UNIWERSYTET
IM. ADAMA MICKIEWICZA
W POZNANIU

Wydział Matematyki i Informatyki

Kamil Tyrek, Mateusz Hypś, Jakub Kozubal
Numer albumu: 434797, 434699, 434726

Projekt i implementacja gry „The Lore: Story of the fallen warrior”

Project and implementation of game „The Lore: Story of the fallen warrior”

Praca inżynierska na kierunku **informatyka**
napisana pod opieką
dr Bartłomieja Przybylskiego

Poznań, styczeń 2021

Poznań, 26 stycznia 2021 r.

Oświadczenie

Ja, niżej podpisany **Kamil Tyrek, Mateusz Hypś, Jakub Kozubal**, student Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt. *Projekt i implementacja gry „The Lore: Story of the fallen warrior"* napisałem samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania rozprawy lub jej części innym osobom, ani nie odpisywałem tej rozprawy lub jej części od innych osób. Oświadczam również, że egzemplarz pracy dyplomowej w wersji drukowanej jest całkowicie zgodny z egzemplarzem pracy dyplomowej w wersji elektronicznej. Jednocześnie przyjmuję do wiadomości, że przypisanie sobie, w pracy dyplomowej, autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego stanowi podstawę stwierdzenia nieważności postępowania w sprawie nadania tytułu zawodowego.

[TAK] wyrażam zgodę na udostępnianie mojej pracy w czytelni Archiwum UAM

[TAK] wyrażam zgodę na udostępnianie mojej pracy w zakresie koniecznym do ochrony mojego prawa do autorstwa lub praw osób trzecich

Spis treści

Rozdział 1

Wstęp

1.1. Cel i założenia projektu

Celem projektu jest stworzenie gry platformowej, zawierającej elementy zręcznościowe oraz łamigłówki. Głównym założeniem projektu jest gra, która zaciekawi swoją fabułą oraz trudnością. Wstępnie prosty zamysł samouczka, w większości gier jest raczej elementem wprowadzającym do rozgrywki, która powinna robić się trudniejsza w im dalszym etapie rozgrywki się znajdujemy, w przypadku naszej gry jest odwrócony. Samouczek poza wprowadzaniu poszczególnych elementów rozgrywki, którymi jest między innymi przedstawienie sterowania oraz funkcjonalności i wstępnych mechanik jest również wymagający, od gracza zależy jakie umiejętności w trakcie rozgrywki będzie rozwijać, aby przejście kolejnych poziomów było łatwiejsze (z perspektywy gracza). Dużą wagę w projekcie przywiązujemy do mini-gier, które występują w trakcie przechodzenia poszczególnych poziomów. Występują dwa rodzaje mini-gier: opcjonalne (te które przejść możemy w celu sprawdzenia siebie oraz zdobycia punktów doświadczenia) oraz wymagane, które trzeba przejść, aby znaleźć się w dalszym etapie gry. Użytkownik posiada do dyspozycji punkty doświadczenia, drzewko umiejętności oraz ekwipunek. Punkty doświadczenia możemy wydawać bezpośrednio w drzewku umiejętności, w którym zadaniem gracza jest wybranie odpowiednich umiejętności zależnie od tego jaką strategię rozgrywki chce przyjąć. Warto pamiętać jednak, że im głębiej będziemy rozwijać daną gałęź tym umiejętności będą bardziej pomocne co sprawia, że gracz musi zastanowić się dobrze nad decyzjami dotyczącymi rozwijania konkretnych umiejętności w drzewku. Ekwipunek służy do zdobywania przedmiotów potrzebnych w trakcie rozgrywki m.in. do otwierania drzwi czy rozpoczęcia opcjonalnej mini-gry. W projekcie są wykorzystane 2 style tworzenia poziomów: ortographic i perspective. Wszystkie animacje w projekcie są tworzone z użyciem technologii inverse

kinematics co pozwala na stałe dodawanie nowych animacji i poprawianie już istniejących.

1.2. Organizacja pracy

Charakter projektu sprawił, iż nie można w naszym zespole jasno podzielić typów zadań, które realizujemy w ramach projektu. Wynikiem tego jest to, iż nowe funkcjonalności są realizowane zazwyczaj przez jedną osobę od początku do końca, nie ma podziału, podobnego jak w przypadku aplikacji z frontendem i backendem.

Jako metodykę pracy przyjęto Scrum. Głównymi powodami tej decyzji jest doświadczenie części zespołu w tej metodyce oraz przejrzystość i rozsądne zarządzanie pracą. Nie rozważano innych metodyk pracy. Przy zarządzaniu pracą wspomaga nas serwis JIRA, który pozwala zarządzać regularne sprinty – w pierwszym semestrze dwutygodniowe, w drugim semestrze tygodniowe.

Kod źródłowy zarządzany jest poprzez GitHub. Dla przejrzystości pracy, każdy commit na GitHub oznaczany jest id zadania na Jirze, dzięki czemu wchodząc w zadanie widzimy commit powiązany z jego rozwiązaniem. W momencie rozpoczęcia zadania deweloper, jeśli następuje potrzeba, tworzy podzadania do zadań na Jirze. Dotyczy to większych zadań, których wykonanie polega na tworzeniu większej ilości funkcjonalności. Dzięki temu realizując nowe zadania, o podobnej budowie, można sugerować się podobnym sposobem działania zadania. Po każdym commicie programista wyznacza ile czasu poświęcił na zadanie, poprzez wbudowaną w Jirze funkcjonalność “Log Work”. Gdy zadanie zostanie zakończone, programista oznacza je statusem “DONE”.

Z racji wybranej metodyki, dokonano również przydzielenia odpowiednich ról członkom zespołu. Podział ról wygląda następująco:

— Kamil Tyrek - Development Team, Scrum Master
— Jakub Kozubal -Development Team, Product Owner
— Mateusz Hypś - Development Team

1.3. Podział prac

1.3.1. Kamil Tyrek - tworzenie elementów logicznych i ich algorytmika

W tym rozdziale zostaną przedstawione sposoby tworzenia elementów logicznych. Część z przedstawionych łamigłówek została użyta w projekcie końcowym. Przykładem są tutaj przesuwane puzzle - rozgrywka polegająca na przesunięciu elementu, celem ułożenia poprawnego obrazka. Algorytmika stojąca za losowaniem kolejności elementów nie jest trywialna, ponieważ źle wylosowana kolejność puzzli powoduje, iż mogą być niemożliwe do ułożenia.

Zostanie przedstawiona też logika mini-gry z ustawieniem odpowiednich ruch, celem połączenia dwóch końców rur. Podobnie jak w poprzednim przykładzie, do rozwiązania tej zagadki potrzebna jest odpowiednia liczba rur danego typu - pionowe, poziome, skrętne.

Następnym przykładem będzie logika stojąca za rozgrywką, która nie jest dostępna w końcowym projekcje, a chodzi tutaj o generowanie labiryntu. Nietrywialnym problemem jest wygenerowanie takiej planszy, aby możliwe było przejście z punktu A do punktu B. W tym rozdziale postaramy się przedstawić rozwiązanie tego problemu, przy użyciu odpowiednich algorytmów.

1.3.2. Mateusz Hypś - Zarządzanie projektem gry komputerowej z wykorzystaniem Agile

W rozdziale zostanie opisany proces zarządzania projektem z wykorzystaniem metodyki zwinnej. Omówione i porównane zostaną podejścia Agile i Agile Game Development oraz zastosowania tych metodyk, w porównaniu z innymi stosowanymi w praktyce. Przedstawione zostaną najważniejszych idee stojące za metodykami Agile m.in. framework SCRUM. Następnie omówione zostanie zastosowanie tych metodyk w projekcie “The Lore”, z uwzględnieniem problemów w trakcie realizacji tego projektu.

1.3.3. Jakub Kozubal - Fizyka postaci

W tym rozdziale zostaną przedstawione sposoby tworzenia poruszania się postaci. Zaznaczone zostaną główne różnice między fizyką rzeczywistą, a tą stosowaną w grach jak i przedstawienie wielu sposobów rozwiązania tej samej funkcjonalności. Ponadto przedstawiony zostanie proces tworzenia postaci. Do tego pojawi się też opisanie problemów takich jak sterowanie postaci w

powietrzu oraz oddziaływanie sił z otoczenia (m.in. poruszające się platformy). Zostanie także przedstawione i przeanalizowane działanie każdej umiejętności występującej w drzewku odpowiedzialnej za poruszanie się.

1.4. Użyta technologia - Unity

W naszym projekcie postanowiliśmy wybrać UNITY jako środowisko do stworzenia naszej gry. Wynikało to z możliwości jakie oferuje oraz z jakości i czytelności, stworzonej przez twórców, oficjalnej dokumentacji. Pozwala nam na tworzenie gier dwuwymiarowych czy trójwymiarowych oraz interaktywnych materiałów, na przykład animacje czy wizualizacje. W razie problemów możemy również wykorzystywać oficjalne forum na którym rzesza użytkowników dzieli się swoimi wskazówkami a także oficjalny sklep – Asset Store, w którym możemy wykupić materiały potrzebne do naszej gry. UNITY działa na każdym systemie operacyjnym, tj. Windows, macOS oraz Linux. Gwarantuje nam również możliwość stworzenia aplikacji nie tylko na komputery osobiste, ale także przeglądarki internetowe, konsole gier wideo oraz urządzenia mobilne. Dzięki aktualizacji silnika do wersji 5.1.1 ta lista wzrasta do 22 platform sprzętowych, w tym gogle wirtualnej rzeczywistości takie jak Oculus Rift. W przeszłości można było tworzyć aplikacje w trzech językach:

— UnityScript (swego rodzaju pochodna JavaScript'u)
— C#
— Boo

Jednak wraz z piątą wersją silnika (wydaną w roku 2015) możliwość pisania w języku Boo została usunięta, pozostała tylko wsteczna kompatybilność w postaci możliwości kompilacji skryptów przez środowisko MonoDevelop. Podobny los dotknął UnityScript, którego wsparcie zakończyło się na wersji 2018.2 (najnowsza wersja stabilna to 2019.3.4). Z tych względów nasz wybór musiał paść na język C#, w którym zostały napisane wszystkie nasze skrypty. Jako jedyny jest wciąż wspierany przez autorów, co zaowocowało drastycznym wzrostem popularności wśród użytkowników.

Rozdział 2

Zarządzanie projektem gry komputerowej z wykorzystaniem Agile

2.1. Czym jest projekt?

Aby odpowiednio przyswoić temat zarządzenia projektem, należy zacząć od zrozumienia paru terminów które są nieodłączną jego częścią.

Projekt jest to termin z pozoru banalny, w końcu spotykamy się z nim wielokrotnie, jednak wytłumaczenie go może sprawiać problemy. Jedną z najlepszych oraz najbardziej wyczerpujących definicji na jaką można trafić jest ta stworzona przez Roberta K. Wysockiego oraz Rudd'a McGary'ego. Są to Amerykańscy specjaliści w temacie zarządzania projektami. W ich książce pod tytułem „Efektywne zarządzanie projektami" definiują projekt jako: „sekwencję niepowtarzalnych, złożonych i związanych ze sobą zadań, mających wspólny cel, przeznaczonych do wykonania w określonym terminie bez przekraczania ustalonego budżetu, zgodnie z założonymi wymaganiami". [1]

Pomimo faktu iż powyższa definicja jest najprawdopodobniej znacznie bardziej złożona od takiej którą słyszymy na co dzień to jest ona bardzo cenna, ponieważ przedstawia najważniejsze cechy każdego projektu.

Składa się z czterech podstawowych elementów, którymi menedżer projektu musi zarządzać symultanicznie. Trzeba pamiętać, że są one ze sobą powiązane. Do wspomnianych elementów należą:
— Zakres
— Zasoby
— Czas
— Pieniądze

2.1.1. Zakres

Zdecydowanie najważniejszy z całej czwórki. Jest on definicją co tak naprawdę projekt ma osiągnąć i co ma w nim zostać zrealizowane. Obrazuje rozmiar projektu, jego cele a także wymagania. Zakres jest nie tylko najważniejszym, ale również najbardziej złożonym. Jakakolwiek zmiana musi zostać odwzorowana w pozostałych trzech elementach. Jest to jeden z powodów dla którego się mówi o współzależności między tą czwórką. Aby to lepiej zrozumieć, można sobie wyobrazić aplikacje, która ma posiadać cztery główne funkcjonalności, zbudowana przez dwa zespoły z budżetem 40 tysięcy złotych. Jeśli zakres projektu się zmieni do przykładowo sześciu funkcjonalności, zadaniem menedżera projektu jest dostosowanie zasobów ludzkich, czasu oraz budżetu w taki sposób aby cel ten został zrealizowany.

2.1.2. Zasoby

Zasoby dzielą się na trzy kategorie:
— Zasoby ludzkie
— Wyposażenie
— Materiały

Zasoby ludzkie

Menadżer projektu musi upewnić się że pracownicy posiadają odpowiednie umiejętności i narzędzia aby ukończyć dane im zadanie. Musi w pełni monitorować czy ma wystarczającą ilość zasobów ludzkich do konkretnego projektu tak aby go ukończyli w ustalonym czasie. Jego zadaniem jest również dopilnowanie aby każda osoba przypisana do zadania doskonale wiedziała i rozumiała co ma zrobić oraz znała wyznaczone terminy. W większych firmach wygląda to inaczej. Pracownicy są podzieleni na grupy którymi zarządza team leader, jest on odpowiedzialny za większość obowiązków które zostały wymienione powyżej. Wynika to z faktu iż menedżer projektu nie jest w stanie zarządzać tak dużym zbiorowiskiem ludzi. Dużym atutem jest również to że team leader najlepiej zna swój zespół, ich umiejętności, możliwości, a także czas którym dysponują. Jest to również duże ułatwienie dla menedżera projektów ponieważ nie musi komunikować się ani nadzorować wszystkich, wystarczy kontakt z team leaderem konkretnych zespołów.

Wyposażenie i materiały

Czasami dochodzi do sytuacji, w której project manager jest odpowiedzialny za pozyskiwanie materiałów i wyposażenie którymi musi zarządzać w taki

sposób aby zespół wykonywał swoją prace w najefektywniejszy sposób. Nie jest to często spotykane zjawisko, w szczególności w dużych firmach w których podzial obowiązków jest mocniej rozdzielony po wielu pracownikach.

2.1.3. Czas

Podobnie jak w przypadku zasobów, czas jest dzielony na trzy grupy
— Podział na zadania
— Harmonogram
— Ścieżka krytyczna (ang. Critical path)

Podział na zadania

Podział na zadania jest pierwszym z trzech kroków do pomyślnego zarządzania czasem. Zadania muszą być tworzone w przemyślany sposób, dobrze przeanalizowane, a także odpowiednio wytłumaczone, tak aby pracownik, który się go podejmie, wszystko zrozumiał za pierwszym razem. Niespełnienie przynajmniej jednego z wyżej wymienionych warunków tworzenia zadań negatywnie wpływa na ich wykonanie. W wielu przypadkach łączy się to z opóźnieniami w realizacji projektu.

Harmonogram

Po pomyślnym stworzeniu zadań przechodzi się do zaplanowania harmonogramu. Tworzy się go poprzez listowanie w odpowiedniej kolejności wszystkich zadań, które muszą zostać wykonane. Niektóre z nich można wykonywać sekwencyjnie, inne z nich mogą nakładać się na siebie, a jeszcze inne mogą zostać wykonane jednocześnie. Kluczem do sukcesu jest ich zrozumienie i poprawne grupowanie. Następnym ważnym czynnikiem jest zrozumienie zależności między nimi, ponieważ niektóre z zadań muszą zostać wykonane w pierwszej kolejności. Ostatnim krokiem tworzenia harmonogramy jest estymacja czasu potrzebnego do ich wykonania, a także przypisanie im odpowiednich zasobów.

Ścieżka krytyczna (ang. Critical path)

Niektóre zadania mają elastyczny termin rozpoczęcia oraz ich zakończenia. Jednak istnieją również takie które tej elastyczności nie mają. Linia przechodząca przez zbiór takich zadań nazywana jest ścieżką krytyczną i wykorzystywana jest do monitorowania w jakim tempie zostają wykonywane zadania w projekcie. W zależności od podziału zadań, istnieje możliwość występowania wielu ścieżek krytycznych. Wszystkie zadania które znajdują się na ich drodze muszą zostać wykonane w terminie, w przeciwnym razie występuje bardzo wysokie prawdopodobieństwo że projekt nie zostanie ukończony na czas.

2.1.4. Pieniądze

Dla najefektywniejszego zarządzania kosztami projektu uwzględnia się jego:

— Koszty
— Wydatki związane z losowymi zdarzeniami
— Zyski

Koszty

Każde zadanie ma określony koszt potrzebny do jego do wykonania, najczęściej bazuje na wydatkach związanych z potrzebnymi zasobami. Każdy z tych wydatków jest estymowany i uwzględniany podczas przygotowania budżetu dla projektu.

Wydatki związane z losowymi zdarzeniami

Podobnie jak w estymacji czasu potrzebnego do wykonania konkretnych zadań, tak samo przy przygotowywaniu budżetu należy uwzględnić pewien bufor. Zostanie on wykorzystany na wypadek losowych zdarzeń, które mogą się w nieoczekiwanym momencie wydarzyć. Najprostszym przykładem w firmie informatycznej może być problem techniczny związany ze sprzętem np. zepsuty komputer.

Zyski

Zyski są to pieniądze, które firma planuje zarobić na wykonanym projekcie, bądź po każdym zakończonym zadaniu. Oczywiście aby projekt został uznany za opłacalny dla biznesu, budżet, który zawiera odpowiednio oszacowane koszty nie może przekraczać pewnego procentu planowanych zysków. Zadaniem menedżera projektu jest oczywiście zminimalizowanie jak najbardziej kosztów produkcji i jak największe zmaksymalizowanie zysku, które firma zarobi po wykonanym projekcie.

2.2. Czym jest zarządzanie projektem

Po dokładnym zaznajomieniu się czym jest projekt, zrozumienie na czym polega zarządzanie projektem staje się trywialne. Jest to proces, którego celem jest ukończenie projektu zgodnie z określonymi wymaganiami (najczęściej czasowymi i budżetowymi), do jego realizacji stosuje się wiedzę, narzędzia, umiejętności, oraz odpowiednie techniki.

Rysunek 2.1. Uproszczony proces realizacji projektu

Powyższy diagram obrazuje proces realizacji projektu w bardzo uproszczonym formacie, ponieważ każdy etap kryje za sobą wiele procesów które trzeba dodatkowo wykonać.

Project manager

2.2.1. Cykl życia projektu informatycznego

2.2.2. Metody pomocne w zarządzaniu projektami informatycznymi

2.2.3. Ryzyko oraz powody niepowodzeń projektów informatycznych

2.3. Zarządzanie projektem gry komputerowej - The Lore

2.3.1. Proces planowania projektu

2.3.2. Wykorzystane systemy i metody projektowe

2.3.3. Rola i wpływ kierownika projektu

2.3.4. Cykl życia projektu The Lore

Rozdział 3

Zagadki logiczne

3.1. Wprowadzenie do zagadek logicznych

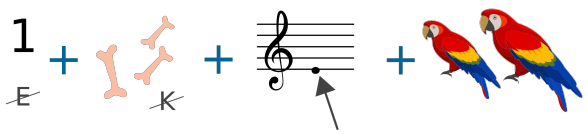
3.1.1. Omówienie zagadnienia

Zagadką logiczną określamy zadanie, którego celem jest odnalezienie odpowiedzi na pytanie, logiczne dojście do rozwiązania problemu czy też czasami abstrakcyjne myślenie. Możemy wyróżnić różne rodzaje zagadek, między innymi graficzne, czego przykładem jest znany, często używany w szkołach podstawowych rebus. Główną ideą takiej rozgrywki jest rozwój swoich intelektualnych możliwości przy dobrej zabawie. [2]

3.1.2. Przykłady logicznych zagadek w prawdziwym świecie

Rebus

Przywołanym już przykładem zagadki może być rebus. To prosta rozgrywka, polegająca na odgadnięciu hasła na podstawie przytoczonego obrazka. Często treści w rebusach mogą być niejednoznaczne, co sprawia, iż nie są one trywialne i wymagają wielu kombinacji haseł, związanych z danym obrazkiem. [3]



Rysunek 3.1. Źródło: Zespół autorski Politechniki Łódzkiej, licencja: CC BY 3.0

Piętnastka - przesuwane puzzle

Kolejnym przykładem może być też piętnastka, która w grze The Lore została zaimplementowana jako przesuwane puzzle rozmiarów 3 na 3. Temat szerzej poruszony będzie w dalszej części pracy, gdzie oprócz założeń związanych z rozwiązaniem zagadki, przedstawiona będzie logika idąca za zrealizowaniem tej zagadki w grze opartej na silniku Unity.

Rysunek 3.2. Źródło: Wikipedia, Sliding Puzzle, Micha L. Rieser - Public Domain

Puzzle

Jedną z najpopularniejszych zagadek logicznych, z którą styczność miał zapewne każdy z nas, są puzzle. Jest to gra, w czasie której rozwiązajemy problem polegający na ułożeniu z dostępnych elementów logicznego obrazka, który zazwyczaj jest dołączony do gry - w postaci fotografii czy też po prostu znajduje na pudełku.

Słowo puzzle pochodzi z XVI wieku, które oznaczało "stan zagubienia". Sama gra ma swoje początki w wieku XVIII, gdy John Spilsbury, grawer i kartograf umieścił mapę na drewnie, następnie przepiłował ją wokół konturów każdego kraju znajdującego się na mapanie. To co powstało postanowił użyć do nauki geografii - ułożenie logicznej mapy uczyło jak wygląda mapa danego regionu. Taka pomoc dydaktyczna zdobyła szybko popularność, wszak była to nauka przez zabawę, używano ją dość często, nawet jeszcze w wieku XIX.

Trudność puzzli zależy zazwyczaj od kształtu elementów i ich ilości. Jeżeli chodzi o kształt, puzzle często mają specyficzny kształt, przez co nie każdy element pasuje do innego, co pozwala nam odrzucić możliwości połączenia różnych par. Ilość elementów zwiększa ilość możliwych oraz logicznych kombinacji danych elementów, przez co zwiększa się czas realizacji zadania. Największa układanka została wykonana przez firmę Educa, posiadała ona aż 42

000 elementów i przedstawia dość bajeczny krajobraz, na którym znajdują się najpopularniejsze budynki z całego świata - Big Ben, Wieża Eiffela czy Krzywą Wieżę w Pizie. Ułożenie tej układanki możemy liczyć w setkach godzin. [5]

Rysunek 3.3. Źródło: Pixnio, Przykład puzzli - Public Domain

3.1.3. Wdrożenie zagadek do gry opartej na silniku Unity

Scena w Unity

W Unity możemy podzielić naszą grę na sceny. Scena to obiekt zawierający nasze menu, czy dane środowisko gry (na przykład poziom gry). Dobrą praktyką jest, aby każdy level gry był osobą sceną, co skróci czas jego ładowania. W scenie możemy umieścić swoje obiekty, skrypty czy grafiki. [6]

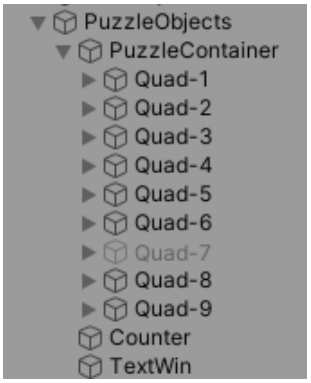
3.1.4. Obiekt - GameObject

Obiekt, zwany w Unity GameObject, to klasa podstawowa dla wszystkich podmiotów w scenach Unity. Do każdego obiektu możemy przypinać kolejne obiekty, tworząc hierarchiczność, która może się przydać w odpowiedniej realizacji projektu. [7]

3.1.5. Sposoby dodawania mini-gier

W toku pracy nad projektem The Lore rozpatrywano dwa sposoby dodawania mini-gier do gry:

— Każda minigra jest w osobnej scenie
  — W przypadku rozbudowanych poziomów nie powiela szerokiej listy obiektów sceny



Rysunek 3.4. Hierarchia obiektów. Przykład z projektu The Lore

- Pozwala na mniejsze pobieranie zasobów w danej jednostce czasu - połączenie działającego poziomu z algorytymką mini-gry może bywać uciążliwe na słabszych komputerach.
- Każda minigra zawiera się w scenie poziomu gry
  - Unikamy dłuższego ładowania się poziomu gry. Po zakończeniu mini-gry w przypadku osobnej sceny, cały poziom musi załadować się od nowa.

Po wewnętrznej dyskusji wybrano opcję wydzielenia do osobnej sceny. W toku testowania tego rozwiązania uznano, iż nie jest ono bardzo uciążliwe pod kątem czasu ładowania poziomu, zaś pozwala na zachowanie porządku w projekcie. Jak się okazało już w przypadku projektowania poziomu 1. gry, ta decyzja była właściwa. Z racji sporego rozbudowania tego obiektu i dużej liczby obiektów, dodanie mini-gier do tej sceny mogłoby spowodować chaos organizacyjny.

3.2. Przesuwane puzzle

3.2.1. Omówienie zagadnienia i ogólne założenia

Przesuwane puzzle to układanka, złożona zazwyczaj z kwadratowej liczby elementów, najczęściej jest to szesnaście pól. Pola są jednakowych rozmiarów i oznaczone są liczbami od 1 do (n-1), gdzie n to liczba dostępnych pól w układance. Jedno pole jest puste, pozwala to na przeniesienie sąsiednich elementów puzzli względem siebie. Rozgrywka kończy się, gdy ułożymy puzzle w odpowiedniej kolejności, według rosnącego porządku liczb lub powstania odpowiedniego obrazka. Trudno określić kto odpowiada za stworzenie zaga-

daki. Wiadomym jest, że w 1878 roku pochodzący ze Stanów Zjednoczonych Samuel Loyd wypromował układankę, jednak prawdopodobnie nie jest to jego pomysł. Dość popularną nazwą na rozgrywkę jest "piętnastka", określającą ilość dostępnych pól w najpopularniejszym ułożeniu - 4x4. **[?]**

W grze The Lore gracze staną przez rozwiązaniem zagadki gdzie do dyspozycji mamy dziewięć pól. Podczas projektowania układanki w grze uznano, iż zagadka może być dość trudna, a korzyści płynące z rozwiązania jej będą nieadekwatne do poświęconego czasu, stąd ilość pól jest mniejsza niż w najpopularniejszej wersji rozgrywki.

3.2.2. Algorytmika

Pojedynczy element puzzle - PuzzleBlock

Każdy pojedynczy element puzzla jest zainicjalizowany jako obiekt, który określamy jako PuzzleBlock. Obiekt posiada koordynaty, które określają jego położenie w przestrzeni na układance.

Rysunek 3.5. Koordynaty każdego pola. Przykład z gry The Lore

Tak jak na załączonym przykładzie, wartość x rośnie w prawą stronę, zaś Y w górę, gdzie x dotyczy położenia w poziomie, a y w pionie. W założeniach przedstawiona została logika, która mówiła, iż element może zostać przemieszczony wtedy i tylko wtedy, gdy pole obok niego jest wolne, czyli nie posiada żadnego PuzzleBlock - elementu z liczbą lub obrazkiem. Dla lepszego efektu wizualnego, w grze The Lore element porusza się stopniowo, aby sprawiał wrażenie, iż porusza się realistycznie. Z racji, iż w Unity każda akcja wykonuje

się podczas pokazywania kolejnej klatki, element porusza się w minimalnym stopniu przez sekundę.

Plansza - Puzzle. Losowanie kolejności puzzli

Plansza rozgrywki posiada 8 elementów PuzzleBlock oraz jedno puste pole. Algorytm powinien wylosować dla 8 pól ich położenie na planszy - tak jak w wyżej przedstawionym przykładzie. W tym celu na początku losujemy dla każdego bloku wartość od 0 do 8. W C# możemy to zrobić przy użyciu funkcji System.Random().Next(a, b). Przykładowo:

Wyciąg 3-1. Fragment klasy `Puzzle.cs`

```
1  private int randomPosition()
2  {
3      int pos = 0;
4      do
5      {
6          pos = new System.Random().Next(0, 9);
7      }
8      while (isOnBoard[pos]);
9      isOnBoard[pos] = true;
10     return pos;
11 }
```

Gdzie isOnBoard[pos] jest tablicą, która weryfikuje, czy dane pole nie jest już zajęte. Jeśli jest, ponownie losujemy wartość dla PuzzleBlock. Oczywiście to nie koniec - z tej wartości musimy stworzyć położenie w postaci (x, y), gdzie x to położenie w poziomie, a y w pionie. W tym celu jedna z tych wartości będzie resztą dzielenia wylosowanej pozycji przez trzy (ilość pól w linii), a druga ilorazem całkowitym pozycji i liczby trzy. Kolejną ważną rzeczą będzie weryfikacja, czy obecne ułożenie puzzli jest wykonalne.

Plansza - Puzzle. Weryfikacja kolejności puzzli

Jak się okazuje niektóre ułożenia puzzli powodują, iż nie ma żadnego rozwiązania tego problemu. Z przypadkiem takiej sytuacji spotykaliśmy podczas pierwszych testów mini-gry puzzle.

Jak się okazuje, występuje tutaj dość prosta zasada. Łamigłówka przesuwanych puzzli z 8 elementami jest rozwiązywalna wtedy i tylko wtedy, gdy liczba inwersji stanu początkowego jest parzysta. Czym jest zaś w tym przypadku liczba inwersji?

Inwersją nazwiemy parę liczb, której wartości są w odwrotnej kolejności, niż w zakładanym stanie końcowym. [8] Przyjmijmy taką sytuację:

Rysunek 3.6. Puzzle bez rozwiązania. Przykład z gry The Lore

1	2	3
6	4	5
7	8	

W tym przypadku liczba inwersji wynosi dwa. Elementami zbioru inwersji są pary (6,4) oraz (6,5) - jak wiadomo, liczba 6 jest w kolejności po cyfrach 4 oraz 5. W tym przypadku rozwiążemy puzzle.

1	2	3
6	5	4
7	8	

W tym przypadku liczba inwersji wynosi już trzy. Elementami zbioru są pary (6,5), (6,4) oraz (5,4). Takich puzzli nie da się rozwiązać. W projekcie platformowej gry w Unity zastosowaliśmy prostą weryfikacje tej sytuacji.

Wyciąg 3-2. Fragment klasy `Puzzle.cs`

```
1         int inversions = 0;
2         for (int i = 0; i < numbersOrdered.Length - 1; i++) {
3             for (int j1 = i + 1; j1 < numbersOrdered.Length - 1; j1++) {
4                 if (numbersOrdered[j1] > numbersOrdered[i]) {
5                     inversions++;
6                 }
7             }
8
9         }
```

Gdzie **inversions**, to liczba znalezionych inwersji, a **numberOrdered** to lista kolejności puzzli w stanie wejściowym. Jeżeli wartość **x** znajduje się w liście

numberOrdered przed wartością **y** i jest od niej większa, to wtedy zwiększamy licznik inwersji o jeden. Oczywiście pozostaje nam prosta weryfikacja, czy liczba inwersji jest nieparzysta - w takim wypadku puzzle będą nierozwiązywalne..

Wyciąg 3-3. Fragment klasy `Puzzle.cs`

```
1         if (inversions % 2 == 1)
2         {
3             pab.restartPuzzleButton();
4         }
```

Gdzie **pab** jest obiektem odwołującym się do akcji **PuzzleActionsButton**, zawierającym przycisk restartButton - ten sam, który dostępny jest w grze w celu zrestrartowania obecnego ułożenia puzzli.

3.2.3. Przedstawienie przykładu w grze The Lore

W grze The Lore zagadka przesuwanych puzzli pełni rolę opcjonalnej rozgrywki, za której rozwiązanie gracz otrzymuje punkty doświadczenia. Mini-grę możemy rozpocząć znajdując się w pobliżu charakterystycznej płytki, wyróżniającej się podświetleniem.

Rysunek 3.7. Miejsce rozpoczęcia puzzli. Przykład z projektu The Lore

W momencie naciśnięcia przycisku akcji kamera zbliża się do obiektu, czyli podświetlonej płytki. Następnie widzimy animację otwierającej się płytki sprawiającej wrażenie, iż ścianka jest przesuwana przez gracza.

Po skończonej animacji rozpoczynamy mini-grę. Oprócz bloczków z kolejnością, na ekranie widzimy również dwa przyciski i cyfrę zero. Przyklad: Gdzie **Exit** kończy rozgrywkę, przenosząc nas z powrotem w miejsce, w którym roz-

Rysunek 3.8. Animacja rozpoczęcia puzzli. Przykład z projektu The Lore

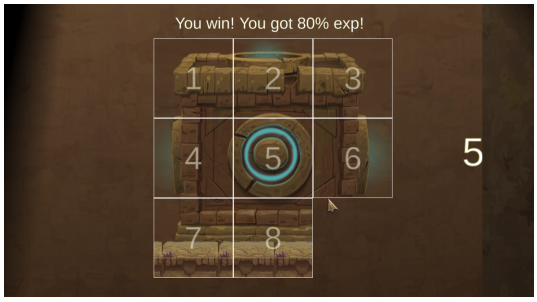
Rysunek 3.9. Cały interfejs mini-gry puzzle. Przykład z gry The Lore

poczynaliśmy zagadkę, **Restart Puzzle** powoduje ponowne rozlosowanie puzzli, odwołując się do akcji w klasie **PuzzleActionsButtons**, czyli tej samej, która uruchamiana jest w przypadku, gdy puzzle nie mają rozwiązania. Dodatkowo po prawej stronie widnieje informacja ile ruchów do tej pory wykonaliśmy rozwiązując zagadkę. Jest to dość istotna informacja dla gracza, gdyż to od ilości ruchów zależy ile punktów doświadczenia zdobędzie za rozwiązanie tej zagadki. Zastosowany wzór dla gry The Lore wygląda następująco:

$$y = \begin{cases} 5 & \text{gdy } x >= 250 \\ 100 - (x/25) * 10 & \text{gdy } x < 250 \end{cases}$$

Gdzie **x** jest liczbą wykonanych ruchów. Po zakończeniu mini-gry jesteśmy informowani o ilości zdobytego doświadczenia. Ponownie widzimy animację przesuwającej się płytki, tym razem zamykającej się.

Każdą mini-grę puzzle można wykonać w grze tylko jeden raz. Ponowne próby są niemożliwe, a o tym, iż zagadka została zakończona, jesteśmy informowani przez fakt, iż płytka symbolizująca mini-grę nie jest podświetlona.



Rysunek 3.10. Wygrana rozgrywka puzzle. Przykład z gry The Lore

Rysunek 3.11. Wygaszona płytka. Przykład z gry The Lore

3.2.4. Przedstawienie przykładu w innych grach

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam quis commodo odio, vel pulvinar risus. Quisque ac lacus urna. Aliquam ullamcorper leo non nisl fringilla.

3.3. Zagadka z rurami

3.3.1. Omówienie zagadnienia

3.3.2. Algorytmika

3.3.3. Przedstawienie przykładu w grze The Lore

3.3.4. Przedstawienie przykładu w innych grach

3.4. Zagadka z otwieraniem skrzyni

3.4.1. Omówienie zagadnienia

3.4.2. Algorytmika

3.4.3. Przedstawienie przykładu w grze The Lore

3.4.4. Przedstawienie przykładu w innych grach

3.5. Labirynt

3.5.1. Omówienie zagadnienia

3.5.2. Algorytmika

3.5.3. Przedstawienie przykładu w innych grach

Rozdział 4

Fizyka postaci

4.1. Tworzenie postaci

4.1.1. Omówienie zagadnienia procesu tworzenia postaci

4.1.2. Proces graficzny tworzenia postaci

4.1.3. Wdrożenie postaci do projektu w Unity

4.1.4. Sposoby animowania postaci

4.2. Poruszanie się

4.2.1. Omówienie zagadnienia

4.2.2. Fizyka w grach 2D, a w prawdziwym świecie

4.2.3. Poruszanie się oraz kolizje postaci

4.2.4. Skakanie

4.2.5. Otoczenie wpływające na fizykę postaci

4.2.6. Umiejętności związane z poruszaniem się

Bibliografia

[1] Robert K. Wysocki, Rudd McGary: Efektywne zarz[1]dzanieprojektami. Wydanie III, ISBN: 83-7361-861-9, dostęp w internecie 26.01.2020r. http://pdf.onepress.pl/efzapr/efzapr-1.pdf

[2] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. https://pl.wikipedia.org/wiki/Zagadka

[3] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. https://pl.wikipedia.org/wiki/Rebus

[4] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. https://en.wikipedia.org/wiki/15_puzzle

[5] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. https://pl.wikipedia.org/wiki/Puzzle

[6] Unity Documentation, dostęp 09.01.2020r. https://docs.unity3d.com/Manual/CreatingScenes.html

[7] Unity Documentation, dostęp 09.01.2020r. https://docs.unity3d.com/ScriptReference/GameObject.html

[8] Geeks for geeks. Check instance 8 puzzle solvable, dostęp 26.01.2020r. https://www.geeksforgeeks.org/check-instance-8-puzzle-solvable/