Programowanie Komputerów

AEiI

Prosta gra RPG v2

(Rozwinięcie Interfejsu Graficznego + QOL)

Autor	Mateusz Górecki
Prowadzący	dr hab. inż., prof. PŚ Roman Starosolski
Rok akademicki	2021/2022
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	4
Termin laboratorium	Wtorek 11:45-13:15
Sekcja	21

1.Temat

Rozwinięcie projektu(**gry RPG**) utworzonego podczas kursu PK2 o **interfejs graficzny** oraz rozbudowanie projektu o różnego typy **QOL** rozwiązania, które niebyły jeszcze dostępne w starej wersji gry.

2.Analiza tematu

Interfejs graficzny pozwola na **przyjemniejszą** oraz **łatwiejszą** względem tekstowej wersji rozgrywkę. Dodając w miejsce konsolowego wyboru opcji(np. od 1-5) rozbudowany interfejs zaopatrzony w **przyciski**, **elementy przeciągane**(np. przedmioty w ekwipunku), oraz ogólne graficzne przedstawienie rozgrywki.

W wyżej wymienionym celu zdecydowałem się na użycie biblioteki **SFML** dzięki której byłem wstanie utworzyć wypełni funkcjonalny system **UI**.

Rozbudowaniu/polepszeniu uległ również proces tworzenia nowego konta(przy pomocy biblioteki **<regex>** dodanie **walidacji złożoności hasła**) oraz mechanizm odczytu plików przy pomocy (biblioteki **<filesystem>**).

Obszernie została również użyta biblioteka **<ranges>**, która pozwoli nam na łatwiejsze przemieszczanie po strukturach przechowywujących obiekty oraz pomogła w procesie zawijania tekstu w polach go wyświetlających.

Nowo dodane zaś klasy zostały załączone w postaci (dodanych w c++20) **modułów**.

3.Specyfikacja zewnętrza

a. Opis

1 . **Interfejs początkowy**: Logowanie, Tworzenie nowej postaci

Tworzenie postaci/konta: login, hasło, wybór ras (z podanych posiadających różne statystyki), klasy (z np. 3 Wojownik/Czarodziej/Zwiadowca).

Logowanie: za pomocą loginu i hasła.

Po zalogowaniu/stworzeniu postaci: Gracz trafia do właściwego **interfejsu rozgrywki.**

2. **Interfejs rozgrywki**: Quest, Kupiec, Lochy, Podgląd postaci

Quest: spotkanie **przeciwnika**/losowego zdarzenia(event) dający **przedmiot-złoto-doświadczenie.**

Kupiec: oferuje kilka(2-3) **przedmiotów** które możemy od niego zakupić za odpowiednią ilość **złota** (oferta zmienia się po każdym wyruszeniu na **Quest'a**).

Lochy: miejsce w którym możemy podjąć walkę z potężniejszymi przeciwnikami(**boss**) za, których pokonanie otrzymujemy lepsze **nagrody** lub legendarne **przedmioty**. Po pokonanie odpowiednie ilości(ok. 10) przeciwników na danym poziomie lochów, odblokowujemy jego (lochu) kolejne **poziomy** z jeszcze potężniejszymi przeciwnikami.

Podgląd postaci: Daje możliwość podglądu naszej postaci, ubieranie/wymianę wyposażenia/oręża (broni, hełmu, zbroi itd.).

3. **Systemy rozgrywki**: postać gracza, przeciwnicy, system walki, przedmioty

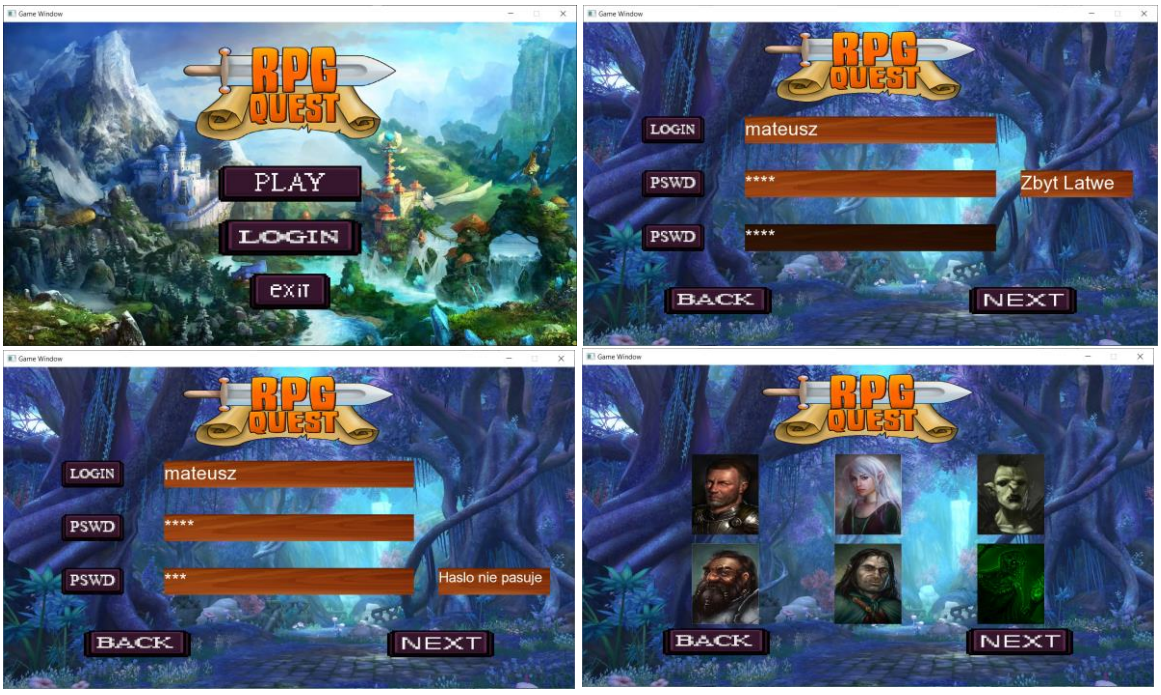
Postać gracza: posiada odpowiednie **statystyki** (zależne od wybranej **klasy i rasy**, **poziomu** i posiadanego **ekwipunku**), może posiadać założone przedmioty oraz przechowywać do **6** nie założonych przedmiotów ekwipunku.

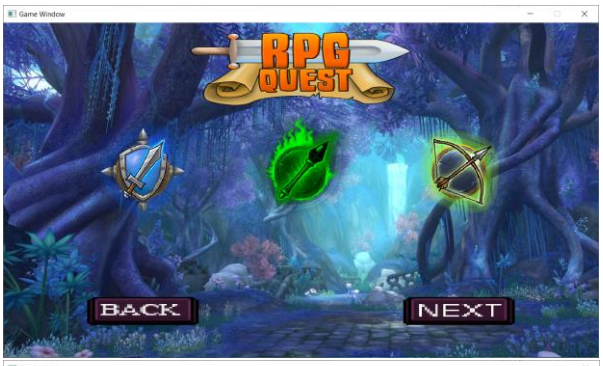
Przeciwnicy: analogicznie do postaci gracza posiadają odpowiednie **statystyki** (tu zależne już tylko od **poziomu** i typu(**klasy i rasy**) przeciwnika) oraz odpowiedni dla swojego typu(**klasy**) rodzaj ataków, z ekwipunku może posiadać tylko broń na potrzeby swojej **klasy.**

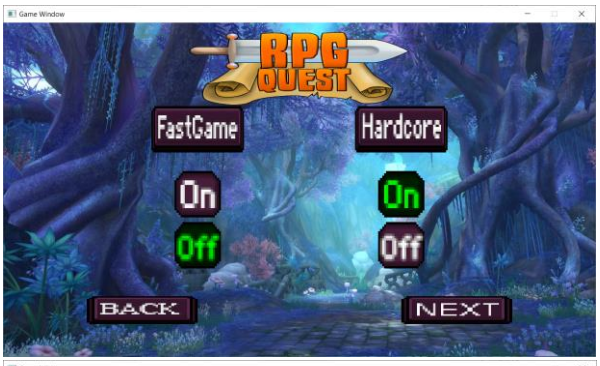
Przedmioty: posiadają nazwy, odpowiednie **statystyki** (zwykłe: 2-3, legendarne: 3+, dla broni jeszcze dodatkowo zakres zadawanych obrażeń), cena za jaką możemy go sprzedać **kupcowi.**

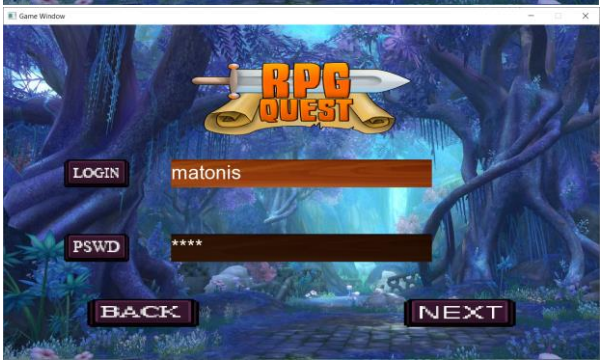
System walki: na początku losowanie kto zaczyna, następnie na podstawie **statyk** kolejno wyprowadzane są ataki **naszej postaci** i **przeciwnika** do momentu utraty całego zdrowia przez **przeciwnika** lub **naszą postać** .

b. Graficzne przedstawienie

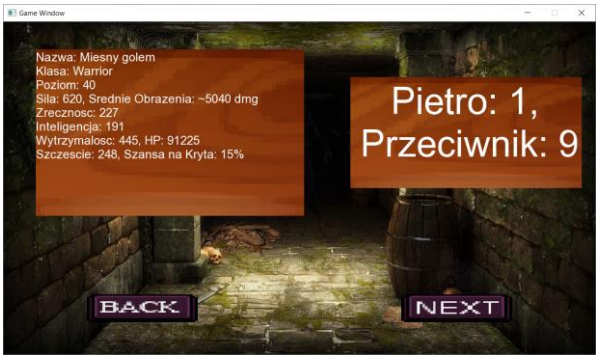



RPG QUEST
BACK
NEXT

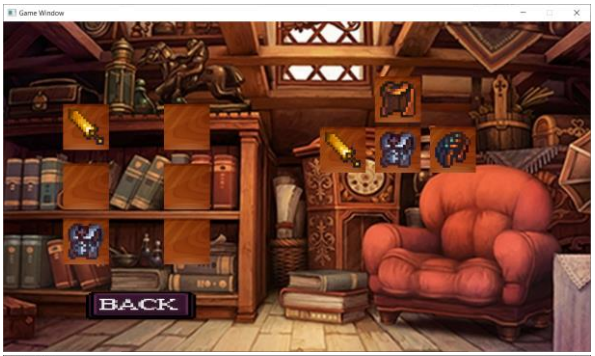

RPG QUEST
FastGame
Hardcore
On
Off
On
Off
BACK
NEXT

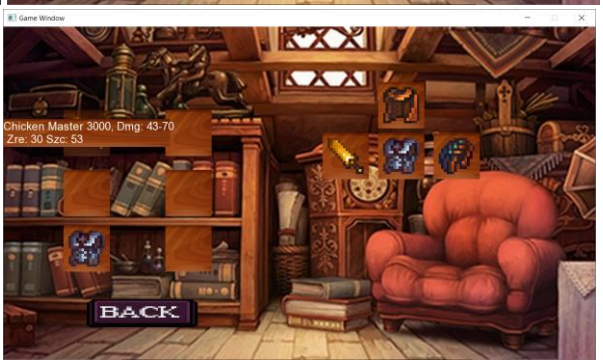

RPG QUEST
LOGIN
matonis
PSWD
****
BACK
NEXT


EXIT

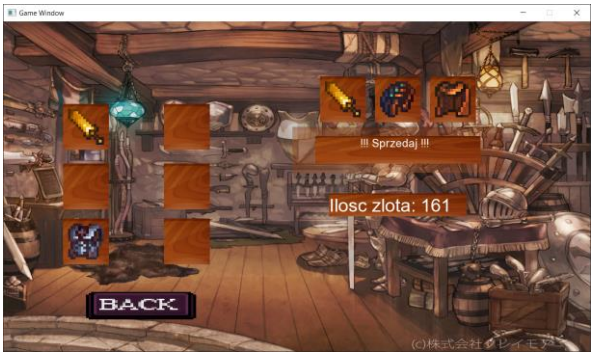

Nazwa: Miesny golem
Klasa: Warrior
Poziom: 40
Sila: 620, Srednie Obrazenia: ~5040 dmg
Zrecznosc: 227
Inteligencja: 191
Wytrzymalosc: 445, HP: 91225
Szczescie: 248, Szansa na Kryta: 15%
Pietro: 1,
Przeciwnik: 9
BACK
NEXT

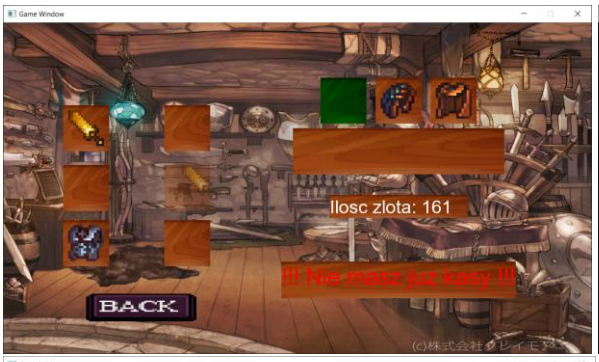

Rasa: Ork, Klasa: Hunter
Poziom: 27, Doswiadczenie: 36985 / 40500
Sila: 96
Zrecznosc: 388, Srednie Obrazenia: ~2189 dmg
Inteligencja: 99
Wytrzymalosc: 354, HP: 39648
Szczescie: 142, Szanasa na trafienie krytyczne: 13%
Pancerz: 387, Redukcja: 14%
Zloto: 161
BACK
NEXT

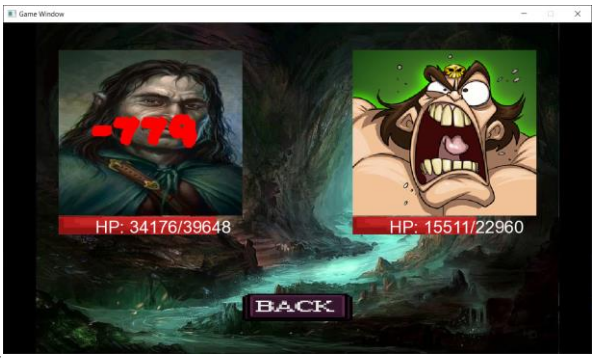

BACK

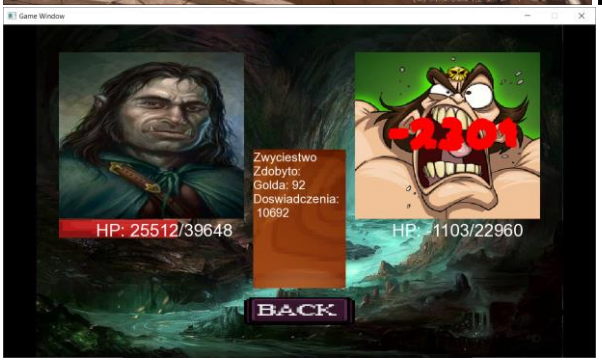

Chicken Master 3000, Dmg: 43-70
Zre: 30 Szc: 53
BACK


1 Do sklepu
2 Idz grac w kosci
3 Do lasu
Quest: Musisz pokonac Wielkiego Koscianego Golema.
Exp: 10692
Gold: 92
Enemy:
Nazwa: Zlowrogi demon
Klasa: Warrior
Poziom: 27
Sila: 180, Srednie Obrazenia: ~1121 dmg
Zrecznosc: 199
Inteligencja: 156
Wytrzymalosc: 164, HP: 22960
Szczescie: 144, Szansa na Kryta: 13%
BACK
NEXT


!!! Sprzedaj !!!
Ilosc zlota: 161
BACK



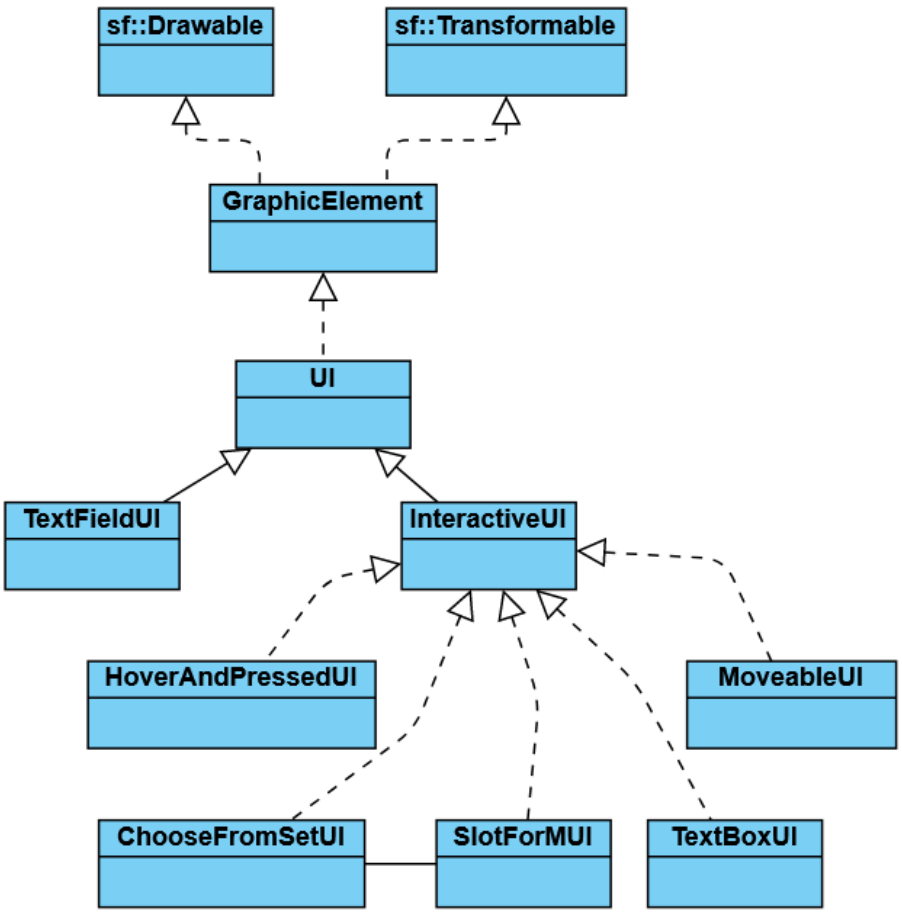




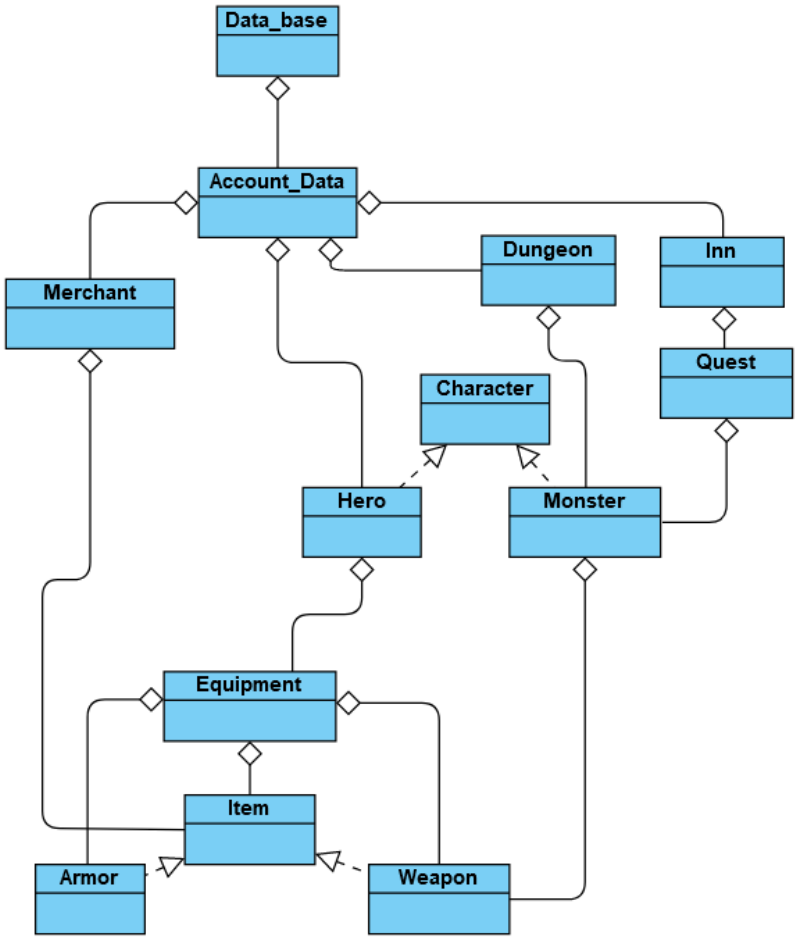
4.Specyfikacja wewnętrzna

a. Diagramy klas

- **Systemu UI**



- **Systemu Rozgrywki**



b. Opis najważniejszych klas

- **Systemu UI**

GraphicElement – klasa abstrakcyjna (**dziedzicząca z klas biblioteki SFML,** dzięki czemu łatwiej wyświetlać na ekranie obiekty dziedziczący z niej klas) pozwalającej na przechowywać podstawowe informacje o elemencie graficznym. Dziedziczy z niej klasa odpowiedzialne za **UI** (oraz w przyszłości w ramach rozbudowy rozgrywki inne elementy graficzne nie związane z interfejsem użytkownika).

UI – klasa odpowiedzialna za element „**User Interfejs'u**" w założeniu jest to **statyczny element**, zawierający jakiś kształt i texture. Klasami z niej dziedziczącymi jest element zawierający dodatkowo tekst(**TextFieldUI**) oraz klasa **InteractiveUI** odpowiedzialna za interaktywne elementy interfejsu.

InteractiveUI – klasa odpowiedzialna za interaktywne elementy interfejsu użytkownika, zawierająca podstawowe funkcja takie jak: sprawdzenie czy kursor myszy znajduję się na obiektem, zmianę tekstury, czy koloru. Dziedziczą z niej klasy odpowiedzialnie za przyciski pojedyncze(**HoverAndPressedUI**), wyboru z kilku opcji(**ChooseFromSetUI**), pola do wprowadzania tekstu(**TextBoxUI**) oraz elementy przeciągane(**MoveableUI**, **SlotForMUI**).

- ## Systemu Rozgrywki

  ***(jest zamieszczony w załączonym na końcu sprawozdaniu)**

c. Opis najważniejszych struktur

- ## Systemu UI

  Najważniejszymi strukturami **sytemu UI** są dwa **kontenery**(tu **vector'y**) przechowujące kolejno elementy **nieinteraktywne** oraz **interaktywne** interfejsu użytkownika. Przy ich pomocy dzięki przechowywaniu informacji o obecnie wyświetlanym **numerze** ekrany(w kodzie nazywanym **Layer'em**) możemy sprawnie **wybrać odpowiednie** elementy do narysowania na ekranie oraz w łatwy sposób **obsłużyć** wszystkie wymagane dla nich interakcje.

- ## Systemu Rozgrywki

  ***(jest zamieszczony w załączonym na końcu sprawozdaniu)**

d. Wykorzystane technik obiektowe

- **Dziedziczenie** – wykorzystane pomiędzy utworzonymi klasami **UI**, ułatwia ono tworzenie coraz to nowych bardziej rozwiniętych klas oraz jednoczenie nie powtarzanie tego samego kodu. Pozwoliło ono również na wykorzystanie funkcjonalności klas biblioteki SFML(**sf::Drawable** i **sf::Transformable**)
- **Polimorfizm** - wykorzystany został przeze mnie głównie podczas przechowywania utworzonych różnego rodzaju(klas) obiektów UI przechowywanych w dwóch vektor'ach wskaźników na klasy bazowe, dzięki którym następnie w łatwy sposób obsługuje ich wyświetlanie oraz wspólne interakcji na działania użytkownika.
- **Ranges** – w moim projekcie użyłem ich głownie do ograniczenia popranych na potrzeby pętli (for(&auto:…)) elementów vektor'a przy pomocy **ranges::views** oraz odpowiednio utworzonych filtrujących funkcji lambda.

```
for (auto& x : interfaceInteractiveUI[4] | std::views::filter(viewsFuncChooseFromandIsPressed))
```

- **Regex** – został wykorzystany w celu sprawdzania poprawności i złożoności(**jedna duża i mała litera oraz jedna cyfra i znak specjalny**) hasła wprowadzanego przez użytkownika. Użyłem go również w procesie zawijania tekstu wykorzystując regex::replace (dodając w odpowiednie miejsca '\n').

```
std::regex txt_regex("^(?=(.*[a-z]){1,})(?=(.*[A-Z]){1,})(?=(.*[0-9]){1,})(?=(.*[!@#$%^&*()\-__+.]){1,}).{4,}$");

t_p = tmp;
if (std::regex_match(tmp, txt_regex))
```

- **Filesystem** – został wykorzystany do tworzenia ścieżek do plików/zasobów potrzebnych do poprawnego uruchomienia rozgrywki.

- **Moduły** – zostały użyte jako zamiennik plików(**.h i .ccp**) dla nowo dodanych do projektu klas UI, co pozwala w przyszłości na łatwiejsze ich wykorzystanie w innych projektach.

e. Podstawowy schemat działania programu

*** Znacząco opisany w podpunkcie 3.a oraz przedstawiony w 3.b**

5.Testowanie i uruchamianie

Podczas długiego procesu tworzenia programu testowałem kolejno dodawane do niego funkcjonalności nowo dodanych elementów UI. W trakcie tych testów zlokalizowałem wiele błędów, z których wszystkie **udało mi się pomyślnie wyeliminować**. Najwięcej z nich dotyczyło funkcjonowania klasy odpowiedzialnej za **przesuwane elementy**(występowały tu błędy głownie z powrotem elementu na swoje miejsce/"slot" oraz sczepianie się elementów w jeden) oraz klasy odpowiedzialnej za **wyświetlanie tekstu**(tu tak gównie problem zawijania).

6.Wnioski

Dzięki pracy nad tym projektem zdecydowanie wzrosły moje umiejętności w zakresie wykorzystania nowo dodawanych w standardzie bibliotek takich jak: **ranges**, **filesystem** czy modułów. Miałem okazję również lepiej zapoznać się tworzeniem programu w **wersji graficznej**(nie konsolowej) i co za tym idzie lepiej poznać użytą w projekcie bibliotekę **SFML**.

7.Link do sprawozdania z projektu Gra_RPG (PK2)

https://polslpl-my.sharepoint.com/:b:/g/personal/mategor195_student_polsl_pl/ESZPkuv6QzNAnX0FqurKlicBRPBPuLh5o-OHEY5E3yk8WQ?e=pzzxND