powstawać w innych krajach azjatyckich, a następnie w Europie i Ameryce Północnej. Do Polski pierwsze escape roomy dotarły w 2014 roku i od tego czasu cieszą się dużym zainteresowaniem zarówno wśród młodzieży, jak i dorosłych.

Z czasem, obok komercyjnych pokoi zagadek, zaczęły pojawiać się również wersje edukacyjne, dostosowane do potrzeb szkół i uczelni. Edukacyjne escape roomy różnią się od wersji rozrywkowych tym, że ich głównym celem nie jest zabawa, lecz przekazanie wiedzy oraz rozwijanie określonych kompetencji. Zagadki w tego typu pokojach są projektowane w taki sposób, aby uczestnicy mogli przyswajać treści z wybranych dziedzin nauki podczas rozwiązywania interaktywnych zadań. Coraz częściej wykorzystywane są one także w środowiskach cyfrowych oraz w wirtualnej rzeczywistości, co dodatkowo zwiększa ich dostępność i atrakcyjność.

2.2.2. Escape roomy jako metoda aktywizacji uczniów

Jednym z największych wyzwań współczesnej edukacji jest utrzymanie wysokiego poziomu zaangażowania uczniów oraz motywowanie ich do aktywnego uczestnictwa w zajęciach. W tym kontekście escape roomy stanowią skutecną metodę dydaktyczną, która pozwala na połączenie nauki z emocjonującą formą zabawy. Dzięki swojej interaktywnej strukturze i zespołowemu charakterowi, pokoje zagadek mobilizują uczestników do współpracy, komunikacji oraz kreatywnego rozwiązywania problemów.

Z punktu widzenia dydaktyki escape roomy wspierają rozwój kompetencji kluczowych, takich jak logiczne myślenie, umiejętność analizowania i syntetyzowania informacji, podejmowanie decyzji pod presją czasu oraz zarządzanie zadaniami w zespole. Dodatkowo umożliwiają uczniom zastosowanie zdobytej wiedzy teoretycznej w praktyce, co znacząco wpływa na trwałość zapamiętywania materiału i lepsze zrozumienie omawianych treści.

W szczególności istotne znaczenie mają escape roomy realizowane w środowisku rzeczywistości wirtualnej. Dzięki immersijnemu charakterowi takich aplikacji uczestnicy mogą zanurzyć się w wirtualnym świecie, w którym zagadki matematyczne są częścią spójnej fabuły. Tego typu doświadczenie nie tylko zwiększa atrakcyjność nauki, ale również wzmacnia motywację wewnętrzną uczniów, którzy postrzegają naukę jako przygodę i wyzwanie, a nie obowiązek.

2.3. Wirtualna rzeczywistość (Konrad Czarnecki)

Współczesna edukacja coraz śmielej sięga po nowoczesne technologie, które pozwalają nie tylko wzbogacić proces nauczania, ale również znaczowo zwiększyć zaangażowanie uczniów. Jednym z najbardziej dynamicznie rozwijających się obszarów w tym zakresie są technologie immersywne, do których zalicza się wirtualną rzeczywistość, rzeczywistość rozszerzoną oraz rzeczywistość mieszaną. Ich wspólną cechą jest zdolność do tworzenia interaktywnych środowisk, w których użytkownik ma wrażenie pełnego zanurzenia w generowanym komputerowo świecie.

W kontekście edukacyjnym szczególnie istotne jest połączenie immersji z interaktywnością, czyli możliwością bezpośredniego wpływu na elementy wirtualnego środowiska. Interaktywne aplikacje VR sprzyjają aktywnej nauce poprzez angażowanie użytkownika w proces rozwiązywania problemów, podejmowania decyzji i wykonywania zadań w czasie rzeczywistym. Taka forma edukacji nie tylko zwiększa efektywność przyswajania wiedzy, ale również pozytywnie wpływa na motywację i postawy uczniów wobec nauki.

2.3.1. Wirtualna rzeczywistość i jej zastosowanie w nauce

Wirtualna rzeczywistość to technologia umożliwiająca tworzenie komputerowo generowanych środowisk trójwymiarowych, z którymi użytkownik może wchodzić w interakcję w czasie rzeczywistym. Dzięki zastosowaniu gogli VR oraz kontrolerów ruchu możliwe jest odwzorowanie ruchów użytkownika w przestrzeni wirtualnej, co pozwala na pełne zanurzenie w symulowanym środowisku. Technologie VR znajdują zastosowanie nie tylko w rozrywce, ale również w przemyśle, medycynie, wojsku oraz, coraz częściej, w edukacji.

W środowisku edukacyjnym wirtualna rzeczywistość oferuje szerokie możliwości w zakresie tworzenia interaktywnych laboratoriów, symulacji zjawisk fizycznych, rekonstrukcji historycznych czy wirtualnych wycieczek. Uczniowie mogą dzięki niej eksplorować trudno dostępne miejsca, takie jak wnętrze ludzkiego organizmu, przestrzeń kosmiczną, odległe zakątki świata lub historyczne budowle, co znacząco wzbogaca tradycyjny proces dydaktyczny.

W przypadku dydaktyki matematyki VR pozwala na wizualizację skomplikowanych zagadnień geometrycznych, przestrzennych oraz analitycznych w atrakcyjnej i przystępnej formie. Uczniowie mogą w wirtualnym środowisku manipulować bryłami, obserwować zmiany funkcji w czasie rzeczywistym czy uczestniczyć w interaktywnych grach logicznych i escape roomach matematycznych. Dzięki temu abstrakcyjne treści stają się bardziej zrozumiałe i łatwiejsze do przyswojenia, co przekłada się na lepsze wyniki w nauce oraz pozytywnie wpływa na rozwój umiejętności analitycznego myślenia.

2.3.2. Wpływ interaktywności na zaangażowanie gracza

Interaktywność stanowi jeden z kluczowych elementów nowoczesnych metod dydaktycznych, w tym również aplikacji wykorzystujących technologię wirtualnej rzeczywistości. Oznacza ona możliwość aktywnego uczestniczenia ucznia w procesie dydaktycznym poprzez bezpośrednie oddziaływanie na środowisko edukacyjne, podejmowanie decyzji oraz realizację zadań w czasie rzeczywistym. W przeciwieństwie do pasywnego odbioru treści w tradycyjnych formach nauczania, interaktywne środowiska angażują uczniów na wielu płaszczyznach — poznawczej, emocjonalnej i motorycznej.

Z perspektywy pedagogicznej interaktywność sprzyja rozwijaniu kompetencji poznawczych,

takich jak krytyczne myślenie, umiejętność analizy danych, wyciągania wniosków czy rozwiązywanie problemów. Uczniowie uczestniczący w interaktywnych lekcjach częściej angażują się w zadania, są bardziej zmotywowani do podejmowania wyzwań i wykazują większą samodzielność w poszukiwaniu rozwiązań.

W przypadku zastosowań wirtualnej rzeczywistości, interaktywność przybiera różnorodne formy — od prostych gestów i ruchów wykonywanych za pomocą kontrolerów, przez manipulowanie obiektyami w przestrzeni wirtualnej, aż po rozwiązywanie zagadek i wykonywanie eksperymentów. Szczególnie efektywne okazują się aplikacje edukacyjne, które łączą elementy gry z nauką, umożliwiając użytkownikom rywalizację, zdobywanie punktów, odblokowywanie kolejnych poziomów czy rozwiązywanie zagadek fabularnych.

Środowiska edukacyjne o wysokim stopniu interaktywności znacząco zwiększą motywację wewnętrzną oraz podnoszą poziom satysfakcji z nauki. Uczniowie mają również większą łatwość w przyswajaniu wiedzy oraz częściej uczestniczą w zajęciach, co pozytywnie wpływa na ogólne efekty dydaktyczne.

W kontekście projektowanej aplikacji typu escape room dla jaskini rzeczywistości wirtualnej, wysoki poziom interaktywności będzie kluczowym elementem wpływającym na atrakcyjność i skuteczność dydaktyczną opracowanego rozwiązania. Możliwość bezpośredniego wpływu na otoczenie, rozwiązywania zagadek matematycznych w przestrzeni wirtualnej oraz współpracy z innymi uczestnikami w czasie rzeczywistym stworzy warunki sprzyjające aktywnej, angażującej naucie i rozwijaniu kompetencji.

3. ANALIZA ISTNIEJĄCYCH ROZWIĄZAŃ

3.1. *Wirtualne escape roomy*

3.2. *Przegląd gier edukacyjnych z elementami VR*

3.2.1. *Blockerzz*

3.2.2. *Mage Math*

3.2.3. *EdScape XR*

3.3. *Wnioski z analizy dostępnych rozwiązań*

4. PROJEKT SYSTEMU

4.1. Wymagania funkcjonalne i niefunkcjonalne (Konrad Czarnecki)

Przed przystąpieniem do implementacji aplikacji edukacyjnej w formie escape roomu w środowisku rzeczywistości wirtualnej konieczne było precyzyjne określenie wymagań funkcjonalnych i niefunkcjonalnych, jakie powinna spełniać projektowana aplikacja. Zdefiniowanie wymagań pozwala na kontrolowanie procesu realizacji projektu, zapewnia jego zgodność z oczekiwaniami użytkowników końcowych oraz umożliwia przeprowadzenie testów walidacyjnych w końcowej fazie prac.

4.1.1. Wymagania funkcjonalne

Wymagania funkcjonalne określają, jakie działania użytkownik może wykonać w aplikacji oraz jakie funkcje powinna ona realizować. W przypadku projektowanej aplikacji escape room VR wymagania te obejmują:

Wyświetlanie wirtualnego środowiska escape roomu – Aplikacja powinna umożliwiać użytkownikowi poruszanie się po wirtualnym pokoju zagadek, obejmującym zestaw pomieszczeń związanych z różnymi działami matematyki.

Realizacja 13 zagadek matematycznych – Aplikacja musi zawierać 13 interaktywnych zagadek, po jednej dla każdego z wybranych działów matematyki: liczby rzeczywiste, wyrażenia algebraiczne, równania i nierówności, układy równań, funkcje, ciągi, trygonometria, planimetria, geometria analityczna, stereometria, kombinatoryka, rachunek prawdopodobieństwa i statystyka oraz optymalizacja i rachunek różniczkowy.

Interaktywność zagadek – Każda zagadka powinna wymagać od użytkownika aktywnej interakcji z otoczeniem wirtualnym, np. manipulowania obiektyami, wpisywania odpowiedzi czy przedstawiania elementów.

System weryfikacji poprawności odpowiedzi – Aplikacja powinna sprawdzać poprawność rozwiązań podawanych przez użytkownika oraz wyświetlać komunikaty informujące o sukcesie lub błędzie.

Podpowiedzi dla użytkownika – Każda zagadka powinna posiadać system podpowiedzi (tekstowych lub dźwiękowych), które użytkownik może wywołać w razie trudności z rozwiązaniem zadania.

Śledzenie postępu użytkownika – System powinien zapamiętywać, które zagadki zostały już rozwiązane, i odblokowywać dostęp do kolejnych pomieszczeń w ustalonej kolejności.

Zakończenie rozgrywki – Po rozwiązaniu wszystkich 13 zagadek aplikacja powinna wyświetlić ekran podsumowujący z informacją o ukończeniu escape roomu.

Obsługa urządzeń VR i interfejsów w Laboratorium – Aplikacja musi być kompatybilna z systemami śledzenia ruchu i projekcji wykorzystywanymi w Laboratorium Zanurzonej Wizualizacji Przestrzennej, w tym z systemami typu CAVE oraz kontrolerami ruchu.

4.1.2. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne opisują cechy jakościowe systemu oraz warunki, jakie powinien spełniać, aby zapewnić poprawne działanie i pozytywne doświadczenia użytkowników. W kontekście projektowanej aplikacji escape room VR wymagania niefunkcjonalne obejmują:

Kompatybilność z infrastrukturą LZWP – Aplikacja musi działać poprawnie w warunkach technicznych Laboratorium Zanurzonej Wizualizacji Przestrzennej, współpracując z systemem projektijnym CAVE oraz wykorzystywanymi kontrolerami ruchu.

Wydajność – Aplikacja powinna działać płynnie, z minimalnym opóźnieniem renderowania obrazu i reakcji na ruchy użytkownika, zapewniając co najmniej 60 klatek na sekundę w środowisku VR.

Jakość grafiki i dźwięku – Wirtualne środowisko escape roomu powinno charakteryzować się realistyczną i spójną oprawą wizualną oraz odpowiednio dobranymi efektami dźwiękowymi wspomagającymi immersję.

Łatwość rozbudowy i modyfikacji – Struktura projektu powinna umożliwiać przyszłą rozbudowę aplikacji o nowe zagadki, pomieszczenia lub funkcjonalności bez konieczności przebudowy istniejącego kodu i modeli.

Przenośność kodu i zasobów – Wszystkie pliki źródłowe, modele oraz zasoby dźwiękowe powinny być przechowywane w repozytorium GitHub w sposób umożliwiający łatwe przenoszenie projektu pomiędzy stanowiskami roboczymi oraz serwerami laboratorium.

Dokumentacja – Projekt musi być opatrzony szczegółową dokumentacją techniczną i użytkową, opisującą strukturę aplikacji, sposób instalacji, obsługi oraz instrukcje dotyczące przyszłej rozbudowy.

4.2. Scenariusz gry (Konrad Czarnecki)

Po uruchomieniu aplikacji użytkownik znajduje się w wirtualnym holu startowym, który pełni funkcję ekranu powitalnego. W tym miejscu użytkownik może zapoznać się z zasadami działania aplikacji oraz sterowaniem w środowisku VR.

W momencie rozpoczęcia gry uruchamiany jest licznik czasu, który będzie rejestrował całkowity czas potrzebny na ukończenie escape roomu. Po wybraniu opcji rozpoczęcia gry użytkownik przenosi się do pierwszego z 13 wirtualnych pomieszczeń. Każde pomieszczenie jest utrzymane w odmiennym stylu wizualnym i zawiera jedną interaktywną zagadkę matematyczną z przypisanego działu.

W każdym pokoju gracz ma możliwość poruszania się po przestrzeni VR, wchodzenia w interakcje z elementami zagadki, wywoływanie podpowiedzi (jeśli jest dostępna), przechodzenia do kolejnego pokoju po poprawnym rozwiązaniu zagadki.

Działanie każdej zagadki może wymagać różnych form interakcji — takich jak przeciąganie obiektów, wpisywanie wyników na wirtualnej klawiaturze, wskazywanie elementów przestrzeni, czy manipulowanie wirtualnymi narzędziami (np. suwaki, przełączniki, dźwignie). Po rozwiązaniu ostatniej, trzynastej zagadki użytkownik zostaje przeniesiony do wirtualnego pokoju podsumowań. W tym miejscu aplikacja prezentuje takie informacje jak łączny czas ukończenia gry i liczbę wykorzystanych podpowiedzi.

W przypadku przerwania rozgrywki przez użytkownika lub awarii systemu, aplikacja powinna zapewniać możliwość powrotu do holu głównego lub całkowitego zamknięcia gry bez ryzyka utraty

integralności danych systemu. Rozwiążanie to gwarantuje bezpieczeństwo użytkownika oraz stabilność aplikacji w środowisku CAVE.

Tak skonstruowany scenariusz rozgrywki umożliwia nie tylko wprowadzenie użytkownika w świat wirtualnej rzeczywistości, ale także zapewnia dynamiczny, uporządkowany i logiczny przebieg interakcji. Dzięki systematycznemu rozwiązywaniu zagadek i przemieszczaniu się między pokojami, aplikacja utrzymuje wysoki poziom zaangażowania i motywacji użytkowników do ukończenia całej gry.

4.3. *Model przypadków użycia*

4.4. *Projekt architektury systemu*

4.5. *Projekt interfejsu użytkownika i środowiska gry*

5. TECHNOLOGIE I NARZĘDZIA

Realizacja projektu aplikacji edukacyjnej typu escape room w środowisku rzeczywistości wirtualnej wymagała doboru odpowiednich narzędzi oraz technologii, które umożliwiły stworzenie interaktywnej, atrakcyjnej wizualnie i funkcjonalnej aplikacji kompatybilnej z systemami dostępnymi w Laboratorium Zanurzonej Wizualizacji Przestrzennej.

5.1. *Silnik gry (Konrad Czarnecki)*

Do stworzenia aplikacji wirtualnego escape roomu zdecydowano się na wykorzystanie silnika Unreal Engine 5, który jest jednym z najpopularniejszych środowisk do tworzenia gier komputerowych oraz aplikacji wirtualnej rzeczywistości. Unreal Engine, rozwijany przez firmę Epic Games, umożliwia tworzenie zaawansowanych wizualnie, interaktywnych projektów 3D oraz VR dzięki nowoczesnemu systemowi renderowania, rozbudowanemu edytorowi oraz wsparciu dla technologii immersyjnych.

Silnik ten oferuje użytkownikom możliwość programowania logiki aplikacji zarówno w języku C++, jak i z wykorzystaniem wizualnego systemu skryptowego Blueprint, co znacząco przyspiesza proces tworzenia prototypów i ułatwia implementację interakcji w środowisku VR. Unreal Engine zapewnia również bogaty zestaw narzędzi do tworzenia animacji, efektów specjalnych, obsługi dźwięku oraz integracji z zewnętrznymi bibliotekami.

Podczas analizy możliwych rozwiązań rozważano również wykorzystanie silnika Unity, który podobnie jak Unreal Engine jest szeroko stosowany w branży gier i aplikacji VR. Unity charakteryzuje się dużą elastycznością, wsparciem dla wielu platform oraz dostępnością licznych wtyczek i rozszerzeń. W porównaniu z Unreal Engine, Unity posiada mniej rozbudowany natywny system graficzny oraz wymaga większego nakładu pracy przy tworzeniu zaawansowanych efektów wizualnych.

Ostatecznym argumentem przemawiającym za wyborem Unreal Engine była kwestia kompatybilności — Laboratorium Zanurzonej Wizualizacji Przestrzennej, w którym aplikacja miała zostać wdrożona, nie wspiera nowszych wersji Unity, natomiast Unreal Engine zapewniał pełną zgodność z istniejącą infrastrukturą sprzętową i programową laboratorium.

5.2. *Laboratorium Zanurzonej Wizualizacji Przestrzennej (Konrad Czarnecki)*

Laboratorium Zanurzonej Wizualizacji Przestrzennej (LZWP) to specjalistyczne środowisko badawczo-edukacyjne wyposażone w systemy rzeczywistości wirtualnej, umożliwiające tworzenie i testowanie aplikacji immersyjnych w warunkach kontrolowanych. W skład laboratorium wchodzą między innymi systemy typu CAVE, czyli pomieszczenia projekcyjne z ekranami scieniymi i podłogowymi, które otaczają użytkownika obrazem 3D wyświetlonym z kilku projektorów.

LZWP wyposażone jest również w systemy śledzenia pozycji i ruchu użytkownika oraz kontrolery umożliwiające interakcję z wirtualnym środowiskiem. Dzięki temu laboratorium stanowi doskonałe zaplecze do testowania aplikacji edukacyjnych VR oraz prowadzenia badań nad efektywnością i ergoniografią rozwiązań immersyjnych.

5.3. Środowisko 3D (Konrad Czarnecki)

W procesie tworzenia aplikacji niezbędne było opracowanie modeli 3D reprezentujących obiekty, elementy wystroju oraz interaktywne przedmioty pojawiające się w wirtualnym escape roomie. Do tego celu wykorzystano Blender — darmowe oprogramowanie do modelowania trójwymiarowego, animacji, teksturowania oraz renderowania.

Blender oferuje szeroki zakres narzędzi umożliwiających tworzenie szczegółowych modeli 3D, generowanie animacji oraz przygotowywanie materiałów i tekstur kompatybilnych z silnikami do tworzenia gier. Dzięki wsparciu dla formatów eksportowych takich jak FBX i GLTF, modele stworzone w Blenderze mogły zostać bezproblemowo zimportowane do Unreal Engine i wykorzystane w aplikacji VR.

5.4. System kontroli wersji (Konrad Czarnecki)

Dla zapewnienia bezpieczeństwa danych oraz efektywnego zarządzania projektem zastosowano system kontroli wersji Git wraz z usługą hostingową GitHub. GitHub umożliwia przechowywanie kodu źródłowego, modeli 3D, plików dźwiękowych oraz dokumentacji projektowej w repozytorium zdalnym z możliwością współdzielenia zasobów pomiędzy członkami zespołu.

Dzięki systemowi kontroli wersji możliwe było śledzenie historii zmian, zarządzanie gałęziami projektowymi oraz szybkie przywracanie poprzednich wersji w przypadku wystąpienia błędów. W projekcie GitHub pełnił również rolę platformy do przechowywania backupów.

6. ORGANIZACJA PRACY (JAN WALCZAK)

Zarządzanie zespołem i dobra organizacja pracy to dwa kluczowe aspekty podczas pracy grupowej. Metodyczne działanie pozwala usprawnić i uporządkować pracę projektową i komunikację w zespole [1]. Zespół, który implementował omawiane w tej pracy rozwiązanie, składał się z trzech osób: Jana Walczaka, Konrada Czarneckiego i Andriego Demyshyna.

6.1. System kontroli wersji (Jan Walczak)

6.1.1. Zastosowane rozwiązanie

System kontroli wersji pomaga śledzić historię zmian dokonywanych przez uczestników projektu. Ułatwia zarządzanie procesem tworzenia projektu, zapewnia integralność danych oraz gwarantuje, że dokonywane zmiany są na bieżąco zapisywane i udostępniane uczestnikom [2].

Do wdrożenia kontroli wersji w omawianym projekcie użyto internetowej platformy GitHub. Organizacja pracy sprowadza się do opracowania zasad dotyczących tzw. „Flow“. „Flow“ pochodzi od angielskiego słowa „workflow“ (ang. przepływu pracy). Taki przepływ ma na celu opisanie i wdrażanie zmian zachodzących w danym projekcie tak, aby ułatwić ich zrozumienie nie tylko deweloperom, ale także użytkownikom, którzy śledzą zmiany zachodzące w repozytorium [2].

6.1.2. Kontrola przepływu

Przepływ zaczyna się od zdefiniowania problemu, poprzez opisanie problemu lub funkcjonalności, których będzie dotyczyć implementowana zmiana. Na platformie GitHub odbywa się to po przez tworzenie Issues (zagadnienia). Pozwalają one również na planowanie i prowadzenie dyskusji na temat danego zagadnienia z innymi uczestnikami projektu lub użytkownikami [3]. Dodatkową zaletą jest możliwość bezpośredniego powiązania zmian dokonanych w kodzie z unikalnym numerem, który jest nadawany każdej takiej dyskusji przez platformę GitHub. W omawianym projekcie zdecydowano się na wprowadzanie zasad dotyczących formułowania tytułu i treści Issue. Tytuł powinien być napisany w języku angielskim, tak żeby każda osoba przeglądająca repozytorium mogła powiązać problem i dyskusję ze zmianą w aplikacji. Dodatkowo powinien być zwięzły i dotyczyć wyłącznie jednej zmiany. Opis i dyskusja mogą być napisane w języku polskim, tak żeby maksymalnie usprawnić komunikację zespołu. Issue może znajdować się w dwóch stanach: zamknięte – czyli stan, w którym praca została zakończona i otwarte – czyli stan, w którym praca nad danym zagadnieniem nadal trwa.

<input type="checkbox"/>	Open	1	Closed	43
<input type="checkbox"/>	<input checked="" type="checkbox"/> Add lifes system for arithmetric room	#41 · by janwalec was closed on Nov 18, 2025		
<input type="checkbox"/>	<input checked="" type="checkbox"/> Add playable mechanics to the arithmetic sequence room	#39 · by janwalec was closed on Nov 17, 2025		
<input type="checkbox"/>	<input checked="" type="checkbox"/> Add arithmetic sequence room	#37 · by janwalec was closed on Nov 16, 2025		

Rysunek 6.1: Zrzut ekranu z platformy GitHub – tablica z przykładowymi Issues do projektu.

W repozytorium znajduje się tak zwana główna linia (*ang. main line of development*). Jest to liniowa historia zmian, które zostały wprowadzone do projektu. Deweloperzy mogą wprowadzać swoje zmiany, które są definiowane tym, w jaki sposób (w którym punkcie w historii zmian) odłączyły się od tej linii. W ten sposób można tworzyć rozgałęzienia (*ang. branches*) względem głównej linii [4]. Główna gałąź (*ang. main branch*), może umożliwić deweloperom zorientowanie się czy zmiana, którą chcą wprowadzić, będzie mogła być połączona z aktualnie istniejącą wersją aplikacji. Ważnym elementem przepływu jest więc zdefiniowanie tego w jaki sposób każda zmiana będzie dołączana do głównej gałęzi. Podczas implementacji projektu zdecydowano się na bezpośrednie powiązanie gałęzi z Issues, tj. nazywanie ich tymi samymi tytułami oraz wiązanie ich przez linkowanie unikalnego identyfikatora nadawanego każdemu Issue.

6.1.3. Łączenie gałęzi

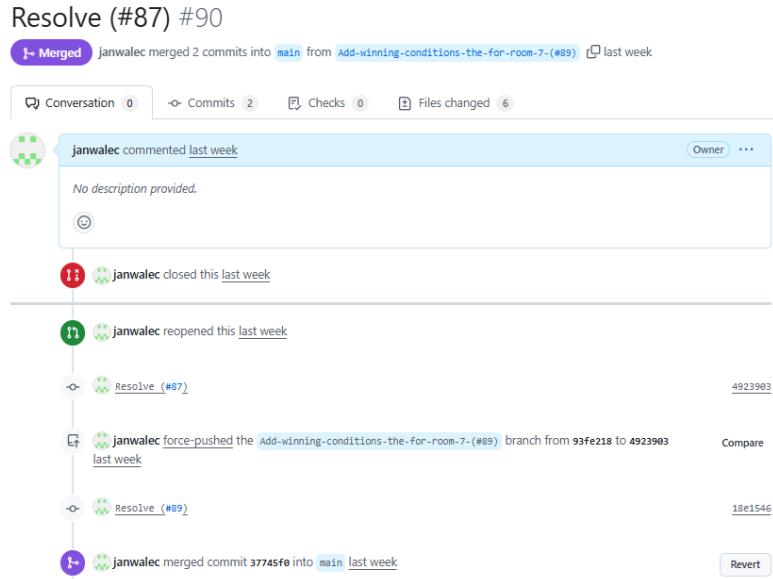
Domyślnie gałęzie mogą być wiązane bez żadnych ograniczeń. GitHub umożliwia tworzenie zbioru zasad (*ang. rulesets*) dotyczących wiązania gałęzi deweloperów z innymi gałęziami. Zasady pozwalają na określenie, które grupy są upoważnione do jakich czynności związanych z gałęzią oraz definiują jakie zasady muszą spełnić aby powiązać swoją gałąź, czyli wprowadzaną zmianę, do danej linii. [5]. W projekcie zastosowano następujące zasady, dotyczące głównej linii:

- `Restrict deletions` – nie pozwala na usuwanie powiązanych referencji.
- `Require a pull request before merging` – powiąż z gałęzią przez „pull request“.
- `Block force pushes` – nie pozwól na wymuszenie nadpisania zmian.

W celu spełnienia zasady `Require a pull request before merging` każda wprowadzana zmiana musi zostać przygotowana do połączenia. Odbywa się to przez mechanizm `Pull request`. Umożliwia on wgląd we wprowadzane modyfikacje innym uczestnikom projektu przed ich finalnym załatwieniem i wdrożeniem [6]. W projekcie określono, że żeby zmiana została połączona z główną gałęzią, musi być zatwierdzona przez minimalnie jednego uczestnika projektu, niebędącego autorem. Dodatkowo, zmiana musi być możliwa do powiązania bez tak zwanych konfliktów, czyli fragmentów plików, których system kontroli wersji nie mógł samodzielnie scalić ze zmianami [6].

Zmiany wprowadzane przez deweloperów składają się z serii migawek `Commit`. `Commit` zawiera wiadomość, unikalny identyfikator oraz listę plików i zmian, które są z nim powiązane. Pozwala na wielokrotny zapis zmian podczas pracy w ramach jednej gałęzi. W projekcie zostało nałożone ograniczenie, wymuszające na uczestnikach scalanie migawek w jedność za pomocą mechanizmu `rebase i squash` [6].

Po poprawnym utworzeniu i scaleniu Pull request utworzona wcześniej gałąź może zostać usunięta. Jest to bezpieczne, ponieważ gałąź zostaje dołączona do głównej linii jako Commit i historia zachowuje swój liniowy charakter. Deweloper powinien oznaczyć Issue jako zamknięte. Pull request otrzyma status Merged automatycznie.



Rysunek 6.2: Zrzut ekranu z platformy GitHub – tablica z przykładowym Pull Request

7. ANALIZA DYDAKTYCZNA I POTENCJAŁ EDUKACYJNY

7.1. *Motywacja uczniów do nauki matematyki*

7.2. *Możliwość personalizacji i różnicowania poziomu trudności*

7.3. *Wpływ imersji na naukę*

8. OPIS TEORETYCZNY ZAGADEK

W poniższym rozdziale znajduje się teoretyczny opis zagadek, który stanowi punkt wejściowy do implementacji projektu inżynierskiego. Stanowi on podstawę teoretyczną i nakreśla charakter pracy. Opisy poszczególnych zagadek zostały zredagowane na podstawie ogólnodostępnej podstawy programowej dla szkół średnich z przedmiotu matematyka. Jest to kluczowy aspekt całego projektu inżynierskiego, gdyż aplikacja końcowa ma być skierowana do uczniów szkół średnich, w związku z czym wymagany zakres wiedzy nie może wykraczać poza podstawę programową.

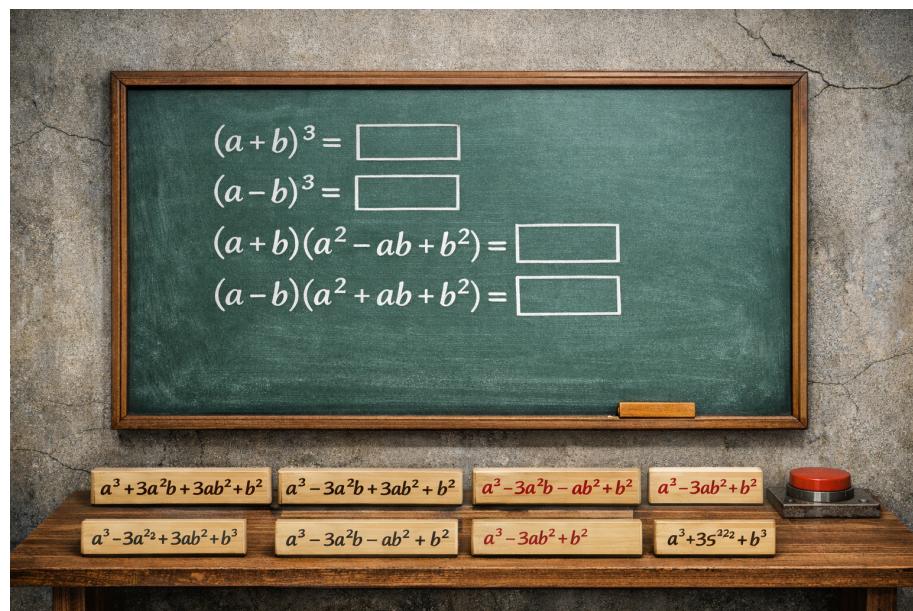
8.1. Zadanie 1 – Wzory skróconego mnożenia (Andrii Demyshyn)

8.1.1. Cel zadania

Celem zadania jest utrwalenie wiedzy z zakresu wzorów skróconego mnożenia oraz rozwijanie umiejętności rozpoznawania poprawnych zależności algebraicznych. Zadanie ma charakter wprowadzający i pozwala uczestnikowi na przypomnienie podstawowych wzorów omawianych w programie nauczania matematyki na poziomie szkoły średniej. Dodatkowo zadanie wspiera rozwój logicznego myślenia oraz umiejętność analizy struktury wyrażeń algebraicznych.

8.1.2. Zasady działania

Zadanie polega na dopasowaniu początków wzorów skróconego mnożenia do ich poprawnych zakończeń. Uczestnik otrzymuje zestaw rozpoczętych wyrażeń algebraicznych, w których brakują prawe strony równań. Równocześnie dostępny jest zbiór możliwych zakończeń wzorów, wśród których znajdują się zarówno poprawne, jak i niepoprawne zapisy algebraiczne.



Rysunek 8.1: Przykładowy wygląd pierwszego zadania – wzory skróconego mnożenia

Celem użytkownika jest wybranie właściwych elementów i połączenie ich w taki sposób, aby

utworzyć kompletne i matematycznie poprawne wzory skróconego mnożenia. Zadanie wymaga znajomości podstawowych własności działań algebraicznych oraz umiejętności odróżniania poprawnych wzorów od błędnych.

8.1.3. *Zakończenie zadania*

Zadanie uznaje się za zakończone w momencie poprawnego uzupełnienia wszystkich wzorów skróconego mnożenia. Prawidłowe rozwiązanie potwierdza, że uczestnik posiada wymaganą wiedzę teoretyczną oraz potrafi ją zastosować w praktyce poprzez rozpoznawanie i kompletowanie wyrażeń algebraicznych.

8.2. *Zadanie 2 – Planimetria (Jan Walczak)*

Po ukończeniu pierwszego zadania pojawia się tabela z trzema kolumnami. Dwie skrajne kolumny zawierają nazwy figur geometrycznych. Środkowa kolumna jest pusta i będzie uzupełniana przez gracza odpowiednimi symbolami.

8.2.1. *Cel zadania*

Celem gracza jest poprawne ułożenie relacji między figurami w kolejnych rzędach. Przykładowo, relacją taką jest to, że „każdy kwadrat jest rombem” lub „każdy kwadrat jest prostokątem”.

8.2.2. *Zasady działania*

Gracze przechodzą przez kolejne rzędy sekwencyjnie. Dopóki nie ułożą pierwszej relacji poprawnie, to nie mogą ułożyć kolejnej. Po poprawnym ułożeniu rzędu (wstawieniu odpowiedniego symbolu) podświetla się on na zielono, sygnalizując graczowi, że może przejść do kolejnego punktu. Symbole, które układają gracze są następujące:

- > to <
- > to nie <

Kwadrat	> to <	prostokąt
Prostokąt	> to nie <	Kwadrat
romb	[]	kwadrat
...

Tabela 8.1: Przykładowa tabela zawierająca zależności między figurami geometrycznymi.

Gracze kierują się swoją wiedzą matematyczną oraz następującymi własnościami figur:

- liczba boków,
- długości boków,
- kąty (proste lub nie),
- cechy charakterystyczne danej figury.

8.2.3. *Zakończenie zadania*

Po poprawnym uzupełnieniu wszystkich relacji zadanie uznaje się za zakończone. Gracz może przejść do kolejnego zadania.

8.3. Zadanie 3 – Nierówności (Konrad Czarnecki)

Po wejściu do kolejnego pomieszczenia gracz widzi przed sobą most zawieszony nad przeświącią. Po drugiej stronie znajduje się zamknięte przejście, do którego gracz musi się dostać. Na bocznych ścianach wypisany jest układ nierówności: $2x - 5 < 9$; $x + 1 \geq 4$. Na płytach mostu umieszczone są różne liczby z zakresu liczb całkowitych (np.: 1, 2, 3, 4, 5, 6, 8, 7, 10).

8.3.1. Cel zadania

Gracz musi rozwiązać układ nierówności i wyznaczyć przedział, który spełnia oba warunki. Następnie powinien przeprowadzić postać przez most, poruszając się wyłącznie po płytach z wartościami należącymi do tego przedziału. Rozwiązywanie układu

1. Rozwiązywanie pierwszej nierówności:

- $2x - 5 < 9$
- $2x < 14$
- $x < 7$

2. Rozwiązywanie drugiej nierówności:

- $x + 1 \geq 4$
- $x \geq 3$

3. Wspólny przedział:

- $x \in [3; 7)$

Gracz musi wybrać wyłącznie płytki z wartościami większymi lub równymi 3 i mniejszymi od 7, np. 3, 4, 5, 6.

8.3.2. Zasady działania

- Gracz poruszając postacią, przechodzi przez kolejne płytki.
- Poprawna płytka – postać przechodzi dalej.
- Błędna płytka – płytka zapada się lub podświetla na czerwono, a postać wraca na początek mostu. Gracz może próbować dowolną liczbę razy, aż do skutecznego przejścia na drugą stronę.

8.3.3. Zakończenie zadania

Zadanie zostaje uznane za zakończone, gdy graczowi uda się przejść na drugą stronę mostu.

8.4. Zadanie 4 – Funkcje (Konrad Czarnecki)

Zadania 4 i 5 są realizowane w jednym pomieszczeniu. Gracz widzi dwie duże tablice, umieszczone na ścianach. Na obu naniesiona jest siatka układu współrzędnych. W układzie współrzędnych pojawiają się trzy punkty, np.:

- Punkt A(1,2)
- Punkt B(3,4)
- Punkt C(5,6)

oraz wzór funkcji z losowo wybranymi współczynnikami. Poniżej widoczne są elementy służące do sterowania wykresem funkcji. Dają one możliwość zmiany współczynników wylosowanego wzoru.

8.4.1. Cel zadania

Celem gracza jest dobranie odpowiednich współczynników funkcji tak, aby przeszła ona przez wszystkie wyświetlane punkty.

8.4.2. Zasady działania

Podczas wybierania współczynników funkcji przez gracza, jej wykres na tablicy zmienia się na bieżąco zgodnie z ustawionym wzorem. Wykres funkcji powinien być losowany z wcześniej zdefiniowanej puli par typu funkcja – punkty, tak aby po każdorazowym uruchomieniu zadania gracz czuł, że ma przed sobą nowe wyzwanie.

8.4.3. Zakończenie zadania

Rozwiążanie jest sprawdzane na bieżąco i gdy gracz dobierze współczynniki funkcji poprawnie, tj. przetnie ona wszystkie wybrane punkty, zadanie zostaje uznane za rozwiązane. W takim przypadku tablica się blokuje i podświetla na zielono.

8.5. Zadanie 5 – Geometria analityczna (Konrad Czarnecki)

W układzie współrzędnych pojawiają się dwa wykresy funkcji. Są one losowane z pewnej określonej puli, podobnie jak w zadaniu 4.

8.5.1. Cel zadania

Zadaniem gracza jest znalezienie punktu przecięcia dwóch funkcji prostych wyświetlanych w układzie współrzędnych. Określenie puli wyklucza możliwość prostych równoległych, które nie mają ze sobą żadnych punktów wspólnych lub w całości się pokrywają. W tych przypadkach zadanie byłoby niemożliwe do rozwiązania. Gracz będzie wpisywał swoją odpowiedź (współrzędne punktu przecięcia) na klawiaturze znajdującej się pod tablicą.

8.5.2. Zasady działania

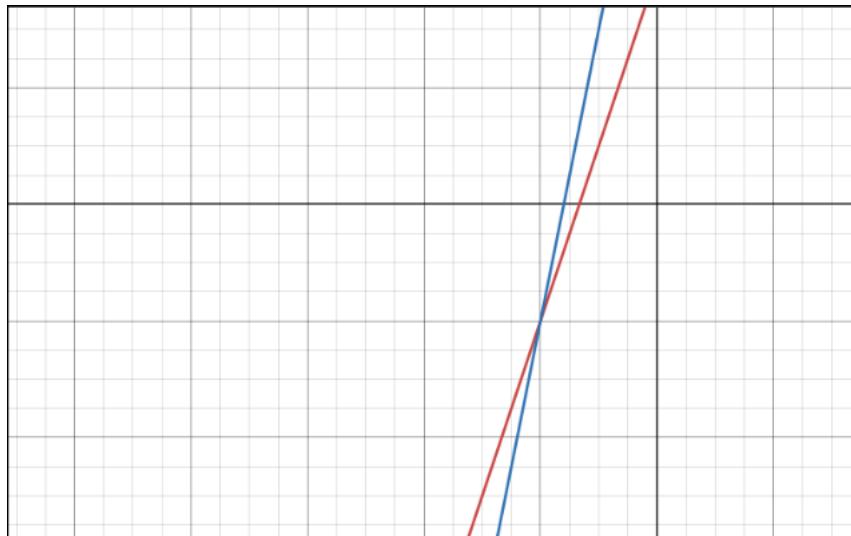
Skala osi współrzędnych jest dobrana tak, by gracz nie mógł odczytać z niej rozwiązania; musi rozwiązać układ równań dysponując dwoma wzorami funkcji. Rysunek 8.2 obrazuje dwa wykresy:

- $y = 3x + 4$ (czerwony)
- $y = 5x + 8$ (niebieski)

Gracz dysponuje tymi wzorami i na ich podstawie musi rozwiązać układ równań. Wykresy powinny być dobrane tak, aby obie współrzędne punktu przecięcia były liczbami całkowitymi.

8.5.3. Zakończenie zadania

Rozwiążanie jest sprawdzane na bieżąco i gdy gracz dobierze współrzędne punktu przecięcia funkcji poprawnie, zadanie zostaje uznane za rozwiązane. W takim przypadku tablica się blokuje i podświetla na zielono. Jeśli gracz wykonał wcześniej zadanie czwarte (znajdujące się w tym samym pokoju), przechodzi do następnego zadania.



Rysunek 8.2: Przykładowy wygląd tablicy z przecinającymi się wykresami

8.6. *Zadanie 6 – Kombinatoryka (Andrii Demyshyn)*

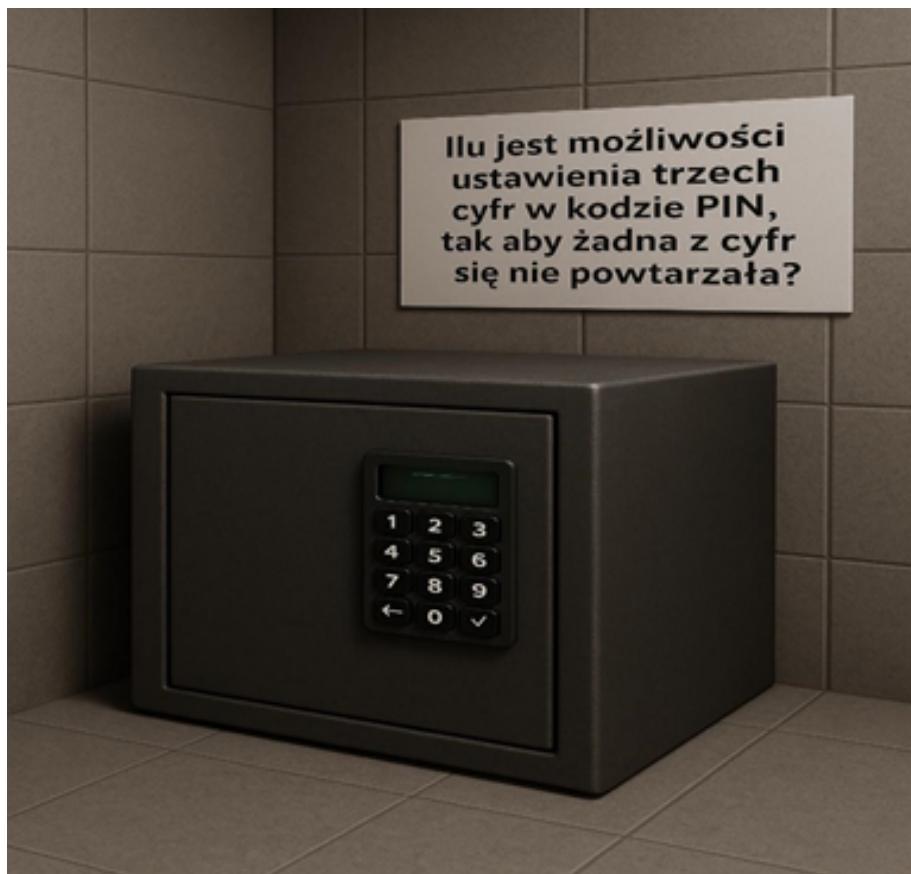
Gracz widzi na środku pokoju zamknięty sejf.

8.6.1. *Cel zadania*

Gracze zostają poinformowani, że muszą wyliczyć, ile jest możliwych do ułożenia, trzycyfrowych „kodów” do sejfu bez powtarzających się cyfr.

8.6.2. *Zasady działania*

Rozwiązaniem jest kod, a nie liczba – oznacza to, że początkową cyfrą w trzycyfrowym kodzie może być zero. Powinno być to klarownie zakomunikowane graczowi np. na tabliczce nad sejfem.
Rozwiązaniem zadania jest: $10 \times 9 \times 8 = 720$



Rysunek 8.3: Przykładowy wygląd sejfu występującego w zadaniu

8.6.3. Zakończenie zadania

Po poprawnym wpisaniu kodu sejf automatycznie się otwiera, co utwierdza gracza w przekonaniu, że poprawnie wykonał zadanie. W środku znajdują się elementy potrzebne do wykonania następnego zadania.

8.7. Zadanie 7 – Liczby rzeczywiste i działania na zbiorach liczbowych

(Jan Walczak)

W kolejnym zadaniu gracze znajdują skrzynię, na której powierzchni – z każdej strony – narysowane są liczby: -3; 7; 12; 2; 5; 0; 10; 3,75; $\sqrt{2}$.

8.7.1. Cel zadania

Skrzynia wyposażona jest w mechaniczny zamek z pięcioma polami na cyfry, obok których znajdują się symbole zbiorów:

- N – liczby naturalne,
- Z – liczby całkowite,
- R – liczby rzeczywiste,
- W – liczby wymierne,
- NW – liczby niewymierne,

Gracz musi uważnie obejrzeć skrzynię i policzyć ile liczb, zapisanych na skrzyni, należy do jakiego zbioru. Po wpisaniu odpowiednich liczb, obok symboli zbiorów, skrzynia otwiera się, a gracz otrzymuje kilka symboli:

- zawieranie się zbiorów – \subset ,
- zbiór pusty – \emptyset ,
- iloczyn zbiorów – \cap .

Tym samym gracz przechodzi do drugiego etapu zadania. W drugim etapie w pokoju ukazuje się plansza z symbolami zbiorów, takimi jak na skrzyni. Celem gracza jest ułożenie poprawnej relacji między nimi tj. zbiór liczb naturalnych zawiera się w zbiorze liczb całkowitych, zbiór liczb całkowitych zawiera się w zbiorze liczb rzeczywistych itd.

8.7.2. Zasady działania

W pierwszym etapie zadania gracz powinien policzyć ile liczb ze skrzyni należy do danego zbioru i wpisać odpowiednią odpowiedź na kłódce. Przykładowo:

- N (naturalne): 7; 12; 2; 5; 10 – 5 liczb,
- C (całkowite): -3; 7; 12; 2; 5; 0; 10 – 7 liczb,
- R (rzeczywiste): wszystkie liczby – 9 liczb,
- W (wymierne): -3; 7; 12; 2; 5; 0; 10; 3,75 – 8 liczb,
- NW (niewymierne): $\sqrt{2}$ – 1 liczba.

Kombinacja do ustawienia na kłódce: 5, 7, 9, 8, 1.

W drugim etapie gracz powinien ustawić posiadane symbole między literami symbolizującymi kolejne zbiory i odpowiednio je obrócić, tak aby powstała między nimi poprawna relacja tj.

- $N \subset Z \subset R$,
- $NW \cap W = \emptyset$,
- $NW \cap R$.

8.7.3. Zakończenie zadania

Zadanie zostaje uznane za zakończone, gdy gracz poprawnie przejdzie przez oba etapy. Gracz nie może przejść do etapu drugiego, bez zakończenia etapu pierwszego – jest to wymuszone przez wymaganie otwarcia przez niego skrzyni.

8.8. Zadanie 8 – Znaki funkcji trygonometrycznych (Konrad Czarnecki)

Po ukończeniu zadania ze zbiorami gracz kieruje się do ściany z czterema dużymi okręgami jednostkowymi, oznaczonymi nazwami funkcji:

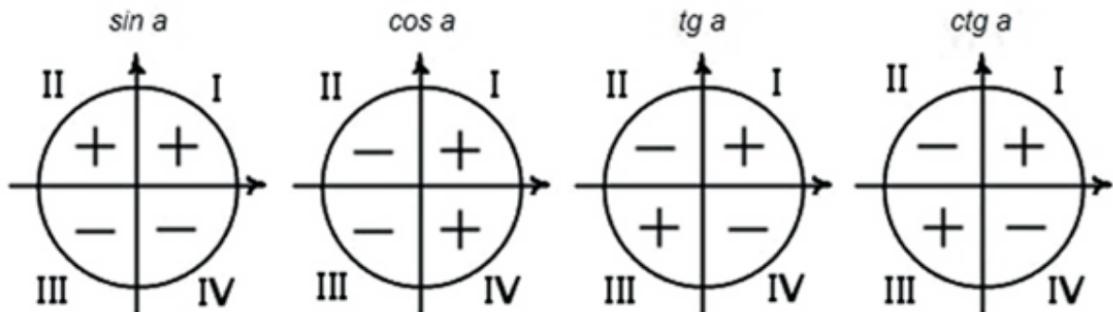
- $\sin(x)$
- $\cos(x)$
- $\tg(x)$
- $\ctg(x)$

Każdy z okręgów podzielony jest na cztery ćwiartki, oznaczone jako I, II, III, IV. Przy każdej ćwiartce znajduje się puste miejsce, które gracz musi wypełnić odpowiednim znakiem:

- „+” (plus) — funkcja przyjmuje wartości dodatnie

- „-” (minus) — funkcja przyjmuje wartości ujemne

Gracz zmienia znak naciskając na niego. Domyślnie wszystkie znaki są ustawione na puste, dopiero po pierwszym naciśnięciu zmieniają się na znak „+”, po kolejnym na „-”, następnie znów na „+”, itd.



Rysunek 8.4: Ćwiartki układów jednostkowych ze znakami funkcji trygonometrycznych

8.8.1. Cel zadania

Celem gracza jest poprawne ustawienie znaków funkcji trygonometrycznych w każdej ćwiartce układu współrzędnych.

8.8.2. Zasady działania

Gracz przeciąga kostki „+” i „-” i umieszcza je w odpowiednich miejscach na planszy. Może dowolnie poprawiać swój wybór, dopóki nie zatwierdzi odpowiedzi.

8.8.3. Zakończenie zadania

Po poprawnym ułożeniu wszystkich znaków cianana rozświetla się na zielono.

8.9. Zadanie 9 – Ciągi liczbowe (Jan Walczak)

Poniżej opisane zadanie będzie podobne do gry wyprodukowanej przez Nintendo na platformę Pegasus.

8.9.1. Cel zadania

Zadaniem gracza będzie odpowiednio szybko obliczyć kolejne wyrazy ciągu arytmetycznego z podanego wzoru. Będzie on wyposażony w laserowy pistolet, którym będzie musiał strzelać w odpowiednio oznaczone kaczki.

8.9.2. Zasady działania

Na ekranie pojawia się formuła ciągu arytmetycznego lub geometrycznego, np.: $a_n = 2n + 1$ (ciąg arytmetyczny) wybranego z predefiniowanej puli. Gracz będzie musiał wybrać i strzelić do kaczki oznaczonej odpowiednią wartością kolejnych wyrazów ciągu. Gra przyspiesza (kaczki lecą coraz szybciej) razem z postępem w zadaniu. Aby ułatwić graczowi zadanie, na górze ekranu będzie podany nie tylko wzór ciągu, ale też aktualny numer wyrazu tego ciągu. Przykładowo, dla

wzoru $a_n = 2n + 1$:

- $n = 1$ – gracz musi trafić w kaczkę z liczbą 3,
- $n = 2$ – gracz musi trafić w kaczkę z liczbą 5

8.9.3. Zakończenie zadania

Gracz będzie zdobywał punkty za każdy poprawnie wybrany wyraz ciągu. Gra zakończy się po upływie określonego czasu lub jeśli gracz pomyli się trzy razy



Rysunek 8.5: gra Duck Hunt

8.10. Zadanie 10 – Prawdopodobieństwo (Andrii Demyshyn)

Gracze podchodzą do stołu ustawionego w rogu pokoju, gdzie siedzi postać Morfeusza. Na stole stoją dwa identyczne pojemniki, a obok leży 100 tabletów — 50 czerwonych i 50 niebieskich. Pojawia się komunikat: „Pomóż Morfeuszowi zwiększyć jego szanse na powrót do rzeczywistości. Podziel tabletki między dwa pojemniki tak, aby miał jak największą szansę na wybranie czerwonej.”

8.10.1. Cel zadania

Gracze muszą znaleźć optymalny układ, czyli: W pierwszym pojemniku umieścić jedną czerwoną tabletkę, W drugim pojemniku umieścić 49 czerwonych i 50 niebieskich tabletów (lub odwrotnie). Rozwiążanie działa niezależnie od tego, który pojemnik traktujemy jako „pierwszy”, a który jako „drugi”). Taki układ osiągnie maksymalne prawdopodobieństwo wylosowania czerwonej tabletki – 74,75

8.10.2. Zasady działania

Gracze mogą: przeciągać tabletki do pojemników w dowolny sposób, testować różne rozkłady, sprawdzać procent szans na wygraną, który wyświetla się po każdym rozłożeniu.



Rysunek 8.6: Przykładowy wygląd w zadaniu

Po każdym rozkładzie system oblicza i wyświetla szansę na wygraną. Jeśli gracz nie osiągnie 74,75%, pojawia się komunikat: „Możesz to zrobić lepiej! Spróbuj jeszcze raz.” Jeśli gracz osiągnie 74,75% system gratuluje i zalicza zadanie.

8.10.3. Zakończenie zadania

Po poprawnym rozłożeniu tabletek i osiągnięciu maksymalnej szansy 74,75%, Morfeusz wstaje od stołu, uśmiecha się i mówi: „Dziękuję. Dzięki Wam mam szansę wrócić do rzeczywistego świata.” Po chwili Morfeusz znika, rozpuszczając się w powietrzu niczym hologram lub efekt teleportacji, symbolizujące jego powrót do rzeczywistości.

8.11. Zadanie 11 – Optymalizacja i rachunek różniczkowy (Autor)

a

8.11.1. Cel zadania

a

8.11.2. Zasady działania

a

8.11.3. Zakończenie zadania

8.12. Zadanie 12 – Układy równań (Andrii Demyshyn)

Na środku pomieszczenia znajduje się stół z trzema przezroczystymi pojemnikami, oznaczonymi kolorami: Czerwony (x), Zielony (y), Niebieski (z). Bezpośrednio nad stołem, na ścianie, umieszczona jest duża żarówka, która zapala się, gdy gracze prawidłowo uzupełnią pojemniki. Obok znajduje się instrukcja techniczna: „Aby uruchomić instalację świetlną, należy załadować dokładnie 12 kulek. Czerwona kulka jest dwa razy bardziej wydajna niż zielona. Czerwona i dwie zielone razem dają o 9 jednostek energii więcej niż niebieska.”

8.12.1. Cel zadania

Gracze mają za zadanie umieścić odpowiednią liczbę kolorowych kulek w każdym pojemniku, aby spełnić wszystkie trzy warunki jednocześnie.

8.12.2. Zasady działania

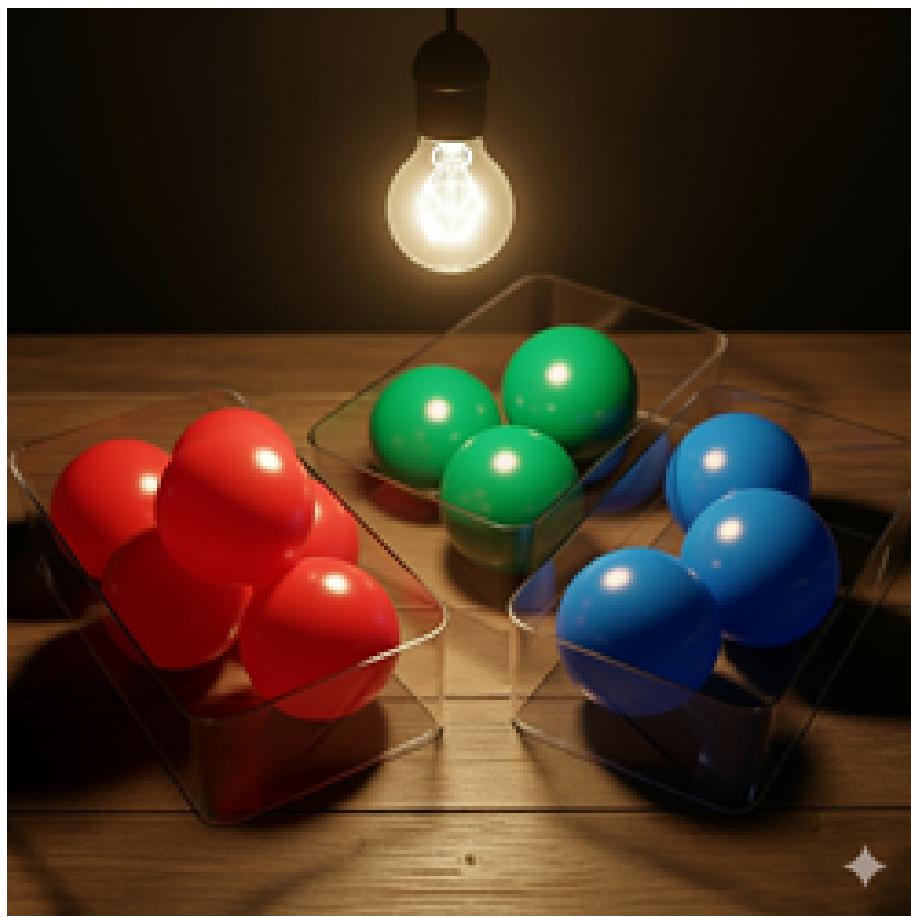
- Gracze wkładają kolorowe kulki do pojemników,
- System na bieżąco sprawdza poprawność ustawienia,
- W razie błędu wyświetlany jest komunikat o błędzie..„Niepoprawne ustawienie. Spróbuj ponownie.”

$$\begin{cases} x + y + z = 12 & x \text{ -- liczba kul czerwonych} \\ x = 2y & y \text{ -- liczba kul zielonych} \\ x + 2y - z = 9 & z \text{ -- liczba kul niebieskich} \end{cases}$$

- Czerwony (x) = 6

- Zielony (y) = 3

- Niebieski (z) = 3



Rysunek 8.7: Przykładowy wygląd zadania

8.12.3. Zakończenie zadania

Po poprawnym ułożeniu żarówka zapala się nad stołem sygnalizując zakończenie zadania.

8.13. Zadanie 13 – Stereometria(Jan Walczak)

Bryły przestrzenne mogą sprawić uczniom trudność, ponieważ wymagają przejścia od rysunku dwuwymiarowego tj. takiego na kartce papieru, do wyobrażenia sobie ich w przestrzeni trójwymiarowej.

8.13.1. Cel zadania

Celem zadania jest zapoznać uczniów z podstawowymi wartościami brył geometrycznych poprzez ich samodzielne odkrywanie.

8.13.2. Zasady działania

Gracz wchodzi do ciemnego pokoju i jest wyposażony w źródło światła np. pochodnię lub latarkę. Na środku pokoju znajduje się jedna z brył, wylosowanych z danej puli: prostopadłościan, ostrosłup, graniastosłup, kula lub sześcian. Na ścianie pokoju znajduje się panel z pytaniami np.:

- Ile ścian ma dana bryła?
- Ile krawędzi ma dana bryła?

- Co jest podstawą danej bryły?

Gracz obchodząc bryłę dookoła i rozświetlając ją latarką powinien móc udzielić odpowiedzi na te pytania.

8.13.3. Zakończenie zadania

Gdy gracz udziela poprawnych odpowiedzi na kolejne pytania zostają one podświetlone na zielono, sygnalizując graczy, że wykonał zadanie poprawnie. Po uzupełnieniu wszystkich odpowiedzi pokój rozświetla się, a gracz może przyjrzeć się bryle w pełnym świetle.

8.14. Zadanie 13 – Stereometria (Jan Walczak)

Bryły przestrzenne mogą sprawić uczniom trudność, ponieważ wymagają przejścia od rysunku dwuwymiarowego tj. takiego na kartce papieru, do wyobrażenia sobie ich w przestrzeni trójwymiarowej.

8.14.1. Cel zadania

Celem zadania jest zapoznać uczniów z podstawowymi wartościami brył geometrycznych poprzez ich samodzielne odkrywanie.

8.14.2. Zasady działania

Gracz wchodzi do ciemnego pokoju i jest wyposażony w źródło światła np. pochodnię lub latarkę. Na środku pokoju znajduje się jedna z brył, wylosowanych z danej puli: prostopadłościan, ostrosłup, graniastosłup, kula lub sześcian. Na ścianie pokoju znajduje się panel z pytaniami np.:

- Ile ścian ma dana bryła?
- Ile krawędzi ma dana bryła?
- Co jest podstawą danej bryły?

Gracz obchodząc bryłę dookoła i rozświetlając ją latarką powinien móc udzielić odpowiedzi na te pytania.

8.14.3. Zakończenie zadania

Gdy gracz udziela poprawnych odpowiedzi na kolejne pytania zostają one podświetlone na zielono, sygnalizując graczy, że wykonał zadanie poprawnie. Po uzupełnieniu wszystkich odpowiedzi pokój rozświetla się, a gracz może przyjrzeć się bryle w pełnym świetle.

9. IMPLEMENTACJA ZAGADEK

9.1. Zadanie 1 – wzory skróconego mnożenia (Andrii Demyshyn)

W tym zadaniu zrealizowano zagadkę poświęconą wzorom skróconego mnożenia. Zadaniem gracza jest prawidłowe połączenie początków i końcówek wzorów skróconego mnożenia poprzez umieszczenie odpowiednich klocków w przeznaczonych dla nich miejscach.

Kiedy gracz pojawia się w pokoju, widzi przed sobą tablicę z początkami wzorów skróconego mnożenia do uzupełnienia, a także klocki z różnymi formułami, panel informacyjny oraz przycisk do sprawdzenia odpowiedzi.

9.1.1. Opis obiektów pokoju

Matematyczna tablica BP_MathBoard

Głównym obiektem pokoju jest matematyczna tablica zrealizowana w postaci blueprintu aktora BP_MathBoard. Tablica zawiera siedem wzorów do uzupełnienia:

1. Kwadrat sumy

$$(a + b)^2 = a^2 + 2ab + b^2$$

2. Kwadrat różnicy

$$(a - b)^2 = a^2 - 2ab + b^2$$

3. Różnica kwadratów

$$(a - b)(a + b) = a^2 - b^2$$

4. Sześcian sumy

$$(a + b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$$

5. Sześcian różnicy

$$(a - b)^3 = a^3 - 3a^2b + 3ab^2 - b^3$$

6. Rozkład sumy sześciennów

$$a^3 + b^3 = (a + b)(a^2 - ab + b^2)$$

7. Rozkład różnicy sześciennów

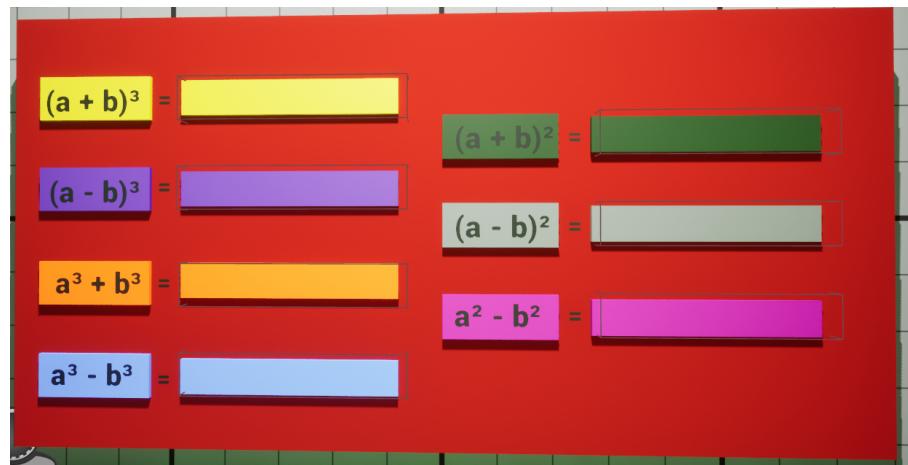
$$a^3 - b^3 = (a - b)(a^2 + ab + b^2)$$

Naprzeciw każdego wzoru, po znaku „=”, znajduje się odpowiedni slot, do którego należy umieścić końcówkę wzoru. Każdy slot składa się z następujących elementów:

- wyróżionego kolorem obszaru,
- komponentu Box Collision,
- komponentu Scene Component -- Point,

- oczekiwanej wartości typu bool Slot[i].IsNoFull, która informuje, czy slot jest zajęty (domyślnie ustalona na True),
- własnego materiału slotu Slot[i].HighLightMaterial.

Każdemu slotowi odpowiada tylko jeden poprawny obiekt z formułą.



Rysunek 9.1: Wygląd BP_MathBoard

9.1.2. Zakończenie zadania

9.1.3. Zakończenie zadania

W obiekcie BP_MathBoard zaimplementowano również funkcję AreAllSlotsFilled, która sprawdza, czy wszystkie sloty zostały uzupełnione oraz czy zostały uzupełnione poprawnie.

Prostopadłościan z formułą BP_FormulaBlock

Wszystkie formuły są przedstawione jako interaktywne obiekty 3D w postaci BP_FormulaBlock typu Blueprint Actor. Gracz może podnosić i przenosić te obiekty po pokoju oraz umieszczać je w slotach tablicy.

Każdy obiekt BP_FormulaBlock składa się z:

- komponentu StaticMesh,
- komponentu TextRender z odpowiednią formułą,
- komponentu Box Collision,
- identyfikatora FormulaID.

Występują zarówno poprawne obiekty BP_FormulaBlock, zawierające prawidłowe końcówki wzorów, jak i obiekty fałszywe (fikcyjne). Obiekty fałszywe wizualnie nie różnią się od poprawnych, lecz zawierają formuły oraz identyfikatory FormulaID, które nie pasują do żadnego wzoru. Zostały one dodane w celu zwiększenia poziomu trudności oraz zmylenia gracza. Formuły te są niemal poprawne, jednak zawierają drobne błędy, takie jak niewłaściwe znaki, współczynniki lub mieszanie typów wzorów.

Przycisk sprawdzenia BP_CheckMath

Na poziomie zaprojektowano przycisk BP_CheckMath, który umożliwia sprawdzenie poprawności uzupełnienia tablicy BP_MathBoard. Gracz decyduje, kiedy chce sprawdzić rozwiązanie, a po kliknięciu przycisku aktywowana jest funkcja AreAllSlotsFilled.

Panel informacyjny BP_MathText

Instrukcja zadania wyświetdana jest za pomocą blueprintu aktora BP_MathText, w którym umieszczono blueprintowy widget W_MathText.



Rysunek 9.2: Wygląd pokoju algebraicznego

9.1.4. Przebieg zadania

Po rozpoczęciu zadania gracz znajduje się w pokoju i widzi na panelu informacyjnym BP_MathText komunikat:

„Połącz odpowiednie początki i końce wzorów skróconego mnożenia. Przeciągnij klocki we właściwe miejsca.”

Jeżeli gracz podniesie obiekt z formułą i spróbuje umieścić go w slocie, w momencie wejścia obiektu w obszar Box_Collision slotu, obiekt tymczasowo przejmuje materiał slotu. Jeżeli w tym momencie gracz upuści obiekt, zostaje on automatycznie umieszczony w slocie. Rozwiązanie to zostało zastosowane w celu ułatwienia precyzyjnego umieszczania obiektów. Jeżeli gracz wyciągnie obiekt z obszaru kolizji slotu, jego materiał zostaje przywrócony do pierwotnego stanu.

W momencie umieszczenia obiektu w slocie tablicy BP_MathBoard, zmienna Slot[i].IsNoFull przyjmuje wartość False, co uniemożliwia umieszczenie innego obiektu w tym samym slocie do momentu usunięcia aktualnie umieszczonego klocka.

Gracz w dowolnym momencie może nacisnąć przycisk BP_CheckMath, który po interakcji odzwierciedla animację wcisnięcia oraz wywołuje funkcję AreAllSlotsFilled. Funkcja ta w pierwszej kolejności sprawdza, czy wszystkie sloty zostały uzupełnione. Jeżeli nie, do widgetu W_MathText wysyłany jest komunikat:

„Nie wszystkie formuły są jeszcze uzupełnione. Połącz każdą część początkową z odpowiednim zakończeniem.”

Jeżeli wszystkie sloty są uzupełnione, funkcja sprawdza poprawność wzorów poprzez porównanie identyfikatora `FormulaID` obiektu z identyfikatorem `ExpectedID` slotu. W przypadku poprawnego dopasowania wyświetlany jest komunikat:

„Świetnie! Poprawnie połączyleś wzory skróconego mnożenia.”

Zadanie zostaje zaliczone, a gra przechodzi do kolejnego poziomu.

W przeciwnym przypadku wyświetlany jest komunikat:

„Części formuł zostały połączone niepoprawnie. Spróbuj dobrać inne pary.”

Gracz może ponawiać próby do momentu poprawnego rozwiązania zadania.

9.2. Zadanie 2 – *Planimetria (Jan Walczak)*

9.2.1. Problemy i różnice w realizacji zadania w praktyce

Podczas implementacji zadania, zgodnie z teorią opisaną w podrozdziale 8.2, wystąpiło kilka problemów. Przede wszystkim zauważono, że odpowiedzi są zero-jedynkowe. Przykładowo: uczeń ma do uzupełniania relację typu „każdy kwadrat ... prostokątem”. Jeśli odpowie niepoprawnie, tj. zaznaczy odpowiedź „nie jest”, a zostanie od razu zapytany ponownie, o tę samą relację, to od razu wyklucza jedną z odpowiedzi. Tym samym zadanie zatraca swoją wartość edukacyjną – uczeń może rozwiązać całe zadanie, stosując jedynie metodę eliminacji.

Rozwiążanie tego problemu, które zostało zaimplementowane, to zdefiniowanie puli takich relacji i po udzieleniu przez ucznia odpowiedzi, każdorazowe losowanie relacji innej niż ta poprzednia. W ten sposób uniemożliwia się uczniowi stosowanie zasady eliminacji i wymusza na nim prawidłowe podejście.

Kolejnym problemem była prezentacja zadania. Zwykła tabela, którą uczeń miałby uzupełniać, mogłaby wydać mu się mało ciekawa. Tym samym postanowiono wizualnie usprawnić zagadkę. Na środku zostały umieszczone dwa przyciski:

- każdy
- nie każdy

Uczeń zostaje poinstruowany, że po obu ścianach pokoju zostaną wyświetlane różne figury geometryczne. Po lewej stronie, patrząc od przycisków – figury oznaczone kolorem czerwonym, po prawej stronie – figury oznaczone kolorem zielonym. Liczba figur jest stała. Każdorazowo typ wyświetlanych figur jest wybierany z określonej puli i wyświetlany w pseudolosowej konfiguracji – losowane jest ich położenie, obrót oraz rozmiar. Zadaniem ucznia jest uzupełnianie kolejnych relacji poprzez wybieranie odpowiednich przycisków. Relacja, wyświetlana na ścianie pokoju, przedstawia się jako: „każdy typ figury, narysowany kolorem czerwonym, jest równocześnie typem figury oznaczonym kolorem zielonym”. Dodatkowo tekst jest odpowiednio pokolorowany tak, aby uczeń nie miał wątpliwości, że chodzi o typ figury, wyświetlanej tymże kolorem na ścianach.

9.2.2. Implementacja struktury danych przechowującej relację

Na początku pracy należało zdefiniować strukturę, przechowującą relacje, czyli pytania, na które uczeń będzie odpowiadał. Relacja taka zawiera następujące pola:

- `Every` – wartość logiczna
- `FigureA` – ciąg tekstowy, reprezentujący pierwszą figurę w relacji

- FigureB – ciąg tekstowy, reprezentujący drugą figurę w relacji

Wartość zmiennej Every odpowiada na pytanie, „czy każda figura typu pierwszego (FigureA) jest równocześnie figurą typu drugiego (FigureB)?“. Struktura zawiera również funkcję słownikową, czyli taką, która tłumaczy wartości tekstowe na liczbowe.

9.2.3. Implementacja kontrolerów

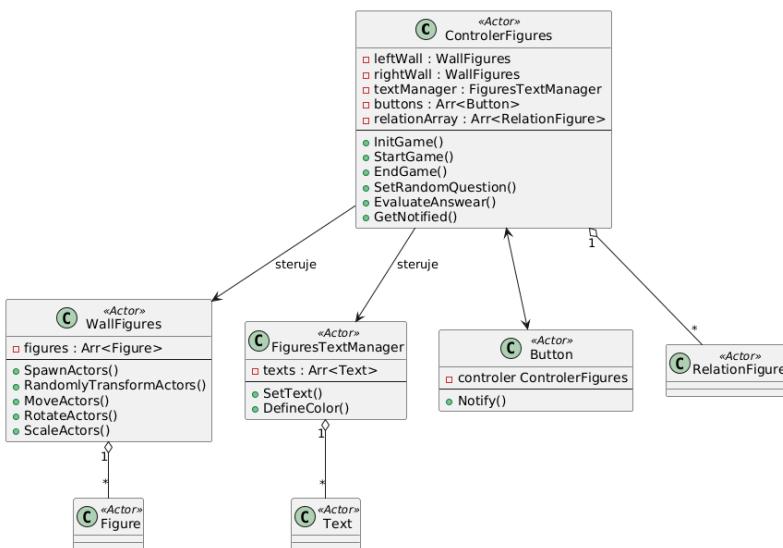
Aby zarządzać zagadką została zaimplementowana seria kontrolerów i menedżerów.

- ControllerFigures (główny kontroler)
- FiguresTextManager
- WallFigures

WallFigures to najprostszy z kontrolerów, wykonuje polecenia głównego kontrolera. Jego zadaniem jest wyświetlanie żądanych figur i ich losowe ustawianie w świecie gry (obracanie, skalowanie, rozmieszczanie). W projekcie występują jego dwie instancje: kontroler lewy i prawy. Odpowiadają one za odpowiednie ściany, na których wyświetlane są figury.

FiguresTextManager jest odpowiedzialny za zarządzanie tekstem wyświetlonym na ekranie. Wykonuje polecenia głównego kontrolera. Jego zadaniem jest odpowiednie wyświetlanie i kolorowanie tekstu.

ControllerFigures, czyli kontroler główny, jest najważniejszym elementem zagadki. Zawiera referencję do pozostałych kontrolerów, zarządza obiektyami wyświetlonymi na scenie i kontroluje przebieg zadania. Dodatkowo zawiera dwustronną referencję do przycisków (Button), które wysyłają do niego powiadomienia o tym, że zostały naciśnięte. Kontroler zawiera też predefiniowaną tablicę relacji typów figur opisanych wcześniej.



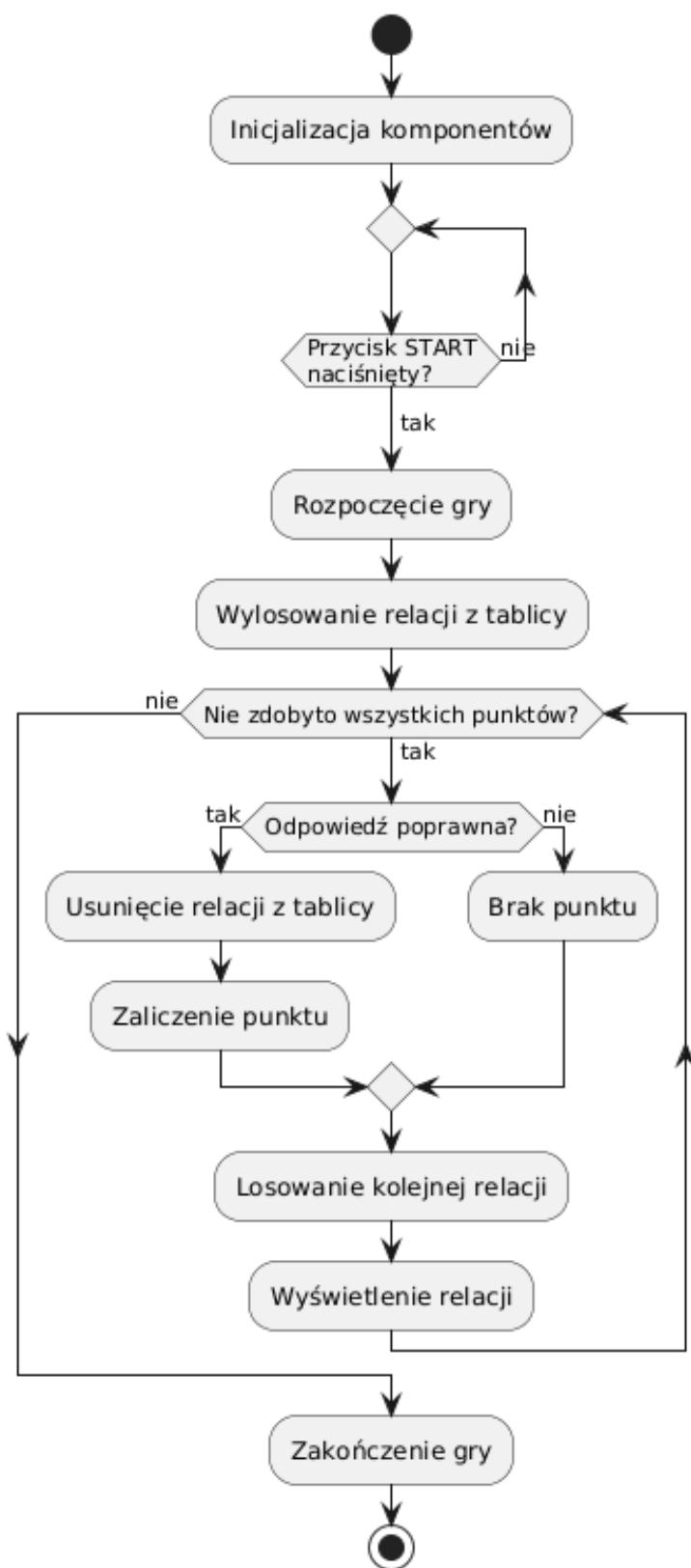
Rysunek 9.3: Diagram UML zawierający najważniejsze elementy kontrolerów dla zadania 2 – planimetria.

9.2.4. Przebieg zadania

Główny kontroler jest odpowiedzialny za inicjalizację zadania. Uruchamia on funkcję `InitBlueprint`, odpowiedzialną za wywołanie odpowiednich funkcji podległych elementów, takich jak: ustawienie tekstu, inicjalizacja ścian i przycisków.

Kontroler oczekuje na otrzymanie powiadomienia od przycisku odpowiedzialnego za uruchomienie gry. Kiedy otrzyma powiadomienie, przesiane mu za pośrednictwem odpowiedniej funkcji, uruchamia grę, tj. wywołuje funkcje podrzędnych komponentów, ustawiając im odpowiednią widoczność w świecie gry.

Z predefiniowanej tablicy relacji zostaje wybrana jedna, która zostaje wyświetlona graczowi. Główny kontroler czeka na powiadomienia od przycisków. Kiedy zostaje powiadomiony o tym, że jeden z nich został naciśnięty sprawdza poprawność odpowiedzi. Jeśli odpowiedź jest poprawna, usuwa relację z tablicy i losuje kolejną. Jeśli nie jest poprawna, losuje kolejną relację do wyświetlenia graczowi i nie zalicza punktu. Gra kończy się po poprawnym uzupełnieniu przez gracza wszystkich relacji.



Rysunek 9.4: Diagram przepływu dla zadania 2 – planimetria

9.3. Zadanie 3 – Nierówności (Konrad Czarnecki)

9.3.1. Problemy i różnice w realizacji zadania w praktyce

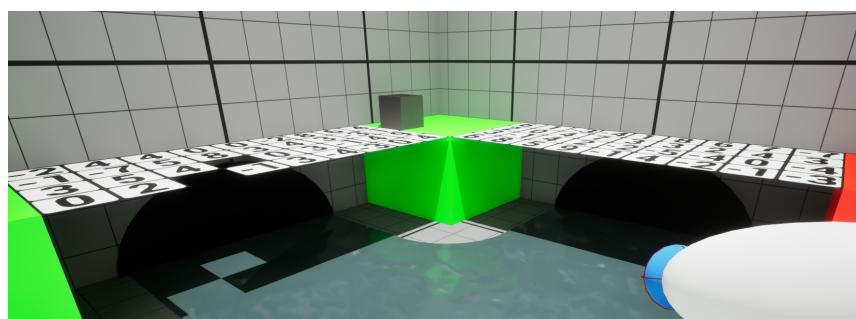
Podczas implementacji zadania, zgodnie z teorią opisaną w podrozdziale 8.3, ze względów estetycznych zmieniono postać, która miała być przeprowadzana przez most, na metaliczną kostkę, która zamiast przesuwania się po kolejnych kafelkach, obraca się wzdłuż swojej krawędzi, tym samym zmieniając swoją pozycję.

9.3.2. Główny kontroler

Zadaniem głównego kontrolera jest zarządzanie przebiegiem całego zadania. Porusza kostką w kierunkach wskazanych przez gracza, sprawdza poprawność ruchów, informuje o porażce lub zwycięstwie, zrzuca kafelki, które nie spełniają wylosowanej nierówności, jeśli gracz pokierował na nie kostkę, a także blokuje niedozwolone ruchy, takie jak opuszczenie mostu przez jego krawędź lub wejście w dziurę po kafelku, który wcześniej spadł – takie błędy świadcząby o przypadkowych, niezamierzonych akcjach, a nie o braku matematycznej wiedzy.

Kiedy gracz dotrze do punktu kontrolnego w połowie mostu, główny kontroler zapisuje tę informację i podświetla punkt kontrolny na zielono (rys. 9.5), aby w przypadku porażki w dalszej części mostu, kostka mogła zostać cofnięta do tego punktu, a nie do samego początku. Odblokowanie punktu kontrolnego uniemożliwia również cofnięcie się do pierwszej sekcji mostu.

Główny kontroler przechowuje także zmienną `real Position`, która określa aktualną pozycję kostki na moście względem kafelków, niezależnie od jej fizycznego położenia w świecie gry, które może być niewystarczająco precyzyjne. Zatem, gdy gracz przesunie kostkę o jeden kafelek w przód, wartość parametru y zmiennej `real Position` zostanie zwiększona o jeden. Cofanie zmniejszy wartość tego parametru o jeden, a ruchy w lewo i prawo odpowiednio zmienią wartość parametru x .



Rysunek 9.5: Punkt kontrolny w pokoju 3

9.3.3. Wybór kafelków i układ nierówności

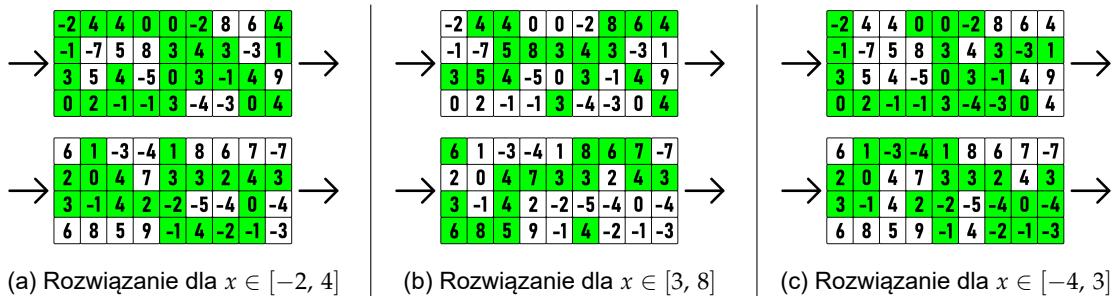
Gracz może przesuwać kostkę jedynie po kafelkach, które są oznaczone liczbami spełniającymi wybrany układ nierówności. W celu zwiększenia różnorodności rozgrywki, układ nierówności jest wybierany w sposób losowany ze zdefiniowanego wcześniej zbioru.

Lista układów nierówności oraz ich rozwiązań:

- $3x - 4 < 11; x + 1 > -2$; rozwiązania całkowite: $-2, -1, 0, 1, 2, 3, 4$
- $2x - 5 > -1; x - 3 < 6$; rozwiązania całkowite: $3, 4, 5, 6, 7, 8$

- $x + 7 > 2$; $-x + 1 > -3$; rozwiązania całkowite: $-4, -3, -2, -1, 0, 1, 2, 3$

W związku z faktem, że zbiory liczb spełniające układ nierówności różnią się w zależności od układu, który zostanie wylosowany, a liczby na kafelkach zawsze pozostają te same, należało odpowiednio dobrać kafelki, tak aby w każdym z przypadków istniała droga przez obie sekcje mostu (rys. 9.6).



Rysunek 9.6: Ścieżki przejścia przez obie sekcje mostu dla każdego z układów nierówności

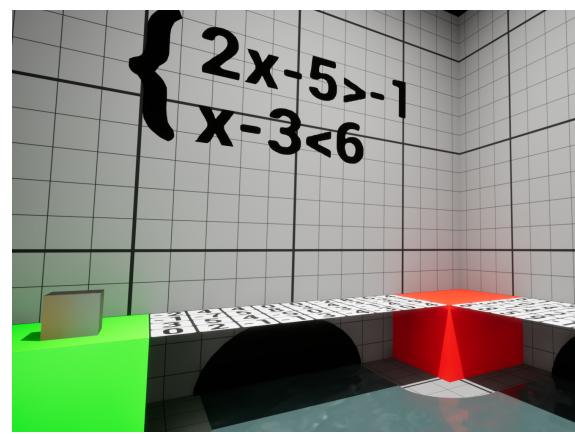
9.3.4. Przebieg zadania

Zadanie rozpoczyna się wraz z wywołaniem funkcji `Put On Tiles`, która ustawia kafelki z liczbami w odpowiednich pozycjach, budując z nich most, a także w losowy sposób wybiera układ nierówności, który gracz będzie musiał wykorzystać. Następnie główny kontroler wywołuje funkcję `Died`, która ustawia kostkę na początku mostu w pozycji wyjściowej (rys. 9.10). Most dzieli się na dwie sekcje oddzielone bezpiecznym punktem kontrolnym – jeśli gracz przegra w drugiej sekcji, kostka zostanie cofnięta do punktu kontrolnego, a nie do samego początku.

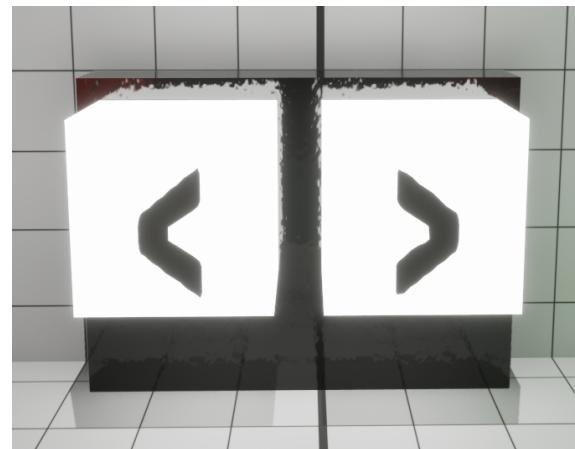
Po wstępnej konfiguracji rozpoczyna się właściwa część zagadki. Gracz dysponuje czterema przyciskami (rys. 9.8), odpowiadającymi czterem kierunkom (w przód, w tył, w lewo i w prawo), które po naciśnięciu wywołują funkcję `turn Side` z odpowiednim parametrem w głównym kontrolerze. Jego zadaniem jest przeprowadzenie kostki przez most, poruszając się jedynie po kafelkach, których liczby spełniają wylosowaną nierówność. Jeśli kostka stanie na kafelku, który nie spełnia nierówności, kafelek i kostka spadają w przepaść, a gracz musi zacząć od początku, lub od punktu kontrolnego, w zależności, w której sekcji mostu popełnił błąd.

Niemogliwe jest wypadnięcie z mostu poprzez wyjście poza jego krawędzie, lub wejście w dziurę po kafelku, który wcześniej spadł – gra uniemożliwia takie ruchy.

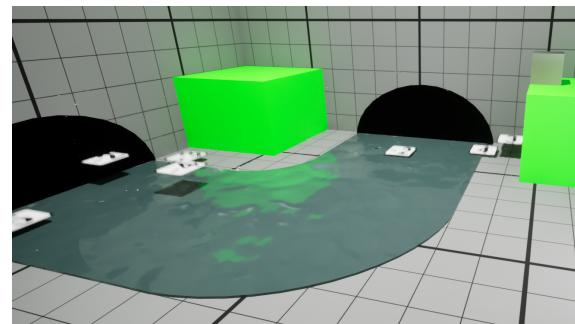
Po przejściu obu sekcji mostu, wszystkie pozostałe kafelki spadają do przepaści, a zagadka zostaje uznana za rozwiązana (rys. 9.9).



Rysunek 9.7: Most podzielony na dwie sekcje w pokoju 3



Rysunek 9.8: Przyciski do poruszania kostką w pokoju 3



Rysunek 9.9: Rozwiązywany pokój 3

9.4. Zadanie 4 – Funkcje (Konrad Czarnecki)

9.4.1. Problemy i różnice w realizacji zadania w praktyce

Jedyną istotną zmianą zadania względem opisu teoretycznego w podrozdziale 8.4 było rozdzielenie zagadki czwartej i piątej na dwa oddzielne pokoje. Pierwotnie oba zadania miały być realizowane w jednym pomieszczeniu, jednak aby udostępnić graczowi większą przestrzeń, zdecydowano się to zmienić.

9.4.2. Wybór dostępnych punktów

Zadanie składa się z trzech części. W każdej liczba punktów, które należy przeciąć wykresem funkcji zwiększa się o jeden, gracz otrzymuje dodatkowy suwak, pozwalający mu modyfikować następny współczynnik. Losowe punkty, które gracz ma za zadanie przeciąć, są wybierane z wcześniej zdefiniowanego zbioru. Zbiór ten został dobrany w taki sposób, aby współczynniki, które należy dobrać zawsze były liczbami całkowitymi, aby poziom trudności rósł wraz z kolejnymi częściami zadania, oraz by gracz musiał wykorzystywać wszystkie dostępne suwaki. Zatem części drugiej (z dwoma punktami do przecięcia i dwoma suwakami) nie da się rozwiązać, wykorzystując jedynie pierwszy suwak odpowiedzialny za współczynnik C. Analogicznie, część trzecia (z trzema punktami do przecięcia i trzema suwakami) wymaga wykorzystania wszystkich dostępnych suwaków.

Lista możliwych punktów do przecięcia w każdej części zadania:

1. Część pierwsza (jeden punkt):

- $P_1(4; 3)$; rozwiązanie: $f(x) = 3$
- $P_1(-3; 5)$; rozwiązanie: $f(x) = 5$
- $P_1(5; -1)$; rozwiązanie: $f(x) = -1$
- $P_1(2; -4)$; rozwiązanie: $f(x) = -4$
- $P_1(-3; 2)$; rozwiązanie: $f(x) = 2$

2. Część druga (dwa punkty):

- $P_1(-1; 2); P_2(0; 5)$; rozwiązanie: $f(x) = 3x + 5$
- $P_1(1; -2); P_2(3; 2)$; rozwiązanie: $f(x) = 2x - 4$
- $P_1(-2; 1); P_2(-3; 3)$; rozwiązanie: $f(x) = -2x - 3$
- $P_1(-1; 6); P_2(1; -2)$; rozwiązanie: $f(x) = -4x + 2$

3. Część trzecia (trzy punkty):

- $P_1(-1; 5); P_2(1; 5); P_3(0; 2)$; rozwiązanie: $f(x) = 3x^2 + 0x + 2$
- $P_1(-1; -2); P_2(0; -2); P_3(-2; 4)$; rozwiązanie: $f(x) = 3x^2 + 3x - 2$
- $P_1(0; 2); P_2(2; 0); P_3(-1; -3)$; rozwiązanie: $f(x) = -2x^2 + 3x + 2$
- $P_1(-1; 6); P_2(-2; 4); P_3(-3; -2)$; rozwiązanie: $f(x) = -2x^2 - 4x + 4$

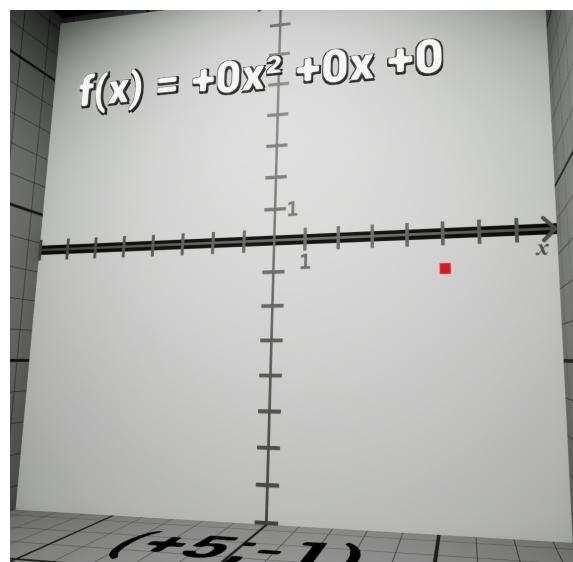
9.4.3. Przebieg zadania

Zadanie rozpoczyna się zainicjalizowaniem i wstępna konfiguracją kanwy, na której wyświetlane będzie układ współrzędnych. Następnie główny kontroler wywołuje funkcję Random Points, która losowo wybiera współrzędne punktu, który gracz będzie musiał przeciąć funkcją, z wcześniej

zdefiniowanego zbioru i wypisze je na podłodze pomieszczenia. Wzór funkcji $f(x) = Ax^2 + Bx + C$ (gdzie współczynniki A , B i C są domyślnie ustawione na 0) zostaje wypisany na ścianie, a jej wykres, jak i wybrany wcześniej punkt narysowany w kanwie (rys. 9.10).

W pierwszej części gracz ma do dyspozycji jeden suwak – odpowiadający za współczynnik C . Jego zadaniem jest przesunięcie wykresu funkcji w górę lub w dół tak, aby przeciął on wylosowany punkt. Po rozwiązaniu pierwszej części, główny kontroler ponownie wywołuje funkcję Random Points, tym razem losując współrzędne dwóch punktów, a gracz otrzymuje drugi suwak – odpowiadający za współczynnik B . Teraz jego zadaniem jest ustalenie odpowiednich wartości obu współczynników, tak aby wykres funkcji przeciął oba punkty jednocześnie. Ostatnia część jest analogiczna do poprzednich. Random Points losuje trzy punkty, a gracz dostaje do dyspozycji trzeci suwak – odpowiadający za współczynnik A (rys. 9.11) i musi dostosować wzór paraboli, przecinając wszystkie trzy punkty.

W momencie zakończenia wszystkich trzech części zadania, zagadka zostaje uznana za rozwiązana.



Rysunek 9.10: Zainicjowany układ współrzędnych wraz z jednym punktem, który gracz musi przeciąć wykresem funkcji w pokoju 4



Rysunek 9.11: Wszystkie suwaki dostępne w ostatnim etapie w pokoju 4

9.5. Zadanie 5 – Geometria analityczna (Konrad Czarnecki)

9.5.1. Problemy i różnice w realizacji zadania w praktyce

Tak samo jak w przypadku zagadki 9.4 jedyną istotną zmianą zadania względem opisu teoretycznego w podrozdziale 8.5 było rozdzielenie zagadki czwartej i piątej na dwa oddzielne pokoje.

9.5.2. Kontroler przycisków

Przyciski wyświetlane po obu stronach kanwy dysponują własnym kontrolerem, który powiadamia kontroler główny za każdym razem, gdy któryś z nich zostanie naciśnięty. Kontroler ten obsługuje też zmiany wprowadzane przez gracza, np. jeśli gracz naciśnie przycisk odpowiadający współczynnikowi x punktu przecięcia, równemu 3, a następnie zmieni zadanie i wciśnie przycisk 4 dla współczynnika x , przycisk 3 zostanie automatycznie odznaczony. Analogicznie działa to dla przycisków współczynnika y .

W celu zachowania przestronności pomieszczenia, zdecydowano się zrezygnować z przycisków o wartościach ujemnych i zamiast nich wprowadzić przycisk oznaczony symbolem \pm , który po naciśnięciu zmienia znaki wszystkich innych przycisków w danej grupie (x lub y) na przeciwnie.

9.5.3. Wybór dostępnych wzorów funkcji

Zadanie składa się z trzech części. W każdej części gracz musi wyliczyć punkt przecięcia wykresów funkcji i wcisnąć odpowiadające mu przyciski. W celu zwiększenia poziomu trudności, w pierwszej części obie funkcje są funkcjami liniowymi, w drugiej jedna z nich jest funkcją liniową, a druga kwadratowa, a w trzeciej obie funkcje są kwadratowe.

Losowe funkcje, których punkt przecięcia gracz ma za zadanie wyznaczyć, są wybierane z wcześniej zdefiniowanego zbioru. Zbiór ten został dobrany w taki sposób, aby funkcje zawsze przecinały się tylko w jednym miejscu (w przypadku funkcji kwadratowych drugi punkt przecięcia zawsze znajduje się w dużej odległości poza zakresem widoczności kanwy).

Lista możliwych wzorów przecinających się funkcji w każdej części zadania:

1. Część pierwsza (dwie funkcje liniowe):

- $f_1(x) = 2x - 3; f_2(x) = -x + 6$; rozwiązanie: $P(3; 3)$
- $f_1(x) = -2x - 8; f_2(x) = 2x + 4$; rozwiązanie: $P(-3; -2)$
- $f_1(x) = -9x - 6; f_2(x) = 2x + 5$; rozwiązanie: $P(-1; 3)$

2. Część druga (jedna funkcja liniowa, jedna kwadratowa):

- $f_1(x) = x^2 - 10x + 23; f_2(x) = 6x - 32$; rozwiązanie: $P(5; -2)$
- $f_1(x) = x^2 + 10x + 29; f_2(x) = -6x - 26$; rozwiązanie: $P(-5; 4)$
- $f_1(x) = x^2 - 6x + 14; f_2(x) = 8x - 19$; rozwiązanie: $P(3; 5)$

3. Część trzecia (dwie funkcje kwadratowe):

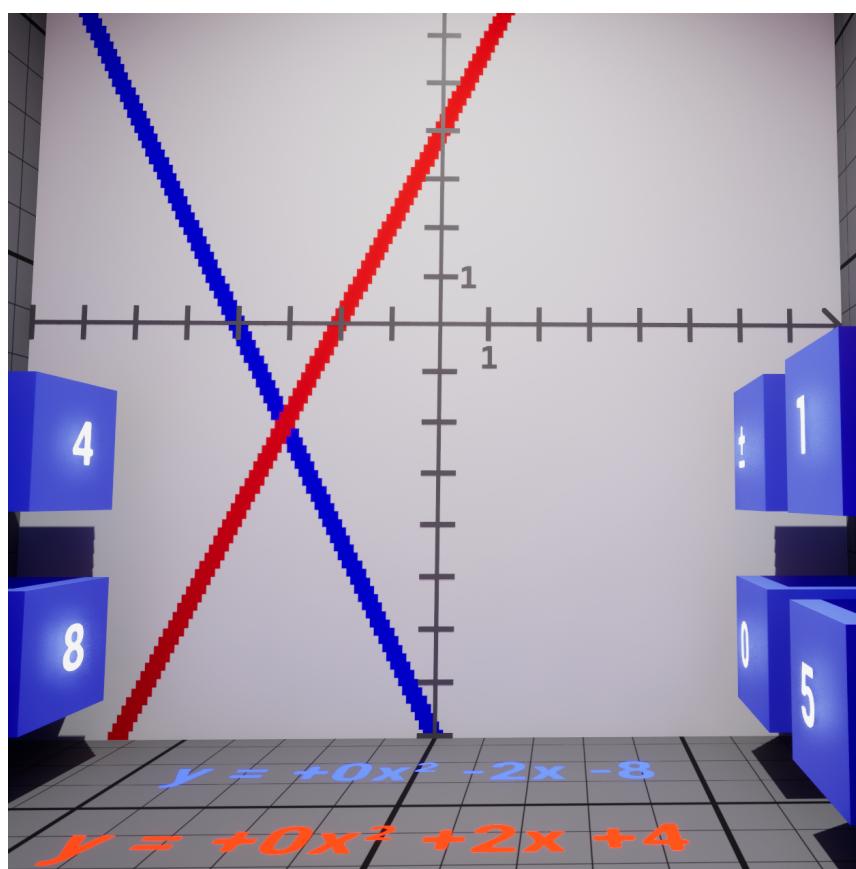
- $f_1(x) = 2x^2 - 31x + 103; f_2(x) = x^2 - 10x + 23$; rozwiązanie: $P(5; -2)$
- $f_1(x) = 2x^2 - 19x + 36; f_2(x) = x^2 - 4x + 10$; rozwiązanie: $P(2; 6)$
- $f_1(x) = 2x^2 + 23x + 46; f_2(x) = x^2 + 6x + 4$; rozwiązanie: $P(-3; -5)$

9.5.4. Przebieg zadania

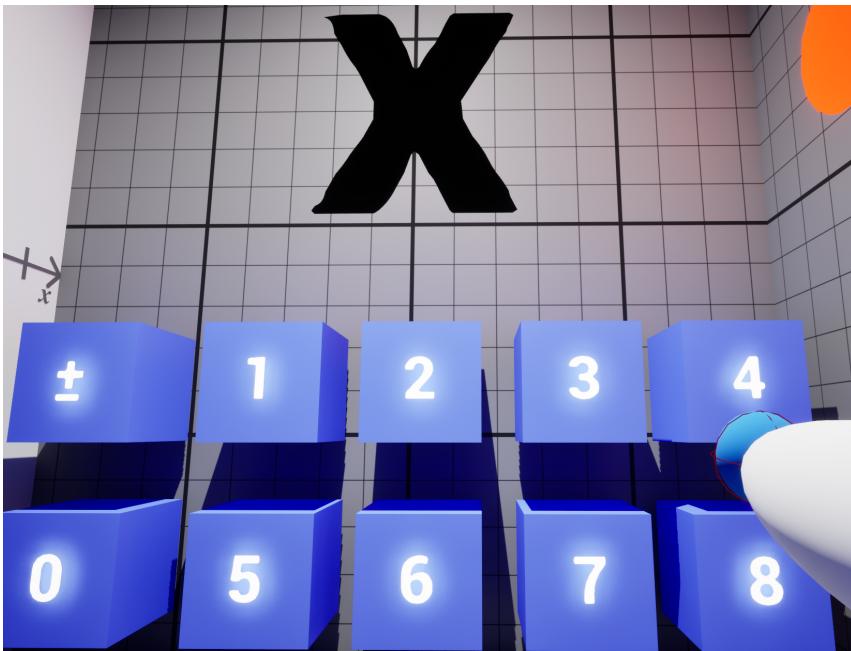
Podobnie jak w przypadku czwartej zagadki, zadanie rozpoczyna się zainicjalizowaniem i wstępna konfiguracją kanwy, na której wyświetlane będą układ współrzędnych oraz wzory funkcji. Następnie główny kontroler wywołuje funkcję `Random Points`, która losowo wybiera wzory dwóch przecinających się funkcji z wcześniej zdefiniowanego zbioru. Wzory funkcji zostają wypisane na podłodze, a ich wykresy narysowane w kanwie (rys. 9.12).

Gracz ma do dyspozycji rzędy przycisków po prawej i lewej stronie kanwy, których zadaniem jest odpowiednio modyfikacja współczynników x (rys. 9.13) i y . Za plecami gracza znajdują się trzy rozświetlone na czerwono lampy. Po rozwiązaniu każdej części zadania, jedna z lamp zmienia kolor na niebieski, a funkcja `Random Points` wybiera nowy zestaw funkcji.

W momencie zakończenia wszystkich trzech części zadania i rozświetlenia wszystkich lamp na niebiesko, zagadka zostaje uznana za rozwiązana.



Rysunek 9.12: Zainicjowany układ współrzędnych wraz z przecinającymi się wykresami funkcji w pokoju 5



Rysunek 9.13: Przyciski odpowiedzialne za współrzędną X w pokoju 5

9.6. Zadanie 6 – Kombinatoryka (Amdrii Demyshyn)

9.6.1. Różnice realizacji zadania w praktyce

W koncepcji teoretycznej sejf zawierał jedną, stałą zagadkę matematyczną. Takie rozwiązanie powodowało, że po jednorazowym rozwiązyaniu zadania element ten tracił swoją funkcję edukacyjną oraz rozgrywkową. W związku z tym zdecydowano się na rozbudowanie pierwotnej koncepcji poprzez wprowadzenie wielu zagadek kombinatorycznych obsługiwanych przez jeden, uniwersalny mechanizm sejfu.

9.6.2. Zaimplementowane zagadki

Zaimplementowano dwanaście zagadek kombinatorycznych, z których każda posiada przypisany indeks. Podczas uruchomienia gry indeks zagadki jest losowo generowany z przedziału od 0 do 11. Następnie treść wybranej zagadki wyświetlana jest za pomocą blueprintu aktora BP_RiddleSafe, w którym umieszczony jest widget typu WB_ZagadkaText. Poprawna odpowiedź ustawiana jest jako zmienna CurrentCode.

1. **Zagadka 1:** „PIN to liczba wszystkich możliwych 3-cyfrowych kodów PIN, w których żadna cyfra się nie powtarza.” **Odpowiedź:** 720
2. **Zagadka 2:** „PIN to liczba wszystkich możliwych 4-cyfrowych kodów PIN, w których żadna cyfra się nie powtarza.” **Odpowiedź:** 5040
3. **Zagadka 3:** „PIN to liczba wszystkich możliwych 4-cyfrowych kodów PIN bez powtórzeń, gdzie pierwsza cyfra nie może być zerem.” **Odpowiedź:** 4536
4. **Zagadka 4:** „PIN to liczba wszystkich 4-cyfrowych kodów PIN, w których dokładnie jedna cyfra występuje dwa razy, a pozostałe dwie cyfry są różne od niej i od siebie (np. 1213).”
Odpowiedź: 4320

5. **Zagadka 5:** „PIN to liczba wszystkich 4-cyfrowych kodów PIN, które składają się z dokładnie dwóch różnych cyfr (obie muszą wystąpić co najmniej raz).” **Odpowiedź:** 2100
6. **Zagadka 6:** „PIN to liczba wszystkich 4-cyfrowych kodów PIN będących palindromem (ABBA).”
Odpowiedź: 100
7. **Zagadka 7:** „PIN to liczba wszystkich 6-cyfrowych kodów PIN będących palindromem (ABC-CBA).” **Odpowiedź:** 1000
8. **Zagadka 8:** „PIN to liczba wszystkich 3-cyfrowych kodów PIN, w których dokładnie jedna cyfra występuje dwa razy.” **Odpowiedź:** 270
9. **Zagadka 9:** „PIN to liczba wszystkich 4-cyfrowych kodów PIN, w których dokładnie dwie cyfry są parzyste.” **Odpowiedź:** 3750
10. **Zagadka 10:** „PIN to liczba wszystkich 4-cyfrowych kodów PIN, w których suma cyfr wynosi dokładnie 10.” **Odpowiedź:** 286
11. **Zagadka 11:** „PIN to liczba wszystkich 4-cyfrowych kodów PIN, w których cyfry są ustawione w ścisłe rosnącym porządku.” **Odpowiedź:** 210
12. **Zagadka 12:** „PIN to liczba wszystkich 5-cyfrowych kodów PIN, w których cyfry są ustawione w ścisłe malejącym porządku.” **Odpowiedź:** 252

9.6.3. Sejf

Sejf został zrealizowany jako obiekt typu Blueprint Actor. Składa się on z komponentów StaticMesh tworzących obudowę sejfu oraz klawiaturę numeryczną, a także z wyświetlacza w postaci widgetu WBP_SafeDisplay, który prezentuje wprowadzane cyfry oraz komunikaty zwrotne.

Klawiatura sejfu zawiera:

- przyciski numeryczne od 0 do 9, których wcisnięcie powoduje wyświetlenie cyfry na ekranie,
- przycisk DEL, umożliwiający usunięcie ostatnio wprowadzonej cyfry,
- przycisk CHECK, służący do zatwierdzenia kodu.

Gracz wchodzi w interakcję z sejfem poprzez naciskanie przycisków. Aktualnie wprowadzany kod jest na bieżąco prezentowany na wyświetlaczu, co umożliwia kontrolę poprawności danych.

Każdy przycisk sejfu składa się z komponentu wizualnego typu StaticMesh oraz komponentu Box Collision, odpowiedzialnego za wykrywanie interakcji. Po wcisnięciu przycisku odtwarzana jest animacja oraz wywoływanie zdarzenia przekazujące informację o wcisniętym elemencie do głównej logiki sejfu.



Rysunek 9.14: Wygląd sejfu

9.6.4. Zakończenie zadania

Po wciśnięciu przycisku CHECK rozpoczyna się proces weryfikacji wprowadzonego kodu z wartością zmiennej CurrentCode. Jeżeli wartości są zgodne, sejf zostaje otwarty, co sygnalizuje poprawne rozwiązanie zagadki. W przypadku wprowadzenia błędnego kodu sejf pozostaje zamknięty, a na wyświetlaczu pojawia się komunikat „ERROR”. Gracz może ponowić próbę rozwiązywania zagadki.

9.6.5. Podsumowanie

Zastosowanie jednego, rozszerzalnego mechanizmu sejfu umożliwiło implementację wielu zadań kombinatorycznych w spójnym środowisku interaktywnym. Takie rozwiązanie znacząco zwiększa regrywalność gry oraz poprawia jej walory edukacyjne.

9.7. Zadanie 7 – Liczby rzeczywiste i działania na zbiorach liczbowych

(**Jan Walczak**)

9.7.1. Problemy i różnice w realizacji zadania w praktyce

Podczas implementacji zadania, zgodnie z teorią opisaną w podrozdziale 8.7, należało wprowadzić względem niej kilka poprawek. Pierwotny mechanizm zakładał, że zamek umieszczony na skrzyni będzie wyposażony w pola, w które uczeń wpisywałby odpowiednie cyfry – liczby elemen-

tów danego podzbioru. W praktyce okazało się to niemożliwe. Maksymalna liczba jednocyfrowa to 9, tym samym zadanie jest ograniczone do wyświetlania maksymalnie 9 liczb rzeczywistych w pokoju.

Zamysł został zmieniony i zamiast kłódki postanowiono umieścić elektroniczny zamek, wyposażony w klawiaturę. W ten sposób otrzymano możliwość umieszczenia więcej niż 9 liczb w pokoju. Uczeń staje również przed dodatkowym wyzwaniem – wywnioskować kolejność wpisywanych liczb zgodnie z podpowiedzią, umieszczoną na ścianie (patrz rysunek 9.15).



Rysunek 9.15: Zrzut ekranu – skrzynia z zamkiem oraz podpowiedź na ścianie dla zadania 7.

Zmienione zostały również liczby, które są rozmieszczone w pokoju. Aktualnie pojawiają się liczby: $15; \pi; 4,5; \sqrt{2}; \frac{1}{4}; -3; 1; -7; \frac{1}{3}$; 9 Tym samym kombinacja prezentuje się następująco (kolejność zgodna z podpowiedzią na rysunku 9.15):

- N (naturalne): $15; 1; 9$ – 3 liczby,
- C (całkowite): $15; 1; 9; -3; -7$ – 5 liczb,
- R (rzeczywiste): wszystkie liczby – 10 liczb,
- W (wymiernie): $15; 1; 9; -3; -7; 4,5; \frac{1}{4}; \frac{1}{3}$ – 8 liczb,
- NW (niewymiernie): $\pi; \sqrt{2}$ – 2 liczby.

Tym samym kombinacja do otwarcia sejfu to 351082.

Ostatnią rzeczą, która została zmieniona jest koncepcja drugiej części zadania. Mechanizm obracania bloczków został uznany za niepotrzebny, ponieważ wybrane relacje są jednoznaczne – są czytane zawsze od lewej do prawej strony. Niepoprawny obrót bloczka od razu sugeruje błędą odpowiedź.

Spośród wszystkich relacji zostały wybrane dwie:

- $N \subset Z \subset R$,
- $NW \cap W = \emptyset$,

Zostały one wybrane przez swoją charakterystykę. Pierwsza z nich pokazuje, jak kolejne zbiory zawierają się w sobie nawzajem, a druga pokazuje rozłączność zbiorów liczb wymiernych i niewymiernych. Gracz ma do dyspozycji 6 bloczków:

- dwa bloczki z symbolem \subset ,

- dwa bloczki z symbolem \emptyset ,
- dwa bloczki z symbolem \cap

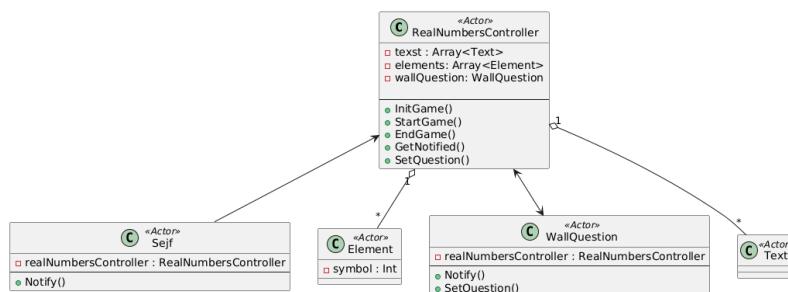
Bloczki mają mu służyć do uzupełnienia dwóch relacji wymienionych wyżej. Po ułożeniu relacji, poprawnie bądź nie, bloczki pozostają do dyspozycji gracza.

9.7.2. Implementacja zadania

Został zaimplementowany jeden, główny kontroler `RealNumbersController`. Zarządza on przebiegiem zadania, położeniem elementów na scenie oraz wyświetlaniem tekstu. Zawiera odpowiednie referencje do komponentów i odpowiada za poprawną inicjalizację.

Komponent `Element` jest bloczkiem z odpowiednim symbolem relacji, `Sejf` jest skrzynią, która przetrzymuje bloczki, a `WallQuestion` to płaska tablica (ścianka), na której wyświetlane są pytania w drugiej części zadania.

`Sejf` zawiera jednostronną referencję do głównego kontrolera. Kontroler może zostać przez niego powiadomiony, że gracz poprawnie wpisał kod. W ten sposób kontroler może sterować dalszym przebiegiem zadania. `WallQuestion` zawiera dwustronną referencję do głównego kontrolera. Mechanizm powiadomień działa tutaj podobnie ale kontroler może odpowiadać na otrzymywane powiadomienia, poprzez wywoływanie odpowiednich funkcji.



Rysunek 9.16: Diagram UML zawierający najważniejsze elementy dla zadania 7 – liczby rzeczywiste i działania na zbiorach liczbowych.

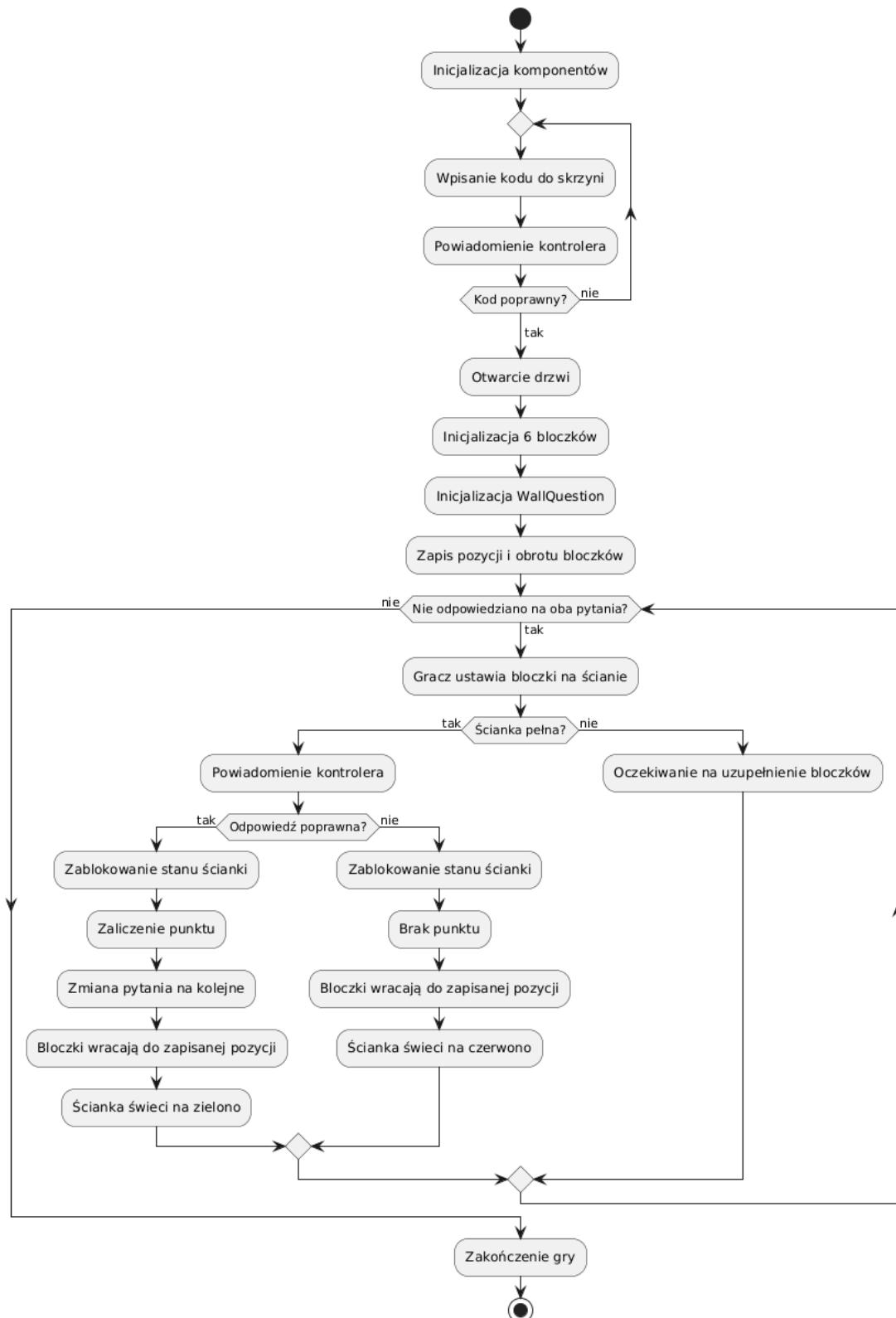
9.7.3. Przebieg zadania

Zadanie rozpoczyna się od pierwszej części – wpisania kodu do skrzyni. Kontroler inicjalizuje komponenty, a na ścianach i na skrzyni znajdują się wyświetlane liczby opisane w 9.7.1. Gracz powinien policzyć ile liczb, z jakiego zbioru, jest wyświetlanych w pokoju i poprawnie wpisać kod. W doborze kolejności wpisywanych liczb pomaga mu podpowiedź umieszczona na ścianie. Po poprawnym wpisaniu kodu, kontroler główny zostaje powiadomiony przez instancję skrzyni, że zadanie może przejść do następnego etapu. Drzwi się otwierają a kontroler inicjalizuje 6 bloczków, opisanych w 9.7.1 i ściankę z pytaniemi `WallQuestion`. Zapisuje pozycję oraz obrót bloczków w świecie gry, tak aby możliwy był powrót do stanu początkowego.

Ścianka wykrywa kolizję z bloczkami, które gracz na niej umieszcza. W momencie kiedy zostaje uzupełniona w całości, tj. bloczki wypełniają miejsca na odpowiedzi, powiadamia kontroler o rozwiązaniu zadania. Kontroler decyduje czy odpowiedź gracza jest poprawna. W przypadku gdy jest, to przesyła do `WallQuestion` powiadomienie o zmianie pytania na kolejne i zalicza punkt. Jeżeli odpowiedź nie jest poprawna, to również powiadamia o tym ściankę, ale nie zalicza punktu. W obu przypadkach bloczki wracają na wcześniej zisaną pozycję. W zależności od poprawności udzielonej odpowiedzi, ścianka rozświetla się kolorem zielonym lub czerwonym, tak żeby

gracz miał pewność, że udzielona odpowiedź jest poprawna. Podczas sprawdzania stanu ścianki, tj. ustawienie bloczków przez gracza pytanie zostaje zablokowane, a gracz nie może modyfikować swojej odpowiedzi (dokładać lub zabierać bloczków).

Kiedy gracz odpowie na oba pytania poprawnie, poziom zostaje uznany za zaliczony.



Rysunek 9.17: Diagram przepływu dla zadania 7 – liczby rzeczywiste i działania na zbiorach liczbowych.

9.7.4. Implementacja mechanizmu uzupełniania ścianki

Ścianka, czyli `WallQuestion`, wykrywa kolizję z bloczkami, czyli `Element`, w dwóch miejscach, w których będą one docelowo umieszczone. Kiedy gracz przytrzyma bloczek nad jednym z dwóch wskazanych miejsc, uruchamiana jest odpowiednia funkcja `StartCollision()`. W zależności od tego gdzie został umieszczony bloczek, sprawdzany jest aktualny stan ścianki – czy w danym miejscu gracz umieścił już bloczek. Jeśli tak, to ścianka nie pozwoli mu umieścić w tym miejscu kolejnego bloczka, dopóki nie wyjmie poprzedniego. Dzieje się tak za sprawą dwóch zmiennych wartości logicznych: `ActorHovered` i `ActorLockedIn`. Oznaczają one kolejno: stan, w którym gracz trzyma bloczek nad ścianką i stan, w którym gracz umieścił bloczek na ścianie. W przypadku gdy `ActorLockedIn` jest ustawione na wartość 0, to zapisywana jest referencja bloczku, który został właśnie umieszczony.

Dodatkowo informacja o tym, że gracz puścił bloczek, czyli zamierza go umieścić w danym miejscu, jest odpowiednio zapisywana w instancji bloczka. Kiedy bloczek znajduje się w ręce gracza otrzymuje sygnaturę `held`. Kiedy gracz go puści, sygnatura jest odbierana. Wyżej wymienione wartości logiczne są ustawiane dopiero wtedy, kiedy sygnatura najpierw istniała (w instancji bloczka), a potem została z niej usunięta.

W przypadku, gdy gracz chce usunąć bloczek ze ścianki, sytuacja jest analogiczna. Tym razem jednak, sprawdzamy czy bloczek, który zabiera gracz, jest tym znajdującym się na ścianie, poprzez porównanie referencji.

9.8. Zadanie 8 – Znaki funkcji trygonometrycznych (Konrad Czarnecki)

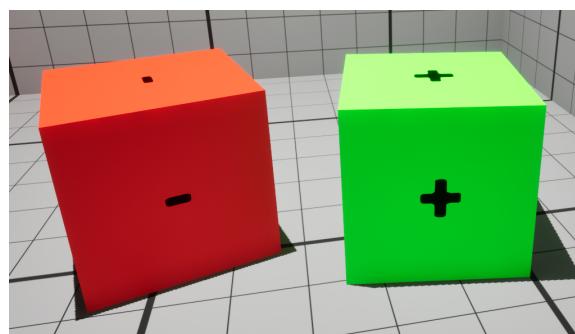
9.8.1. Problemy i różnice w realizacji zadania w praktyce

Podczas implementacji zadania zgodnie z opisem teoretycznym zawartym w podrozdziale 8.8 wprowadzono zmiany poprawiające estetykę i grywalność zagadki. Pierwotnie wszystkie cztery układy jednostkowe (oznaczone $\sin(x)$, $\cos(x)$, $\tg(x)$ i $\ctg(x)$) miały znajdować się na jednej ścianie, ale w celu uzyskania większej przejrzystości każdy z nich umieszczono na oddzielnej. Dodano ruchome sześciiany w dwóch wariantach: zielone – oznaczone symbolami „+” oraz czerwone – oznaczone symbolami „–” (rys. 9.18). Przyciski do zmiany znaku funkcji umieszczone w czterech ćwiartkach układów jednostkowych zostały zastąpione lukami (rys. 9.19), które gracz musi wypełnić wyżej wspomnianymi sześciianami. Taka zmiana zwiększa poziom interakcji gracza z zagadką, co bezpośrednio przekłada się na wyższy poziom zaangażowania oraz lepszą czytelność mechanik zadania.

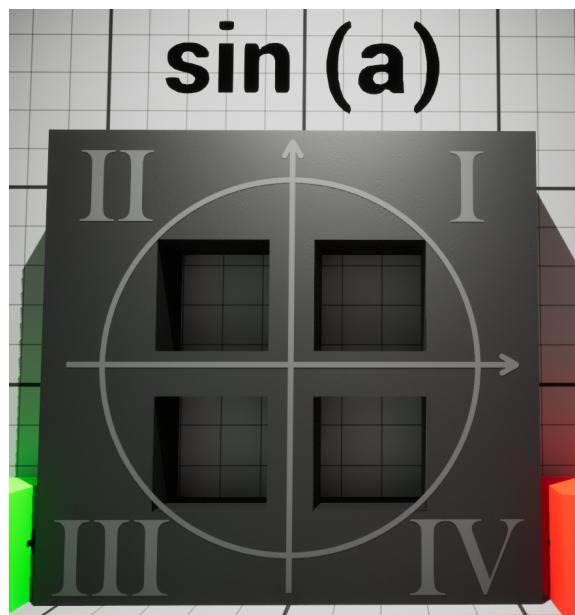
Ze względu na ograniczony rozmiar Jaskini tylko jeden układ jednostkowy jest widoczny w danym czasie. Po wypełnieniu jego ćwiartek sześciianami opatrzonymi odpowiednimi symbolami, układ jednostkowy chowa się, a zza innej ściany płynnie wysuwa się następny – ich kolejność jest losowa. W związku z tym liczba sześciianów z symbolami dostępnych w jednym czasie w pomieszczeniu mogła zostać ograniczona do czterech – dwa oznaczone „+” i dwa oznaczone „–”. Pozwala to rozwiązać każdy poszczególny układ jednostkowy (rys. 8.4).

9.8.2. Implementacja sześciianów

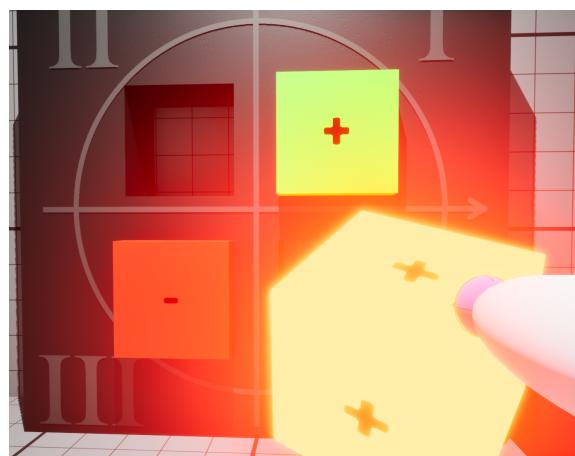
Sześciiany dysponują własnym kontrolerem, który powiadamia kontroler główny za każdym razem, gdy któryś sześciian zostaje umieszczony w luce. Gracz może również wyjąć wcześniej umieszczony sześciian z luki i włożyć go w inną. Ponadto kontroler zapobiega umieszczeniu dwóch



Rysunek 9.18: Ruchome sześciany w pokoju 8



Rysunek 9.19: Układ jednostkowy w pokoju 8



Rysunek 9.20: Sześciian po wykryciu luki w pokoju 8

sześciianów w tej samej luce, a także wyrzuceniu ich poza obszar gry. W takim wypadku sześcian automatycznie pojawia się na środku pokoju, zapewniając ciągłość zagadki.

Kiedy gracz zbliży się do luki w układzie jednostkowym, trzymając sześcian, kolor sześcianu zmieni się na jaskrawo żółty, informując gracza, że luka została wykryta (rys. 9.20). Jeśli w takim momencie gracz puści sześcian, uruchomi się płynna animacja, w której ten automatycznie obróci się do odpowiedniego kąta i umieści w luce.

9.8.3. Przebieg zadania

Zadanie rozpoczyna się wywołaniem funkcji `Random Order` przez główny kontroler. Funkcja ta w losowy sposób ustala kolejność czterech układów jednostkowych funkcji trygonometrycznych, które gracz będzie musiał rozwiązać. Następnie wykonuje się funkcja `Reset Boxes Position`, która umieszcza ruchome sześciany w rogach pomieszczenia, również w losowy sposób.

Po tej wstępnej konfiguracji rozpoczyna się właściwa część zadania. Pierwszy układ jednostkowy wyłania się zza ściany, a gracz podnosi sześciany i umieszcza je w odpowiednich lukach. Za każdym razem wywoływana jest funkcja `Box Inside` informująca główny kontroler, że sześcian został umieszczony wewnętrz układow i sprawdzającą, czy położenie wszystkich sześciianów jest poprawne. Kiedy to nastąpi układ jednostkowy chowa się, nazwa funkcji trygonometrycznej nad nim rozświetla się na zielono, dając graczu do zrozumienia, że element zagadki został rozwiązyany poprawnie, sześciany cofają się do swoich pozycji startowych funkcją `Reset Boxes Position`, a następnie pojawia się kolejny układ jednostkowy do rozwiązania.

Kiedy gracz wypełni wszystkie układy poprawnie, zagadka zostaje uznana za rozwiązana.

9.9. Zadanie 9 – Ciągi liczbowe (Jan Walczak)

9.9.1. Problemy i różnice w realizacji zadania w praktyce

W przeciwieństwie do zadań opisanych w podrozdziałach 9.2 i 9.7 w tym zadaniu nie pojawiło się dużo rozbieżności między teorią, a praktyczną implementacją.

Miejsce, w którym wprowadzono modyfikację, to warunki zakończenia gry. Zadanie, przez swój projekt, stawia bardziej na szybką analizę, niż na długie zastanawianie się nad odpowiedzią. Jest bezpośrednio inspirowane prawdziwą grą, w której największą wartością, wyróżniającą gracza, jest jego czas reakcji. Z tego powodu zdecydowano się na usunięcie limitu czasowego. Gra przyspiesza liniowo, razem z postępem, tj. liczbą zestrzelonych przez gracza kaczek.

Dodatkowo zwiększo liczbę żyć, tj. możliwości na pomyłkę gracza do 4 zamiast 3. Jest to spowodowane wprowadzeniem dodatkowej mechaniki gry – jeśli kaczka wyleci poza zasięg gracza (za ścianę), gracz traci życie. Gra kończy się jedynie w momencie utraty wszystkich życia – co musi nastąpić, ponieważ prędkość kaczek będzie rosła tak długo, jak toczy się rozgrywka.

9.9.2. Implementacja pistoletu laserowego

Implementacja pistoletu laserowego była kluczowym punktem, niezbędnym do stworzenia całego poziomu.

Aby poprawnie zaimplementować taki laser potrzebne są trzy wartości:

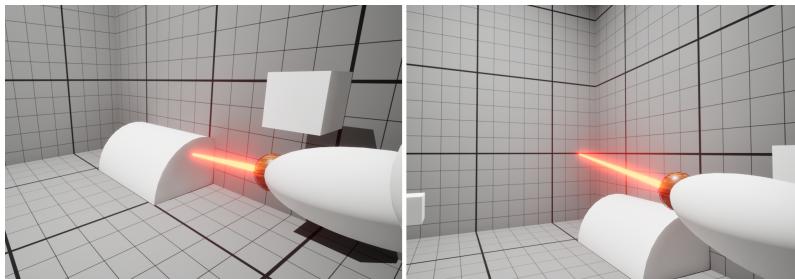
- punkt początkowy, od którego zaczynamy rysowanie lasera,
- punkt końcowy, do którego laser będzie rysowany,
- wektor kierunkowy, według którego laser będzie rysowany.

Za punkt początkowy został obrany koniec ręki gracza, tj. kontrolera, którym steruje. Z ustawienia kontrolera w świecie gry można również wyznaczyć wektor kierunkowy. Należy pobrać z kontrolera wartość tego wektora, przy użyciu systemowej funkcji Unreal Engine `Get Forward Vector`. Należy zapisać ten wektor, razem z punktem początkowym. Obliczenie punktu końcowego jest nieco bardziej skomplikowane. Polega na początkowym pomnożeniu wektora kierunkowego przez inny wektor, o dużych wartościach poszczególnych współrzędnych, a następnie dodaniu wyniku tego działania do współrzędnych punktu początkowego. W projekcie zastosowano wektor $(1000, 0; 1000, 0; 1000, 0)$. Tak obliczony punkt należało chwilowo zapisać jako punkt końcowy.

Następnie, przy użyciu systemowej funkcji `Line Trace by Channel`, należało wyznaczyć punkt, w którym linia, poprowadzona od punktu początkowego do końcowego, przecina się ze światem przedstawionym w grze. Funkcja ta zwraca wynik w postaci wartości typu struktury `Hit Result`, który można rozbić na składowe i pozyskać z niego `Impact Point`, czyli rzeczywisty punkt końcowy oraz `Distance`, czyli wartość zmienoprzecinkową, reprezentującą długość wyznaczonej linii.

Dzięki obliczonym wartościom można narysować laser, przebiegający od kontrolera gracza, do pierwszego napotkanego obiektu w grze. W tym celu kontroler ma na stałe przypisany obiekt `Static Mesh`, z siatką statyczną w kształcie cylindra, który jest rozciągany zgodnie z obliczonymi wartościami. Nałożony jest na niego materiał, emitujący czerwone światło. Na końcu rozciągniętej siatki statycznej umieszczona jest niewidzialna kula, złączona z systemem kolizji. Dzięki niej można wykrywać, w jaki komponent na scenie aktualnie celuje gracz.

Na rysunku 9.21 widać, że laser faktycznie wykrywa otoczenie, w które celuje gracz i zmienia odpowiednio swoją długość. Zmianie ulega również kierunek, w którym celuje gracz.



Rysunek 9.21: Porównanie rysowania lasera w przypadku kontaktu z przykładowym elementem otoczenia i ścianą.

9.9.3. Implementacja zadania

Zaimplementowany został jeden, główny kontroler `SequenceController`. Jego zadaniem jest inicjalizacja gry, zarządzanie przebiegiem gry, wyświetlanie tekstu i zarządzanieinstancjami kaczek. Kontroler wybiera pseudolosowo jeden z trzech dostępnych wzorów, na kolejny wyraz ciągu arytmetycznego, według którego będą obliczane kolejne elementy. Dostępne są wzory:

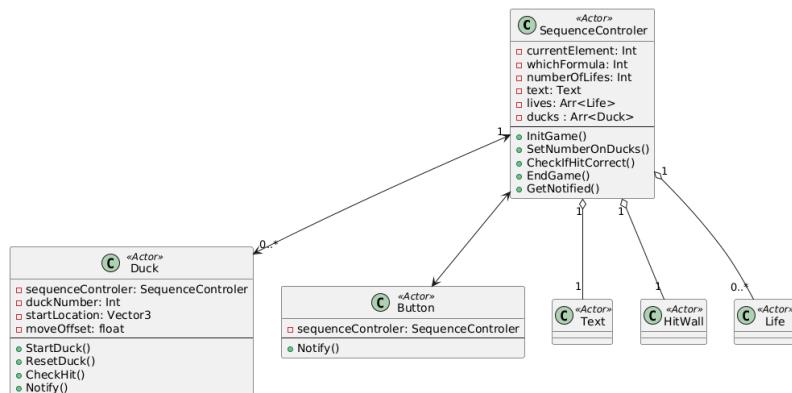
- $a_n = 3n + 4$
- $a_n = 2n + 3$
- $a_n = 5n + 7$

Komponent `Duck` implementuje funkcjonalności związane ze sterowaniem pojedynczej kaczki. Zawiera dwustronną referencję, do obiektu głównego kontrolera i przesyła powiadomienia, kiedy zostanie zestrzelony przez gracza, przy użyciu pistoletu laserowego. W instancji kaczki zapisy-

wana jest pozycja początkowa, numer, który aktualnie jest na niej wyświetlany i wartość zmienno-przecinkowa, czyli droga, którą pokona kaczka w każdej jednostce czasu w grze.

Gracz zostaje również zapoznany z zasadami działania gry. Kiedy jest gotowy do rozpoczęcia gry naciska przycisk, podobny do tego opisanego w podrozdziale 9.2. Przycisk działa na tej samej zasadzie i również posiada dwustronną referencję do instancji kontrolera.

W zagadce występują jeszcze dwa komponenty: `HitWall` i `Life`. Główny kontroler zawiera tablicę referencji do 4 instancji obiektów `Life`. Reprezentują ilość pozostałych życia gracza i nie pełnią w logice działania programu żadnej istotnej roli, poza tą wizualną. `HitWall` jest obiektem niewidocznym, znajdującym się poza zasięgiem gracza. Lecące kaczki będą wykrywać ten obiekt i sygnalizować to głównemu kontrolerowi zderzenie tak, jakby zostały trafione. Jeżeli kaczka uderzy w ten komponent, wyśle specjalny typ powiadomienia, mówiący o tym, że graczowi nie udało się jej zestrzelić. Jeśli kaczka zawierała poprawny wyraz ciągu to kontroler odczyta to jako błąd gracza i odejmie mu jedno życie. W takim przypadku, jeżeli gracz ma jeszcze pozostałe życie, wyświetli się kolejny wyraz ciągu, a kaczki zaczną lecieć od początku.



Rysunek 9.22: Diagram UML zawierający najważniejsze elementy dla zadania 9 – Ciągi liczbowe.

9.9.4. Przebieg zadania

Zadanie rozpoczyna się od inicjalizacji komponentów przez główny kontroler. Początkowo graczu zostaje wyświetlona instrukcja do zadania. Kiedy gracz będzie gotowy może rozpocząć grę naciskając przycisk „Start“. Kontroler zostaje w takiej sytuacji powiadomiony przez instancję przycisku i rozpoczyna grę. W sposób pseudolosowy zostaje wybrany jeden z trzech wzorów na kolejny n -ty wyraz ciągu arytmetycznego. Kontroler ustawia również poprawny wyraz ciągu, początkowo $n = 0$, na jednej z trzech kaczek. Pozostałe kaczki otrzymują wyrazy większe lub mniejsze od poprawnego wyrazu, również w sposób pseudolosowy. Dzieje się to wedle zasady, że wybrane wyrazy mogą być odległe jedynie o 2 jednostki na osi współrzędnych od poprawnego wyrazu ($a_n \pm 2$).

Kaczki pokonują stałą jednostkę odległości, w każdej jednostce czasu wykonywania się programu, wyrażoną za pomocą liczby zmienoprzecinkowej. Jednostka ta jest tym większa, im większy jest aktualny numer wyrazu ciągu arytmetycznego. Oznacza to, że będą przyspieszać razem z przebiegiem gry. Do tak obliczonej stałej wartości, dodawana jest pseudolosowa wartość zmienoprzecinkowa wybrana pseudolosowo z przedziału $[0, 5; 0, 7]$.

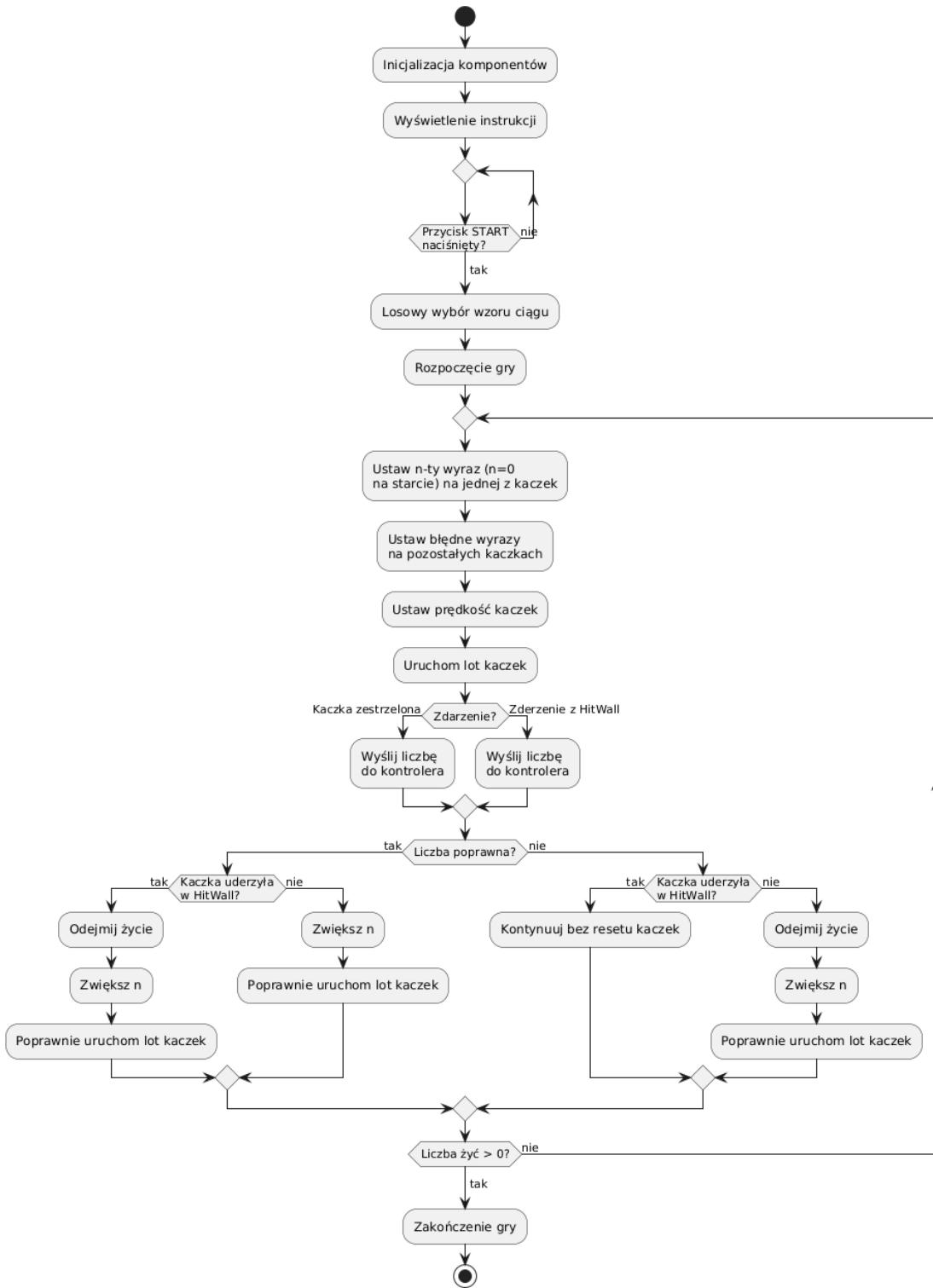
Kaczki poruszają się od lewej do prawej strony ekranu, aż do zderzenia się z instancją obiektu `HitWall`. Kaczki mogą zostać zestrzelone przez gracza, zanim zderzą się z wyżej wymienionym

obiektem. W obu przypadkach do kontrolera wysyłane jest powiadomienie odpowiedniego rodzaju: kaczka została zestrzelona lub kaczka zderzyła się z obiektem `HitWall`. W powiadomieniu przekazana jest również liczba całkowita, która znajdowała się na danej kaczce.

Kontroler po otrzymaniu w powiadomieniu liczby oraz informacji, czy dana kaczka została zestrzelona, podejmuje decyzję:

1. Jeśli liczba jest poprawnym wyrazem ciągu – zlicza punkt i ustawia ponownie kaczki. Zwiększa też numer wyrazu ciągu.
2. Jeśli liczba nie jest poprawna – odejmuje jedno życie, ale nie ustawia ponownie kaczek.
3. Jeśli kaczka, przechowująca poprawny wyraz ciągu, uderzy w obiekt `HitWall` (wyloci poza zasięg gracza) – odejmuje jedno życie i ponownie ustawia kaczki. Zwiększa też numer wyrazu ciągu.

Ponowne ustawienie kaczeek polega na obliczeniu kolejnego wyrazu ciągu arytmetycznego, ustawieniu go na jednej z nich, ustawieniu dwóm pozostałym kaczkom błędного wyrazu oraz na cofnięciu ich do stanu początkowego. Gra kończy się w momencie, gdy gracz straci wszystkie 4 życia.



Rysunek 9.23: Diagram przepływu dla zadania 9 – Ciągi liczbowe

9.10. Zadanie 10 – Prawdopodobieństwo (Andrii Demyshyn)

9.10.1. Praktyczna realizacja zadania

Praktyczna realizacja zadania nie różni się zasadniczo od tego, co zostało zaplanowane w teorii, niewielkie zmiany zostały opisane poniżej.

Design pokoju

Modele trójwymiarowe wykorzystane w projekcie zostały pobrane z platformy Sketchfab¹ na licencji Creative Commons i dostosowane do wymagań silnika Unreal Engine.

Początkowo planowano, że zagadka zostanie zrealizowana w stylu filmu „Matrix”. W tym celu pobrano i dodano do projektu fotel i stół w stylu filmu. Niestety nie udało się znaleźć postaci Morfeusza w bezpłatnym dostępie, którą można by dodać do projektu, dlatego postanowiono znaleźć i pobrać podobną postać, którą następnie można było przerobić na styl filmu. Udało się znaleźć i pobrać taką postać, a następnie przerobiono jej tekstury, aby styl odpowiadał postaci Morfeusza z filmu.



Rysunek 9.24: Rysunek przedstawia postać przed i po modyfikacji.

Postanowiono również stylizować pokój na styl „Matrixa”. W tym celu stworzono teksturę materiału z neonowo-zielonymi symbolami na czarnym tle, podobnymi do tych, które widzieliśmy w filmie. Następnie teksturę dodano do stworzonego materiału, który został użyty na obiektach ścian. Aby zwiększyć immersję gracza, dodano ruch tekstury na obiektach, dzięki czemu wygląda to nie jak ściany, ale jak dane poruszające się w ciemności.

Obiekty sceny

Na scenie na początku gry, oprócz stołu, krzesła oraz postaci Morfeusza, gracz widzi również tekst zagadki wyświetlany nad postacią, dwa pojemniki oraz zestaw 100 tabletek.

Treść zagadki przedstawiona graczowi ma następującą formę:

¹<https://sketchfab.com>

Morpheus: Agenci Matrixa mnie schwytali. Przede mną leży 50 czerwonych i 50 niebieskich pigułek — czerwona mnie ocali, a niebieska pozostawi mnie na zawsze w Matriksie. Mogę rozłożyć ją w dowolny sposób do dwóch identycznych pudełek. Agent sam wybierze jedno pudełko i na ślepo wyciągnie z niego jedną pigułkę. Jak powinieneś je rozłożyć, aby maksymalnie zwiększyć szansę na wyciągnięcie czerwonej?

Aktor Morfeusza – BP_Morpheus Pobrała postać Morfeusza została umieszczona w obiekcie typu Blueprint Actor BP_Morpheus. Do obiektu postaci dodano widżet typu Blueprint Widget WBP_MorpheusText. Na tym widżecie wyświetlana jest informacja dla gracza na każdym etapie gry. Aby uzyskać interaktywność, dodano animacje. Animacje postaci zostały pobrane z platformy Mixamo². Pobrały i dodano trzy animacje: Sitting_Idle_Anim, Sit_To_Stand oraz Salute. Domyslnie wykorzystywana jest animacja Sitting_Idle_Anim.

Tabletki – BP_Pil Na stole gracz widzi przed sobą 100 tabletów w kolorze czerwonym i niebieskim. Tabletki są realizowane jako obiekty typu Blueprint Actor BP_Pil. Tabletki mają komponent StaticMesh w kształcie kapsułki, komponent kolizji BoxCollision, materiał w kolorze czerwonym lub niebieskim, identyfikator ID oraz, w zależności od potrzeb, tagi BP_Red_Pil lub BP_Blue_Pil.

Spawner – BP_Pil_Spawner Dla wygody na scenie umieszczono aktora typu Blueprint Actor BP_Pil_Spawner. Przy starcie gry obiekt ten tworzy na stole 50 czerwonych i 50 niebieskich tabletów z identyfikatorem „Table”.

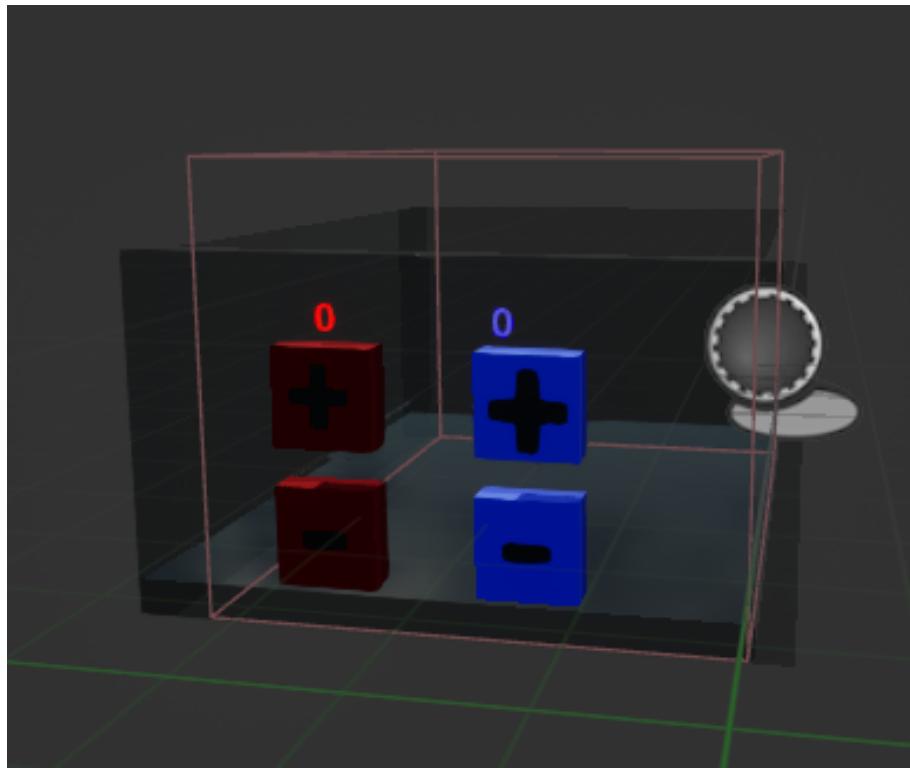
Kontenery – BP_PillContainer Kontenery zostały zaimplementowane jako obiekt typu Blueprint Actor BP_PillContainer. Ściany kontenera zostały zaimplementowane przez komponenty typu StaticMesh ze szklanym materiałem, dzięki czemu gracz może zobaczyć wszystko, co znajduje się i dzieje się wewnątrz kontenera.

Początkowo zakładano, że gracz będzie ręcznie przenosił tabletki ze stołu do kontenera, co zostało zaimplementowane, jednak ręczne przenoszenie 100 tabletów nie jest zbyt wygodne. W związku z tym dodano do kontenerów przyciski, które po naciśnięciu przenoszą tabletki do kontenera. Do kontenera dodano cztery przyciski: + czerwony, - czerwony, + niebieski oraz - niebieski. Każdy przycisk został utworzony z komponentu StaticMesh, komponentu kolizji BoxCollision oraz komponentu tekstowego TextRender.

Kiedy gracz naciśnie przycisk „+”, przycisk zaczyna teleportować tabletki odpowiedniego koloru do momentu, dopóki gracz nie zwolni przycisku. Po teleportacji pojemnik przypisuje tabletce swój identyfikator. Aby uniknąć konfliktów, teleportacja jest tak skonstruowana, że najpierw sprawdza i teleportuje tabletki ze stołu, a następnie, gdy się skończą, teleportuje tabletki z innego kontenera. W przypadku przycisku „-” kontener przenosi tabletki z siebie na stół i nadaje im identyfikator stołu.

Nad przyciskami w pojemniku znajduje się również widżet, który pokazuje, ile tabletka każdego koloru znajduje się w pojemniku.

²<https://www.mixamo.com>



Rysunek 9.25: Rysunek przedstawia wygląd kontenera

Przycisk sprawdzania – BP_CheckButton Na scenie, na piedestale, znajduje się przycisk sprawdzania etapów zadania, zaimplementowany jako obiekt typu Blueprint Actor BP_CheckButton. Obiekt składa się z komponentów StaticMesh w formie sfery oraz komponentu kolizji typu BoxCollision.

Po wcisnięciu przycisku w pierwszej kolejności sprawdzane jest, czy wszystkie 100 tabletki zostały umieszczone w obydwu kontenerach. Jeżeli nie wszystkie tabletki zostały umieszczone w kontenerach, gracz otrzymuje przez widżet komunikat: „Morpheus: Umieść wszystkie pigułki w pojemnikach.”. W przeciwnym przypadku sprawdzane jest, czy którykolwiek z pojemników jest pusty. Żaden pojemnik nie może być pusty. Jeżeli jeden z nich jest pusty, gracz otrzymuje komunikat: „Morpheus: Żaden pojemnik nie może być pusty. Umieść tabletki w obu pojemnikach.”.

Jeżeli powyższe warunki zostały spełnione, obliczane jest prawdopodobieństwo wylosowania czerwonej pigułki według wzoru:

$$P = \left(\frac{R_A}{N_A} + \frac{R_B}{N_B} \right) \cdot \frac{1}{2} \cdot 100\%$$

Jeżeli prawdopodobieństwo wylosowania czerwonej pigułki wynosi mniej niż 74,74%, gracz otrzymuje komunikat: „Morpheus: Aktualne prawdopodobieństwo czerwonej pigułki: {P}%. Możesz zrobić to lepiej. Spróbuj ponownie.”.

Jeżeli prawdopodobieństwo wynosi powyżej 74,74%, gracz otrzymuje komunikat końcowy, zadanie zostaje zaliczone, uruchamiane są animacje Sit_To_Stand oraz Salute, po czym postać Morfeusza znika ze sceny.

9.11. Zadanie 11 – Optymalizacja i rachunek różniczkowy (Autor)

9.12. Zadanie 12 – Układy równań (Andrii Demyshyn)

9.12.1. Różnice realizacji zadania w praktyce

W tym zadaniu zrealizowano zagadkę matematyczną opartą na osiągnięciu zadanej wartości energetycznej niezbędnej do zapalenia żarówki. Zadaniem gracza jest umieszczenie odpowiedniej liczby kulek każdego koloru w pojemnikach w taki sposób, aby łączna wartość energetyczna kulek była równa wartości wskazanej nad żarówką.

W trakcie opracowywania zadania zdecydowano się na zmianę logiki zagadki. Początkowo planowano implementację rozwiązania posiadającego tylko jedno poprawne rozwiązanie, co nadałoby grze charakter jednorazowy. We wcześniejszej koncepcji docelowa wartość energii była stała, a gracz musiał umieścić dokładnie 12 kulek w kontenerach. W toku realizacji podjęto decyzję o zwiększeniu różnorodności rozgrywki poprzez losowanie wartości docelowej z przedziału od 18 do 51. W obecnej wersji zagadki gracz sam decyduje o liczbie umieszczanych kulek, pod warunkiem że ich łączna wartość energetyczna odpowiada liczbie wyświetlanej nad żarówką. Dzięki tym zmianom zagadka stała się bardziej zróżnicowana i wielokrotnego użytku.

Po wejściu do pokoju gracz widzi lampa, trzy pojemniki na kulki, zestaw kolorowych kulek oraz panel informacyjny z instrukcją zadania. Nad lampą wyświetlana jest docelowa wartość energii, którą należy osiągnąć.

9.12.2. Opis obiektów sceny

Panel informacyjny BP_RiddleText

W pokoju umieszczono panel informacyjny w postaci blueprintu aktora **BP_RiddleText**, wewnątrz którego znajduje się blueprintowy widget **ZagadkaLamp**. W widżecie wyświetlana jest instrukcja zadania:

„Aby uruchomić oświetlenie, należy włożyć do pojemników taką liczbę kulek, aby łącznie wygenerowana moc była dokładnie równa wartości wyświetlonej nad lampą. Jeśli moc będzie zbyt niska lub zbyt wysoka — instalacja nie zadziała. Pojemniki przyjmują tylko kulki w swoim kolorze.

Parametry energetyczne kulek: Czerwona kula ma o 3 jednostki energii więcej niż zielona. Niebieska kula ma o 1 jednostkę energii więcej niż czerwona. Trzy zielone kulki mają o 2 jednostki energii mniej niż jedna niebieska.”

Kulki

W scenie zaprojektowano trzy typy kulek: czerwone, zielone oraz niebieskie. Zostały one zrealizowane za pomocą trzech blueprintów aktorów: BR, BB oraz BG, gdzie pierwsza litera jest skrótem od słowa „Ball”, a druga od pierwszej litery koloru.

Każdy obiekt kulki składa się z:

- komponentu **StaticMesh** w postaci sfery,
- komponentu **Box Collision**,
- tagu **Ball**,

- tagu odpowiadającego kolorowi kulki (Green, Blue, Red).

Kulki są obiektami fizycznymi, które gracz może podnosić, przenosić oraz umieszczać w kontenerach.

Kontenery

W scenie zaimplementowano trzy kontenery w postaci obiektów typu Blueprint Actor: BP_Container_Red, BP_Container_Blue oraz BP_Container_Green.

Ściany każdego kontenera składają się z czterech komponentów typu StaticMesh z materiałem szklanym, dna wykonanego z komponentu StaticMesh w odpowiednim kolorze oraz komponentu Box Collision.

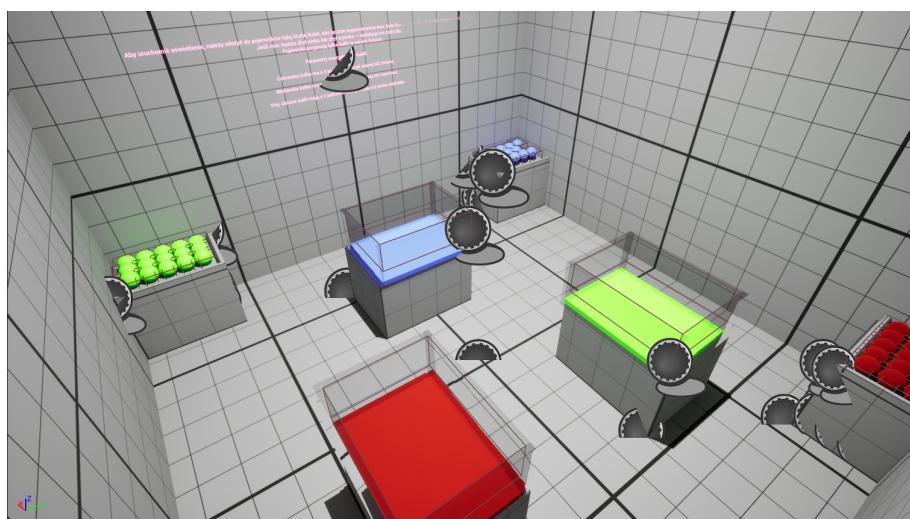
Kontenery przyjmują oraz przekazują do żarówki wartości energetyczne kulek odpowiadających im koloru. Mechanizm ten został zrealizowany w taki sposób, że w momencie wejścia obiektu w obszar Box Collision kontener sprawdza, czy obiekt posiada tag Ball oraz tag odpowiadający kolorowi kontenera. Jeżeli warunki są spełnione, kontener dodaje wartość energetyczną kulki do swojej łącznej wartości, która domyślnie wynosi 0. W przeciwnym przypadku nie jest wykonywana żadna akcja.

Jeżeli kulka opuszcza obszar kontenera, jej wartość energetyczna zostaje odjęta od sumy. Aktualne wartości energetyczne kontenerów są na bieżąco przekazywane do obiektu BP_Lamp.

Żarówka

Żarówka została zrealizowana jako blueprint aktora BP_Lamp. Obiekt składa się z komponentów:

- StaticMesh,
- TextRender,
- PointLight.



Rysunek 9.26: Wygląd pokoju z kulami

9.12.3. Zakończenie zadania

Komponent TextRender wyświetla wartość energii, którą gracz musi osiągnąć. Wartość ta jest generowana losowo przy każdym uruchomieniu gry z przedziału od 18 do 51.

W obiekcie `BP_Lamp` zaimplementowano funkcję `ApplyTotal`, która porównuje sumę wartości energetycznych kulek przekazywanych przez kontenery z wygenerowaną wartością docelową. W przypadku zgodności wartości żarówka zmienia kolor, zapala się, a zadanie zostaje zaliczone. W przeciwnym przypadku lampa pozostaje wyłączona.

9.13. Zadanie 13 – Stereometria (Jan Walczak)

9.13.1. Problemy i różnice w realizacji zadania w praktyce

Zadanie nie odbiega za bardzo od wcześniej opracowanego opisu teoretycznego. Zmieniono ilość wyświetlanego brył geometrycznych. Według opisu graczu miałaby być przedstawiana tylko jedna bryła. W praktyce takie zadanie byłoby bardzo krótkie. Zdefiniowano więc pulę złożoną z 5 brył:

1. kuli,
2. stożka,
3. walca,
4. sześciianu,
5. ostrosłupa prawidłowego czworokątnego.

Do każdej z wymienionych brył wybrano 3 pytania. Dotyczą one kolejno:

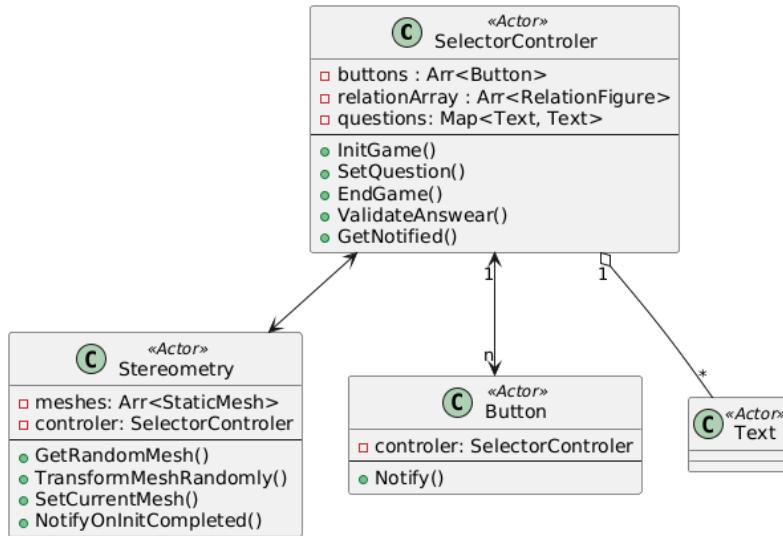
- rodzaju bryły,
- wzoru na obliczenie powierzchni bryły,
- wzoru na obliczenie objętości bryły.

Gracz musi odpowiedzieć na wszystkie 3 pytania prawidłowo. Jeśli się pomyli, że zbioru wybierana jest inna bryła. Inna niż ta, o którą pytano go przed chwilą. W ten sposób uniknięto sytuacji, w której gracz odpowiadałby losowo i przechodził zadanie metodą eliminacji.

9.13.2. Implementacja zadania

W zadaniu został zaimplementowany jeden główny kontroler `SelectorController`. Jego zadaniem jest inicjalizacja gry, zarządzanie przebiegiem gry, wyświetlanie tekstu i wybór odpowiednich pytań, wyświetlanego graczu. Zawiera on dwustronną referencję do obiektu `Stereometry` oraz dwustronną referencję do trzech obiektów `Button` (takich, jak te opisane w podrozdziałach 9.2 i 9.13). Zawiera także predefiniowany zbiór pytań, dotyczących każdej z brył, reprezentowany przez mapę typu `pytanie:poprawna odpowiedź`.

Obiekt `Stereometry` odpowiada za poprawne wyświetlanie brył. Zawiera 5 obiektów typu siatki statycznej, które wyświetla w świecie gry, wybierając bryłę odpowiadającą aktualnie zadawanemu pytaniu. Jest też odpowiedzialny za pseudolosowy mechanizm wyboru brył. Przed wyświetleniem obiekt `Stereometry` w sposób pseudolosowy obraca bryłę, zmienia jej rozmiar, oraz położenie w świecie gry. W tym przypadku nie było potrzeby implementowania mechanizmu powiadomień, ponieważ rodzaj komunikatów wysyłanych pomiędzy obiektami `Stereometry` i `SelectorController` jest prosty i zawsze jednakowy (żądanie wyświetlenia nowej bryły). Dodatkowo, obiekt `Stereometry` zawiera referencję do kontrolera tylko dlatego, że jego inicjalizacja jest dłuża (prawdopodobnie przez wczytywanie siatek statycznych brył, z nałożoną teksturą odbijającą światło). Musi więc powiadomić kontroler główny, że zakończył swoją inicjalizację, aby ten mógł zacząć wykonywać na nim operacje, przez wywołanie odpowiednich funkcji.

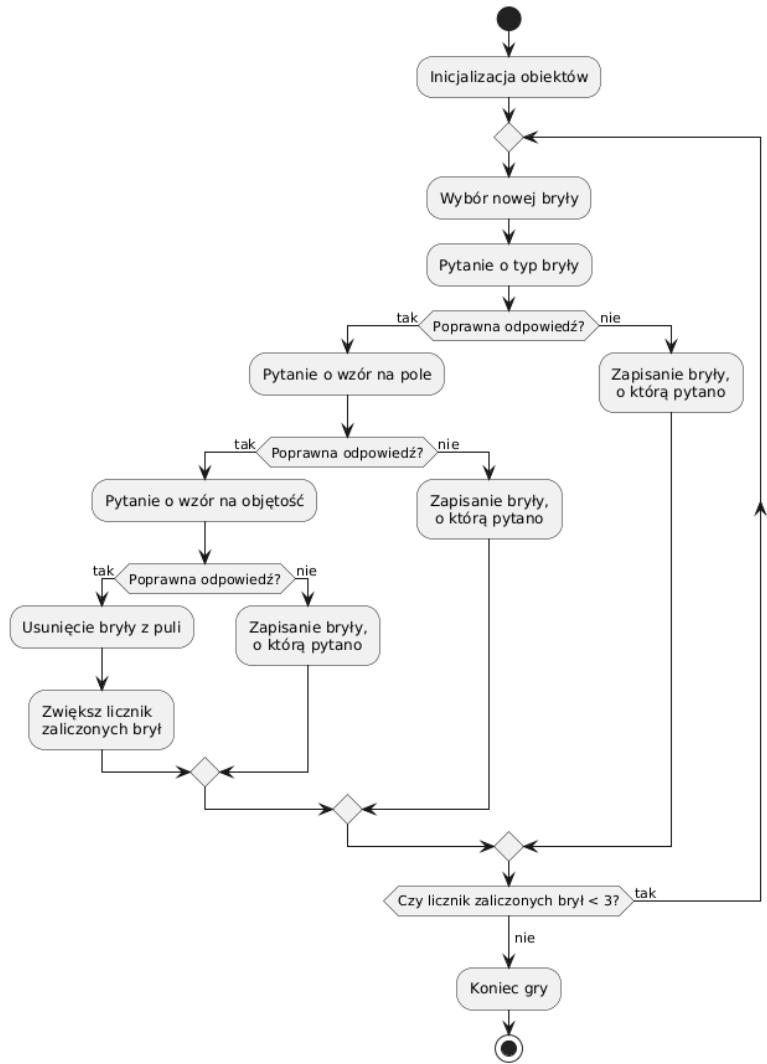


Rysunek 9.27: Diagram UML zawierający najważniejsze elementy kontrolerów dla zadania 13 – stereometria.

9.13.3. Przebieg zadania

Kontroler inicjalizuje podrzędne mu obiekty i rozpoczyna grę. W sposób pseudolosowy wybierana jest jedna z 5 predefiniowanych brył, poprzez wywołanie odpowiedniej funkcji obiektu Stereometry. Gracz ma do wyboru 3 przyciski, które zawierają odpowiedzi A, B lub C. Pytania są wyświetlane sekwencyjnie – najpierw zadawane jest pytanie o typ bryły, potem o wzór na obliczenie jej pola powierzchni, a na końcu o wzór na obliczanie jej objętości.

Jeśli gracz pomyli się podczas udzielania odpowiedzi, kontroler ponownie wywoła funkcję odpowiedzialną za wybór kolejnej bryły. W takim przypadku może się ona pojawić ponownie w zadaniu, ale nie od razu po udzieleniu błędnej odpowiedzi. Jeśli gracz odpowie na wszystkie 3 pytania prawidłowo, bryła zostaje uznana za zaliczoną i jest usuwana z puli dostępnych do wylosowania brył. Gra kończy się kiedy gracz poprawnie odpowie na pytania dotyczące 3 różnych brył.



Rysunek 9.28: Diagram przepływu dla zadania 13 – stereometria

10. PODSUMOWANIE

10.1. *Ocena realizacji celów*

10.2. *Propozycje rozwoju projektu*

10.3. *Wnioski końcowe*

WYKAZ LITERATURY

- [1] A. Koszlajda, *Zarządzanie projektami IT. Przewodnik po metodach*. Helion, 2010.
- [2] I. GitHub. "About Git", (**urldate** 2026-01-02). URL: <https://docs.github.com/en/get-started/using-github>
- [3] I. GitHub. "About issues", (**urldate** 2026-01-02). URL: <https://docs.github.com/en/issues/tracking-your-work-with-issues/learning-about-issues/about-issues>
- [4] G. community. "Branches in a Nutshell", (**urldate** 2026-01-02). URL: <https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell>
- [5] I. GitHub. "About rulesets", (**urldate** 2026-01-02). URL: <https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-rulesets/about-rulesets>
- [6] I. GitHub. "Pull requests", (**urldate** 2026-01-02). URL: <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests>

SPIS RYSUNKÓW

6.1 Zrzut ekranu z platformy GitHub – tablica z przykładowymi Issues do projektu.	22
6.2 Zrzut ekranu z platformy GitHub – tablica z przykładowym Pull Request	23
8.1 Przykładowy wygląd pierwszego zadania – wzory skróconego mnożenia	25
8.2 Przykładowy wygląd tablicy z przecinającymi się wykresami	29
8.3 Przykładowy wygląd sejfu występującego w zadaniu	30
8.4 Ćwiartki układów jednostkowych ze znakami funkcji trygonometrycznych	32
8.5 gra Duck Hunt	33
8.6 Przykładowy wygląd w zadaniu	34
8.7 Przykładowy wygląd zadania	36
9.1 Wygląd BP_MathBoard	39
9.2 Wygląd pokoju algebraicznego	40
9.3 Diagram UML zawierający najważniejsze elementy kontrolerów dla zadania 2 – planimetria.	42
9.4 Diagram przepływu dla zadania 2 – planimetria	44
9.5 Punkt kontrolny w pokoju 3	45
9.6 Ścieżki przejścia przez obie sekcje mostu dla każdego z układów nierówności	46
9.7 Most podzielony na dwie sekcje w pokoju 3	47
9.8 Przyciski do poruszania kostką w pokoju 3	47
9.9 Rozwiążany pokój 3	47
9.10 Zainicjowany układ współrzędnych wraz z jednym punktem, który gracz musi przeciąć wykresem funkcji w pokoju 4	49
9.11 Wszystkie suwaki dostępne w ostatnim etapie w pokoju 4	49
9.12 Zainicjowany układ współrzędnych wraz z przecinającymi się wykresami funkcji w pokoju 5	51
9.13 Przyciski odpowiedzialne za współrzędną X w pokoju 5	52
9.14 Wygląd sejfu	54
9.15 Zrzut ekranu – skrzynia z zamkiem oraz podpowiedź na ścianie dla zadania 7.	55
9.16 Diagram UML zawierający najważniejsze elementy dla zadania 7 – liczby rzeczywiste i działania na zbiorach liczbowych.	56
9.17 Diagram przepływu dla zadania 7 – liczby rzeczywiste i działania na zbiorach liczbowych.	58
9.18 Ruchome sześciany w pokoju 8	60
9.19 Układ jednostkowy w pokoju 8	60
9.20 Sześcian po wykryciu luki w pokoju 8	60
9.21 Porównanie rysowania lasera w przypadku kontaktu z przykładowym elementem otoczenia i ścianą.	62
9.22 Diagram UML zawierający najważniejsze elementy dla zadania 9 – Ciągi liczbowe.	63
9.23 Diagram przepływu dla zadania 9 – Ciągi liczbowe	65
9.24 Rysunek przedstawia postać przed i po modyfikacji.	66

9.25 Rysunek przedstawia wygląd kontenera	68
9.26 Wygląd pokoju z kulkami	70
9.27 Diagram UML zawierający najważniejsze elementy kontrolerów dla zadania 13 – stereometria	72
9.28 Diagram przepływu dla zadania 13 – stereometria	73

SPIS TABEL

8.1 Przykładowa tabela zawierająca zależności między figurami geometrycznymi. . . . 26