AGH Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Fizyki i Informatyki Stosowanej

Praca inżynierska

Tomasz Piech

Kierunek studiów: **informatyka stosowana**

Gra RPG 2D napisana w silniku UNITY implementująca rozwój postaci.

Opiekun: **dr inż. Janusz Malinowski**

Kraków styczeń 2021

Oświadczenie studenta Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy. Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. – Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

……

(czytelny podpis studenta)

Merytoryczna ocena pracy przez opiekuna:

Merytoryczna ocena pracy przez recenzenta:

SPIS TREŚCI

Wstęp

1.1.Wprowadzenie

Znaczenie gier komputerowych bardzo wyraźnie wzrosło w ostatniej dekadzie XXI wieku.

Bez wątpienia zaczęły one odgrywać znaczną rolę w strefie kulturowej, stały się nową dziedziną rozrywkowo- kulturową, która wywiera istotny wpływ na sposób spędzania wolnego czasu, szczególnie u młodych ludzi. Można więc mówić, w odniesieniu do sfery relaksu, rozrywki i zabawy w naszych czasach, o erze gier komputerowych.

Z pewnością na popularność tej formy spędzania wolnego czasu ma wpływ, tak poprawa jakości sprzętu komputerowego, jak i atrakcyjność samych gier, ich grafika, fabuła, elementy edukacyjne.

Początkowo, wobec gier komputerowych wysuwano wiele zarzutów. Nie były one kojarzone z czymś wartościowym, przypisywano im negatywny wpływ na psychikę młodych graczy, powodowanie uzależniania od komputera czy nabieranie złych nawyków .

Krytyka ta, nie była całkiem nieuzasadniona, ponieważ w latach 80-tych i 90-tych ubiegłego wieku pojawiało się na rynku wiele gier niskiej jakości, nie tylko technicznej ale też etycznej, mających wręcz elementy demoralizujące. Później, jak również obecnie, duże i znane firmy, dbające o swoją renomę, bardzo starannie dobierają treści swoich produktów, kładąc nacisk na elementy przygodowe, edukacyjne, twórcze. Dlatego największą popularnością wśród współczesnych graczy cieszą się gry typu RPG. Jednak wybór gier, dokonywany zwłaszcza przez młode osoby, powinien być kontrolowany przez osoby dorosłe – rodziców, nauczycieli, opiekunów. Ważne są przy tym takie cechy grającego jak: wiek, dojrzałość psychiczna i emocjonalna, poziom intelektualny, czas przeznaczony na grę. Trzeba zdawać sobie sprawę z tego, że zbyt częsty kontakt z grami, może mieć znaczny wpływ na rozwój i kształtowanie się osobowości i zachowań młodej osoby.

Odpowiednio dobrana gra pod względem treści i wszystkich możliwości gracza, może przynosić wiele korzyści. Nie tylko zapewnia ciekawą rozrywkę, ale pomaga kształtować pewne umiejętności i cechy: kształcenie umiejętności manualnych, strategicznych, rozwijanie wyobraźni, zdolność samodzielnego podejmowania decyzji i ponoszenia ich konsekwencji, szybki refleks. Wartościowe gry komputerowe RPG posiadają elementy edukacyjne, służą uczeniu myślenia dedukcyjnego, analizowania, wyciągania wniosków, odczytywania

wskazówek , konsekwencji w dążeniu do celu, podporządkowania się ustalonym zasadom, a w grach zespołowych, umiejętności współpracy i szanowania decyzji pozostałych graczy.

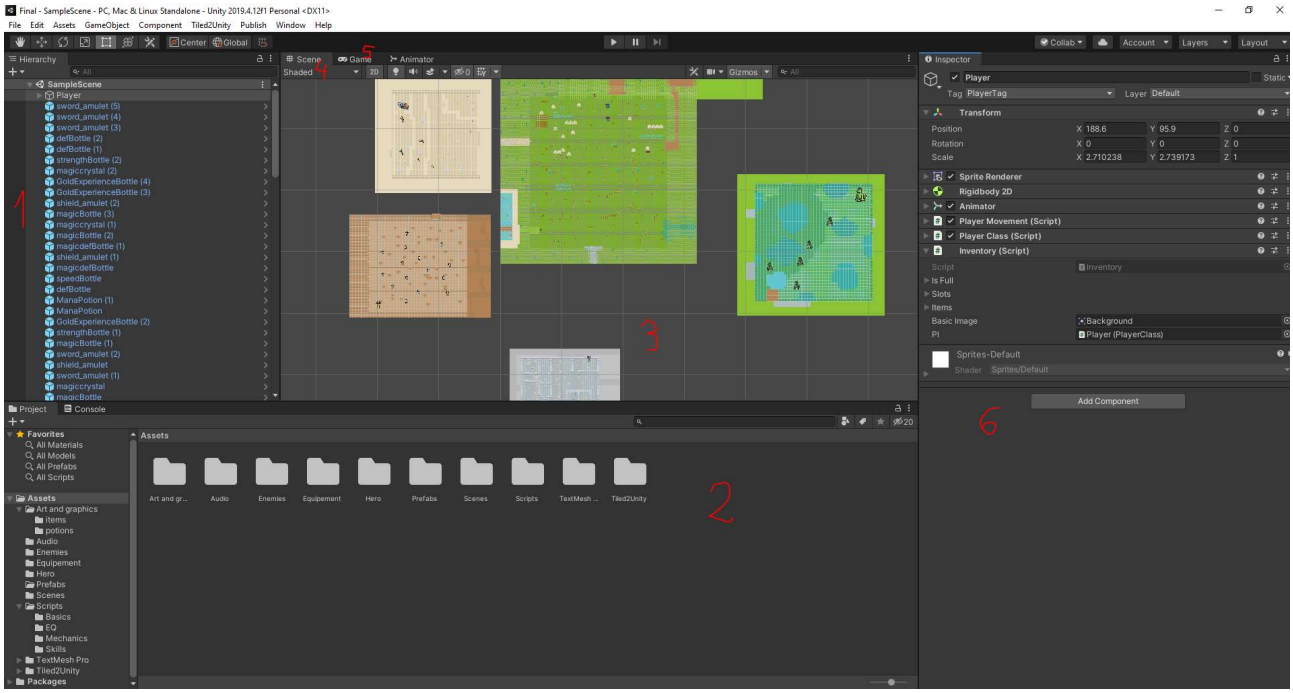
Podsumowując, można stwierdzić, że w obecnym czasie gry komputerowe stały się nieodłącznym sposobem na spędzanie wolnego czasu wielu ludzi, dlatego, aby nie stały się czymś destrukcyjnym dla naszego życia, a wręcz przeciwnie, przyczyniały się do rozwoju i dostarczały pozytywnych emocji, należy starannie i mądrze je wybierać oraz odpowiedzialnie z nich korzystać .

1.2 Cel pracy

Celem mojej pracy było zapoznanie się z silnikiem Unity, możliwościami jakie zapewnia, udoskonalenie mojej znajomości języka C# oraz zaprezentowanie części umiejętności nabytych przeze mnie podczas nauki. W napisanej przeze mnie pracy, użyte zostały m .in: programowanie obiektowe, polimorfizm, różne techniki przesyłania danych pomiędzy obiektami, umiejętność analizy kodu, zarządzanie grafiką za pomocą zarówno kodu, jak i programu.

Efektem jest krótka gra RPG ("role-playing game"), którą zatytułowałem "Final Dream" ("Ostatni sen"). Celem gry jest zapewnienie krótkotrwałej rozrywki użytkownikowi oraz sprawdzenie jego umiejętności strategicznych i spostrzegawczości. Gracz wciela się w śpiącego bohatera, uwięzionego we własnym śnie. Ma do pokonania szereg wyzwań, które mają doprowadzić do jego wybudzenia.

1.3 Podstawowe narzędzie – edytor Unity



(na obrazku: 1 – lista obiektów dostępnych na scenie, 2 – miejsce dodawania i pobierania zasobów, 3- główna scena, 4- zakładka głównej sceny, 5 – zakładka widoku gry, 6 – właściwości wybranego przedmiotu)

Silnik Unity oferuje edytor graficzny, za pomocą którego można łatwo i szybko zarządzać obiektami. Dostępne są opcje ustawiania położenia wewnątrzgrowych obiektów, zarządzania kamerą, dodawania skryptów, oraz właściwości do obiektów. Edytor oferuje też opcję uporządkowania swoich zasobów poprzez stworzenie drzewka folderów.

Oprócz edycji sceny, istnieje też możliwość przełączenia się na tryb rozgrywki, na którym możemy przetestować to, co do tej pory stworzyliśmy.

Unity ma funkcję szybkiego dostępu do plików udostępnianych przez innych twórców, zarówno do tych darmowych, jak i za uiszczeniem opłaty.

Do wyboru jest też platforma, na której stworzony program będzie działał. Do wyboru są między innymi: plik wykonywalny, odpalany na komputerach, aplikacja na system android czy aplikacja mobilna WebGL.

Rozdział 1. Zarys historyczny

1.1 Zarys historii i geneza gier RPG

Dużą rolę, tak w tworzeniu gier typu RPG, jak i uczestniczeniu w nich, odgrywa wyobraźnia. To dzięki niej można wcielić się w jakąś postać pochodzącą ze świata realnego lub też zupełnie fantastyczną, mityczną, baśniową.

W grach fabularnych występuje postać: Mistrz Gry (MG), (z ang. Game Master - GM lub Referee, Storyteller- Narrator , Dungeon Master – Mistrz Podziemi). Jego funkcją jest nadzorowanie przebiegu rozgrywki i pełnienie roli narratora, który opisuje to, co postacie widzą, słyszą, czują . Gra, zaś polega często na dialogu między graczami a Mistrzem Gry. Rozgrywający informują Mistrza o tym, co ich postacie wykonują, a on przekazuje im rezultaty tych działań.

Warto zauważyć, że gry fabularne mają przygotowany ogólny scenariusz, obejmujący główne wydarzenia, wątki, postacie, które pojawiają, się w trakcie gry. Ciekawe jest jednak to, że ów plan nie musi być ściśle realizowany, gracze, bowiem mają swobodę w podejmowaniu decyzji, wyborów, co ma istotny wpływ na dalszy przebieg akcji. Muszą jednak trzymać się konwencji stworzonego w grze świata oraz, zazwyczaj, praw fizyki.

Istotą gier RPG jest dialog między Bohaterami Graczami (BG) i Mistrzem Gry (MG). Ten ostatni jest nie tylko twórcą scenariusza, ale poza nadzorowaniem przebiegu rozgrywki, przejmuje także rolę pozostałych postaci niewybranych przez graczy, tzw. Bohaterów Niezależnych (BN) (z ang. Non-Player Character- NPC). Są to często istoty fantastyczne, ale także byty obdarzone inteligencją, wchodzące w interakcję z BG.

Gry typu narracyjnego, w odróżnieniu do innych gier, nie polegają na rywalizacji między graczami a Mistrzem, czy między sobą. Nie ma tu też strony wygrywającej i przegrywającej. Satysfakcję grającym daje wspólne tworzenie ciekawej fabuły, rozwijającej się w nieznanym dla wszystkich kierunku.

Gracze, po wyborze postaci, wyposażają je w charakterystyczne dla nich cechy fizyczne(np. siła, kondycja) i psychiczne (np. inteligencja, wiedza) oraz w atrybuty specjalne, jakieś wyjątkowe umiejętności. Wszystkie te informacje są nanoszone na kartę postaci.

W sytuacjach konfliktowych, gdy nie można rozstrzygnąć od razu ich wyniku, stosuje się mechanikę. Najczęściej jest to rzut kostką (są do dyspozycji kostki od 6-ściennych do 100-ściennych), który ma służyć określeniu wyniku. Sytuacje trudne do rozstrzygnięcia np.(walki,

upadki), mają przypisaną konkretną liczbę, która umożliwia określenie szansy powodzenia danej czynności.

Jedna rozgrywka jest nazywana sesją i trwa przeciętnie od 4 do 6 godzin. Aby mogła się odbyć, potrzeba co najmniej jednego gracza i MG, jednak najczęściej uczestniczą drużyny składające się z prowadzącego i 3 do 4 graczy.

Istnieje bardzo wiele systemów RPG (świat, zwany settingiem plus mechanika) w różnych konwencjach. Może to być fantasy, science fiction, horror oraz różne ich kombinacje. Bywają światy inspirowane znanymi komiksami, telewizyjnymi kreskówkami itp. Bywają także gry RPG odstające od typowego schematu. Jednakże, wszystkie one służą doskonałej rozrywce, rozwojowi wyobraźni i umiejętności porozumienia się oraz współpracy grupowej [1].

1.2 Zarys historii gier cRPG, rynku i technologii towarzyszących

Jako początek ery gier komputerowych uważa się stworzony w 1958 roku Tennis for Two, a druga połowa lat 70-tych, to początek rewolucji w grach wideo na domowe komputery, konsole do gier i automaty do gier. Konsole pojawiły się w 1972 roku. Były to proste urządzenia, zdolne do uruchamiania tylko Ponga i innych podobnych gier, wstępnie wbudowanych w sprzęt. Konsole drugiej generacji zostały wyposażone w układy ROM, umożliwiając firmom opracowywanie nowych gier na swoje urządzenia. Komputery domowe , mimo że znacznie bardziej wszechstronne niż konsola do gier, były drogie, niezwykle skomplikowane w obsłudze i nie oferowały swoim użytkownikom czegoś innego niż proste aplikacje i gry. Przez kilka następnych lat, rynek odbiorców stale rósł, po czym wraz z Space Invaders, eksplodował w 1979 roku, rozpoczynając złoty wiek gier cRPG. W ciągu pięciu lat, gry wideo przeszły od Ponga do bogatego systemu z wieloma gatunkami, platformami i odbiorcami.[2]

Gdy na początku lat 80, na rynku komputerów domowych pojawiły się IBM, przemysł technologiczny zmienił się radykalnie. Podczas, gdy Apple zniechęcał niezależnych programistów, IBM udostępnił wszystkie informacje i otwartą architekturę. IBM stał się wiodącym standardem, dzięki obszernemu katalogowi oprogramowania i komponentów, ale nadal były to produkty drogie . Kluczowe znaczenie w popularyzacji komputerów domowych, miały nowe urządzenia z niższej półki, takie jak Commodore 64, ZX Spectrum. Podłączało się je do zwykłych telewizorów. Zaczęły być używane jako narzędzia edukacyjne i automaty do gier. W międzyczasie rynek konsoli stał się bardzo chaotyczny. Wiele firm publikowało

własne gry, ale większość z nich miała słabą jakość lub była tanią kopią popularnych tytułów, więc klienci przestali je kupować. W tym czasie w Japonii, tego samego dnia - 15 lipca 1983 roku, dwie firmy wypuściły swoje pierwsze konsole: Nintendo Famicom i Sega SG-1000.[3]

Po krachu w 1983 roku, konsole do gier wideo stały się mało popularne, ale Sega i Nintendo wpadły na pomysł, aby oferować swój produkt, nie jako konsole do gier wideo, ale jako zabawki. Osiągnięciem Nintendo stał się jej „znak firmowy ", który gwarantował dobrą jakość każdej opublikowanej gry, a chip blokujący uniemożliwił innym firmom wydawanie nielicencjonowanych gier. Nintendo gwarantowało jakość każdej opublikowanej gry, a programiści zostali zmuszeni do podpisywania umów z firmami konsolowymi. Pojawiła się nowa generacja komputerów osobistych, na czele z Commodore Amiga i Atari ST. Bardzo poprawiono wydajność, dźwięk i grafikę, a wyjątkową rewolucją było wprowadzenie myszy i graficznego interfejsu użytkownika. Dzięki temu komputery domowe stały się bardziej intuicyjne i dostępne.[4]

Wczesne lata 90 są często określane jako złoty wiek gier. W ciągu zaledwie kilku lat, wymyślono lub udoskonalono nowe gatunki, które dały początek ciągłym seriom i klasycznym tytułom. Zaczęły się rozpowszechniać gry typu „shareware", które można było wypróbować za darmo, a następnie kupić w wersji pełnej. Był to sposób, w jaki małe firmy mogły bezpośrednio promować swoje gry. Wiele magazynów z grami zaczęło dołączać płyty CD-ROM wypełnione tytułami shareware, a także dema i zwiastuny gier, pomagając im rozpowszechniać się jeszcze bardziej. Wspomagane przez nową technologię, taką jak grafika VGA i procesory Intel i386, komputery PC okazały się bardzo mocnym segmentem sprzętu używanego do gier[5].

W drugiej połowie lat 90. kontynuowano wydawanie nowych gier, a nowością stała się grafika 3D. Konkurencja rozwijała rynek kart akcelerujących 3D, kart dźwiękowych, a komputery PC weszły do głównego nurtu[6].

Lata 90. były czasem wielkich skoków technologicznych, a początek nowego tysiąclecia, czasem drastycznych zmian w biznesie. Playstation 2 biło wszelkie rekordy sprzedaży, ostatecznie stając się najlepiej sprzedającą się konsolą wszech czasów, ale koszty rozwoju wymyślnej grafiki 3D, wciąż rosły. Tworzenie gier było wysoce dochodowym, ale także niezwykle ryzykownym biznesem. Podczas, gdy komputery domowe były bardziej popularne niż kiedykolwiek wcześniej, gry na PC zostały przyćmione sukcesem PS2. Mówiło się o „śmierci gier komputerowych" i chociaż gry na PC nie umarły, to zdecydowanie ucierpiały. W 2000 roku pojawił się nowy pretendent, Microsoft. Xbox. Był on bramą dla firm, które chciały

spróbować swoich sił na kwitnącym rynku konsol. Aby je ułatwić, Xbox (skrót od „DirectX Box”), został zaprojektowany tak, aby doświadczeni programiści PC mogli z nim łatwo pracować.[7]

W pierwszej dekadzie XXI wieku, branża gier została niemal całkowicie zdominowana przez kilku gigantycznych wydawców, takich jak EA, Activision i Ubisoft, oraz trzech producentów konsol - Nintendo, Sony i Microsoft. Ta sytuacja, w połączeniu z ciągle wysokimi kosztami produkcji, doprowadziła do stagnacji. Wszystkie gry musiały wyjść w wersji funkcjonującej na każdej możliwej platformie. Pojawienie się siódmej generacji konsoli, jeszcze bardziej podniosło koszty rozwoju.[8]

Nowa dekada przyniosła powszechną demokratyzację branży gier. Programiści uzyskali nawet darmowy dostęp do silników do gier. Rozwój gier mobilnych i gier niezależnych wydawców, zmienił także sposób myślenia o kupowaniu gier. Na obecnym rynku funkcjonują gry zarówno od jednoosobowych twórców gier niezależnych, średnich studiów i dużych studiów, doświadczonych programistów, a także od tych, którzy mają pomysł w głowie i darmowy silnik w swoich komputerach [9]

Rozdział 2 Technologia wykonania gry

2.1 Implementacja technologii

W silniku Unity, do tworzenia wewnątrzgrowych skryptów wykorzystuje się język C#. Jest to obiektowy, wieloparadygmatowy język kompilowalny, stworzony dla firmy Microsoft. Unity implementuje do niego własną bibliotekę "UnityEngine", zawierającą liczne obiekty odczytywane przez program, ułatwiające użytkownikowi interakcję.

W mojej grze wytworzyłem ponad 60 klas reprezentujących różne obiekty, mające swoje odzwierciedlenie w przedmiotach wewnątrzgrowych, funkcjach mechanicznych, postaciach oraz umiejętnościach.

W poniższych podpunktach postaram się omówić działanie określonych mechanik, zawartych w moim programie.

a) Postacie i przechowywane przez nie dane

W celu zachowania polimorfizmu, zacząłem od napisania klasy bazowej dla postaci. Klasa "Character" służy jako schemat przechowujący podstawowe statystyki każdej postaci. Należą do nich: siła ataku, zdrowie maksymalne, zdrowie posiadane przez postać obecnie, ilość magicznej energii, umiejętności magiczne, ochrona przed atakami fizycznymi, ochrona przed atakami magicznymi, szansa zadania obrażeń krytycznych oraz poziom danej postaci. Poza poziomem, wszystkie te statystyki mają dwie składowe: wartość bazową oraz wartość dodaną. Wartość bazowa jest niezmienna podczas walk, rośnie wraz z rozwojem postaci oraz dzięki znalezionym przez gracza miksturom.

Wartość dodana jest wartością "chwilową", resetowaną po każdej walce. Wpływają na nią umiejętności użyte podczas walki przez postać oraz przez przeciwnika, a także użyte przedmioty jednorazowego użytku, określone w samej grze jako "amulety", o których wspomnę w następnych punktach.

Klasa "PlayerClass" jest pierwszą klasą pochodną po klasie "Character". Jest ona reprezentacją postaci sterowanej przez gracza. Przechowuje dodatkowe informacje takie jak: lista znanych przez bohatera umiejętności, kilka zmiennych logicznych typu "bool, odpowiadających stanowi w jakim znajduje się gracz oraz ważne dla rozgrywki elementy świata. W tej klasie zawarty jest również podstawowy schemat rozwoju, który można zmienić w trakcie gry, rozmawiając z jednym z trzech "nauczycieli". Klasa ta posiada też możliwość wyświetlenia ekwipunku, jaki gracz posiada.

Klasa "Enemy" reprezentuje przeciwnika. Oprócz przechowywanych statystyk, posiada odwołanie do wykorzystywanej w walce umiejętności, wartość doświadczenia dodawanego graczowi po wygranej walce oraz informację o tym, że zostaje pokonana.

Występują w niej również metody "OnTriggerEnter2D" oraz "OnTriggerExit2D", odpowiadające za zarządzanie "polem widzenia" postaci. Jeśli gracz w nie wejdzie, rozpoczyna się walka. Przeciwnik przesyła stosowną informację do obiektu zarządzającego systemem walki.

b) Postaci niezależne

W celu informowania gracza o zadaniach oraz o świecie przedstawionym, stworzyłem klasę "interactionObject". Początkowo miała ona reprezentować dowolny obiekt, z którym gracz mógł wejść w interakcję, a same postaci niezależne, miały być klasą pochodną. Jednak wraz ze zmianą podejścia do ekwipunku i systemu interakcji, postanowiłem rozdzielić ten schemat na dwie różne części. Klasa ta, podobnie jak "Enemy", obsługuje "pole widzenia" obiektu, (nie jest obowiązkowe), a wymaga nadania komunikatu zatwierdzającego poprzez wciśnięcie przycisku "E". Aktywuje się wtedy pole, na którym wyświetla się komunikat postaci oraz możliwości dialogowe, wybierane przyciskami 1 – 3, wprowadzanymi z klawiatury.

c) Przedmioty

Podczas gry, gracz ma możliwość zbierania do swojego ekwipunku różnych przedmiotów.

W celu zachowania polimorfizmu, stworzyłem klasę "GameItem". Po tej klasie dziedziczą pozostałe klasy, które można przechowywać. Są to "amulety", "mikstury" oraz "kryształy".

Mikstury po "wypiciu" zwiększają jedną ze statystyk gracza na stałe o daną, niewielką wartość.

Dla urozmaicenia rozgrywki, wprowadziłem jeszcze trzy mikstury"specjalne": miksturę zdrowia, miksturę "many" (mocy magicznej, omówionej w innym rozdziale) oraz miksturę doświadczenia. Skorzystanie z dwóch pierwszych przywraca zasoby, których gracz używa podczas walki. Ostatnia mikstura z wymienionych, dodaje do puli doświadczenia, 10% ilości wymaganej do osiągnięcia następnego poziomu.

Amulety, po wykorzystaniu, dodają wartość procentową, jednak jest ona tracona po najbliższej walce. W przeciwieństwie do mikstur, których można "wypić" dowolną ilość, dla każdej statystyki, może być użyty tylko jeden amulet.

Ostatnim rodzajem przechowywanego przedmiotu, jest "keyItem" ("przedmiot kluczowy"). Są to "kryształy", których zebranie wymagane jest do przywołania ostatniego przeciwnika i ukończenia rozgrywki.
Każdy przedmiot posiada również krótki opis oraz zmienne logiczne, opisujące możliwości interakcji z nim.

d) Ekwipunek

Jak wspomniałem we wcześniejszych punktach, podczas tej krótkiej przygody, gracz spotyka na swojej drodze wiele niebezpieczeństw. W celu zwiększenia efektywności pokonywania ich, na mapach zostały rozmieszczone różnego rodzaju przedmioty. Gra, jednak została skonstruowana w taki sposób, że niektóre z nich są efektywniejsze, gdy wykorzystane zostaną w późniejszym etapie lub w określonym momencie. Dla możliwości przechowania tych przedmiotów oraz ułatwienia interakcji z nimi, utworzone zostały klasy "Inventory" oraz "InventoryItemOptions". Odpowiadają one za przechowywanie obiektów w tablicy, odczytywaniu ich wartości oraz możliwość wykorzystania ich lub odłożenia w celu zwolnienia miejsca w ekwipunku. Równocześnie gracz może posiadać maksymalnie 36 przedmiotów.

e) Dialogi

Informacje o świecie oraz instrukcje, zostały umieszczone w systemach dialogowych. Rozmawiając ze wspomnianymi wcześniej postaciami niezależnymi, można dowiedzieć się różnych ciekawostek, mających za zadanie wprowadzić zamierzony klimat oraz zasugerować kolejność pokonywania przeszkód. Podczas dialogu można zadawać pytania, wybierając jedną z podanych opcji. Jest też możliwość wcześniejszego pominięcia rozmowy. Zdobywanie informacji jest obligatoryjne.

W celu obsługi tego mechanizmu, powstał obiekt "DialogueScript" aktywujący pole tekstowe i wybrane opcje oraz przekazujący odpowiedzi.[16]

f) Misje

Jak wiadomo założeniem gier jest zapewnienie rozrywki poprzez postawienie pewnego rodzaju wyzwania. W celu obsługi tego zagadnienia zaimplementowane zostały klasy "Quest", "QuestHander" oraz "QuestTrigger". Pierwsza reprezentuje podstawowe zadanie oraz przechowuje informacje o jego aktywności, nagrodzie w postaci doświadczenia i wartości logicznej jego wykonania. Druga, przechowuje obiekty "Quest" i zarządza nimi, a trzecia obsługuje aktywację lub zakończenie tych obiektów, po dojściu gracza na dany obszar.[17][18][19][20]

g) Rozwój postaci

W grach RPG ("Role-play games") istotną rolę odgrywa możliwość rozwijania swojej postaci i wpływu gracza na jej formę. W moim programie przygotowałem niewielki system, który pozwala użytkownikowi na decydowanie o statystykach podstawowych postaci oraz o wyborze schematu rozwoju i zdobyciu umiejętności. Bohater posiada wspomniany wcześniej "schemat" (początkowo pusty), który wpływa na zwiększanie się wartości podstawowych statystyk, przy zdobyciu poziomu. Oprócz schematu, gracz co poziom, dostaje trzy punkty rozwoju, które może wydać w zależności od własnej decyzji (zdrowie i "mana" zwiększają się o 10, za każdy wydany punkt).

Schemat ten można otrzymać, wchodząc w interakcję z jednym z trzech "nauczycieli". Jednak po wybraniu go, pozostali nauczyciele znikają. Schematy te odpowiadają planowanym "stylom" walki: bazującym na wytrzymałości, na atakowaniu wręcz oraz na umiejętnościach magicznych.

Oprócz schematu rozwoju, gracz dostaje też przygotowany wcześniej zestaw umiejętności, z których może korzystać podczas walki. Umiejętności pierwszej z wymienionych klas, opierają się na wzmacnianiu swojej odporności i redukcji prędkości przeciwnika.

Druga klasa pozwala na zwiększenie swojego ataku, szansy na trafienie krytyczne i redukcji ochrony przeciwnika. Trzecia, zaś daje możliwość zadania sporych obrażeń magicznych oraz nakładania różnorakich efektów na siebie lub wroga, lecz kosztem magicznej energii.

h) Umiejętności

Wspomniane wcześniej umiejętności są pochodnymi klasami obiektu "Skill". We wspomnianej wcześniej instancji "PlayerClass" istnieją dwie listy odnoszące się do tego zagadnienia. W pierwszej przechowywane są same obiekty umiejętności, w drugiej zaś, krótkie opisy. Utworzony został też obiekt "AllSkillsHandler". Ma on na celu udostępniać graczowi oraz przeciwnikom obiekty pochodne klasy "Skill", z których mogą korzystać i się ich uczyć.

i) "Studnie"

W głównej części sceny istnieją dwie studnie. Jedna po interakcji przywraca zdrowie do pełni, druga robi to samo z "maną".

2.2 Implementacja i omówienie mechanik

Silnik Unity zapewnia spore możliwości w implementowaniu funkcji wewnątrzgrowych. Dzięki temu programista może urozmaicać swoje programy w różnoraki sposób, tak aby przedstawione przez niego rozwiązania, były coraz bardziej zaskakujące dla odbiorcy.

W tej części pracy, chciałem zaprezentować moje pomysły na implementację tych zagadnień.

a) Walka

W napisanej przeze mnie grze obowiązuje system turowy. Zawsze zaczyna gracz, następnie jest kolej przeciwnika. Do wyboru są cztery opcje: atak fizyczny, ochrona, rzucenie zaklęcia lub próba ucieczki.
Atak: obrażenia fizyczne liczone są według następującego, zaproponowanego przeze mnie wzoru:

$$dmg=\lceil user.atttack*difference*crit*0.01\rceil$$

Gdzie: dmg- to obrażenia łączne, user.attack- to umiejętności ofensywne atakującego, difference oznacza (100 + user.attack – target.def), w którym z kolei target.def- to umiejętności defensywne celu, "crit" przyjmuje wartość 2, przy powodzeniu losowego testu na obrażenia krytyczne lub 1, jeśli ten test się nie powiedzie.

Ochrona: Na najbliższą turę, umiejętności obrony i obrony magicznej, rosną dwukrotnie. Dotyczy to statystyk podstawowych, (co oznacza, że statystyki dodatkowe nie mają wpływu).

Zaklęcie:

Po naciśnięciu przycisku odpowiedzialnego za wybór, na ekranie pojawia się rozwijany pasek, z którego wybieramy umiejętność oraz przycisk zatwierdzający wybór. Należy jednak uważać, gdyż, jeśli nie posiadamy odpowiedniej ilości "many", zaklęcie się nie powiedzie . W celu utrudnienia wprowadzono opcję,że przeciwnikom nie kończy się "mana" (z wyjątkiem używania umiejętności o nazwie "HyperBeam", która wykorzystuje stałe 50% całkowitej "many").

Obrażenia magiczne liczone są według wzoru:

$$dmg = \lceil user.magic * difference * 0.01 \rceil + spelldmg$$

Gdzie: dmg- to obrażenia łączne, user.magic- to magiczne umiejętności ofensywne atakującego, difference -oznacza (100 + user.magic – target.magicdef), w którym z kolei target.magicdef -to magiczne umiejętności defensywne celu. Spelldmg jest stałą wartością dla wybranego zaklęcia.

Wyjątkiem od tego wzoru, są czary zadające stałe obrażenia. Przykładem jest zaklęcie o nazwie "SacrificeSpell" ("Czar ofiary"), które odbiera stałe 20 punktów zdrowia, w zamian za poświęcone 10.

Próba ucieczki

Jeśli gracz spodziewa się, że sobie nie poradzi, może spróbować uciec. Aby to się udało, statystyka jego szybkości musi być większa od prędkości przeciwnika. Jeśli ta jest mniejsza, bądź równa, ucieczka się nie powiedzie. Gracz nie traci tury przy nieudanej próbie. W ucieczce pomóc mogą czary zwiększające prędkość lub spowalniające przeciwnika. Po udanej ucieczce, gracz pojawia się w początkowej lokacji w swoim "łóżku", a przeciwnik wraca do pełni zdrowia. Umożliwia to jednak skorzystanie z uprzednio wspomnianych "studni" oraz wcześniejsze użycie przedmiotów.

Zwycięstwo w walce

Jeśli przeciwnik straci całe zdrowie, zostaje pokonany, a gracz nagrodzony doświadczeniem i niejednokrotnie możliwością zebrania przedmiotu lub odblokowania ścieżki.

Przegrana

Jeśli to gracz jako pierwszy straci swoje zdrowie, zostaje przeniesiony na początek mapy i zostaje zmuszony do skorzystania ze studni lub z mikstur leczniczych. Podjąłem decyzję o tym, aby nie karać gracza za porażkę w żaden znaczący sposób, ponieważ moim głównym założeniem jest zapewnienie chwilowej rozrywki.

Klasa "BattleHandler" używa modułów "couroutine", które umożliwiają robienie krótkich odstępów czasowych, w celu zwiększenia czytelności.

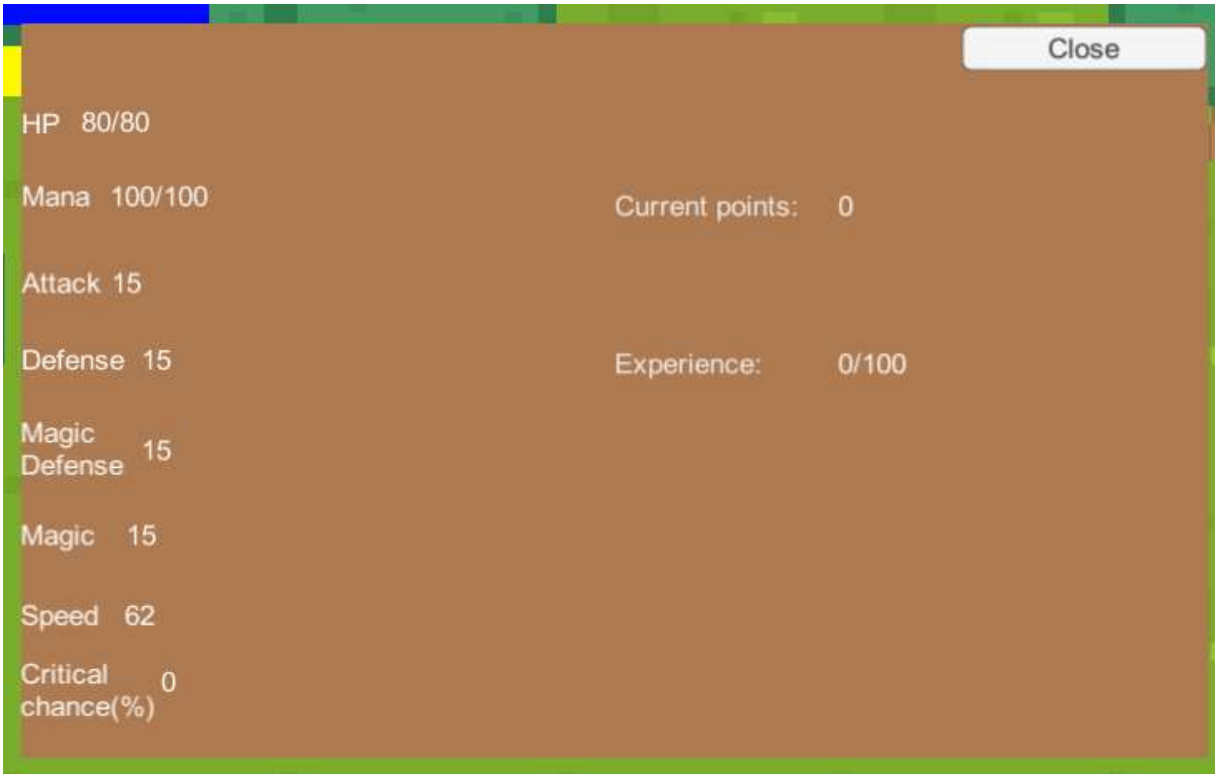
Obiekt ten posiada również zapisane stałe współrzędne dwóch punktów, przenoszących gracza i przeciwnika na "arenę senną", na której toczy się pojedynek. W zależności od wyniku,

zwycięzca zostaje przeniesiony z powrotem na swoje miejsce. Pokonany przeciwnik, jeśli jest graczem, wraca na początek poziomu, jeśli jest obiektem "enemy", zostaje dezaktywowany.

Na powyższym obrazku widzimy przykładową scenę z walki. Czerwony pasek oznacza zdrowie gracza, niebieski -energię magiczną, żółty zaś, wytrzymałość umożliwiającą wykonanie sprintu poza walką. Czerwony pasek w prawym górnym rogu, symbolizuje zdrowie przeciwnika.[12]

b) **Rozwój postaci**

Po zdobyciu poziomu, oprócz statystyk rozwijanych automatycznie zgodnie z jednym z wybranych przez gracza schematów, postać otrzymuje 3 punkty umiejętności do wydania. Można dzięki temu na stałe zwiększyć umiejętności podstawowe o +1 za każdy punkt (lub + 10 do zdrowia i "many"). Na obrazku poniżej widzimy schemat ze statystykami postaci:



Po zdobyciu wystarczającej ilości doświadczenia, bohater może wydać otrzymane punkty wciskając przycisk +

Należy jednak uważać, ponieważ po wciśnięciu przycisku, zmian nie można cofnąć.

Schematy rozwoju:

Wojownik fizyczny:

atak +5,

szansa na zadanie obrażeń krytycznych +5,

zdrowie +5,

prędkość +3,

obrona +3

Czarodziej:

Magia+5,
mana +10,

ochrona magiczna +3,

prędkość +3

Obrońca:
zdrowie +10,

mana +5,

obrona +3,

ochrona magiczna +3

Kod odpowiadający za zarządzanie poziomami:

Klasa "LevelUpHandler"

```
public class LevelUpHandler : MonoBehaviour
{
    public GameObject dBox;
    public Text hptxt;
    public Text manatxt;
    public Text strtxt;
    public Text deftxt;
    public Text magdeftxt;
    public Text magictxt;
    public Text speedtxt;
    public Text exptxt;
    public Text pointstxt;
    public Text critxt;
    public Button atk;
    public Button def;
    public Button mdef;
    public Button magic;
    public Button hp;
    public Button speed;
    public Button mana;
    public Button close;
    public Button crit;
```

```
public PlayerClass PL;
bool isActiveTab;
// Start is called before the first frame update
void Start()
{
    dBox.SetActive(false);
    atk.gameObject.SetActive(false);
    def.gameObject.SetActive(false);
    mdef.gameObject.SetActive(false);
    magic.gameObject.SetActive(false);
    hp.gameObject.SetActive(false);
    speed.gameObject.SetActive(false);
    mana.gameObject.SetActive(false);
    crit.gameObject.SetActive(false);
    close.gameObject.SetActive(false);
    isActiveTab = false;
}
// Update is called once per frame
void Update()
{
    hptxt.text = PL.current_hp + "/" + PL.max_hp;
    manatxt.text = PL.current_mana + "/" + PL.max_mana;
    strtxt.text = PL.attack.ToString();
    deftxt.text = PL.def.ToString();
    speedtxt.text = PL.speed.ToString();
    magdeftxt.text = PL.magic_def.ToString();
```

```
    magictxt.text = PL.magic.ToString();
    pointstxt.text = PL.development_points.ToString();
    critxt.text = PL.crit_chance.ToString();
    exptxt.text = PL.experience + "/" + PL.exp_threshold;
    if (PL.IsInAction == false || isActiveTab) {
        if (Input.GetKeyDown(KeyCode.C))
        {
            if (!isActiveTab)
            {
                PL.IsInAction = true;
                show_my_stats(PL.development_points);
            }
            else
            {
                close_tab();
            }
        }
    }
}
public void close_tab()
{
    dBox.SetActive(false);
    atk.gameObject.SetActive(false);
    def.gameObject.SetActive(false);
    mdef.gameObject.SetActive(false);
    magic.gameObject.SetActive(false);
```

```
    hp.gameObject.SetActive(false);
    speed.gameObject.SetActive(false);
    mana.gameObject.SetActive(false);
    close.gameObject.SetActive(false);
    crit.gameObject.SetActive(false);
    PL.IsInAction = false;
    isActiveTab = false;
}
void show_my_stats(int points)
{
    dBox.SetActive(true);
    close.gameObject.SetActive(true);
    isActiveTab = true;
    if (points > 0) {
        atk.gameObject.SetActive(true);
        def.gameObject.SetActive(true);
        mdef.gameObject.SetActive(true);
        magic.gameObject.SetActive(true);
        hp.gameObject.SetActive(true);
        speed.gameObject.SetActive(true);
        mana.gameObject.SetActive(true);
        crit.gameObject.SetActive(true);
    }
}
```

```
public void increase_hp() {
    if (PL.development_points > 0)
    {
    PL.baseHp += 10;
    PL.current_hp += 10;
    hptxt.text = PL.current_hp + "/" + PL.max_hp;
    PL.development_points -= 1;
    }
    // pointstxt.text = PL.development_points.ToString();
}
public void increase_mana() {
    if (PL.development_points > 0)
    {
        PL.baseMana += 10;
        PL.current_mana += 10;
        manatxt.text = PL.current_mana + "/" + PL.max_mana;
        PL.development_points -= 1; }
```

```
    //pointstxt.text = PL.development_points.ToString();
}
public void increase_attack() {
    if (PL.development_points > 0)
    {
        PL.baseAttack += 1;
```

```
        strtxt.text = PL.attack.ToString();
        PL.development_points -= 1;
    }
   // pointstxt.text = PL.development_points.ToString();
}
public void increase_defence()
{
    if (PL.development_points > 0)
    {
        PL.baseDef += 1;
        deftxt.text = PL.def.ToString();
        PL.development_points -= 1;
    }
   // pointstxt.text = PL.development_points.ToString(); }
```

Klasa przechowuje informacje o polu typu "canvas", otwieranym lub zamykanym po wciśnięciu przycisku "C", o doświadczeniu i punktach rozwoju przechowywanym przez postać oraz o przyciskach i polach tekstowych zawierających się we wspomnianym wcześniej obiekcie reprezentującym "płótno".

Metody typu "increase" wywoływane są po wciśnięciu przycisku "+" na ekranie rozwoju postaci. Podnoszą określoną statystykę, odejmując jeden punkt rozwoju.

Zakomentowane fragmenty początkowo miały za zadanie przesyłać obecną ilość punktów rozwoju do obiektu "canvas", jednak z postępem pracy, rozwiązałem to zagadnienie w inny sposób.

Fragmenty odpowiadające za rozwój w klasie "PlayerClass":

```
if (experience >= exp_threshold) {
            lvl++;
            development_points += 3;
            experience = experience - exp_threshold;
            exp_threshold = lvl * 100;
            baseClassSchema.develop_stats(this);
            //baseClassSchema.add_new_skill(lvl, playerSpells);
        }
```

Na wyżej wymienionym fragmencie, sprawdzam, czy doświadczenie jest wystarczające do "awansu". Jeśli tak, "próg awansu" zwiększa się o 100 punktów, a nadmiarowa różnica zostaje zachowana. Postać zyskuje 3 punkty rozwoju przeznaczone do wydania, w zależności od preferencji użytkownika, statystyki są rozdawane zgodnie z obecnym schematem. Zakomentowany fragment miał za zadanie dodawać umiejętność do listy, po zdobyciu określonego poziomu, jednak z czasem i ta formuła została zmieniona.

```
public void add_new_tank_skills()
    {
        AllSkillsHandler ash = GameObject.FindGameObjectWithTag("AllSkillsHandler")
       .GetComponent<AllSkillsHandler>();
        playerSpells.Add(ash.allskills[3]);
        playerSpells.Add(ash.allskills[12]);
        playerSpells.Add(ash.allskills[14]);
        playerSpells.Add(ash.allskills[15]);
```

```
    playerSpells.Add(ash.allskills[17]);
    playerSpellsNames.Add(ash.descriptions[3]);
    playerSpellsNames.Add(ash.descriptions[12]);
    playerSpellsNames.Add(ash.descriptions[14]);
    playerSpellsNames.Add(ash.descriptions[15]);
    playerSpellsNames.Add(ash.descriptions[17]);
}
```

Funkcja "add_new(...)_skills" zostaje wywołana przy kontakcie z jednym z trzech "nauczycieli" i dodaje do listy, przygotowane umiejętności.

c) **Ekwipunek**

Zbierane przedmioty, napisana przeze mnie funkcja, dodaje do ekwipunku. Obiekt "Inventory", przechowuje informacje o posiadanych przedmiotach oraz daje możliwości ich wykorzystania. Po kliknięciu na dany przedmiot, pojawiają się dwie lub trzy opcje (w zależności od rodzaju przedmiotu).

(Na zdjęciu kolejno są umieszczone: amulet ataku, mikstura siły, magiczny kryształ, amulet magii, mikstura ochrony magicznej i mikstura magii)

Po naciśnięciu na zapełnione pole, dostajemy opcję przeczytania informacji, wyrzucenia lub, jeśli to możliwe, skorzystania z przedmiotu.

Po użyciu opcji wyrzucenie, przedmiot nie znika . W razie potrzeby można po niego wrócić później.

```
public class Inventory : MonoBehaviour
{

    public bool[] isFull;
    public Image[] slots;
    public GameItem[] items;
    public Sprite basicImage;
    public PlayerClass pl;

}
```

Klasa reprezentująca ekwipunek: "tems" – tablica przechowująca przedmioty, Klasa "slots" – to tablica przechowująca poszczególne obiekty odpowiadające za komórki (każda z tych komórek jest obrazkiem w formie przycisku z doczepionym skryptem). Klasa "basicImage" odpowiada za obrazek "podstawowy" (wyświetlany, gdy miejsce w tablicy nie jest zajęte, a podmieniany jest, gdy postać podnosi przedmiot) . Gdy postać wyrzuca przedmiot, tablica "slots" pobiera wspomniany "podstawowy" obrazek z pola "basicImage". Pole klasy PlayerClass jest odwołaniem do obiektu gracza.[13][14]

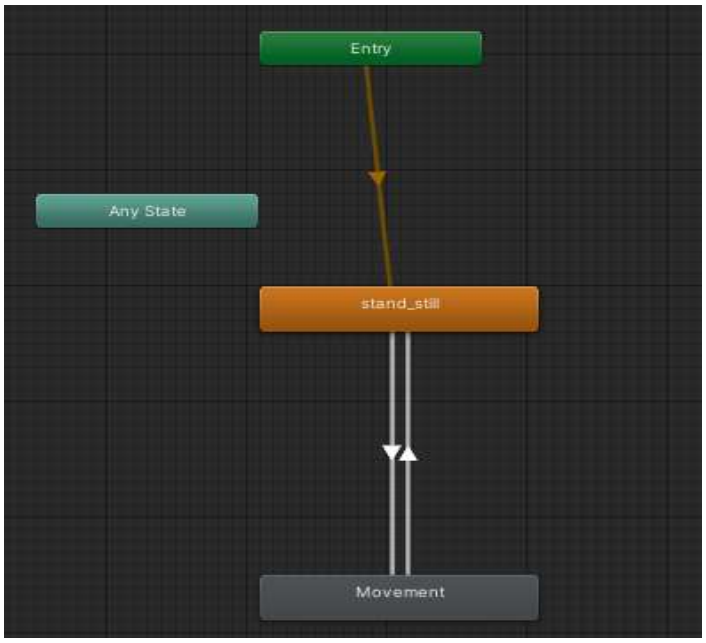
d) **Poruszanie się**

Mechanika ruchu połączona jest z podstawową klasą animacji. Tak długo, jak jeden z przycisków ruchu jest wciśnięty, tak w tym czasie, postać porusza się w danym kierunku. Wartości położenia są w tym czasie stale zwiększane. Zaimplementowałem również możliwość wykonania krótkotrwałego "sprintu". Postać może poruszać się w zwiększonym

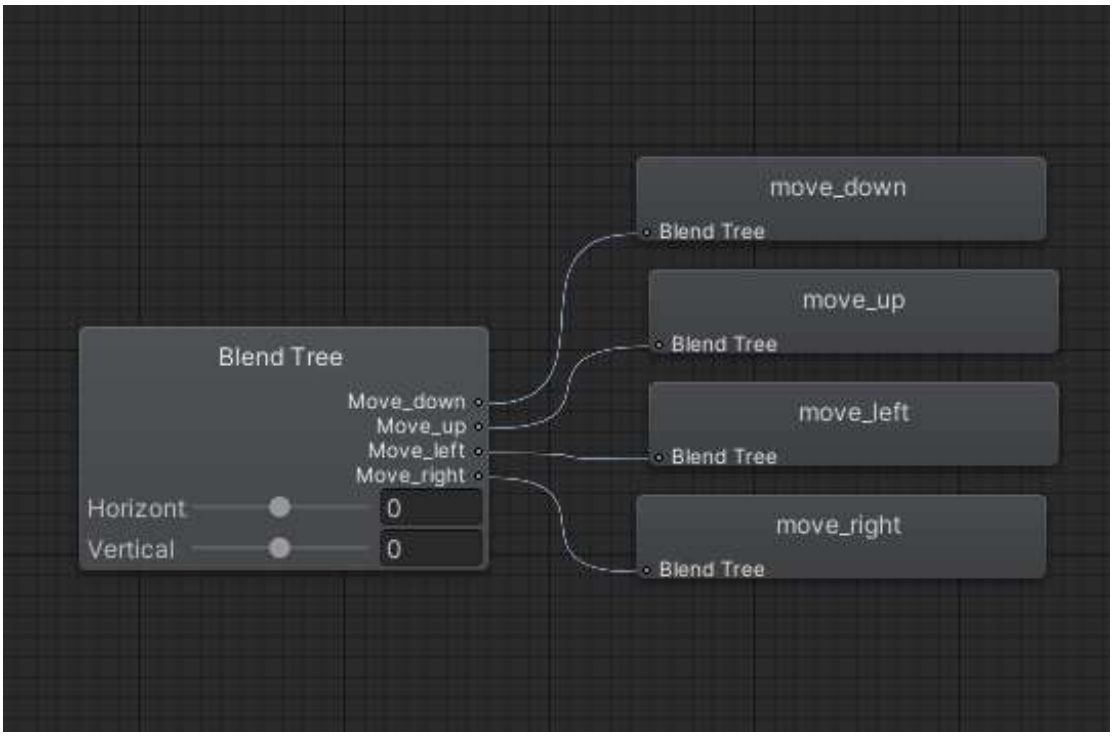
tempie tak długo, jak wystarcza mu na to "wytrzymałości". Gdy ta dojdzie do 0, postać automatycznie zwalnia do standardowej prędkości, a energia zaczyna ładować się automatycznie.

Gdy żaden z przycisków nie jest wciśnięty, odpala się automatyczna animacja stania w miejscu, a postać jest zwrócona twarzą do gracza.[11]

Klasa "PlayerClass zawiera także element "isInAction", którego zadaniem jest zezwolenie lub odmowa dostępu do możliwości poruszania się i wywoływania animacji ruchu u gracza.



Na powyższym schemacie widzimy warunek przejścia pomiędzy obiektem animacji bazowej a obiektem "blend tree", który zawiera pozostałe animacje ruchu, odtwarzane w zależności od przesyłanych wartości w obiekcie "PlayerMovement. Aby jednak warunek rozpatrzenia tych wartości został spełniony, prędkość poruszania się postaci musi być większa od 0 oraz warunek logiczny "isInAction", musi posiadać wartość "false".



Kod odpowiadający za poruszanie się postaci:

```
void Update()
    {


        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");
        anime.SetFloat("Vertical", movement.y);
        anime.SetFloat("Horizontal", movement.x);
        if (movement.sqrMagnitude != 0)
        {
            anime.SetFloat("Speed", 1);
        }
        else
```

```
    {
        anime.SetFloat("Speed", 0);
    }
    if (Input.GetKeyUp(KeyCode.LeftShift))
    {
        Sprint = !Sprint;
    }
    if (Sprint == true && PL.stamina > 0 && movement.sqrMagnitude != 0)
    {
        sprintspeed = 2;
        PL.stamina -= 2;
    }
    else
    {
        Sprint = false;
        sprintspeed = 1;
        PL.stamina += 0.7;
        if (PL.stamina > 1000)
        {
            PL.stamina = 1000;
        }

    }
}
void FixedUpdate()
{
```

```
    if (PL.IsInAction == false) {
        rb.MovePosition(rb.position + movement * MS * 0.4f * sprintspeed);
    }

}
```

Funkcja Update() wywołuje się co klatkę[1]. Pobiera i ustawia wartości płożenia gracza. Jeśli położenie się zmienia, informacja ta przesyłana jest do obiektu "animator", który reaguje na to wyzwoleniem odpowiedniej animacji.

Funkcja FixedUpdate() wywołuje się w regularnych odstępach czasowych (w przeciwieństwie do update, która może mieć problem przy słabszym sprzęcie i gra się "zawiesza"). Jeżeli wartość "IsInAction" jest prawdziwa, gracz nie może zmienić położenia. Jeżeli zaś, tak nie jest, gracz odzyskuje możliwość ruchu.

Rozdział 3 Opis gry

3.1 Instrukcja gry

a) **Sterowanie**

W mojej grze zawarłem bardzo prosty system sterowania postacią: przyciski "W","A","S","D" służą do poruszania się po mapach, lewy shift do włączenia trybu "sprintu".

Po podejściu do obiektu, z którym możliwa jest interakcja, możemy do niej przystąpić, wciskając przycisk "E". Podczas dialogu, opcje wybieramy przyciskami 1 – 3 wprowadzanymi z klawiatury.

Interakcja z przeciwnikami jest automatyczna. Podczas walki, opcje wybieramy myszką.

Podobnie jest z ekwipunkiem, który możemy otworzyć i zamknąć za pomocą przycisku "I". Po naciśnięciu myszką na przedmiot, pojawiają się trzy przyciski, pozwalające na wybór jednej z opcji.

Przycisk "C" wyświetla pole przedstawiające aktualne statystyki gracza. Jeśli liczba punktów rozwoju jest dodatnia, pojawiają się dodatkowe przyciski, umożliwiające stałe zwiększenie statystyk.

b) **Misje**

W mojej grze, warunkiem jej ukończenia, jest pokonanie ostatniego przeciwnika. Jednak, aby do tego doszło, gracz musi wykonać wcześniej 7 innych zadań:

- porozmawiać z pierwszą postacią niezależną,

- porozmawiać z "psem Walterem" (druga postać niezależna, informująca o przedmiotach dostępnych w grze),

- znaleźć 5 kryształów ukrytych w różnych zakątkach świata.

Za każde wykonane zadanie, gracz zostaje nagrodzony punktami doświadczenia. Gdy wcześniejsze zadania zostaną wykonane, na specjalnym polu pojawia się ostateczny przeciwnik ("Final Boss"). Po pokonaniu go, gra się kończy, zostawiając na ekranie krótką notkę.

c) **Dodatkowa pomoc**

Niektórzy przeciwnicy mogą stanowić wyzwanie dla gracza. Ułatwieniem ich pokonania może być podjęcie różnych czynności.
Są to między innymi: odpowiednie przygotowanie do walki poprzez skorzystanie z przedmiotów jednorazowego użytku (określonych w innej części pracy jako "amulety"), "picie" magicznych mikstur zwiększających statystyki na stałe, skorzystanie z pomocy jednego z trzech "nauczycieli", co powoduje dodanie kilku umiejętności do puli dostępnych dla gracza czarów. Postać zyskuje wtedy również schemat rozwoju, zwiększający automatycznie część statystyk, po otrzymaniu poziomu.
Czasem, jednak lepszym rozwiązaniem jest ucieczka i spróbowanie swoich sił z mniej wymagającymi przeciwnikami.
Gdyby było to konieczne, można skorzystać z umieszczonych na mapie dwóch "studni". Po interakcji, pierwsza z nich uzupełnia zdrowie do maksimum, druga energię magiczną (studnia z niebieską obwódką).

(Na obrazku powyżej: gracz stojący obok trzech nauczycieli : 1 - stylu ochronnego, 2 - stylu magicznego, 3- stylu fizycznego)

(Na obrazku powyżej: gracz stojący obok studni mocy magicznej)

(Na obrazku powyżej: gracz stojący obok studni mocy fizycznej)

3.2 Fabuła

Krótka fabuła mojej gry opiera się o motyw bohatera uwięzionego w swoim własnym śnie. Gracz podejmuje działania, mające na celu doprowadzić postać do szczęśliwego końca.

Wybór tematyki sennej, pozostawił mi spore możliwości do kreatywnego podejścia do poziomów. Oprócz standardowego motywu lasu i polany, mających dać graczowi "wytchnienie", pojawia się kilka innych lokacji: nawiedzone ruiny, cmentarz, statek piracki i bagno. Walka z ostatecznym przeciwnikiem odbywa się na mapie polany, w specjalnym miejscu przygotowanym na jego przyzwanie. Po pokonaniu go, pojawia się krótka notatka opisująca dzieje bohatera.

3.3 Inspiracja

Wykorzystaną w grze mechanikę wzorowałem na popularnej serii gier "Pokemon", wychodzącej od roku 1996, wytwarzaną przez studio "GameFreak" [10]. Pierwsza gra z tej serii, posiadała bardzo proste animacje, muzykę, oraz prosty system turowy. Gracz będący "trenerem" tytułowych stworków, miał do wyboru jedną z czterech znanych przez jego

podopiecznego, umiejętności do wykonania podczas tury lub mógł skorzystać z przedmiotu. Każdy ze stworków posiadał swój indywidualny schemat rozwoju, na który można było w pewnym niewielkim stopniu wpływać, poprzez walkę z odpowiednimi typami przeciwników oraz indywidualny zestaw umiejętności, jakich mógł się uczyć.

Tę część chciałem odwzorować, umieszczając w mojej grze turowy system walki, wspomniany schemat rozwoju, danie możliwości graczowi na rozwój postaci zgodnie z własnym wyborem oraz system wpływania na statystyki przez przedmioty jednorazowego użytku.

3.4 Grafiki

Grafiki wykorzystywane w programie można podzielić na 2 typy: złożone oraz proste. Jako "złożone" określam przygotowane przeze mnie mapy, złożone w programie "Tiled" i wyeksportowane za pomocą "Tiled2Unity".[15]

Do stworzenia ich użyłem gotowych "kafelków", udostępnionych przez użytkownika o pseudonimie "Kenney" na jego własnej stronie[G1].

Pozostałe grafiki wykorzystane w programie, pobrałem z profilu "GamesPlusJames" na portalu dropbox[G2], a część została pobrana za pomocą wyszukiwarki Google Grafika i przerobiona przeze mnie w programie GIMP, w celu dodania kanału alpha oraz delikatnych korektach graficznych.

3.5 Napotkane problemy techniczne

Mimo swojej powszechności, silnik Unity ma swoje błędy. Nie wszystkich da się uniknąć, nawet przy starannie zaplanowanej pracy. Negatywnym doświadczeniem, które nabyłem podczas pisania programu, było to, że natknąłem się na kilka niedogodności, które utrudniły mi pracę.

Jednym z nich jest fakt skalowania obiektów "canvas" w zależności od rozdzielczości. Pomimo starannego dopasowania ich rozmiarów, podczas tworzenia, w procesie budowania, wartości te zmieniały się automatycznie, ponieważ silnik chciał "wymusić" odpowiedni rozmiar obszaru "płótna", co ostatecznie skończyło się brakami w czytelności. Innym problemem, który był ciężki do wykrycia z uwagi na swoisty styl występowania, był błąd pobierania informacji o obiekcie z wykorzystaniem funkcji "GameObject.FindObjectWithTag". Funkcja ta bezproblemowo działała podczas testowania gry w edytorze zawartym w silniku, jednak podczas testowania na pliku wykonywanego przez

system Windows, wartość zwracana wynosiła "null". Rozwiązaniem problemu, było dodanie do korzystających z tej funkcji obiektów, pola wskazującego na wyszukiwany obiekt bezpośrednio. Niestety konieczność ręcznego łączenia adresów wpłynęła znacznie na długość tworzenia pracy i zwiększyła ryzyko błędu.

Na poniższych dwóch obrazkach zamieszczam kolejno wygląd okna gry z edytora, a później z pliku wykonywanego przez system Windows:

Podsumowanie

Silnik Unity daje sporo możliwości w rękach doświadczonego programisty oraz oferuje łagodne wejście w "świat" game developingu dla nowych użytkowników. Posiada wiele poradników od samego studia, ma łatwo dostępną i czytelną dokumentację, w której można znaleźć rozwiązanie problemów, dodatkowo wsparte przykładami.

Unity posiada również spore grono użytkowników, którzy chętnie udostępniają sobie wzajemnie projekty oraz pomagają sobie z kodem.

Język C# jest bardzo wygodny w zarządzaniu pamięcią, zaś jego zorientowanie w stronę obiektowości, znacznie ułatwia tworzenie projektu.

Gry zazwyczaj tworzone są przez studia lub małe zespoły projektowe, co wpływa znacznie na odbiór i efektywność rozwiązań. Jednak, dzięki ułatwieniom zawartym w silniku, proces twórczy jest możliwy nawet dla pojedynczych osób chcących próbować swoich sił w tej dziedzinie.

Potencjalny rozwój

Moja gra zawiera możliwości przyszłego rozwoju, wpływającego na nieprzewidywalność rozgrywki oraz jej balans.

W pierwotnym założeniu, zamiast systemu znajdowania przedmiotów, istniał handel, ostatecznie jednak postanowiłem zrezygnować z tej mechaniki. Istnieje jednak łatwa możliwość przywrócenia tego systemu. Do puli nagród, za pokonanie przeciwników oraz wykonanie misji, oprócz doświadczenia, doszłoby wtedy także złoto.

Można dokładać również nowe umiejętności, rozwijając klasę "AllSkillsHandler" oraz zmodyfikować znaczenie statystyk.

Istnieje również możliwość dodania elementów losowych innych niż "szansa na trafienie krytyczne" i wybór akcji przez przeciwnika, np. "szansa powodzenia ataku" lub "wskaźnik mocy", który zwiększałby efektywność przeciwko przeciwnikom danego typu. Wszystko to można bardzo łatwo umieścić, edytując jedynie klasę "BattleHandler".

Wsparcie twórców

Unity udostępnia swój silnik w kilku różnych odsłonach, a powstające na nim gry mają różne licencje. Przykładowo, jeśli twórca chce zacząć zarabiać na swojej aplikacji, może umieścić ją w "Google Store" za uiszczeniem opłaty.

Firma "Unity Technologies" znana jest z dobrego podejścia do twórców oraz możliwości komunikacji z nimi.

Bibliografia

[1] Co to jest RPG ? Marcin ‘Seji’ Segit

[2] CPRG Book Project - Sharing the History of Computer Role-Playing Games

edited by Felipe Pepe – 2018 str. 36

[3] tamże str. 46

[4] tamże str. 68

[5] tamże str. 122

[6] tamże str. 202

[7] tamże str. 262

[8] tamże str. 338

[9] tamże str. 400

[10] https://en.wikipedia.org/wiki/Pokémon_(video_game_series)

Spis i linki do poradników video

[11]https://www.youtube.com/watch?v=whzomFgjT50&list=PLjsG4BVuysGh83Zup6uT8fgpPtqnLYaGv&index=4&t=1129s

[12]https://www.youtube.com/watch?v=_1pz_ohupPs&list=PLjsG4BVuysGh83Zup6uT8fgpPtqnLYaGv&index=5&t=1224s

[13] https://www.youtube.com/watch?v=DLAIYSMYy2g&list=PLjsG4BVuysGh83Zup6uT8fgpPtqnLYaGv&index=10&t=422s

[14]https://www.youtube.com/watch?v=OG7vHstkZqM&list=PLjsG4BVuysGh83Zup6uT8fgpPtqnLYaGv&index=14&t=2s

[15]https://www.youtube.com/watch?v=sVCXgN5_XmY&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm&index=6

[16]https://www.youtube.com/watch?v=BrCxFpZFRS0&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm&index=28&t=305s

[17] https://www.youtube.com/watch?v=oH-OFyJFrC8&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm&index=31

[18]https://www.youtube.com/watch?v=d3yc8RG3Xro&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm&index=32

[19]https://www.youtube.com/watch?v=l31nnI5d6Cw&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm&index=33

[20]https://www.youtube.com/watch?v=IqjjUuX1th8&list=PLiyfvmtjWC_X6e0EYLPczO9tNCkm2dzkm&index=34

Spis grafik

[G1] https://www.kenney.nl/assets/roguelike-rpg-pack

[G2] https://www.dropbox.com/sh/ytmsiisxn2l0134/AACHH1lH4TIIGhy19RAsd6vPa?dl=0

Linki do wykorzystanych grafik:

[G3] <pirat>

https://www.google.com/search?q=pixel+art+pirate&tbm=isch&ved=2ahUKEwj379r17IbuAhWUvKQKHRj8CmYQ2-cCegQIABAA&oq=pixel+art+pirate&gs_lcp=CgNpbWcQAzIECCMQJzIECAAQEzIECAAQEzIECAAQEzIECAAQEzIECAAQEzIECAAQEzIECAAQE1CshAFYrIQBYMOGAWgAcAB4AIABX4gBX5IBATGYAQCgAQGqAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=o2_1X7eSF5T5kgWY-KuwBg&bih=970&biw=1920&client=opera&hs=Pjm#imgrc=qqpOlReIw8wxeM

[G4]<korpoJaszczur>

https://www.google.com/search?q=lizard+pixel+art&tbm=isch&ved=2ahUKEwjp5fq-lYXuAhXkMewKHXNeDr4Q2-cCegQIABAA&oq=lizard+pixel&gs_lcp=CgNpbWcQARgAMgIIADIECAAQHjIECAAQHjIGCAAQBRAeMgYIABAFEB4yBggAEAUQHjIGCAAQBRAeMgYIABAFEB46BQgAELEDOgQIIxAnOgQIABBDOgcIABCxAxBDOggIABCxAxCDAVCHqL4EWMLgvgRg7Oq-BGgAcAB4AIABogGIAfEKkgEEMC4xMpgBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=vo30X-mOOeTjsAfzvLnwCw&bih=970&biw=1920&client=opera&hs=CYX#imgrc=huMikGgmPs_zmM

[G5]<jaszczur>

https://www.google.com/search?q=lizard+pixel+art&tbm=isch&ved=2ahUKEwjp5fq-lYXuAhXkMewKHXNeDr4Q2-cCegQIABAA&oq=lizard+pixel&gs_lcp=CgNpbWcQARgAMgIIADIECAAQHjIECAAQHjIGCAAQBRAeMgYIABAFEB4yBggAEAUQHjIGCAAQBRAeMgYIABAFEB46BQgAELEDOgQIIxAnOgQIABBDOgcIABCxAxBDOggIABCxAxCDAVCHqL4EWMLgvgRg7Oq-BGgAcAB4AIABogGIAfEKkgEEMC4xMpgBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=vo30X-

mOOeTjsAfzvLnwCw&bih=970&biw=1920&client=opera&hs=CYX#imgrc=irzlm22f2cXafM

[G6]<anioł>

https://www.google.com/search?q=angel+statue+pixelart&tbm=isch&ved=2ahUKEwjInca0lYXuAhXuwAIHHT2_C-IQ2-cCegQIABAA&oq=angel+statue+pixelart&gs_lcp=CgNpbWcQA1DtigFY0p0BYLWfAWgBcAB4AIABjwGIAdIFkgEDNC4zmAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=qY30X4ilBu6Bi-gPvf6ukA4&bih=970&biw=1920&client=opera&hs=CYX#imgrc=o0S4JgfOLD4xkM

[G7]<zombie>

https://www.google.com/search?q=pixelart+zombie+png&tbm=isch&ved=2ahUKEwju2fDPk4XuAhVEgqQKHdwkAHMQ2-cCegQIABAA&oq=pixelart+zombie+png&gs_lcp=CgNpbWcQAzoGCAAQBxAeUNRuWM90YNt5aABwAHgAgAFniAGeApIBAzIuMZgBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=yYv0X67RIcSEkgXcyYCYBw&bih=970&biw=1920&client=opera&hs=CYX#imgrc=ZULWe5-GqqGHHM

[G8]<szkielet>

https://www.google.com/search?q=pixel+skeleton+png&client=opera&hs=CYX&sxsrf=ALeKk02po2tAY79Y4XstiVYHZ0ujgJ5lsQ:1609862060865&tbm=isch&source=iu&ictx=1&fir=X289UnfE0YvFcM%252CQHmWwENxLVK5mM%252C_&vet=1&usg=AI4_-kSYFntDgyQnx6afHExmFaPxdgwNjA&sa=X&ved=2ahUKEwiVtJnCk4XuAhWlyIUKHVvlAPoQ9QF6BAgUEAE&biw=1920&bih=970#imgrc=X289UnfE0YvFcM

[G9]<pies Walter>

https://www.google.com/search?q=pixel+art+walter+dog&tbm=isch&ved=2ahUKEwivobDthoXuAhXKKewKHWRIC3MQ2-cCegQIABAA&oq=pixel+art+walter+dog&gs_lcp=CgNpbWcQAzIECAAQEzIICAAQCBAeEBMyCAgAEAgQHhATOgYIABAeEBNQjztYxz9ghkFoAHAAeACAAW2IAfACkgEDMy4xmAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=ZX70X-_aLcrTsAfkkK2YBw&bih=970&biw=1920&client=opera&hs=xzB#imgrc=cLIgA9yenhkTNM

[G10]<amulety>

https://www.google.com/search?q=pixel+art+items&client=opera&hs=T0u&sxsrf=ALeKk02xuLPtWlb-g9A8a8AowZwQo_f4nQ:1609793344059&tbm=isch&source=iu&ictx=1&fir=LO3J-si90JH8OM%252CnSP0atjo_LhMGM%252C_&vet=1&usg=AI4_-kT6gTtZhQsKQu-EbRo4-n8MHPxsXA&sa=X&ved=2ahUKEwj8q7zDk4PuAhWi3eAKHeqsDXwQ9QF6BAgREAE&biw=1920&bih=970#imgrc=LO3J-si90JH8OM

[G11]<flakonik na mikstury>

https://www.google.com/search?q=bottle+pixel+art&tbm=isch&ved=2ahUKEwiutLLFvYLuAhUSpRoKHVraA9UQ2-cCegQIABAA&oq=bottle+&gs_lcp=CgNpbWcQARgAMgQIABBDMgIIADICCAAyBAgAEEMyBAgAEEMyAggAMgIIADICCAAyAggAMgIIADoICAAQsQMQgwE6BQgAELEDOgQIIxAnOgcIABCxAxBDOgQIABATUOCDIFjMnCBg57AgaARwAHgAgAGwAYgBpgiSAQMxLjiYAQCgAQGqAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=FiXzX67RJpLKatq0j6gN&bih=970&biw=1920&client=opera&hs=1e9#imgrc=uB65norbzdVHjM

[G12]<pies Cheems>

https://www.google.com/search?q=pixel+art+cheems&client=opera&hs=xDg&sxsrf=ALeKk02MTSxMW9xW7ST-ZKhimdDavcg6cA:1609180441126&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjhiMqkqPHtAhWGzoUKHa2kBkoQ_AUoAXoECA0QAw&biw=1920&bih=970#imgrc=svkCGarK-GHjSM