



UNIWERSYTET
IM. ADAMA MICKIEWICZA
W POZNANIU

Wydział Matematyki i Informatyki

Kamil Tyrek, Mateusz Hyps, 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

Bibliography	4
Rozdział 1. Wstęp	5
1.1. Cel i założenia projektu	5
1.2. Organizacja pracy	6
1.3. Podział prac	7
1.3.1. Kamil Tyrek - tworzenie elementów logicznych i ich algorytmika	7
1.3.2. Mateusz Hypś - Zarządzanie projektem gry komputerowej z wykorzystaniem Agile	7
1.3.3. Jakub Kozubal - Fizyka postaci	7
1.4. Użyta technologia - Unity	8
Rozdział 2. Zarządzanie projektem gry komputerowej z wykorzystaniem Agile	9
2.1. Zarządzanie projektem informatycznym	10
2.1.1. Proces zarządzania projektem i jego geneza	10
2.1.2. Cykl życia projektu informatycznego	10
2.1.3. Metody pomocne w zarządzaniu projektami informatycznymi	10
2.1.4. Ryzyko oraz powody niepowodzeń projektów informatycznych	10
2.2. Zarządzanie projektem gry komputerowej - The Lore	10
2.2.1. Proces planowania projektu	10
2.2.2. Wykorzystane systemy i metody projektowe	10
2.2.3. Rola i wpływ kierownika projektu	10
2.2.4. Cykl życia projektu The Lore	10
Rozdział 3. Zagadki logiczne	11
3.1. Wprowadzenie do zagadek logicznych	11
3.1.1. Omówienie zagadnienia	11
3.1.2. Przykłady logicznych zagadek w prawdziwym świecie	11
3.1.3. Wdrożenie zagadek do gry opartej na silniku Unity	13
3.1.4. Obiekt - GameObject	13
3.1.5. Sposoby dodawania mini-gier	13
3.2. Przesuwane puzzle	14
3.2.1. Omówienie zagadnienia i ogólne założenia	14
3.2.2. Algorytmika	15

3.2.3. Przedstawienie przykładu w grze The Lore	18
3.2.4. Przedstawienie przykładu w innych grach	20
3.3. Zagadka z rurami	21
3.3.1. Omówienie zagadnienia	21
3.3.2. Algorytmika	21
3.3.3. Przedstawienie przykładu w grze The Lore	21
3.3.4. Przedstawienie przykładu w innych grach	21
3.4. Zagadka z otwieraniem skrzyni	21
3.4.1. Omówienie zagadnienia	21
3.4.2. Algorytmika	21
3.4.3. Przedstawienie przykładu w grze The Lore	21
3.4.4. Przedstawienie przykładu w innych grach	21
3.5. Labirynt	21
3.5.1. Omówienie zagadnienia	21
3.5.2. Algorytmika	21
3.5.3. Przedstawienie przykładu w innych grach	21
Rozdział 4. Fizyka postaci	22
4.1. Tworzenie postaci	23
4.1.1. Omówienie zagadnienia procesu tworzenia postaci	23
4.1.2. Proces graficzny tworzenia postaci	23
4.1.3. Wdrożenie postaci do projektu w Unity	23
4.1.4. Sposoby animowania postaci	23
4.2. Poruszanie się	23
4.2.1. Omówienie zagadnienia	23
4.2.2. Fizyka w grach 2D, a w prawdziwym świecie	23
4.2.3. Poruszanie się oraz kolizje postaci	23
4.2.4. Skakanie	23
4.2.5. Otoczenie wpływające na fizykę postaci	23
4.2.6. Umiejętności związane z poruszaniem się	23
Bibliografia	24

ROZDZIAŁ 1

Wstęp

1.1. CEL I ZAŁOŻENIA PROJEKTU

Celem projektu jest stworzenie gry platformowej, zawierającej elementy zręcznościowe oraz łamigłówek. Głównym założeniem projektu jest gra, która zacieka 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 gałąź 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, skątne.

Następnym przykładem będzie logika stojąca za rozgrywką, która nie jest dostępna w końcowym projekcie, a chodzi tutaj o generowanie labiryntu. Nie trywialnym 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 Hyps - 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 najważniejsze 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. ZARZĄDZANIE PROJEKTEM INFORMATYCZNYM

2.1.1. Proces zarządzania projektem i jego geneza

2.1.2. Cykl życia projektu informatycznego

2.1.3. Metody pomocne w zarządzaniu projektami informatycznymi

2.1.4. Ryzyko oraz powody niepowodzeń projektów informatycznych

2.2. ZARZĄDZANIE PROJEKTEM GRY KOMPUTEROWEJ - THE LORE

2.2.1. Proces planowania projektu

2.2.2. Wykorzystane systemy i metody projektowe

2.2.3. Rola i wpływ kierownika projektu

2.2.4. 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. [1]

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. [2]



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 rozwiązujemy 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 mapie. 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. [4]



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 osobną sceną, co skróci czas jego ładowania. W scenie możemy umieścić swoje obiekty, skrypty czy grafiki. [5]

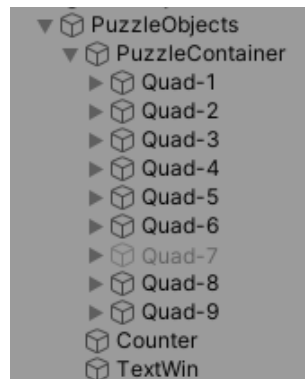
3.1.4. Obiekt - GameObject

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. [6]

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 algorytmką mini-gry może być 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-

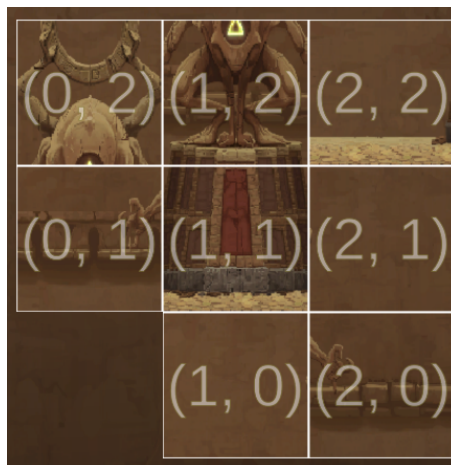
daki. Wiadomym jest, że w 1878 roku pochodzący ze Stanów Zjednoczonych Samuel Loyd wypromował układankę, jednak prawdopodobnie nie jest to jego pomysł. Dość popularną nazwą na rozgrywkę jest "piętnastka", określającą ilość dostępnych pól w najpopularniejszym ułożeniu - 4x4. [?]

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. [7] 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 weryfikację 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 zrestartowania 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.



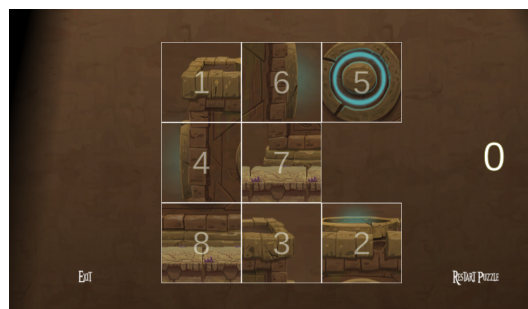
Rysunek 3.7. Miejsce rozpoczęcia puzzli. Przykład z projektu The Lore

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. Przykład: 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:

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

Gdzie x jest liczbą wykonanych ruchów. Po zakończeniu mini-gry jesteśmy informowani o ilości zdobytego doświadczenia. Ponownie widzimy animację przesuwanącej 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 płytki. 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] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. <https://pl.wikipedia.org/wiki/Zagadka>
- [2] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. <https://pl.wikipedia.org/wiki/Rebus>
- [3] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. https://en.wikipedia.org/wiki/15_puzzle
- [4] Wikipedia, Wolna Encyklopedia." Wikimedia Foundation, Inc. July 17, 2002, dostęp 07.01.2020r. <https://pl.wikipedia.org/wiki/Puzzle>
- [5] Unity Documentation, dostęp 09.01.2020r. <https://docs.unity3d.com/Manual/CreatingScenes.html>
- [6] Unity Documentation, dostęp 09.01.2020r. <https://docs.unity3d.com/ScriptReference/GameObject.html>
- [7] Geeks for geeks. Check instance 8 puzzle solvable, dostęp 26.01.2020r. <https://www.geeksforgeeks.org/check-instance-8-puzzle-solvable/>