UNIWERSYTET KARDYNAŁA STEFANA WYSZYŃSKIEGO
W WARSZAWIE

WYDZIAŁ MATEMATYCZNO-PRZYRODNICZY
SZKOŁA NAUK ŚCISŁYCH

Michał Aleksandrowicz
99005
Informatyka

Gra komputerowa RPG w języku C++

Praca licencjacka

Promotor:

Doktor Paweł Łubniewski

WARSZAWA 2020

……
Imię i nazwisko studenta/studentki
……
Nr albumu
……..
Wydział
……
Instytut
……..
Kierunek

Dziekan Wydziału ……..

……..

OŚWIADCZENIE

Świadomy(a) odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w żadnej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Oświadczam, że poinformowano mnie o zasadach dotyczących kontroli samodzielności prac dyplomowych i zaliczeniowych. W związku z powyższym oświadczam, że wyrażam zgodę na przetwarzanie* moich prac pisemnych (w tym prac zaliczeniowych i pracy dyplomowej) powstałych w toku studiów i związanych z realizacją programu kształcenia w Uczelni, a także na przechowywanie pracy dyplomowej w celach realizowanej procedury antyplagiatowej w ogólnopolskim repozytorium pisemnych prac dyplomowych.

…….....
podpis studenta

*Przez przetwarzanie pracy rozumie się porównywanie przez system antyplagiatowy jej treści z innymi dokumentami (w celu ustalenia istnienia nieuprawnionych zapożyczeń) oraz generowanie raportu podobieństwa.

OŚWIADCZENIE

Oświadczam, że niniejsza praca napisana przez
Pana/Panią....., nr albumu ..... została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

…….....
podpis promotora

Spis Treści

Spis ilustracji

Rozdział 1. Wstęp

Gry komputerowe są ważnym elementem obecnego świata. Sięgają po nie ludzie ze wszystkich grup wiekowych a rynek gier komputerowych zyskuje na sile. Twórców oraz pomysłów jest coraz więcej. Wiele gier dość mocno się powtarza, idąc jednym schematem. Są jednak gry które odstają od głównego nurtu i mają swój unikalny charakter. Takim gatunkiem gier są gry RPG (ang. Role-playing games) [1]. Swój początek mają jeszcze przed szeroko dostępnymi komputerami w większości domów i zaczynały na kartkach papieru.

Najczęściej gry te są towarzyską rozgrywką przeznaczoną dla kilku uczestników zabawy. Gracze spotykają się twarzą w twarz. Zdarzają się "sesje gry", w których uczestnicy dodatkowo przebierają się, zgodnie z charakterem rozgrywki, tzw. cosplay [2]. Głównym założeniem gry RPG jest wcielanie się w stworzoną, fikcyjną postać oraz odgrywanie jej roli. Rozgrywka nie ma ograniczeń jeśli chodzi o miejsce akcji, przedstawioną historię oraz styl rozgrywki. Uczestnikom towarzyszy tzw. mistrz gry [3]. Jest to osoba, która opowiada uczestnikom rozgrywki, co aktualnie dzieje się w świecie gry. Jest on narratorem opowieści. Przybliża pozostałym graczom zjawiska, które zachodzą podczas rozgrywki oraz pilnuje zasad gry. Mistrzem gry może zostać każdy z uczestników, lecz jego wiedza i doświadczenie jest kluczowe w odbiorze rozgrywki przez jej uczestników. Mistrz gry często sporządza dodatkowe mapy oraz przynosi rekwizyty, by pomóc uczestnikom wczuć się w opowieść. "Często podczas rozgrywki używane są różne rodzaje kości do gry, które pomagają w bezstronnym określaniu wyników poszczególnych akcji podejmowanych przez bohaterów graczy. Wynik rzutu uwzględniany jest jako kolejny czynnik w równaniach mechaniki systemu" [1].

Tematem mojej pracy licencjackiej jest stworzenie gry komputerowej w języku programowania C++. Gra została zatytułowana jako "Death in the Console" (ang. Śmierć w konsoli) i jest z gatunku RPG. Jest ona utrzymana w stylu tekstowym, a cała treść wyświetlana jest w konsoli poleceń systemu Windows.

Moim założeniem przy tworzeniu gry było przeniesienie rozgrywki na ekrany komputerów, tak aby można było zagrać samemu, a brakujących graczy oraz mistrza gry zastąpił komputer. Aplikacja została stworzona korzystając z paradygmatu programowania ekstremalnego [4] oraz korzysta ona z bibliotek dll, łączonych dynamicznie. Dodatkowo program korzysta z wielu zasobów zewnętrznych, takich jak dźwięki czy ilustracje. Wszelkie opisy wewnątrz gry zostały przygotowane w języku angielskim. Umożliwi to graczom z całego świata zapoznanie się z prezentowaną grą. Swoją pracę podzieliłem na informacje przeznaczone dla finalnego klienta oraz implementację mechanik gry. Podział ten jest potrzebny, gdyż finalny użytkownik produktu nie musi oraz nie powinien znać wszystkich informacji przed rozpoczęciem zabawy. Wiedza ta mogłaby popsuć odbiór programu jako całości.

Rozdział 2. Opis konsumenta

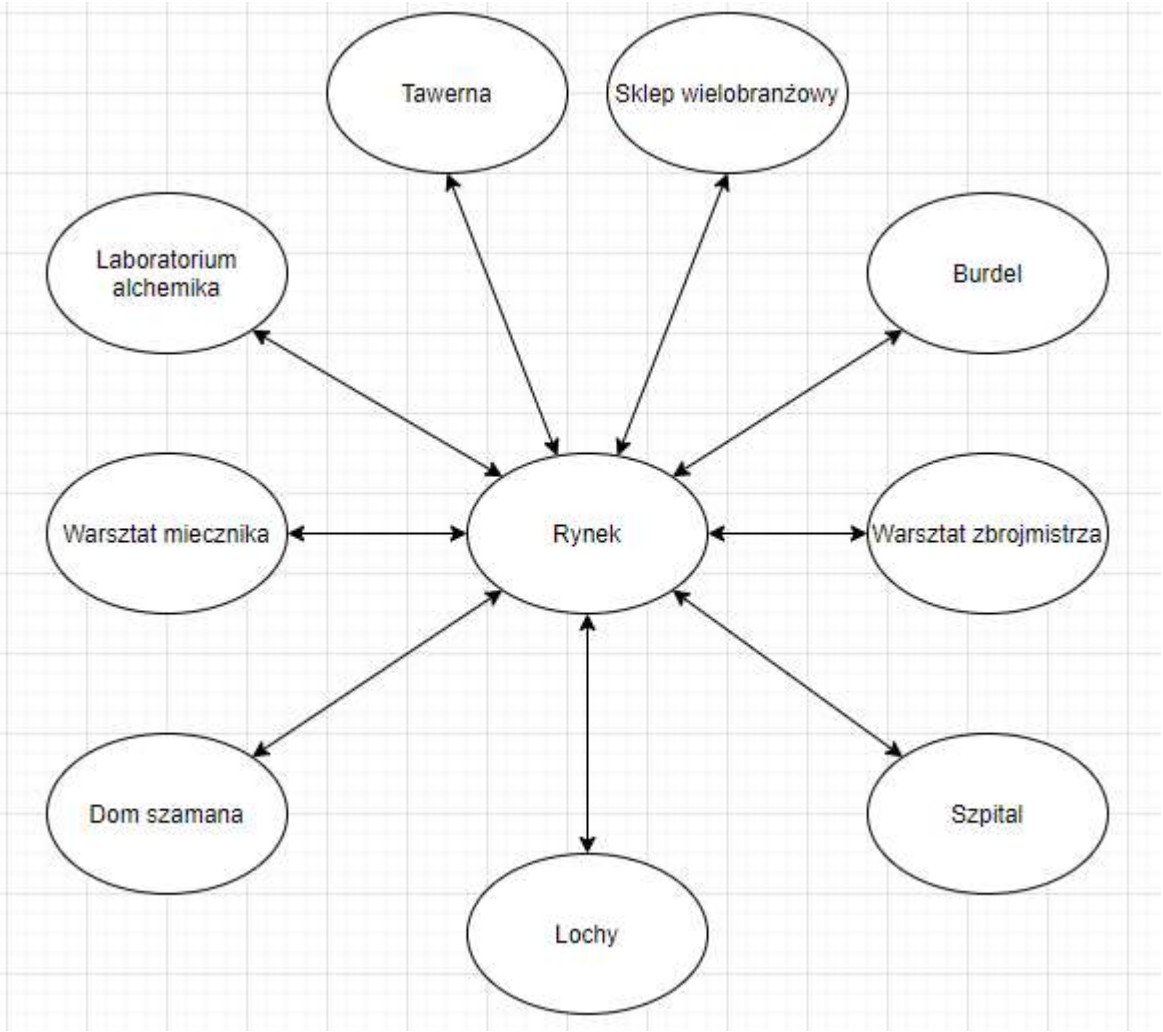
Na dobrą grę z gatunku RPG składa się wiele czynników. Wyszczególnić możemy fabułę, bohaterów oraz świat gry. Informacje te użytkownik powinien znać jeszcze przed rozpoczęciem rozgrywki. Tego typu informacje można często znaleźć na pudełku zawierającym grę lub w serwisach informacyjnych. W tym rozdziale postaram się opisać wyżej wymienione informacje, aby użytkownik rozpoczął swoją przygodę z niezbędną wiedzą na temat świata gry.

2.1. Świat gry

Rozgrywka toczy się w fikcyjnym świecie stylizowanym na świat Fantasy [5]. Świat ten łączy pewne miejsca ze świata prawdziwego z elementami czysto fikcyjnymi. Główna część gry odbywa się w mieście podzielonym na szereg lokacji. Miasto nie zostało jeszcze nazwane ze względu na brak fabuły. Lokacje w grze służą rozgrywce w unikalny dla siebie sposób. Wśród nich można wyszczególnić:

- Rynek,
- Tawerna,
- Sklep wielobranżowy,
- Warsztat zbrojmistrza,
- Warsztat miecznika,
- Laboratorium alchemika,
- Burdel,
- Szpital,
- Dom szamana,
- Lochy.

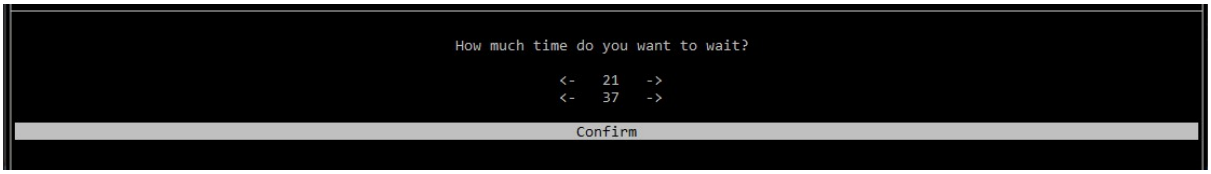
Poruszanie się pomiędzy lokacjami odbywa się za pośrednictwem rynku (Rysunek 2:1). Lokacje opisane wyżej pełnią wiele funkcji dla postaci gracza. Część z nich służy głównie handlowaniu przedmiotami. Inne mają rozwijać gracza oraz jego ekwipunek. Część z nich służy odpoczynkowi oraz rozwijaniu informacji o świecie. Jednak najważniejszą lokacją są niewątpliwie lochy, gdyż prowadzą gracza do celu gry. W lokacjach można spotkać postacie niezależne (ang. NPC) [6]. Oferują one graczowi zadania poboczne oraz wchodzą z nim w liczne interakcje pogłębiające iluzję gry. Unikalną lokacją jest rynek. Jest to lokacja przejściowa. Jednym z elementów które mają ułatwić graczowi poruszanie się po świecie gry jest mapa miasta. Wyświetlana jest ona w interfejsie tekstowym oraz podświetla ona lokację w której aktualnie znajduje się postać gracza.



Rysunek 2:1 - Schemat poruszania się po mieście

2.2. Fabuła

W aktualnej wersji gry fabuła nie została jeszcze zaimplementowana. Aby proces przyspieszyć stworzono mechanikę pod jej prezentację. Polega ona na wyświetlaniu tekstu przekazanego przez programistę znak po znaku z opóźnieniem 50ms pomiędzy znakami. Opóźnienie można łatwo modyfikować według potrzeb. Taki sposób prezentacji imituje ludzką mowę. Postać gracza nie będzie odzywać się bezpośrednio. Jego narracja ograniczy się do wybierania opcji z listy dialogowej. Pod wypowiedzi będzie można podłożyć dźwięk aby pogłębić wrażenia z gry.



Rysunek 2:2 - Okno wyboru po wywołaniu funkcji czekania

2.3. Cel gry

Gra posiada wiele celów. Najważniejszym jest wcielanie się gracza w swoją postać oraz kierowanie jej poczynaniami. Gracz będzie rozwijał swoją postać wedle swojego stylu rozgrywki. Kolejnym celem gry prowadzącym do jej zakończenia jest eksploracja lochów. Wejście do lochów znajduje się od strony rynku miejskiego. Przewidziano dziewięć poziomów oraz jeden dodatkowy na finalną walkę. Przechodząc przez kolejne z nich, gracz będzie zdobywał przedmioty, walczył z coraz groźniejszymi przeciwnikami, zdobywał punkty doświadczenia oraz wykonywał zadania.

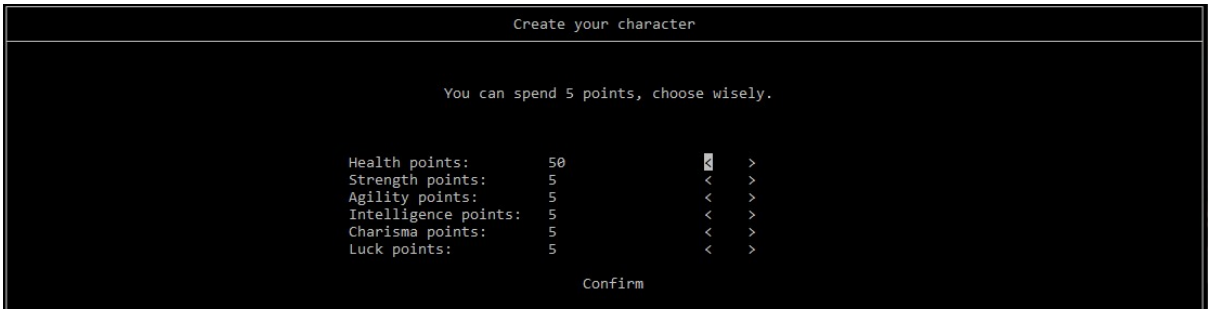
2.4. Upływ czasu

Częstym wymaganiem graczy od gier komputerowych jest poczucie odrealnienia. Jednak część z nich nie może wyobrazić sobie gry komputerowej bez niektórych zjawisk zachodzących w przyrodzie. Jednym z nich jest upływ czasu.

W grze zaimplementowano system dobowy wraz z zegarkiem dla głównej postaci. Godzina określa wiele aspektów rozgrywki. Wśród nich możemy wyróżnić:

- dostęp do niektórych lokacji w mieście,
- wykonywanie zadań,
- statystyki głównej postaci.

Przykładowo po godzinie 20:00, według czasu gry, część z lokacji w mieście będzie zamknięta. Jeżeli jednak postać gracza znajduje się w środku jednej z nich, przykładowo: sklepu lub warsztatu, zostanie ona z niego wyproszona. Lokacja będzie ponownie dostępna od godziny 6:00 rano czasu gry. Ma to symulować cykl dobowy postaci niezależnych które znajdują się w środku i zarządzają danym miejscem. Z najbardziej oczywistych mechanik związanych z upływem czasu, najczęściej doświadczymy uczucia głodu głównej postaci. Będzie ona tracić jeden punkt najedzenia co dwie godziny czasu gry. Dlatego potrzebna jest pamięć o tym zjawisku. Niezaspokojenie głodu może prowadzić do pogorszenia niektórych statystyk oraz w skrajnych przypadkach może uśmiercić postać gracza prowadząc do przedwczesnego zakończenia rozgrywki. Ostatnią z najważniejszych korzyści z wprowadzenia systemu dobowego jest połączenie go z systemem premii oraz kar dla postaci gracza [2.5.3, str. 11]. Aby przyspieszyć upływ czasu, postać gracza może pójść spać lub przeczekać (Rysunek 2:2).



Rysunek 2:3 - Menu tworzenia postaci wywoływane po rozpoczęciu nowej gry

Warto pamiętać, że podczas takich aktywności poziom najedzenia głównej postaci będzie się zmniejszał. Jeżeli gracz będzie chciał czekać na tyle długo, że jego postać odniesie obrażenia spowodowane głodem, zostanie wyświetlony komunikat na ekranie. W grze zaimplementowano również licznik dni od rozpoczęcia rozgrywki, ale nie pełni on aktualnie żadnej funkcji.

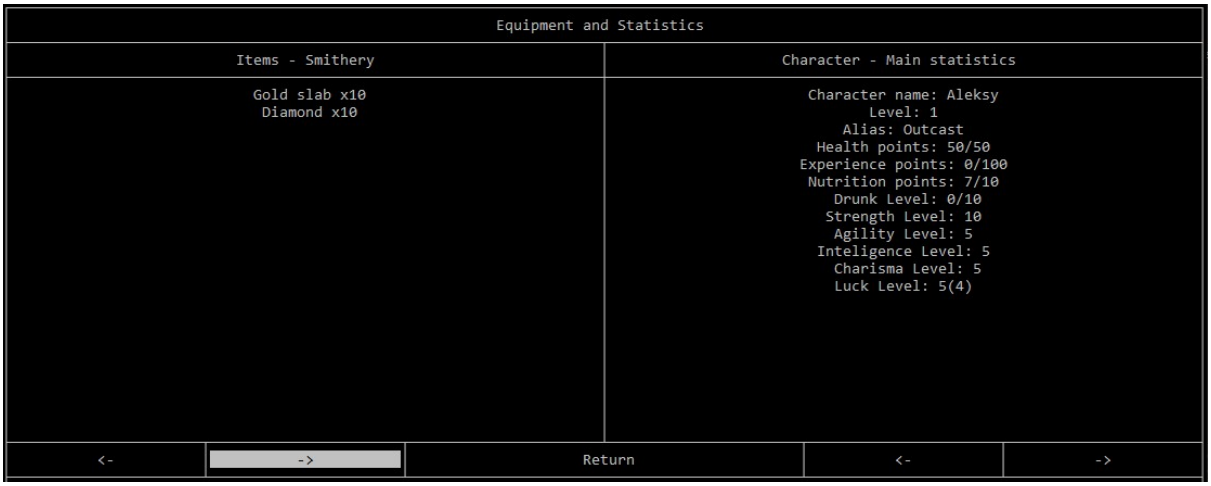
2.5. Postać gracza

Opierając się na założeniach gier RPG, gracz wciela się w postać i odgrywa jej rolę. Dlatego na samym początku rozgrywki tworzony jest obiekt z postacią gracza, a użytkownik konfiguruje jej najważniejsze parametry. Proces ten odbywa się za pośrednictwem specjalnego menu w którym konfiguruje się najważniejsze parametry postaci (Rysunek 2:3).

2.5.1. Statystyki

Systemy wewnątrz gry opierają się głównie na statystykach postaci, dlatego gracz dostał możliwość wstępnej konfiguracji postaci pod swój styl rozgrywki. Do tego celu wykorzystuje on 5 początkowych punktów umiejętności. Należy rozdzielić wszystkie punkty przed rozpoczęciem rozgrywki. Postać gracza posiada następujące statystyki:

- Punkty życia,
- Punkty doświadczenia,
- Punkty najedzenia,
- Punkty upojenia alkoholowego,
- Poziom,
- Tytuł,
- Ilość złota,
- Siła,
- Zręczność,
- Inteligencja,
- Charyzma,
- Szczęście.



Rysunek 2:4 - Edytor ekwipunku

Podczas rozgrywki podane wyżej statystyki będą wzmacniane za sprawą postępu. Każda ze statystyk ma swoją wartość bazową i nie może zostać stale obniżona działaniami gracza. Każda ze statystyk wpływa na rozgrywkę w innym jej aspekcie. Dodatkowo kilka z nich może zostać czasowo zmienionych za pomocą systemu premii oraz kar [2.5.3, str. 11]. Część parametrów związanych z interakcjami postaci gracza w świecie gry, na przykład: znaczniki zadań pobocznych lub zegarek, zostały zapisane w obiekcie postaci gracza dla łatwiejszego manipulowania nimi.

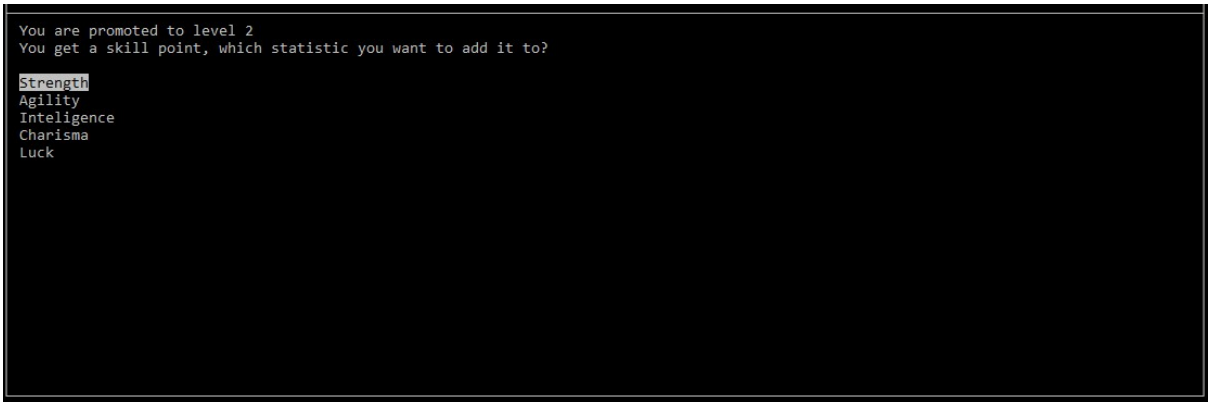
Postać posiada również wyposażenie, które pomoże mu w walce z przeciwnikami. Do tego celu gracz może wykorzystać jedną z sześciu broni dostępnych w świecie gry. Postać gracza posiada również oddzielne miejsca dedykowane posiadanej zbroi. Każda z jej części, jak hełm, napierśnik, rękawice, spodnie oraz buty, traktowana jest osobno i przyszłej wersji gry wpływać będzie na otrzymywane obrażenia.

2.5.2. Ekwipunek

Postać gracza może podczas gry wchodzić w interakcję z różnymi przedmiotami. Część z nich można przechowywać. Jeżeli przedmiot nie posiada swojego dedykowanego miejsca, jakim jest przykładowo: miejsce na broń, zostanie przeniesiony do plecaka. Plecak postaci posiada pewne ograniczenia, które zbliżają rozgrywkę do odrobiny autentyczności i dlatego jest podzielony na 3 kategorie przedmiotów:

- Użytkowe,
- Rzemieślnicze – alchemiczne,
- Rzemieślnicze – kowalskie.

Większość przedmiotów w grze jest jednorazowego użycia. Każda z kategorii plecaka może pomieścić 20 sztuk różnych przedmiotów. Aby odrobinę uprościć model ekwipunku, jeżeli gracz posiada dwa takie same przedmioty, są one umieszczone na jednym miejscu plecaka. Przy każdej manipulacji plecakiem gracza, zostanie wywołana funkcja która posortuje jedną z trzech części plecaka. Cały ekwipunek jaki posiada postać gracza można



Rysunek 2:5 - Menu awansu na nowy poziom

oglądać w edytorze ekwipunku (Rysunek 2:4). Za jego pomocą można uzyskać informacje o danym przedmiocie, użyć go lub wyrzucić.

Aktualnie w grze znajduje się kilka przedmiotów testowych, które mogą zostać zmienione lub usunięte w późniejszych wersjach gry. Używanie przedmiotów nie jest możliwe w każdej sytuacji. Każdy z przedmiotów może posiadać ograniczenia, przykładowo: w miejscu jego użycia. Wartość przedmiotów nie jest wyświetlana w edytorze ekwipunku, więc gracz sam będzie musiał nauczyć się, które przedmioty warto zbierać, a których unikać.

2.5.3. Premie oraz kary

Korzystając z faktu, iż postać gracza posiada liczne statystyki, pojawiła się potrzeba by nagrodzić lub ukarać gracza za szczególne działania. Dlatego w grze zaimplementowano system kar oraz premii dla poszczególnych statystyk gracza.

Polega on na zwiększeniu, bądź zmniejszeniu ilości punktów poszczególnej statystyki. Efekt ten jest nadawany czasowo. Z każdą minutą która upłynęła, licznik efektu zostaje zmniejszony, aż efekt całkowicie zaniknie. Może on zostać wzmocniony lub zniwelowany. System ten ma na celu zasugerowanie niektórych działań oraz pokazanie synergii pomiędzy graczem a światem gry. Postaci niezależne które gracz spotyka na swojej drodze, mogą nałożyć pozytywne efekty lub unieważnić negatywne za opłatą. Po upływie konkretnego czasu dodatkowe efekty zostają zdjęte, a statystyki postaci gracza powracają do stanu sprzed nadania efektów.

2.5.4. Rozwój postaci

Podczas rozgrywki postać gracza zdobywa punkty doświadczenia. Ich różna ilość może być przyznawana za dowolną aktywność w świecie gry. Punkty te przyczyniają się do rozwoju postaci poprzez zwiększenie jej poziomu. Część przedmiotów, obszarów oraz mechanik gry może być w ten sposób blokowana, aby gracz odkrywał je z czasem. Gdy ilość punktów doświadczenia będzie równa lub większa limitowi nowego poziomu, gracz awansuje, zyskując tym sposobem punkt umiejętności. Punkt ten trzeba rozdysponować na jedną z kilku dostępnych statystyk, celem jej ulepszenia (Rysunek 2:5).

W aktualnym stanie gry jedynym limitem dla poszczególnych statystyk jest limit typu Int. Podczas zdobywania nowych poziomów postać gracza otrzymuje również tytuł, który będzie miał wpływ na interakcje z postaciami niezależnymi. Może on być pozytywny jak i negatywny. Dodatkowo wszystkie negatywne statystyki gracza, takie jak kary do poszczególnych statystyk, poziom głodu oraz upojenia alkoholowego, zostają zresetowane do wartości domyślnych.

2.6. Opinie innych graczy

Gra została nieodpłatnie udostępniona kilku osobom z najbliższego środowiska. Zostali oni poproszeni o przekazanie krótkiej opinii na temat gry.

„Jako osoba, która uwielbia gry niezależne, mogę śmiało powiedzieć ,że "Death in the console" wykorzystuje schemat na grę, który od dawna nie był eksploatowany. W niektórych aspektach zdecydowanie potrzebuje polerowania, ale to da się nadrobić odrobiną ciężkiej pracy. Uważam również ,że jak na grę stworzoną przez jedną osobę, zawiera sporo treści oraz wielu możliwości do rozbudowy. Interfejs użytkownika wygląda dość informacyjnie, jak na grę konsolową, ale brakowało mi możliwości zmiany rozdzielczości ekranu. Muszę wspomnieć, że gra zawiera system statystyk. Myślę, że to świetny dodatek do gier indie z elementami RPG. Uważam, że ten komponent pozwala ci stworzyć postać tak, jak tego chcesz, jak lubisz grać. Nie musisz korzystać z tego co zdefiniował za ciebie deweloper. W niektórych momentach brakowało komunikacji. Jako przykład podam początek gry. Wybierając umiejętności lub zadania z menu, brakuje jakichś podstawowych informacji na ich temat.” ~ Konstantin

„Bardzo zainteresowała mnie ta produkcja. Jest to gra w stylu tekstowym co jest w dzisiejszych czasach niezwykle rzadkie, mocno RPG z wieloma elementami charakterystycznymi dla tego gatunku. Starannie napisana z możliwościami eksplorowania wielu lokacji i interakcji w nich. Bardzo podobają mi się rozbudowane statystyki, gdzie nie brakuje np. poziomu najedzenia i upojenia alkoholowego. Gdybym miał coś poprawić, to tylko rozbudować ten projekt jeszcze bardziej, ponieważ uważam, że ma duży potencjał.” ~ Marek

„Utworzona w języku C++ grą komputerowa posiada sporo pozytywnych cech. Przede wszystkim, gra jest dopracowana pod względem szczegółów. Po sprawdzeniu wszystkich możliwości wykonania akcji, dostajemy jakąś informacje zwrotną. Rzadko zdarza się sytuacja, gdzie np. po rozmowie z jakąś postacią, aplikacja przestaje działać. Duże wrażenie sprawiły na mnie kreatywne rozwiązania w aplikacji. Możliwość przejścia lochów z funkcją odsłaniania mapy analogicznie jak w grze Heroes of Might and Magic, symulator walki z bandytami podobny do zaprezentowanego w serii gier Pokemon czy też interakcje z innymi postaciami, które przypominają dialogi w grach serii Diablo. Te zabiegi stosowane są w wielu znanych, profesjonalnych grach, które podbiły serca wielu milionów graczy. Interfejs akcji w grze jest starannie dopracowany, gdzie przeanalizowano dużo rodzajów akcji. Jako, że cała gra wyświetla się w konsoli, użytkownik potrzebuje kilku minut z oswojeniem się z interfejsem, który też jest wyświetlony od razu po uruchomieniu aplikacji. Warto byłoby

dokonać aktualizacji gry o osobne wyświetlanie menu głównego i menu akcji, jakie może dokonać użytkownik. Najlepiej, aby to się działo po naciśnięciu któregoś z przycisków. Po przetestowaniu aplikacji uważam, że ma ona potencjał na zostanie interesującą grą komputerową. Z zaciekawieniem czekam jak wprowadzone zostaną zadania, które stworzą fabularną przygodę." ~ Marcin

2.7. Wymagania sprzętowe

Prawie każda gra, aby została uruchomiona, musi spełnić wymagania sprzętowe. Jest to lista wymagań dla zestawu komputerowego, aby można było uruchomić aplikację [7].

System operacyjny:	Windows 7 lub wyższy
Procesor:	Intel i3 (dowolny) lub wyższy
Pamięć RAM:	512MB
Karta graficzna:	dowolna posiadająca co najmniej 512MB pamięci VRAM
Łącze internetowe.	
Klawiatura.	

Zalecane jest wykorzystanie monitora z możliwie jak najwyższym odświeżaniem np. 144HZ.

2.8. Uruchamianie

Aby uruchomić grę należy wypakować całe archiwum z grą w dowolnym miejscu i podwójnie nacisnąć na plik Gerka.exe. Foldery txt, wav, dll z zawartością oraz plik pdcurses.dll również są potrzebne. Program można również skompilować samemu, korzystając z kodu źródłowego zamieszczonego w systemie kontroli wersji Github [8]. Do tego celu należy skonfigurować projekty pod wykorzystanie bibliotek dll, PDCurses, oraz Winmm. Projekt powinien zostać skompilowany jako 32 bit.

2.9. Sterowanie

Gra obsługiwana jest za pomocą klawiatury. Mysz nie jest obowiązkowym akcesorium. Większość akcji wykonuje się za pomocą klawiszy strzałek oraz zatwierdza przyciskiem ENTER. W przypadku pól tekstowych gra przyjmuje dowolny klawisz wciskany przez gracza.

2.10. Podsumowanie

W rym rozdziale przybliżyłem użytkownikowi końcowemu najważniejsze informacje związane z grą. Będzie on w stanie komfortowo rozpocząć rozgrywkę i poznawać systemy zarządzające grą.

Rozdział 3. Implementacja

W tym rozdziale chciałbym skupić się na przekazaniu wiedzy odnośnie najważniejszych mechanik zaimplementowanych w grze.

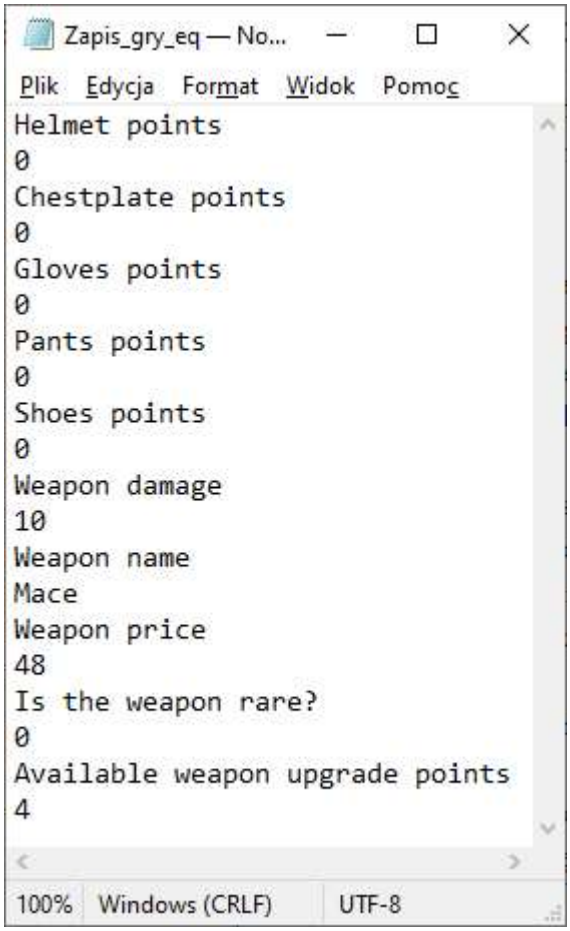
3.1. Zapis gry

Z biegiem czasu, gry stawały się coraz dłuższe i dłuższe, a ilość wolnego czasu przeznaczanego na grę często się nie zmieniała. Gracze lubią czuć postęp w grze oraz nie chcieliby go tracić w momencie końca czasu przeznaczonego na rozgrywkę. Dlatego w grach zaczęto implementować możliwość na zapisanie aktualnego stanu gry i wczytanie go w dowolnym momencie.

W mojej grze również znajduje się podobny system. Jest on oparty na plikach tekstowych znajdujących się w katalogu z grą. Gracz ma możliwość zapisania gry jeżeli znajduje się na rynku. Operacja zapisu nie jest obciążona żadnymi kosztami z punktu widzenia ekonomii gry. Zapis dzieli się na kilka plików tekstowych. Każdy z nich dotyczy innej kategorii danych. Pliki zapisu dzielimy na:

- Podstawowe statystyki postaci gracza,
- Plecak postaci gracza,
- Ekwipunek postaci gracza,
- System premii oraz kar,
- Postacie niezależne,
- Skrzynia na przedmioty.

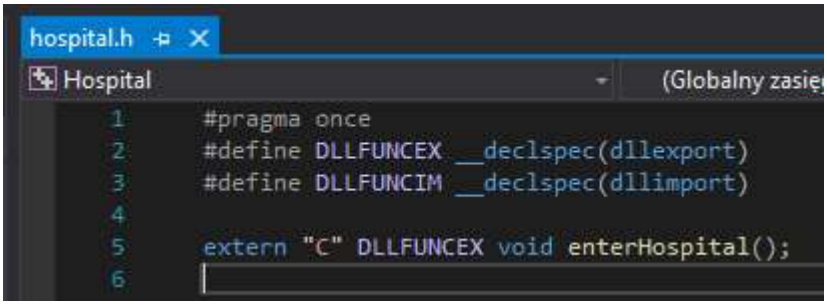
Pliki zapisu posiadają specyficzną strukturę (Rysunek 3:1). Każdy zapisany parametr posiada swój opis. Jest on potrzebny w sytuacji ręcznej manipulacji zapisem gry.



Rysunek 3:1 - Struktura pliku z zapisem gry

Przy operacji zapisu usuwane są również stare pliki z zapisami, znajdujące się w katalogu z grą. Ma to na celu wyeliminować wszystkie problemy ze zgodnością. Zapisy ze starszych wersji gry mogą powodować błędy w nowej wersji aplikacji, dlatego zalecane jest, rozpoczynanie rozgrywki od początku pomiędzy kolejnymi wersjami gry. Gracz będzie również mógł wczytać stan rozgrywki. Dokonać tego można wybierając odpowiednią opcję z menu głównym gry (Rysunek 3:2). Przy starcie aplikacji tworzone są obiekty gracza, postaci pobocznych oraz skrzyni. Posiadają one domyślne statystyki wynikające z konstruktorów danych obiektów.

Rysunek 3:2 - Menu główne gry z podświetloną opcją służącą do wczytania stanu gry



Rysunek 3:3 - Plik nagłówkowy jednej z bibliotek dll

Następnie wywoływana jest funkcja odpowiadająca za wczytanie stanu rozgrywki. Funkcja sprawdza czy pliki z zapisem znajdują się w katalogu z grą. W przypadku braku pliku, statystyki nie zostaną nadpisane. Jednak jeżeli funkcja znajdzie plik z zapisem gry, otwiera go oraz będzie wczytywać kolejne jego linie. Co druga linia zostanie przypisana do konkretnej statystyki. Linie zawierające opis zostają pominięte. W momencie kiedy funkcja napotka koniec pliku, zamyka go i przechodzi do sprawdzenia kolejnego pliku. Po zakończeniu wczytywania wszystkich plików, gra przenosi wszystkie obiekty na rynek.

3.2. Biblioteki DLL

Podczas tworzenia aplikacji zaistniała potrzeba wprowadzenia dodatkowej zawartości. W mojej aplikacji zaimplementowano mechanizm obsługi plików dll, txt oraz wav. Pliki te wykonują w programie wiele funkcji. Przechowują najważniejsze informacje oraz wpływają na odbiór gry przez użytkownika końcowego.

Najważniejszym z rodzajów plików są pliki z rozszerzeniem dll. Skrót DLL [9] oznacza biblioteki łączone dynamicznie. Oznacza to, że kod zawarty w pliku .dll jest łączony z aplikacją, podczas jej działania. Aplikacja, która korzysta z plików .dll będzie łatwo rozszerzalna o dodatkową zawartość. Wynika to z faktu ,że w przypadku nowej wersji biblioteki należy ją tylko podmienić, bez konieczności całkowitego przebudowania aplikacji. Rozwiązanie to sprawdza się przy dużych projektach.

Lokacje w grze zostały podzielone na osobne projekty a każda z nich została zbudowana jako osobna biblioteka DLL. Biblioteki korzystają z części kodu głównej aplikacji więc należy skonfigurować projekt by to umożliwić. Można to zrobić we właściwościach projektu. Najważniejszymi elementami do skonfigurowania są:

- Prekompilowane nagłówki,
- Dodatkowe katalogi plików nagłówkowych,
- Dodatkowe katalogi biblioteki,
- Dodatkowe zależności.

W pliku nagłówkowym dla biblioteki .dll zadeklarowano 2 makra do atrybutów __declspec (Rysunek 3:3). Umożliwiają one import oraz eksport danych pomiędzy biblioteką DLL, a aplikacją docelową. Funkcje oznaczone są słowem extern, co oznacza że mają powiązania zewnętrzne.

```
typedef void(*function)();
HINSTANCE libraryHandler = LoadLibrary("./dll/Brothel.dll");
if (libraryHandler)
{
    function pointer = (function)GetProcAddress(libraryHandler, "enterBrothel");
    if (pointer)
    {
        savePlayerTemp(gracz);
        pointer();
        loadPlayerTemp(gracz);
    }
    else
    {
        vector <string> message = { "This location is unavailable, you have to buy a expansion." };
        tabSubmenuTextOnly(23, 32, message);
    }
    FreeLibrary(libraryHandler);
}
else
{
    vector <string> message = { "This location is unavailable, you have to buy a expansion." };
    tabSubmenuTextOnly(23, 32, message);
}
```

Rysunek 3:4 - Struktura dynamicznego linkowania bibliotek dll

W projekcie z aplikacją docelowa dodano odwołania do projektów bibliotek, aby przy kompilacji projektu startowego, każdy z pozostałych, połączonych projektów został skompilowany. Zapobiegnie to przypadkowi, w którym programista zapomni skompilować np. bibliotekę przed użyciem jej w programie docelowym. Łączenie biblioteki dynamicznie w aplikacji przedstawiono na rysunku (Rysunek 3:4).

Aby połączyć bibliotekę w sposób dynamiczny z aplikacją potrzebne jest określenie typu zwracanego przez bibliotekę. W przypadku zaprezentowanym powyżej, typem zwracanym przez funkcję jest void, a funkcja jest typem wskaźnikowym bez argumentów. W kolejnym kroku tworzymy uchwyt do instancji. Tą instancją będzie wczytana biblioteka DLL. Musimy podać ścieżkę do biblioteki. Dobrą praktyką jest podawanie ścieżki do pliku w sposób względny. Jeżeli biblioteka nie zostanie znaleziona, użytkownik otrzyma komunikat o braku zasobów. Następnie tworzymy wskaźnik typu zdefiniowanego powyżej. Do utworzonego wskaźnika zostanie przypisany adres funkcji którą wpiszemy jako argument. Ważne jest sprawdzenie poprawności wypełnienia wskaźnika. W przypadku gdy wskaźnik jest pusty, użytkownik również otrzyma komunikat o braku zasobów. W programie stworzono funkcję która automatycznie skopiuje najnowsze wersje bibliotek do katalogu w głównym folderze aplikacji. Jeżeli wszystko się powiodło, będziemy mogli użyć naszej funkcji poprzez wskaźnik. Po zakończeniu działania funkcji z biblioteki DLL, należy zwolnić po niej pamięć. Zrobimy to funkcją FreeLibrary. Aby przekazać obiekt gracza oraz postaci niezależnych pomiędzy biblioteką a aplikacją docelową, stworzono funkcje zapisujące i wczytujące ich parametry z plików tekstowych.

W środku biblioteki tworzone są obiekty postaci gracza oraz postaci niezależnych. Nadpisywane są również ich parametry korzystając z wcześniej utworzonych plików tekstowych. Aby uniknąć zmodyfikowania wartości przez użytkownika, nowo utworzone pliki tekstowe zostaną automatycznie usunięte zaraz po ich odczytaniu. W momencie

zakończenia działania biblioteki, program zapisuje parametry postaci gracza oraz postaci niezależnych, by odczytać je w aplikacji docelowej.

3.3. Pliki dźwiękowe

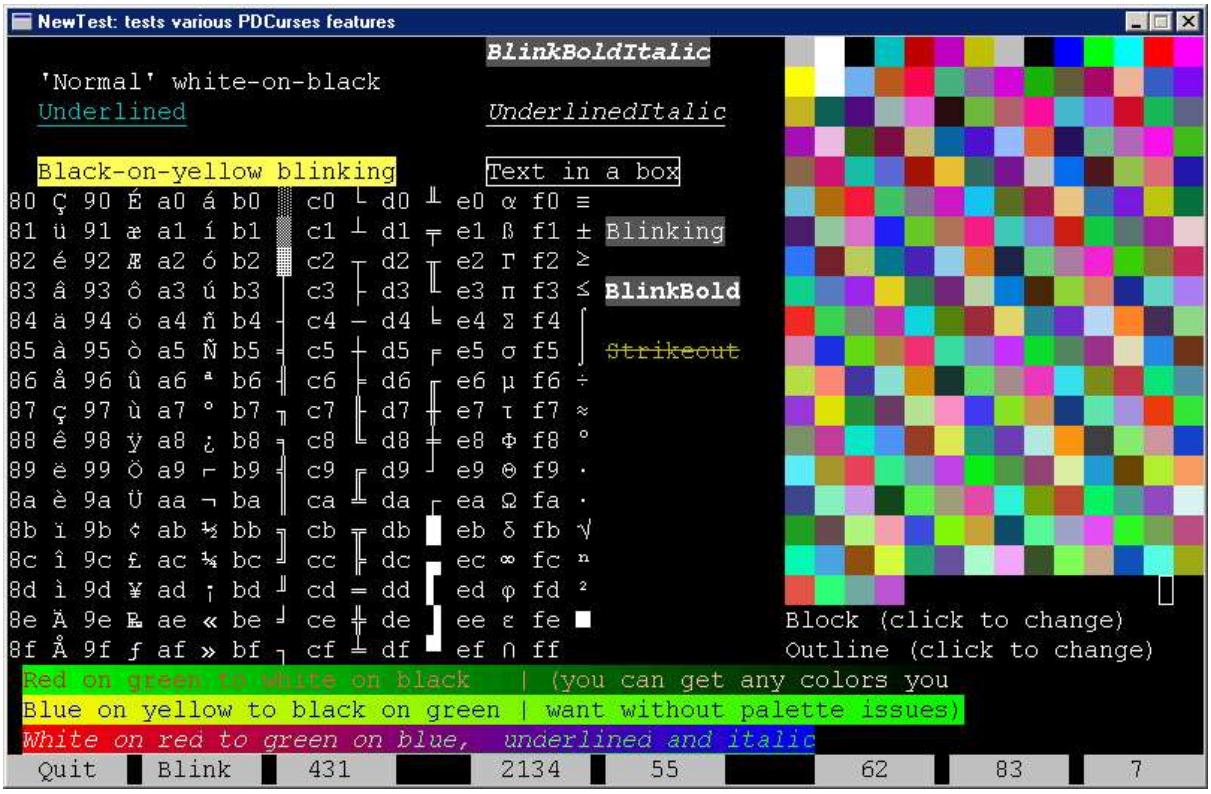
W grze znalazł się mechanizm odtwarzania plików dźwiękowych. Jego funkcjonowanie opiera się o wykorzystanie biblioteki systemu Windows o nazwie Winmm [10]. Pozwala ona na odtwarzanie dźwięków w postaci plików WAV. Biblioteka dołączana jest do projektu w jego właściwościach. Sygnały dźwiękowe są wykorzystywane w grze na podkreślenie konkretnej akcji, przykładowo: rozpoczęcie walki. Pliki z dźwiękami znajdują się w katalogu z grą i zajmują większość objętości całej gry, co jest ich niewątpliwym minusem. Sygnały dźwiękowe wykorzystywane są w całym kodzie aplikacji, dlatego wywołanie funkcji z nimi związanych zostało ustawione na samym początku programu. Dźwięki wywołujemy komendą PlaySound, przekazując ścieżkę do pliku oraz parametr jego odtwarzania. Do najważniejszych parametrów używanych w programie zaliczamy:

- SND_SYNC – działanie programu zostanie wstrzymane do momentu zakończenia odtwarzania dźwięku.
- SND_ASYNC – dźwięk będzie odtwarzany równolegle z działaniem programem.

3.4. Interfejs tekstowy

Celem interfejsu graficznego jest komunikacja użytkownika z aplikacją. Komputer wyświetla dane oraz informacje na ekranie w zrozumiały dla użytkownika sposób. Może on również oczekiwać na odpowiedź użytkownika. Interfejsy tekstowe [11] zostały w dzisiejszych czasach wyparte przez interfejsy graficzne z uwagi na ich trudność obsługi oraz archaiczny wygląd.

Aplikacja korzysta z biblioteki PDCurses [12], która jest wykorzystywana do rysowania interfejsu tekstowego oraz zbierania informacji zwrotnej od gracza. Owa biblioteka działa w trybie 32 bitowym, dlatego aplikacja docelowa oraz biblioteki muszą zostać zbudowane jako 32 bitowe. Biblioteka umożliwia tworzenie okien, rozwijanych list i pozwala korzystać z wielu znaków specjalnych oraz stylów tekstu. Umożliwia również wykorzystanie wielu kolorów wyświetlania tekstu na ekranie konsoli (Rysunek 3:5). Programista może tworzyć kombinacje kolorów do wykorzystania w całym programie. Większość obiektów można umieścić w konsoli poleceń na zasadzie podobnej do układu współrzędnych. Dzięki takiemu rozwiązaniu można pracować nad każdym obiektem, np. oknem osobno, a w końcowej fazie ustawić je wedle uznania w obszarze konsoli. Dużą zaletą biblioteki PDCurses jest to, iż każde okno które jest wyświetlane w konsoli poleceń może zostać odświeżone osobno. Umożliwia to aktualizacje tylko konkretnych części wyświetlanego obrazu, bez potrzeby wymazywania całego ekranu konsoli.



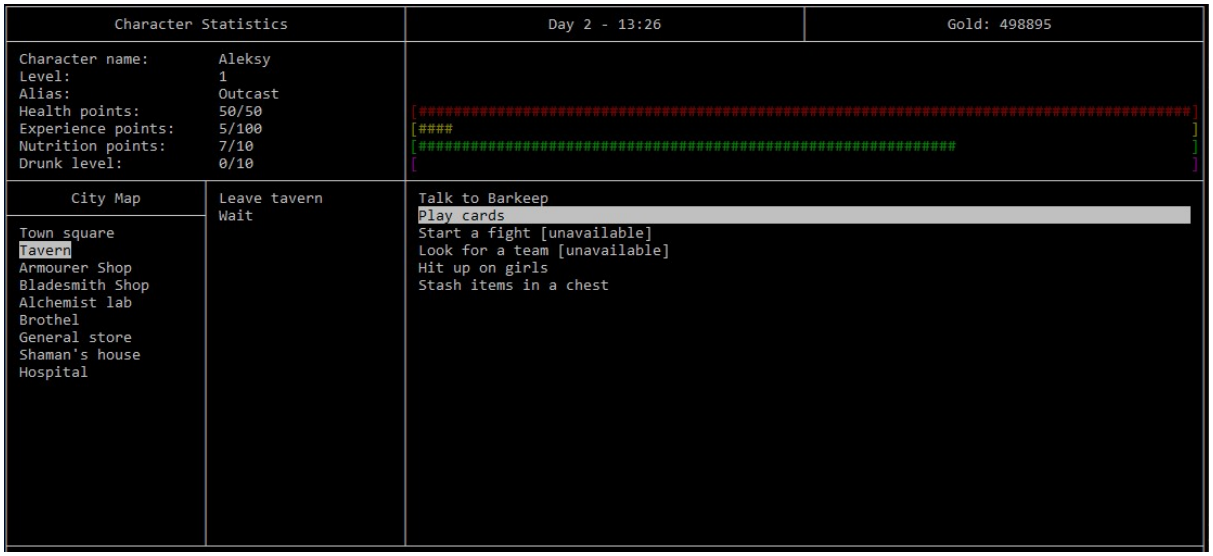
Rysunek 3:5 - Pokaz umiejętności biblioteki PDCurses [13]

W stworzonej aplikacji powstało kilka rodzajów interfejsów tekstowych. Wykorzystywane są one w całym programie. Każdy z nich ma inne zadanie oraz przyjmuje inne dane, przez co nie jest zastępowalny. Najczęstszymi danymi przekazywanymi do funkcji są tablice, bądź kontenery vector [14]. Ich główną zasadą działania jest wyświetlanie tekstu na ekranie oraz oczekiwanie na wciśnięcie klawisza przez użytkownika aplikacji.

Stworzono również kilka podinterfejsów (Rysunek 3:6), które dopełniają komunikację z graczem. Są to proste wyświetlające na ekranie tekst oraz pozwalające wybrać graczowi opcję z krótkiej listy wyboru. Na samym początku działania funkcji, program pobiera maksymalne rozmiary konsoli poleceń oraz tworzy kilka okien, bazując na wcześniejszych pomiarach. Ustawia je w konkretnych miejscach konsoli oraz przystępuje do wypełniania ich treścią. Dla każdego z okienek tworzona jest ramka na krawędziach okna. Nada to odrobiny estetyki oraz oddzieli treść od siebie. Dzięki temu użytkownik programu będzie widział jasny podział na konkretne części interfejsu tekstowego. Konkretne części tekstu mogą zostać podświetlone, by wyróżniały się spośród pozostałego tekstu. W mojej grze służą one określeniu funkcji, która jest w tym momencie aktywna (Rysunek 3:7).

```
string return_progress_bar(int min, int max, int how_long);
void findANewHighlight(int& highlight, string left_side[20], string right_side[20]);
int tab(player gracz, int &highlight, string local, string shorty[20], string menu[20], __int64 ceny[20]);
int tabItemsLeftOnly(int highlight, string top_side, string left_side[21], string right_side[21], vector <string> bottom_side);
int tabItemsLeftAndRight(int highlight, string top_side, string left_side[21], string right_side[21], vector <string> bottom_side);
void tabSubmenuTextOnly(int height, int startPoint, vector<string> myDisplay);
void tabSubmenuFancyTextOnly(int height, int startPoint, vector<string> myDisplay, int delay);
int tabSubmenuOneColumnChoice(int height, int startPoint, vector<string> message, vector<string> options);
string tabSubmenuInputField(int height, int startPoint, string message);
int tabTrade(int highlight, string top_side[3], string left_side[21], string right_side[21], vector <string> bottom_side);
int tabDungeon(player gracz, bool &mode, string local, string shorty[20], char tab[32][114], vector <char> levelMonstersSymbols);
int tabFight(player gracz, string mobName, vector <string> monsterInfo, int& highlight, string playerInfo[18], string shortcuts[20], string actions[20]);
```

Rysunek 3:6 - Lista interfejsów tekstowych w programie razem z ich argumentami



Rysunek 3:7 - Interfejs tekstowy po narysowaniu go na ekranie komputera

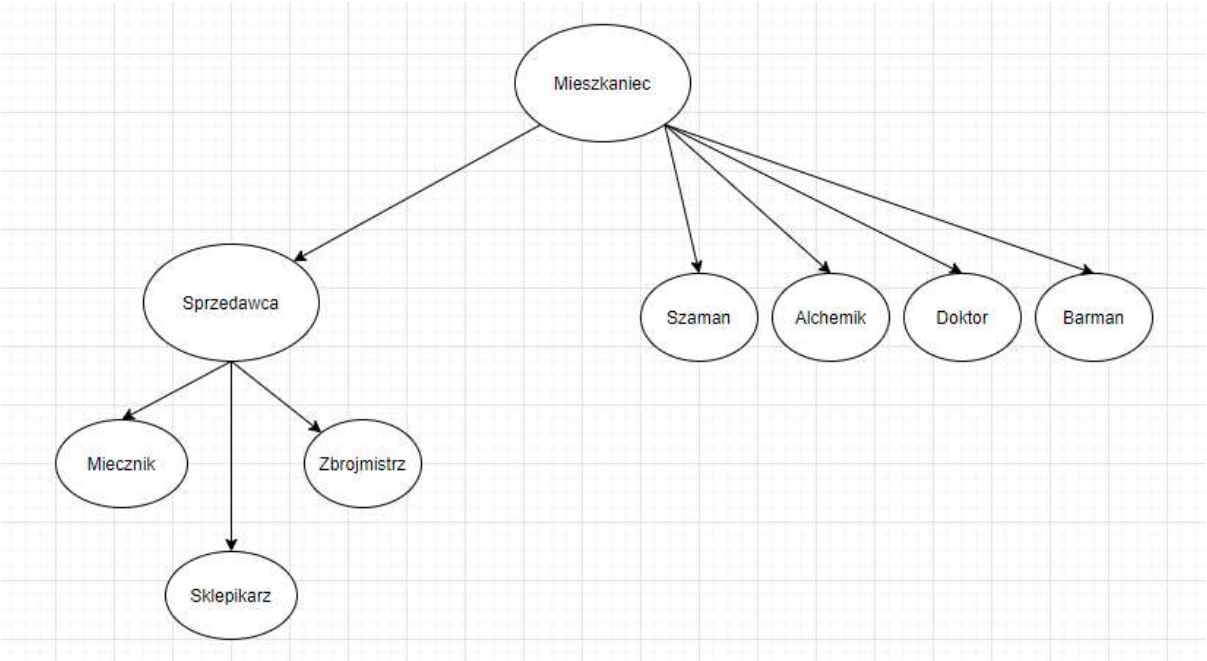
Po wypisaniu wszystkich informacji nastąpi odświeżenie wszystkich okien. W tym momencie interfejs tekstowy będzie oczekiwał na wciśnięcie klawisza z klawiatury. Zależnie od wykorzystanego interfejsu, może on zakończyć działanie lub rozpocząć jeszcze raz cykl działania funkcji w oczekiwaniu na specjalną akcję gracza, np. wciśnięcie klawisza ENTER. Często może zdarzyć się sytuacja gdy prezentowana lista funkcji nie jest pełna. Interfejs tekstowy nie jest skalowany, aby nie powodować zbędnego migania ekranu przy kolejnych cyklach rysowania. Aplikacja zlicza ile opcji znajduje się na liście funkcji oraz podświetla tylko te które są na niej obecne. Puste obszary będą pomijane.

3.5. Postaci niezależne

Aby wypełnić świat gry, zostali stworzeni bohaterowie niezależni (tzw. NPC). Pełnią oni kluczową rolę podczas rozgrywki. Gracz będzie wchodził w interakcję z postaciami niezależnymi, jednak będzie możliwe ukończenie gry bez ich pomocy.

Bohaterowie niezależni znajdują się w lokacjach, do których zostali przydzieleni. Pełnią oni rolę opiekunów tych lokacji. Wśród postaci niezależnych możemy wyróżnić:

- Barman - postać znajdująca się w tawernie. Zajmuje się opowiadaniem historii na temat świata gry, podaje posiłki oraz napoje. Korzystając z jego usług, można również wynająć pokój w tawernie.
- Alchemik - postać znajduje się w laboratorium. Sprzedaje eliksiry. Jego funkcjonalność została ograniczona do sprzedaży dwóch z nich. Postać ta zostanie przebudowana w późniejszej wersji gry.
- Doktor - znajduje się w szpitalu. Potrafi wyleczyć postać gracza z odniesionych obrażeń.
- Szaman - znajduje się w domu szamana. Jego zadaniem jest zdejmowanie negatywnych efektów nadanych na poszczególne statystyki postaci gracza.



Rysunek 3:8 - Schemat tworzenia obiektów postaci niezależnych

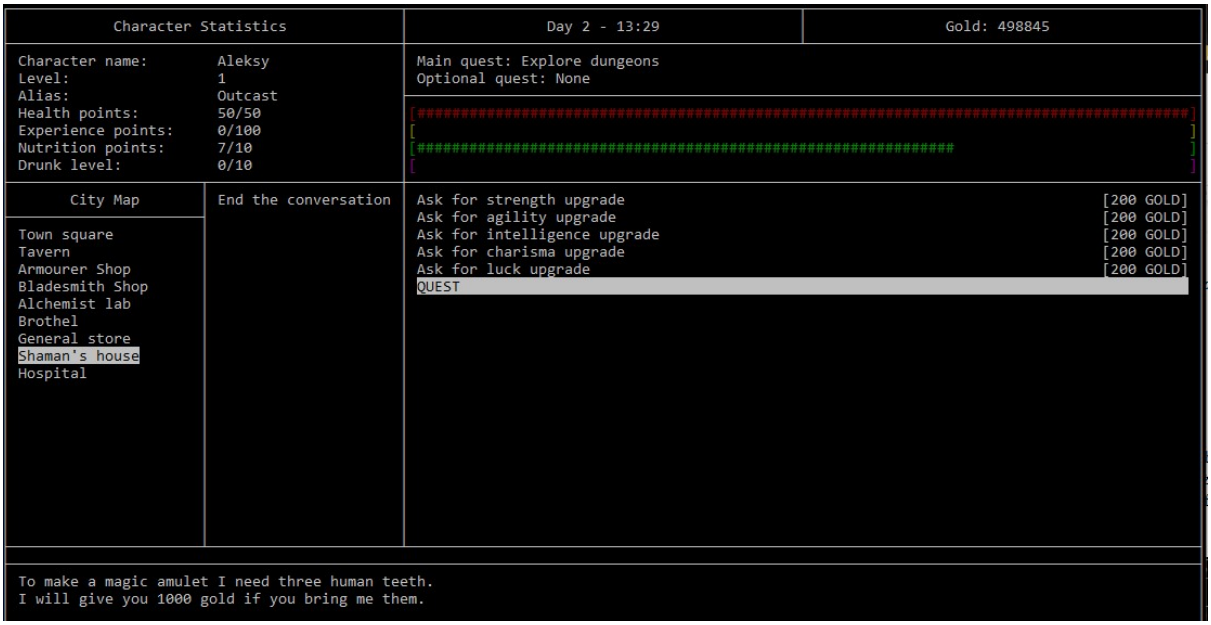
- Miecznik - znajduje się w warsztacie miecznika. Handluje on bronią z bohaterem gracza. Pozwala on również na wzmocnienie posiadanej przez gracza broni o maksymalnie trzy punkty obrażeń.
- Zbrojmistrz - znajduje się w warsztacie zbrojmistrza. Ulepsza on posiadane przez postać gracza części pancerza.
- Sklepikarz - znajduje się w sklepie wielobranżowym. Zajmuje się handlem przedmiotami innymi niż pozostałe postacie niezależne.

Każda z docelowych postaci niezależnych jest utworzona na bazie specjalnego obiektu z bazowymi parametrami oraz funkcjami (Rysunek 3:8). W ten sposób możemy pokazać jeden z paradygmatów programowania obiektowego [15], jakim jest dziedziczenie oraz nie powielanie, raz napisanego kodu.

Każda z postaci niezależnych posiada wiele statystyk. Wśród nich znajdziemy przykładowo:

- Nazwę postaci,
- Poziomy reputacji względem postaci gracza,
- Listę funkcji oraz opcji dialogowych,
- Ceny za poszczególne usługi,
- Ilość posiadanego złota,
- Znaczniki zadań pobocznych.

Postacie niezależne nie będą opuszczały lokacji do których przynależą, więc nie było potrzeby zaimplementowania im statystyk zbliżonych do postaci gracza. Postacie są również nieśmiertelne co może się zmienić. Każda z postaci posiada również dodatkowe, unikalne dla niej funkcje. Funkcje te są skojarzone z lokacją w której przebywa. Aby rozpocząć interakcję z postacią niezależną należy w konkretnej lokacji wybrać opcję rozmowy z nią.



Rysunek 3:9 - Procedura uzyskania zadania pobocznego podczas rozmowy z postacią niezależną

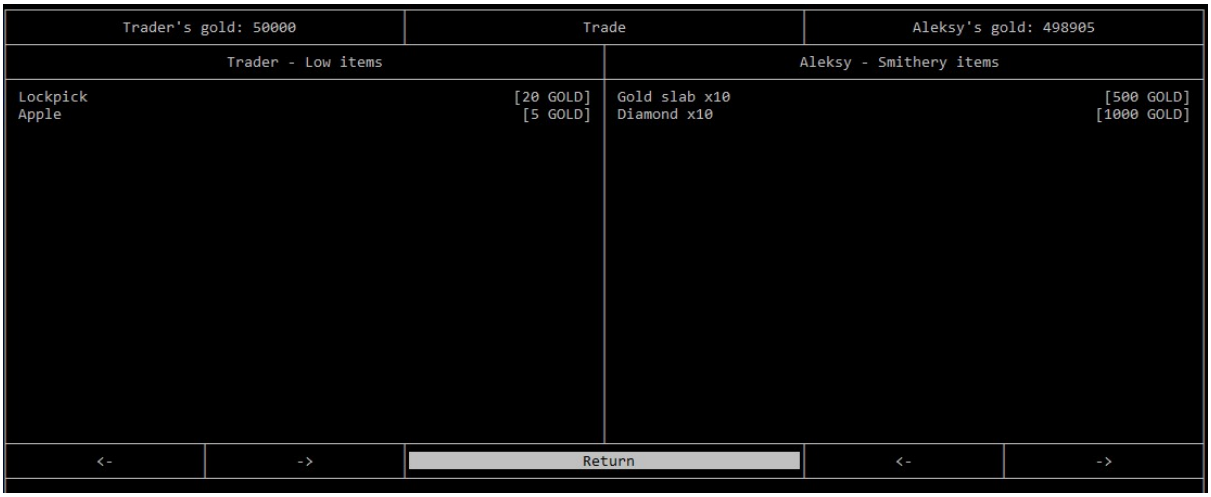
W dalszym kroku zostaną pokazane opcje dialogowe przyporządkowane do postaci niezależnej. Gracz wybierając jedną z nich uzyskuje dostęp do interesującej go formy interakcji. Aby zakończyć rozmowę, gracz może wybrać opcję z menu kontekstowego, znajdującego się obok mapy miasta w interfejsie tekstowym. Część funkcji postaci niezależnych, przykładowo: wynajęcie pokoju w karczmie, automatycznie zakończy rozmowę. Postacie niezależne posiadają znaczniki oraz funkcje związane z zadaniami pobocznymi. Znaczniki określają jakie zadanie poboczne aktualnie trwa, jego status oraz nagrody za jego ukończenie. Wszystkie statystyki postaci niezależnych zostają zapisane do plików tekstowych w momencie zapisania rozgrywki przez gracza.

3.6. Zadania poboczne

Podczas rozgrywki gracz będzie miał możliwość rozegrania dodatkowych zadań pobocznych. Są one nieobowiązkowe z punktu widzenia głównej linii fabularnej. Gracz może mieć aktywne jedno zadanie poboczne naraz. W przyszłości to ograniczenie powinno zostać zniesione. Zadania poboczne mają na celu zaprezentować część świata gry, która nie jest wyeksponowana w zadaniach głównych, ale przede wszystkim rozwinąć postać gracza. Zadania mogą mieć na celu wiele aktywności w świecie gry. Mogą być one ograniczone limitem rozwinięcia głównej postaci lub czasu gry.

Najczęstszym sposobem na rozpoczęcie zadania pobocznego jest interakcja z postaciami niezależnymi. Funkcje związane z zadaniami pobocznymi oznaczone są w interfejsie tekstowym znacznikiem QUEST (Rysunek 3:9).

Gdy zadanie poboczne zostanie rozpoczęte, jego tytuł będzie wyświetlany w interfejsie tekstowym. Od tego momentu, gra będzie śledziła poczynania gracza. Zadanie może



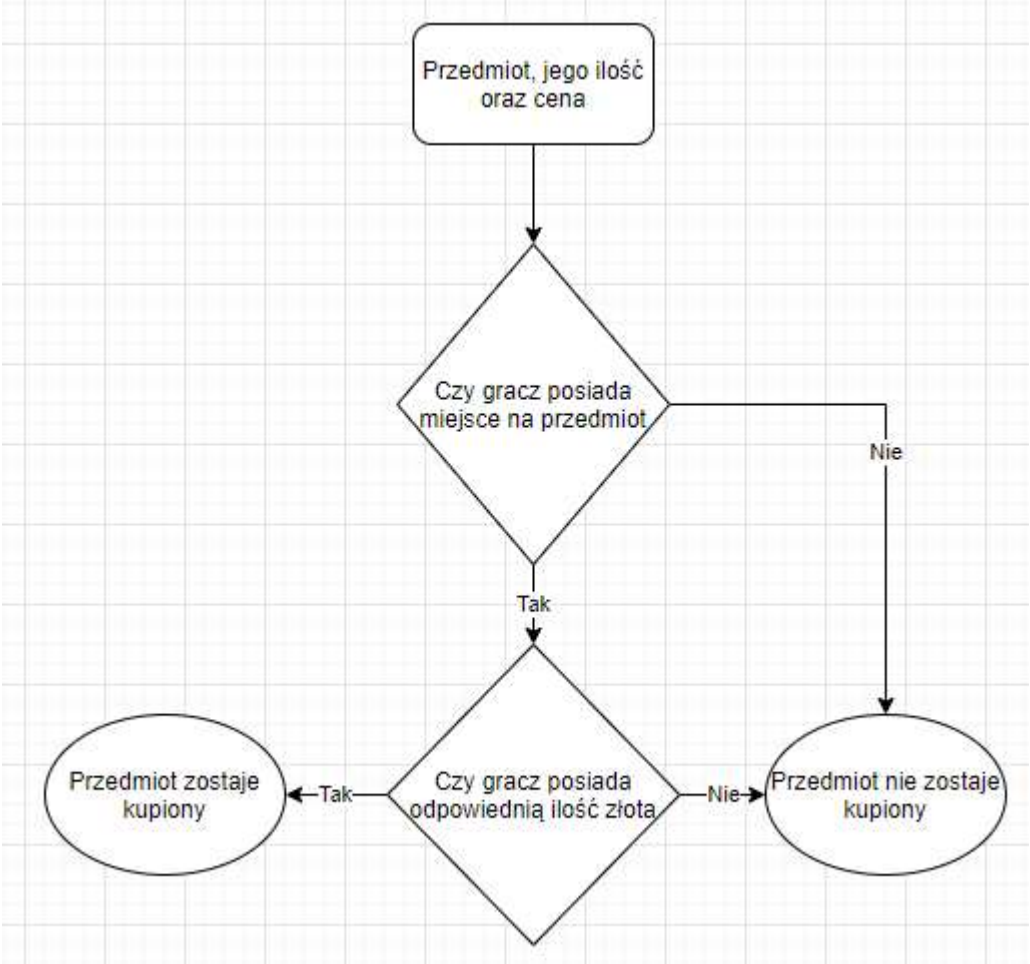
Rysunek 3:10 - Menu handlu pomiędzy postacią niezależną a postacią gracza

zakończyć się sukcesem lub porażką. Jeżeli gra stwierdzi, że gracz wykonał zadanie poboczne, podświetli na zielono tytuł w interfejsie tekstowym. Analogiczne działanie będzie w wyniku porażki, lecz wtedy zadanie zostanie podświetlone kolorem czerwonym. Zadania zakończone porażką będzie można powtórzyć, by gracz nie został odcięty od reszty zawartości z powodu niepowodzenia. Po spełnieniu warunków ukończenia zadania pobocznego, należy wrócić do miejsca lub postaci niezależnej, od której zadanie otrzymano aby odebrać nagrody. Nagrodą za wykonanie zadań pobocznych jest zawsze złoto oraz punkty doświadczenia. W aktualnej wersji gry dostępne jest jedno zadanie poboczne. Polega ono na przyniesieniu 3 ludzkich zębów dla szamana. W zamian za ukończenie tego zadania, postać gracza otrzyma 1000 sztuk złota oraz 100 punktów doświadczenia. Jeżeli uda się graczowi ukończyć zadanie poboczne, znacznik wskazujący na aktualne zadanie poboczne zostanie zmodyfikowany i przestawiony na kolejne z nich.

3.7. Handel

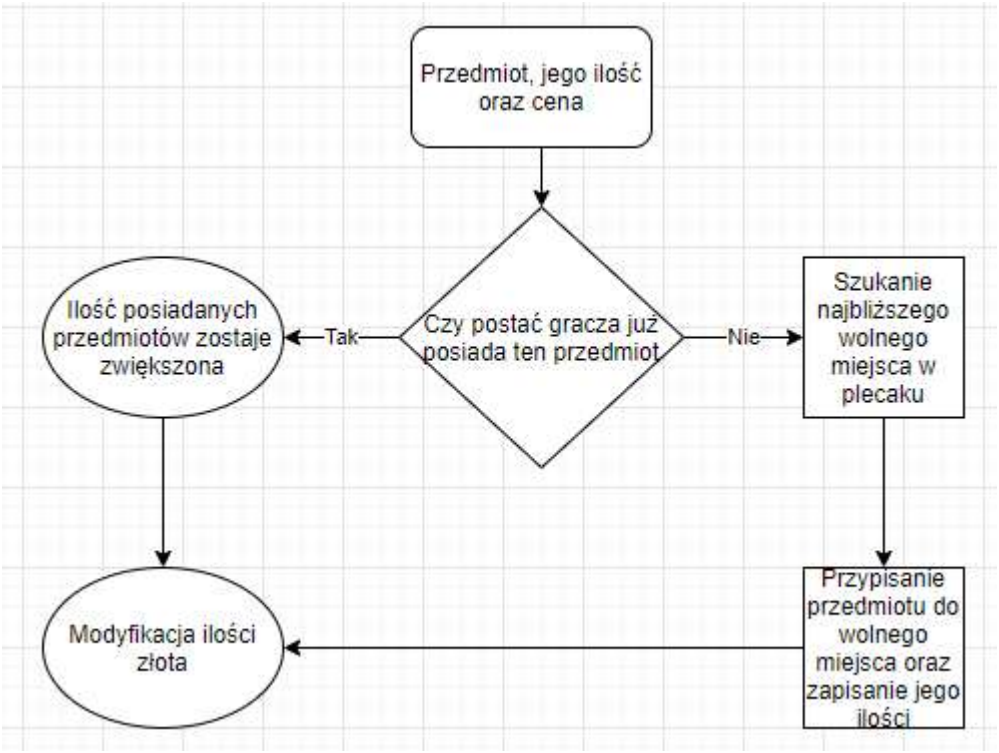
Podczas przygody gracza może mu się zdarzyć sytuacja, w której nie będzie w stanie wziąć przedmiotu ze sobą ze względu na brak miejsca w plecaku. W takiej sytuacji gracz może wyrzucać przedmioty ale jest to nieopłacalne z punktu widzenia ekonomii gry. Dlatego część z postaci pobocznych będzie mogła handlować z graczem (Rysunek 3:10). W chwili rozpoczęcia handlu, postać niezależna określi wartość przedmiotów posiadanych przez obie strony transakcji. Zrobi to na podstawie specjalnych list przedmiotów, dołączonych do gry w postaci plików tekstowych. Jeżeli system nie jest w stanie określić ceny przedmiotu, uznaje go za bezwartościowy. Oferowany przez sprzedawcę towar może różnić się w zależności od poziomu postaci gracza. Część przedmiotów będzie niemożliwa do zakupu przy zbyt niskim rozwinięciu naszej postaci. Ważnym aspektem handlu są statystyki postaci gracza oraz postaci niezależnej. Z każdą wymianą gracz będzie zyskiwał punkty reputacji, które w przyszłości dadzą mu korzystniejsze ceny za oferowany towar.

Podczas sprzedaży przedmiotu przez postać niezależną graczowi, określana jest ilość sztuk, którą jest chętny kupić gracz. Gra sprawdza czy postać gracza posiada wolne miejsce



Rysunek 3:11 - Procedura kupna przedmiotu od strony postaci niezależnej

w plecaku oraz czy ilość złota, którą posiada postać gracza jest w stanie pokryć cenę przedmiotów. Jeżeli wszystkie warunki zostaną spełnione, zostanie odegrany sygnał dźwiękowy oraz przedmiot zostanie zakupiony. Sprzedawca dysponuje nieograniczoną ilością przedmiotów które może zakupić gracz. Kiedy przedmiot zostanie zakupiony, musi on być przetransportowany do ekwipunku postaci gracza (Rysunek 3:11, Rysunek 3:12).

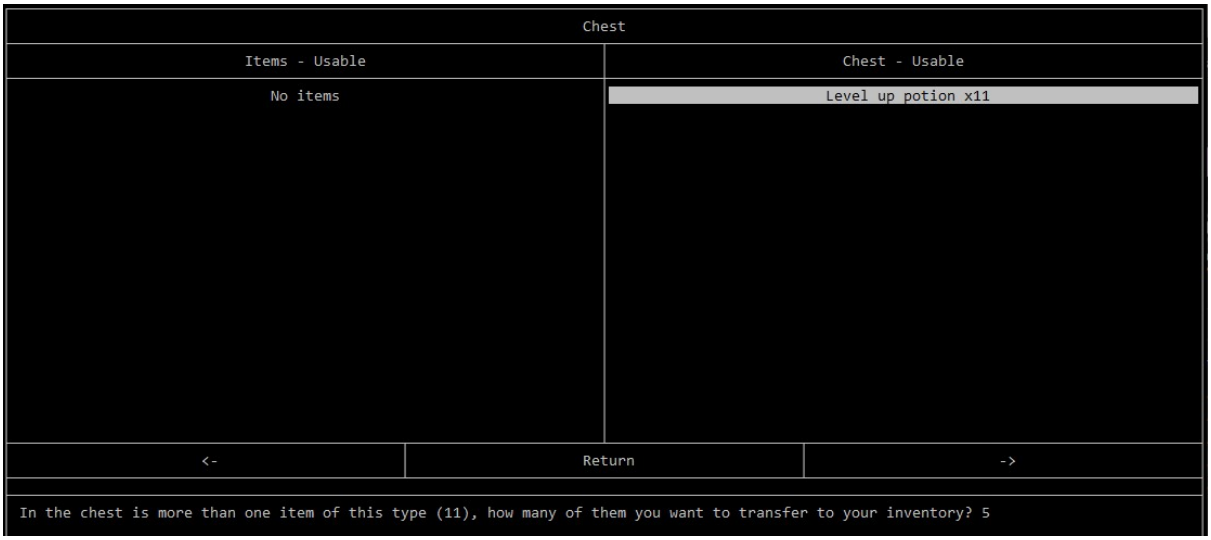


Rysunek 3:12 - Procedura kupna przedmiotu od strony postaci gracza

Gracz również może sprzedawać posiadane przedmioty. Sprzedaż jest permanentna, więc sprzedanych przedmiotów nie da się odkupić. Rozwiązanie to będzie skłaniało gracza do rozważań na temat strategii działania. Część postaci niezależnych nastawiona jest na konkretną branżę i związane z nią przedmioty. Dlatego mogą nie kupić części przedmiotów oferowanych przez gracza. Jednak gdy gracz zaoferuje przedmiot którym będzie zainteresowany kupiec, zostanie sprawdzona ilość posiadanych przedmiotów oraz jego cena. Jeżeli gracz posiada więcej niż jeden przedmiot, zostanie zapytany ile z nich chce zaoferować. W treści komunikatu zostanie wypisana cena za jedną sztukę. Gdy gracz wpisze poprawną wartość, przedmioty zostaną usunięte z jego plecaka. Gdy transakcja kupna lub sprzedaży zostanie zakończona, odegrany zostanie sygnał dźwiękowy, a ilość złota obu uczestników wymiany zostanie zaktualizowana.

3.8. Skrzynia

Z uwagi na ograniczenia plecaka głównej postaci, w grze została zaimplementowana skrzynia na przedmioty. Pełni ona rolę przechowalni na niektóre z przedmiotów posiadanych przez gracza. Skrzynia została podzielona na 3 kategorie, zgodne z podziałem plecaka głównej postaci. Podobnie jak plecak, skrzynia pomieści 20 sztuk przedmiotów z danej kategorii. Korzystając z niej, gracz może przechować część zgromadzonych dóbr, których nie chce sprzedać, a zachować na bardziej odpowiedni moment. W skrzyni nie można przechowywać broni, lecz planowana jest taka możliwość. Przedmioty pomiędzy skrzynią a postacią gracza mogą być przenoszone bez dodatkowych kosztów oraz uwzględniając ich ilość (Rysunek 3:13). Skrzynia znajduje się w karczmie, lecz została ona napisana w taki sposób by można było dodać więcej niż jedną instancję w dowolnej lokacji gry. Skrzynia jest niezniszczalna oraz nie może zostać okradziona, dlatego gracz może czuć się bezpieczny o



Rysunek 3:13 - Procedura przenoszenia przedmiotów w skrzyni

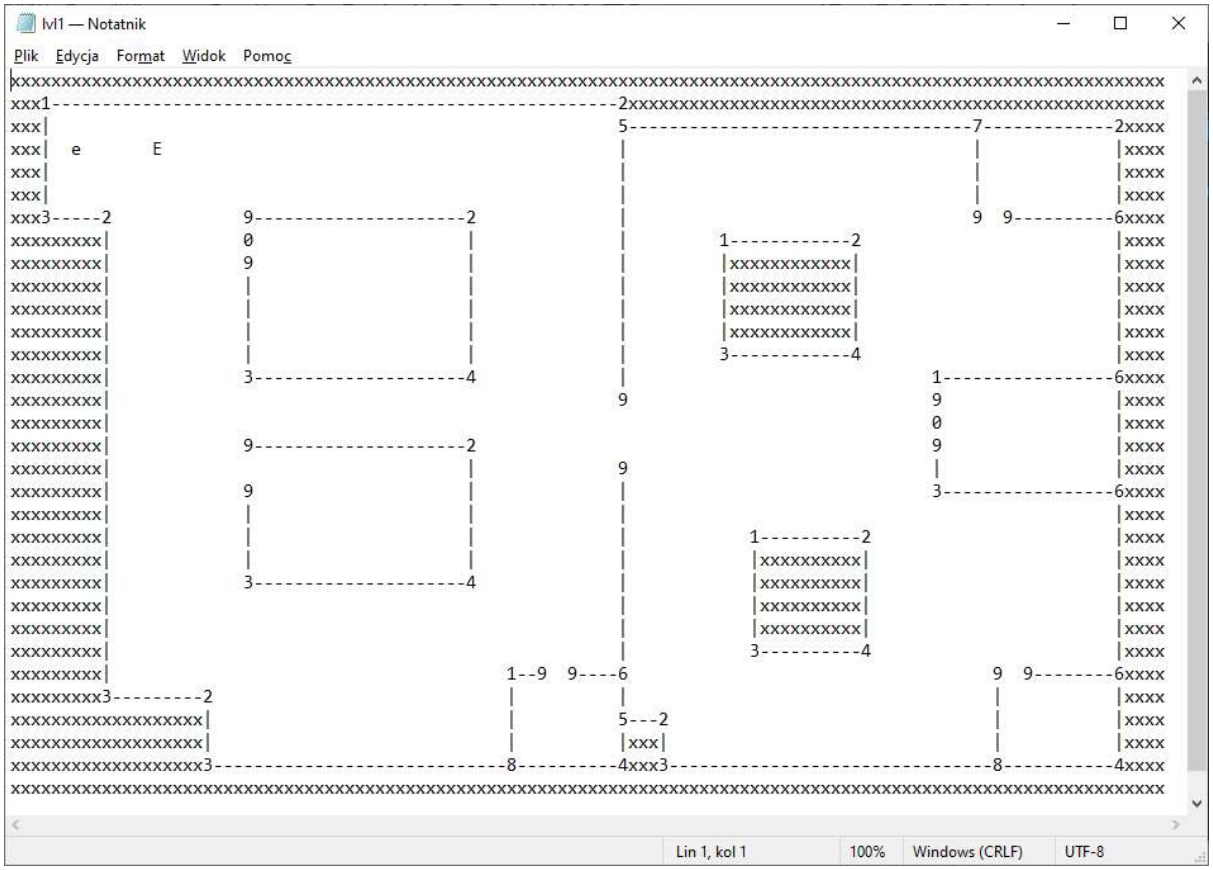
swoje przedmioty. Jej stan również zapisywany jest w momencie zapisu rozgrywki przez gracza.

3.9. Lochy

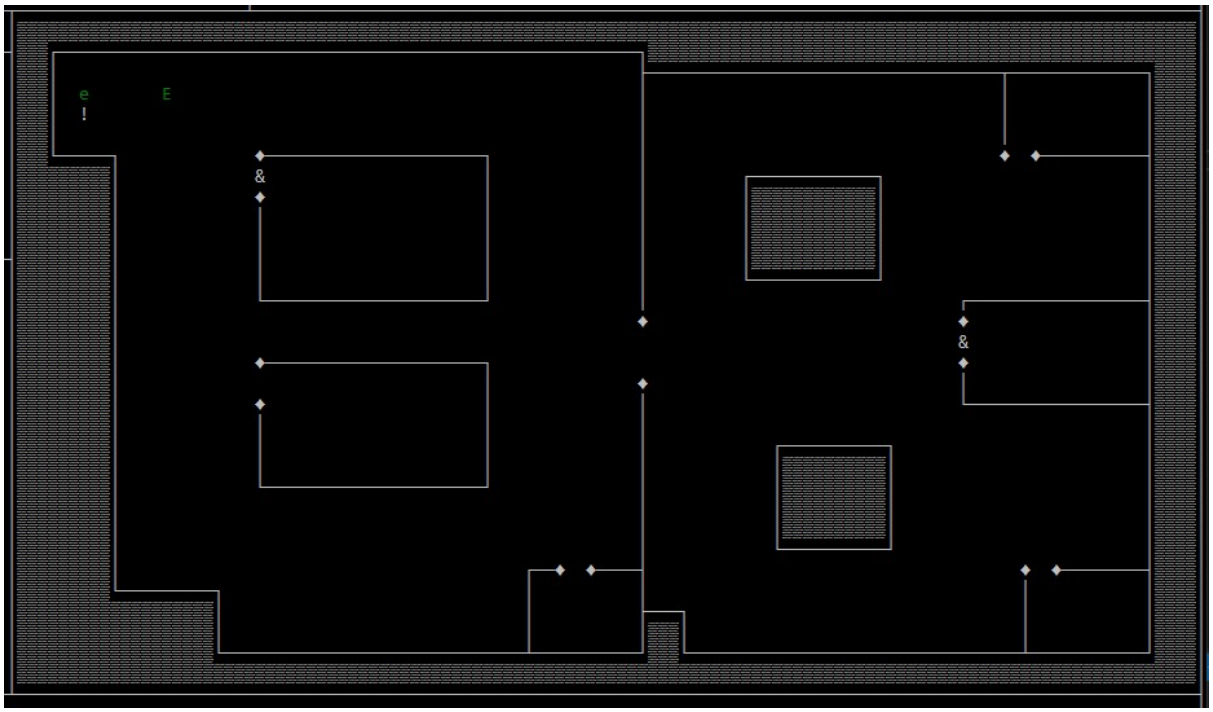
Lochy stanowią główną część rozgrywki w grze. Jest to miejsce tajemnicze i niebezpieczne. Ma za zadanie budować napięcie oraz popychać rozwój postaci gracza do przodu. Jest to seria lokacji o strukturze prostych labiryntów z większymi, bądź mniejszymi pomieszczeniami. Każdy z poziomów jest wypełniony obiektami, które mogą skrywać przedmioty, przeciwnikami oraz elementami dekoracyjnymi. Część z pomieszczeń może być zablokowana przez zamknięte drzwi. Jeżeli postać gracza posiada specjalny przedmiot jakim jest wytrych, będzie mogła sforsować blokadę. Nad lochem czuwa zarządca lochów.

3.9.1. Zarządca lochów

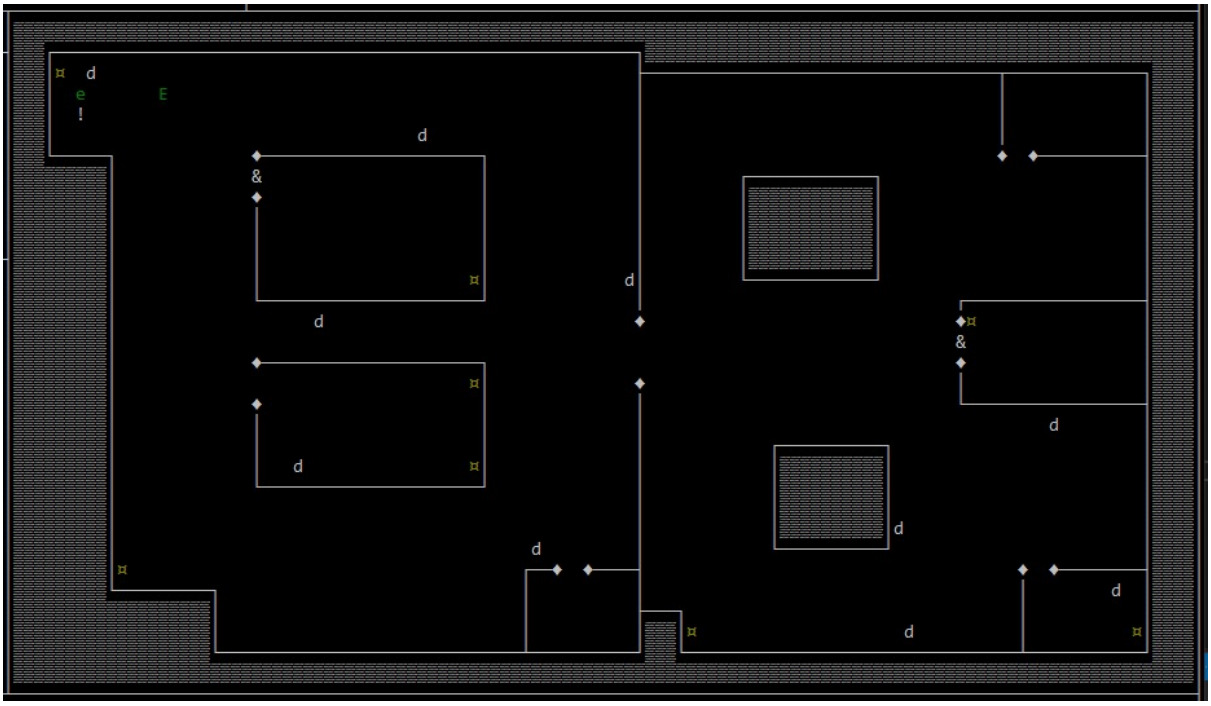
Aby wygodnie operować na danych związanych z lochami, powstał specjalny obiekt klasy. Jest nim zarządca lochów. Przetrzymuje on najważniejsze parametry dotyczące lochów oraz metody z nimi związane. Przechowuje on mapę lochu, listy przedmiotów oraz przeciwników. To jego dane będą wysyłane do interfejsu tekstowego. Każda mapa lochu, która znajduje się w pamięci jest w postaci znakowej tablicy dwuwymiarowej. Pole rozgrywki ma wymiary 114 x 32 kratki. Są to również wymiary wcześniej wspomnianej tablicy. Przy każdorazowym odwiedzaniu lochów przez postać gracza, zarządca określa poziom, na którym powinna toczyć się akcja. Pomagają mu w tym specjalne pliki tworzone w katalogu z grą. Są one przypisane do każdej z lokacji osobno. Jeżeli w katalogu z grą, wspomniane pliki nie zostaną znalezione, gracz zacznie eksplorację lochów od poziomu pierwszego. Każdy poziom lochów wysyłany jest do zarządcy za pomocą plików tekstowych (Rysunek 3:14, Rysunek 3:15). Jest to forma umożliwiająca poprawki w strukturze poziomu, bez potrzeby ponownej kompilacji całej aplikacji.



Rysunek 3:14 - Plik tekstowy zawierający informacje o poziomie lochu



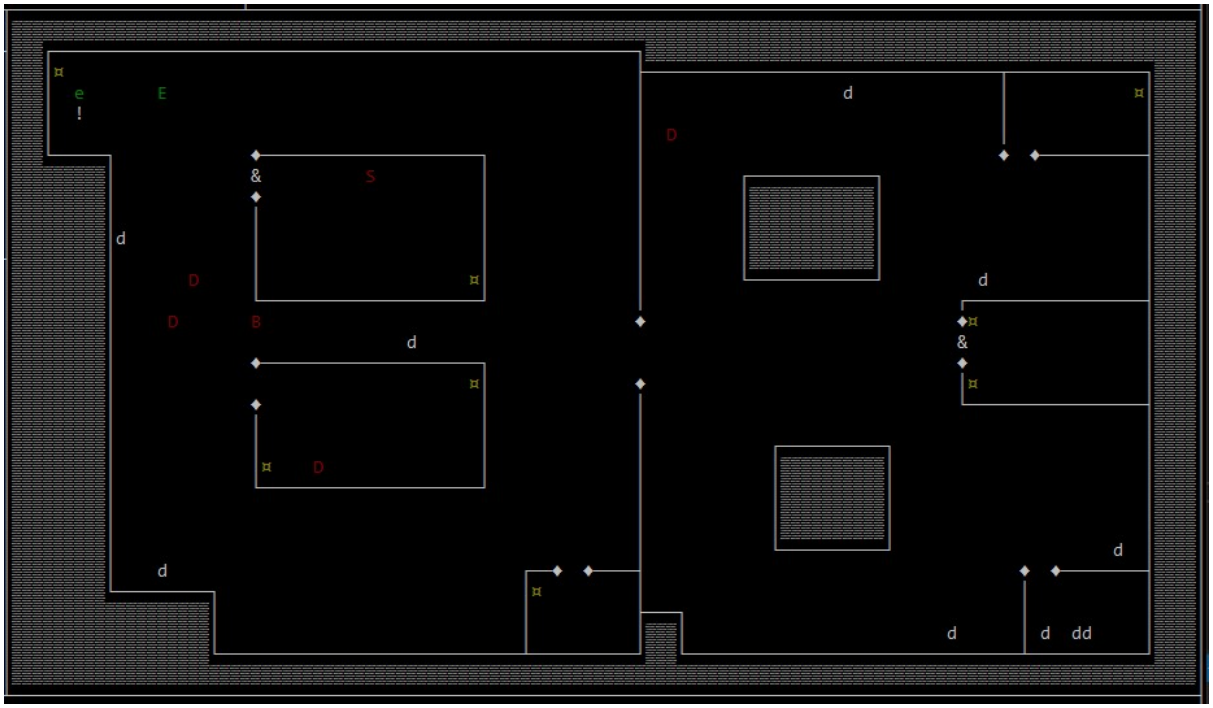
Rysunek 3:15 - Stan lochu po przetworzeniu pliku przez program



Rysunek 3:16 - Stan lochu po wygenerowaniu obiektów dekoracyjnych

Następnym krokiem, po określeniu poziomu, jest wygenerowanie obiektów dekoracyjnych (Rysunek 3:16). Dzielą się one na zniszczalne oraz niezniszczalne. Lampy zaliczane do obiektów niezniszczalnych, będą generowane w rogach pomieszczeń. Ich liczba jest losowa. Przedmiotami niezniszczalnymi są przykładowo: beczki oraz skrzynie. Mogą one skrywać przedmioty lub pozostać puste. Ilość obiektów, szansa na posiadanie przedmiotu wewnątrz oraz jego nazwa określane są w zewnętrznym pliku tekstowym. Obiekty zniszczalne pojawiają się pod ścianami labiryntów, by nie przeszkadzać graczowi w poruszaniu się po lochu.

Ostatni na poziomie pojawiają się przeciwnicy (Rysunek 3:17). Zostaną oni opisani dokładniej w osobnym podrozdziale [3.9.4, str. 31]. Gdy cały loch jest już wygenerowany, dane zostaną przekazane do interfejsu tekstowego, celem ich narysowania na ekranie konsoli poleceń. System na bieżąco interpretuje otrzymane dane oraz zamienia część symboli na inne, zdefiniowane w kodzie aplikacji. Część symboli została narysowana z użyciem kolorów. Ma to na celu łatwiejszą ich identyfikację. Symbole oznaczające zagrożenie, narysowane są kolorem czerwonym. Przejścia na wyższy bądź niższy poziom lochów oznaczone są kolorem zielonym. Obiekty neutralne nie korzystają z kolorów. Postać gracza jest oznaczona symbolem wykrzyknika.



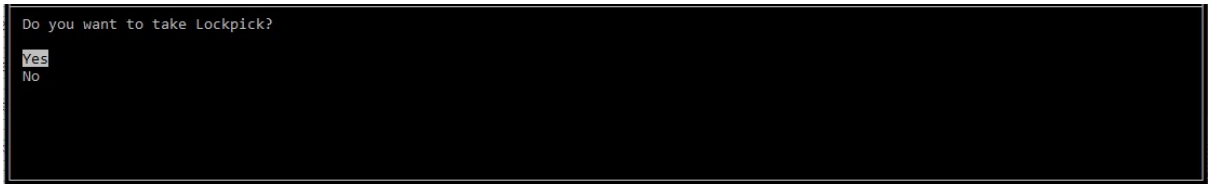
Rysunek 3:17 - Stan lochu po wygenerowaniu przeciwników

3.9.2. Poruszanie się

Poruszanie się po lochach odbywa się w formule turowej [16]. Każde naciśnięcie klawiszy strzałek przez gracza jest traktowane jako oddzielna tura. Postać gracza zawsze porusza się o jedną kratkę w kierunku określonym przez wciśnięty przycisk na klawiaturze. Gracz może również nacisnąć klawisz ESC, który odpowiada za menu podręczne. W nim znajdują się najważniejsze opcje związane z postacią gracza, np. ekwipunek. Gracz może poruszać się tylko w pustej przestrzeni. Na ścianach labiryntu oraz obiektach niezniszczalnych postać gracza zatrzyma się. W momencie, gdy gracz wejdzie w pole, na którym znajduje się obiekt, który można zniszczyć, zostanie wyświetlona stosowna informacja. Gracz zostanie poinformowany o możliwości zniszczenia obiektu oraz zapytany, czy chce to zrobić (Rysunek 3:18).

Jeżeli odpowiedź na pytanie będzie twierdząca, obiekt zostanie zniszczony. W zależności czy posiadał on przedmiot w środku, zostanie on wstawiony w miejsce zniszczonego obiektu. Interfejs tekstowy zakomunikuje ten fakt, umieszczając znak zapytania w miejscu przedmiotu. Gracz będzie mógł podnieść przedmiot lub pozostawić na miejscu (Rysunek 3:19).

Rysunek 3:18 - Komunikat wyświetlany przy próbie zniszczenia obiektu dekoracyjnego



Rysunek 3:19 - Komunikat wyświetlany przy próbie podniesienia przedmiotu leżącego w lochu

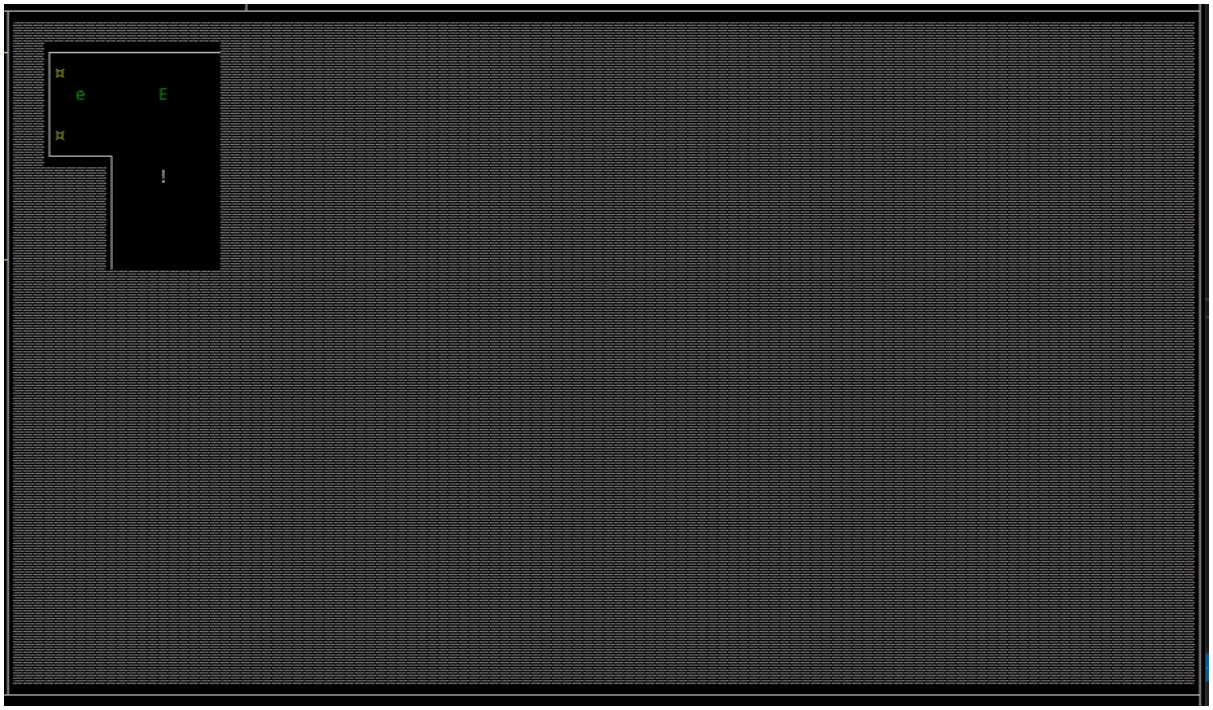
Gracz przechodząc pomiędzy poziomami, będzie rozpoczynał dany poziom od nowa. Dlatego każdy z pozostawionych na ziemi przedmiotów przepadnie. Aby przemieszczać się pomiędzy kolejnymi poziomami lochów, zostały ustalone punkty tranzycji (Rysunek 3:20). Oznaczone są one znakiem “e” oraz “E”. Jest to skrót od entrance (ang. wejście) oraz exit (ang. wyjście). Jeżeli gracz zdecyduje się wrócić na poprzedni poziom, zacznie od wejścia czyli znaku “e”.

W każdej turze po ruchu postaci gracza, następuje ruch przeciwników. Każdy z nich losuje wartość od 0 do 4. W przypadku wartości 0-3 gra przesunie przeciwnika o jedno pole w jednym z czterech kierunków. Wartość 4 odpowiada pozostawieniu przeciwnika w miejscu, bez podejmowania akcji.

3.9.3. Mgła wojny

W świecie rzeczywistym, człowiek nie jest w stanie widzieć obiektów oddalonych od niego na znaczną odległość. Dlatego podobny system zaimplementowano w grze. Mechanika ta jest znana często pod pojęciem mgły wojny [17]. Jest to specjalna bariera która przykrywa część obszaru, której postać gracza nie powinna móc zobaczyć (Rysunek 3:21). Ma ona na celu ukryć część poziomu, by nadać rozgrywce odrobiny wyzwania. Również ukryci we mgle przeciwnicy, wpłyną na ogólne odczucia z grania. Na początku rozgrywki cały ekran gry zostaje nią przykryty. Następnie, z każdym ruchem postaci gracza, mgła rozmywa się. W miejscu które gracz odkrył, ale nie znajduje się w nim bezpośrednio, mgła nie zostaje ponownie dodana.

Rysunek 3:20 - Komunikat wyświetlany przy przejściu pomiędzy poziomami lochu



Rysunek 3:21 - Stan lochu po nałożeniu na niego mgły wojny

3.9.4. Przeciwnicy

W grze występuje wiele typów przeciwników. Każdy z przeciwników ma dwa stany w zależności, czy walczą oni z postacią gracza, czy jedynie przemieszczają się po lochu. Przeciwnicy są tworzeni podczas procesu wczytywania poziomu przez zarządcę lochów. Każdy poziom lochów posiada oddzielną listę przeciwników którzy znajdują się na poziomie. O wariacji ich występowania decyduje los. Listy dla każdego poziomu, w postaci plików tekstowych, przechowują nazwy przeciwników oraz ilość ich występowania na konkretnym poziomie. Są to informacje niezbędne dla zarządcy lochów. Jeżeli lista dla konkretnego poziomu nie zostanie znaleziona w katalogu z grą, przeciwnicy nie zostaną stworzeni. W przypadku walki poza lochami, występowanie przeciwników ustalane jest drogą indywidualną. Do czasu rozpoczęcia walki z przeciwnikiem, ten posiada tylko część ze statystyk. Wśród nich możemy wyszczególnić:

- położenie na mapie lochu,
- nazwa,
- symbol.

Nazwa pobierana jest z listy przeciwników, zaś symbolem jest pierwsza litera nazwy przeciwnika. Każdy z symboli jest dopisywany do listy symboli przeciwników na poziomie. Będzie ona przesyłana do interfejsu tekstowego celem podświetlenia ich na kolor czerwony. Przeciwnicy ustawiani są w losowych, pustych miejscach poziomu, z zastrzeżeniem kilku miejsc bezpośrednio obok postaci gracza. W przeciwnym przypadku mogłoby to prowadzić do sytuacji, gdzie gracz zostaje zaatakowany bezpośrednio po wczytaniu poziomu. Mogłoby to prowadzić do jego niepotrzebnej frustracji oraz obniżyć przyjemność płynącą z rozgrywki. Przeciwnicy tworzeni są na samym końcu procesu wczytywania poziomu, by ilość miejsc na

których mogą zostać umieszczeni, podczas tworzenia, była maksymalnie ograniczona. Przeciwnicy przechodzą do drugiego stanu, w momencie rozpoczęcia walki z postacią gracza, tym samym zyskując dodatkowe statystyki. Ich nazwa zostaje przekazana do funkcji odpowiadającej za tryb walki. Zostanie utworzony obiekt przeciwnika wewnątrz funkcji walki. Korzystając z nazwy przeciwnika, funkcja wyszukuje plik tekstowy ze statystykami odpowiedniego przeciwnika. Jeżeli plik ze statystykami nie został znaleziony, do pamięci zostaną załadowane domyślne statystyki zaimplementowane w kodzie aplikacji. W momencie załadowania pliku, przeciwnik otrzymuje dodatkowe parametry:

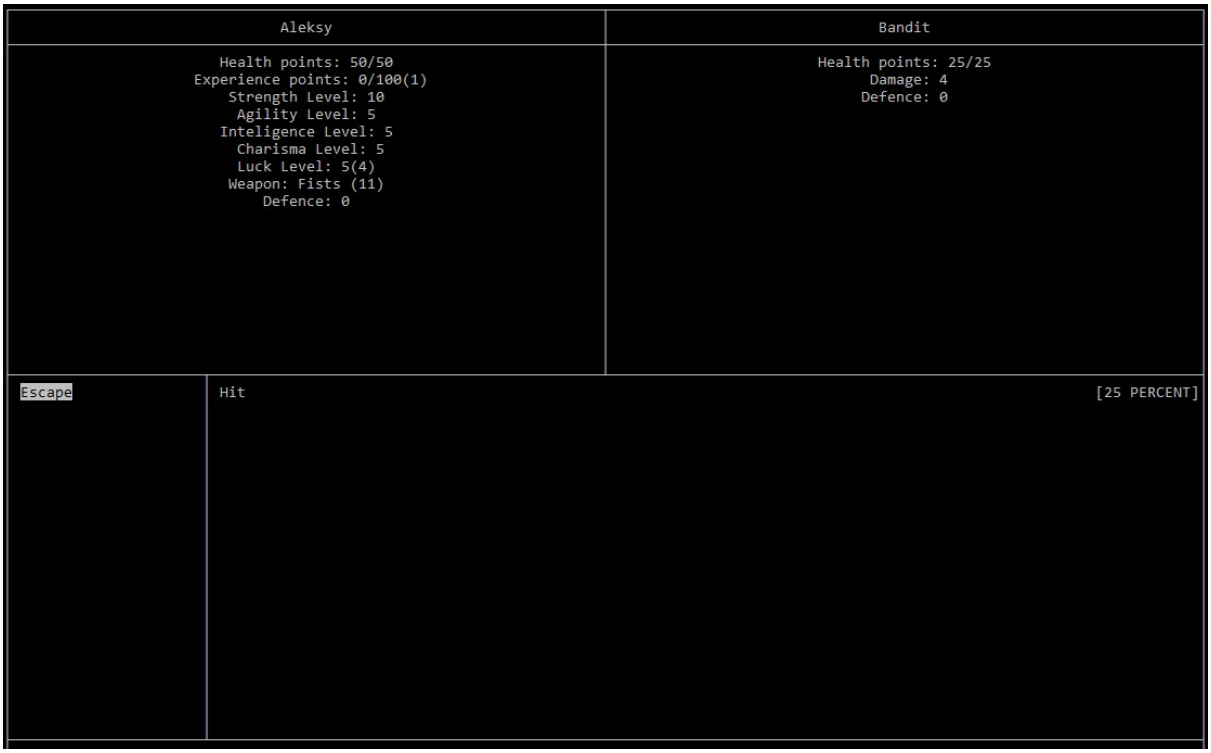
- punkty życia,
- punkty doświadczenia,
- punkty ataku,
- punkty obrony,
- przedmiot,
- ilość złota,
- szanse na otrzymanie obrażeń,
- szanse na zadanie obrażeń,
- listę ze statystykami w formie tekstowej.

Część z parametrów, takich jak ilość złota, szanse na zadanie lub otrzymanie obrażeń jest podzielona na dwie wartości: podstawę oraz dodatek. Podstawa jest określana jako niezmienna wartość, za to dodatek jest liczbą losową z przedziału: 0 do podanej wartości. Podczas pobierania jednego z wyżej wymienionych podzielonych parametrów, pobierana jest suma dwóch składników. Ma to na celu wprowadzić odrobinę losowości w statystykach przeciwników o tej samej nazwie. Trzymany przedmiot jest przyznawany, bądź też nie na podstawie losowania, z uwzględnieniem współczynnika wczytanego z pliku tekstowego. W momencie kiedy przeciwnik ma już wczytane wszystkie statystyki, są one zapisywane w liście w formie tekstu. Będą one wyświetlane w interfejsie tekstowym, podczas walki. Przeciwnicy posiadają również funkcje odświeżającą ową listę, po zadaniu im obrażeń.

3.10. System walki

Podczas zwiedzania lochów postać gracza może zostać zaatakowana. Gracz również może rozpocząć walkę. System walki jest jednym z fundamentów rozgrywki. Rozpoczęcie walki sygnalizowane jest specjalnym sygnałem dźwiękowym. Walka jest metodą na rozwiązanie konfliktów pomiędzy postacią gracza, a jego przeciwnikiem.

Walka odbywa się w systemie turowym. System walki polega na wymianie ciosów, a gracz oraz jego przeciwnik wykonują akcje naprzemiennie. Zależnie od tego kto zainicjował walkę, zacznie on pierwszy. Walka może zostać wszczęta w lochach, jak i w każdej innej lokacji. Postać gracza posiada przy sobie broń. Jest ona określona typem oraz ilością punktów zadawanych obrażeń. Broń można kupić od postaci niezależnych. Planowana jest też możliwość znalezienia jej podczas rozgrywki. Postać posiada również opancerzenie podzielone na poszczególne części. Statystyki broni są ważne podczas walki. Rodzaj broni określa szanse na trafienie przeciwnika. W grze przygotowano mechanizmy pod odbijanie

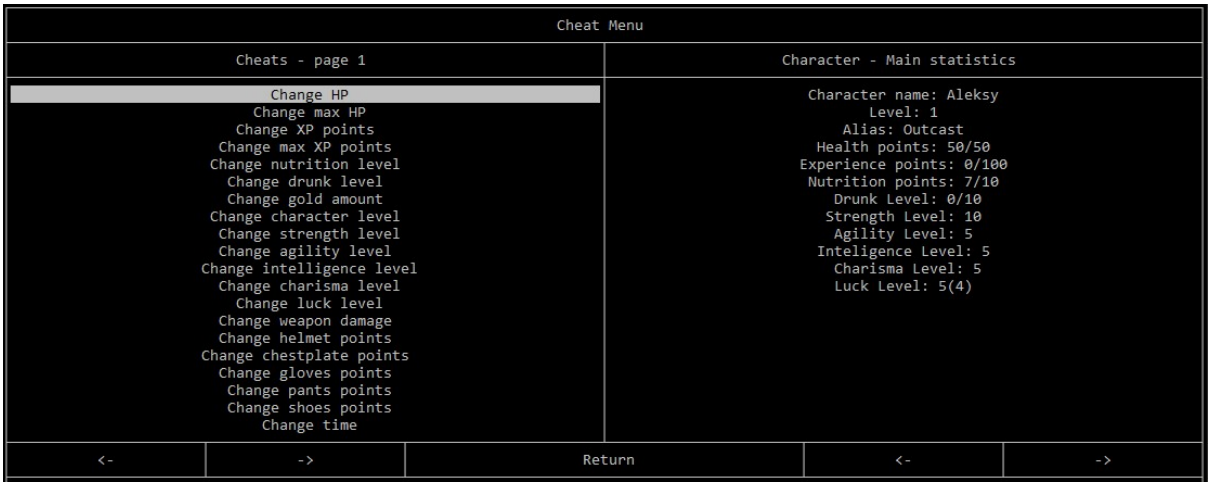


Rysunek 3:22 - Okno systemu walki

ciosów za pomocą pancerza, lecz system okazał się częściowo wadliwy. Zostanie przywrócony w późniejszej wersji gry. Tyczy się to postaci gracza oraz jego przeciwnika. Statystyki postaci gracza jak i przeciwnika wyświetlają się w interfejsie tekstowym. Walka polega na wybraniu przez gracza odpowiadającej mu akcji z listy. System przelicza szanse na wykonanie wskazanej akcji oraz wyświetla ją na ekranie (Rysunek 3:22). W obecnej wersji gry, gracz może uderzyć przeciwnika lub też próbować uciec. Przy każdej rundzie system ponownie przelicza szanse na trafienie przeciwnika przez gracza, z uwagi na losowanie tej wartości.

W przypadku kiedy atak postaci gracza trafił lub też nie, zostanie wyświetlony komunikat oznajmiający wynik losowania. Jeżeli atak postaci trafił w cel, zostanie wylosowana dodatkowa wartość, która ma symulować rzut kością. Wartość ta określa wystąpienie obrażeń krytycznych. Oznacza to podwojenie zadanych obrażeń. Próg zadania obrażeń krytycznych został ustawiony na 5%. Wartość obrażeń zostaje odjęta od wartości punktów życia przeciwnika. Po wykonanym ciosie przez gracza, następuje kolej na przeciwnika. Podobnie jak gracz, losuje on szansę na trafienie postaci gracza z uwzględnieniem statystyk obu postaci. Jeżeli trafi, również ma szansę na zadanie obrażeń krytycznych, a wartość jego ataku zostanie odjęta od wartości punktów życia postaci gracza.

Z walki można również próbować uciec, lecz jest szansa na ponowne jej rozpoczęcie w niedługim czasie. Konflikt najczęściej trwa, dopóki któraś ze stron nie straci wszystkich punktów życia. W planach jest również wprowadzenie walki, która nie ma na celu uśmiercenia przeciwnika. Śmierć postaci gracza oznacza zakończenie rozgrywki. W przypadku wygranej gracza, zostanie odegrany sygnał dźwiękowy. Ilość zdobytego doświadczenia oraz złota zostanie dodana do wartości postaci gracza oraz wypisana na



Rysunek 3:23 - Okno trybu testowego

ekranie. Jeżeli przeciwnik posiadał przy sobie przedmiot, gracz zostanie zapytany czy chce go zabrać czy pozostawić. W przypadku, gdy gracz nie zabierze ze sobą przedmiotu po przeciwniku, zostanie on ustawiony na mapie, podobnie jak przedmioty z obiektów zniszczalnych opisanych w poprzednim rozdziale [3.9.1, str. 26]. W ostatnim kroku jest usunięcie obiektu przeciwnika z listy przeciwników, którą posiada zarządca lochów.

3.11. Testowanie

Napisanie gry komputerowej jest dość dużym wyzwaniem dla człowieka, nawet bardzo dobrze znającego swój fach. Niestety ludzie popełniają błędy, które trzeba poprawić. Proces tworzenia aplikacji potrafi trwać bardzo długi okres czasu ze względu na wiele możliwości, które musi pokryć programista podczas tworzenia programu. Należy bowiem pamiętać, że nie każdy użytkownik działa w ten sam sposób. Umiejętności końcowego użytkownika mogą być za małe, by skutecznie operować posiadanym oprogramowaniem. Dlatego programista musi przewidywać, co może pójść nie tak w każdym aspekcie tworzonej aplikacji.

W mojej aplikacji zamieściłem specjalny tryb pomagający w testowaniu poszczególnych aspektów gry. Odpowiada on za zmianę najważniejszych statystyk postaci gracza. Opcje podzielone są na strony oraz poszczególne funkcje. System został napisany w ten sposób, aby można było dodawać kolejne funkcje dość małym kosztem. W trybie testowym wyświetlane są również statystyki, aby tester miał wgląd w parametry, które go interesują (Rysunek 3:23).

Użycie tego narzędzia polega na wybraniu interesującej nas funkcji oraz wpisaniu z klawiatury nowej wartości. Ważnym aspektem tego narzędzia jest obsługa wyjątków. Zapobiega ona sytuacjom, gdzie wpisane dane mogą powodować awaryjne zatrzymanie aplikacji. Jeżeli podane wartości będą poprawne, aplikacja nadpisze odpowiadający funkcji parametr postaci gracza oraz zasygnalizuje to sygnałem dźwiękowym. Tryb testowy nie będzie dostępny dla końcowego użytkownika.

3.12. Poziom trudności

Ważną częścią gier jest ich poziom trudności. Jest to kwestia, która potrafi podzielić graczy. Aby gracz czuł satysfakcję z grania w grę, powinna ona być dla gracza jakimś wyzwaniem. Część graczy lubi gry, które nie będą zbyt trudne. Inni będą chcieli by gra niosła ze sobą dużo wysiłku oraz zaangażowania. Twórca musi zdecydować którą część konsumentów przyciągnąć, a z której zrezygnować. Może on też próbować stanąć po środku i balansować system pomiędzy obie grupy.

Tworząc swoją grę, wybrałem ostatni sposób z powyższej listy. Duża część przeliczników jest wczytywana z plików tekstowych. Dzięki temu można na bieżąco balansować poziom trudności i zmieniać go ręcznie, jeśli zajdzie taka potrzeba. Wiele wewnętrznych systemów opartych jest o statystyki postaci, więc na początku przygody gdy postać gracza jest jeszcze nierozwinięta, gra może być dość wymagająca. Z czasem gdy gracz nabierze doświadczenia poziom trudności może pozostać w miejscu lub też zostać delikatnie podniesiony, by zapewnić wciąż efekt wyzwania. Poziom trudności jest bardzo ciężko implementowalny w grach RPG, z racji na fakt częstych losowań liczb podczas niektórych aktywności w świecie gry. Można jednak rezygnować z elementów, które mogą frustrować, na przykład Grinding [18]. Jest to sytuacja w której gracz jest zmuszony do powtarzania części gry, dla uzyskania potrzebnej ilości dóbr lub punktów doświadczenia.

3.13. Perspektywy

W zaprezentowanej przeze mnie grze nie udało się do tej pory zaimplementować pewnych treści. Część z nich posiada przygotowane miejsce w aplikacji. Są one kluczowe z punktu widzenia rozgrywki, więc zostaną one dodane w najbliższej przyszłości. Wśród nich możemy wyszczególnić:

- Fabuła - ważnym elementem każdej gry RPG jest wciągająca fabuła oraz dobrze scharakteryzowani bohaterowie niezależni.
- Rozwój postaci - podczas zdobywania nowego poziomu, gracz wybiera jedną statystykę która będzie ulepszona. Nie da się pominąć tego wyboru. Planuję możliwość zachowania punktu rozwoju na “później” oraz możliwość zresetowania obecnie rozdysponowanych punktów. Będzie to umożliwiało graczowi zmianę obecnej taktyki oraz wyposażenia. Planowany jest również limit na poszczególne statystyki. Powyżej pewnej wartości gra stawałaby się zbyt prosta, więc można by zaimplementować możliwość zresetowania poszczególnej umiejętności za istotne korzyści w świecie gry. Kolejnym z elementów gry związanych z rozwojem postaci, są umiejętności. W kodzie przygotowane jest miejsce pod umiejętność, lecz nie jest ono wykorzystywane.
- A* - jest to algorytm określający najkrótszą drogę w grafie [19]. Będzie on przydatny w lochach. W obecnym stanie gry, przeciwnicy poruszają się w losowych kierunkach. Z pomocą tego algorytmu, przeciwnicy będą kierowani do postaci gracza. Z uwagi iż lochy mają konstrukcje labiryntową, ważne jest określenie najkrótszej drogi pomiędzy przeciwnikiem, a postacią gracza, o ile taka istnieje.

- Game loop - jest to mechanika która jest obecnie wykorzystywana w większości silników do gier [20]. Polega ona na tym, że klawisze naciskane przez użytkownika wpływają na dane gry. Gra aktualizuje swój stan i wysyła informacje do renderowania. Po wszystkim proces zapętla się dając płynny obraz użytkownikowi, nawet gdy ten nie wykonuje żadnej akcji. Za pomocą mechaniki Game Loop będzie można również zmodernizować upływ czasu w ciągu rozgrywki.
- System drużyny - planowane jest dodanie funkcji pozwalającej na zrekrutowanie pomocników do walki. Ich statystyki będą losowane. Za ich pomoc, gracz będzie musiał oddawać część zdobytego podczas rozgrywki złota.
- Zadania poboczne - w grze występuje mechanizm zadań pobocznych, lecz są one predefiniowane. Planuję stworzenie mechanizmu do generowania zadań z kilku wariantów. Zadania utworzone w ten sposób można byłoby rozpocząć, wchodząc w interakcje z tablicą ogłoszeń. Ważnym elementem zadań pobocznych będzie również dziennik. W nim będzie można przeczytać opis fabularny, zadania oraz wszystkie niezbędne informacje. Planowana jest również możliwość przekazania graczowi przedmiotów w formie nagrody lub czasowej premii do którejś ze statystyk postaci.
- System walki - aktualnie system walki polega tylko na wymianie ciosów pomiędzy postacią gracza oraz jego przeciwnikiem. Jest to mało satysfakcjonujące, dlatego planuję dodać możliwość użycia przedmiotów lub umiejętności. Przedmioty będą musiały być przypięte do pasa postaci, aby zostały użyte w ciągu jednej tury. Otwarcie ekwipunku i przemieszczenie przedmiotów na pas będzie zużywało turę gracza. Planowane są dodatkowe efekty nakładane na, posiadaną przez gracza, broń lub losowane podczas wymiany ciosów. Podczas działania efektu, przeciwnik będzie otrzymywał dodatkowe obrażenia lub zostanie czasowo wyłączony z walki.
- Przeciwnicy - planowany jest podział przeciwników na kategorie. Podział będzie miał na celu stworzenie grup przeciwników podatnych na konkretny rodzaj ataku oraz niewrażliwych na inny.
- Dodatkowe lokacje - rozgrywka toczy się wokół jednego miasta, lecz miłym dodatkiem byłaby możliwość odwiedzenia dodatkowych miejsc, niezwiązanych z głównym miastem. Generowałyby się tam unikalne przedmioty oraz przeciwnicy do pokonania.
- Lochy - podobnie jak w przypadku dodatkowych lokacji, lochy również są predefiniowane. Ważnym elementem powinny być lochy generowane losowo. Dzięki takiemu rozwiązaniu, zamiast 10 poziomów lochów, można by było generować ich więcej, lecz mogły by one nie być tak szczegółowe.
- Wytwarzanie przedmiotów - podczas rozgrywki gracz zbiera przedmioty, z których będzie można wytwarzać inne przedmioty o specjalnych właściwościach. Będą one wytwarzane za pośrednictwem specjalnego menu. Gracz nie będzie miał możliwości wytworzenie niektórych z nich przed osiągnięciem konkretnego poziomu swojej postaci.
- Postaci niezależne - planowana jest przebudowa mechanik postaci niezależnych, by korzystały jeszcze lepiej z napisanych raz mechanik, przykładowo: handlu przedmiotami. Planowana jest również możliwość wytwarzania przedmiotów oraz wchodzenie w relacje towarzyskie z postaciami niezależnymi.

- JSON - planowane jest przeniesienie części najważniejszych danych z plików tekstowych do plików typu JSON. Utrudni to manipulowanie niektórymi mechanikami przez klienta końcowego aplikacji.
- Testowanie - planuję rozpowszechnianie swojej gry wśród znajomych, celem wykrycia jak największej ilości błędów. Każdy człowiek myśli oraz działa odmiennymi kategoriami, więc działania poszczególnych użytkowników mogą się różnić i dzięki temu sprawdzić więcej funkcji gry.
- Tłumaczenie - w momencie ukończenia gry planowane jest również przetłumaczenie jej na inne języki, między innymi na język Polski. Gracz będzie mógł zmienić wersję językową podczas uruchamiania gry.

3.14. Podsumowanie

W tym rozdziale przedstawiłem implementację najważniejszych mechanik zarządzających grą. Zaprezentowałem ich strukturę oraz poszczególne cechy. Rozdział ten może być również bardzo pomocny ewentualnym programistom przy pracy z kodem źródłowym. W treści rozdziału zamieściłem również moje plany związane z rozbudową oprogramowania jaką jest gra.

Rozdział 4. Zakończenie

Pisząc grę, moim celem było stworzenie produktu, w który sam chciałbym zagrać w wolnym czasie. Cel ten zrealizowałem w niecały rok. Gra jest dość mocno rozbudowana, posiada liczne mechaniki związane z grami RPG oraz pozwala zanurzyć się w świecie wyobraźni. Poszczególne mechanizmy gry zostały przeze mnie opisane w sposób szczegółowy i pozwalający je dobrze zrozumieć. Kod aplikacji jest łatwo rozszerzalny i pozwalający na przebudowę struktury. Bazując na recenzjach użytkowników dowiedziałem się, że gra posiada elementy wykonane w stopniu dobrym oraz inne które należy poprawić. Myśląc o przyszłości, planuję wciąż rozwijać oraz udoskonalać swoją grę. Pomysłów na rozszerzenie jej zawartości jest bardzo dużo. Jest również możliwe powstanie całkowicie nowej gry, jako wersji poprawionej, tzw. Remake [21].

Pisanie gier jest procesem niezwykle ciężkim, pracochłonnym, ale również bardzo satysfakcjonującym. Możliwość rozwoju, jaką dała mi zaprezentowana w tej pracy gra, jest ogromna. Z każdym dniem zauważyłem wzrost umiejętności nie tylko programistycznych, ale również analitycznych czy językowych. Możliwości jakie idą za zdobywaniem nowych osiągnięć programistycznych, pchały mnie na przód. Chwilowe porażki pokazywały mi zagadnienia nad którymi muszę jeszcze popracować.

Chciałbym podziękować swojemu promotorowi, Panu Doktorowi Pawłowi Łubniewskiemu za pomoc w realizacji pracy oraz rozbudzenie pasji jaką jest programowanie.

Rozdział 5. Bibliografia

[1] (2020, Maj) Wikipedia, Gra fabularna. https://pl.wikipedia.org/wiki/Gra_fabularna

[2] (2020, Maj) Wikipedia, Cosplay. https://pl.wikipedia.org/wiki/Cosplay

[3] Khaki. gryfabularne.pl, strona poświęcona graczom. http://www.gryfabularne.pl/#

[4] Robert Madejowicz Marcin Maślanka. (2020, Maj) Encyklopedia Zarządzania, Encyklopedia internetowa. https://mfiles.pl/pl/index.php/Metodyka_Extreme_Programming

[5] (2019, Grudzień) Wikipedia, Fantasy. https://pl.wikipedia.org/wiki/Fantasy

[6] (2019, Wrzesień) Wikipedia, Bohater Niezależny. https://pl.wikipedia.org/wiki/Bohater_niezale%C5%BCny

[7] TechTerms. (2020, Styczeń) techterms.com, Portal Technologiczny. https://techterms.com/definition/system_requirements

[8] Michał Aleksandrowicz. (2020, Maj) Github, Projekt Gerka. https://github.com/AleksyPL/Gerka

[9] (2020, Styczeń) Pomoc techniczna Microsoft. https://support.microsoft.com/pl-pl/help/815065/what-is-a-dll

[10] (2020, Marzec) Wikipedia, Windows legacy audio components. https://en.wikipedia.org/wiki/Windows_legacy_audio_components

[11] (2017, Marzec) Wikipedia, Interfejs tekstowy. https://pl.wikipedia.org/wiki/Interfejs_tekstowy

[12] William McBrine Mark Hessling. (2019, Wrzesień) pdcurses, Strona producenta oprogramowania. https://pdcurses.org/

[13] fanfare00. (2014, Wrzesień) J. Donald McCarthy's Blog, Portal technologiczny. https://jdonaldmccarthy.wordpress.com/2014/09/05/how-to-set-up-pdcurses-in-visual-studio-2013-c/

[14] Praca zbiorowa, *C++ Notes For Professionals*.: Goalkicker, 2018.

[15] (2020, Maj) Wikipedia, Programowanie obiektowe. https://pl.wikipedia.org/wiki/Programowanie_obiektowe

[16] (2018, Październik) Wikipedia, Strategiczna gra turowa. https://pl.wikipedia.org/wiki/Strategiczna_gra_turowa

[17] (2011, Czerwiec) GryOnline, Internetowe kompendium gier. https://www.gry-online.pl/slownik-gracza-pojecie.asp?ID=66

[18] Prometheus. (2012, Listopad) gameplay.pl, Strona poświęcona graczom. https://gameplay.pl/news.asp?ID=72793

[19] Rachit Belwariar. geeksforgeeks, Portal technologiczny. https://www.geeksforgeeks.org/a-search-algorithm/

[20] Robert Nystrom. (2014) gameprogrammingpatterns, Portal technologiczny. https://gameprogrammingpatterns.com/game-loop.html

[21] Łukasz Wrzalik. (2019, Czerwiec) g.e.e.x, Strona poświęcona graczom. https://geex.x-kom.pl/wiadomosci/remaster-remake-reboot-czym-sie-od-siebie-roznia/