**Dokumentacja projektu**

Gra RPG oparta o system klient-serwer.

Wykorzystano C++ oraz bibliotekę SFML.

Autor: **Kamil Łuc**

Grupa: **3**

1. **Opis wykorzystanych metod:**

Kreator postaci to klasa odpowiedzialna za stworzenie nowych bohaterów ich eksport i import, posiada następujące metody:

1. zapiszDoPliku – stworzona postać zostaje zapisana do pliku save.txt do folderu Images
2. wczytajZPliku – podobnie jak wyżej tylko w przeciwna stronę
3. kreator – konstruktor po kolei pytający o imię i poszczególne statystyki
4. kreator(tuParametry) – drugi konstruktor, przypisujący statystkom wartości zapisane w parametrach.

Chat tu działam na funkcjach bezpośrednio napisanych przed main, są to:

1. Server – nawiązuje połączenie jako serwer
2. Client – analogicznie jako klient
3. GetInput – czyta znaki do klawiatury i zapisuje do stringa
4. DoStuff – odpowiada za pakiety i sockety, wysyła dane
5. MainChat – główna funkcja, łączy pozostałe, tworzy osobny wątek do chatu

Pozostała część aplikacji jest umieszczona w main nie są już do metody czy funkcje więc zostaną opisane kolejnych punktach. Nie jest to dobra praktyka ale w przypadku średniej długości projektu napisanego przez jedną osobę ułatwia czytelność, dlatego zdecydowałem się na takie rozwiązanie.

1. **Opis środowiska programistycznego:**

Projekt został napisany w Microsoft Visual C++ 2010 Express, czyli użyty język to C++. Główną biblioteką a właściwie całym ich zbiorem był SFML, konkretnie wersja 2.3.2. Wykorzystano następujące moduły wcześniej wspomnianej technologii:

1. System – odpowiedzialny między innymi za wątki, w projekcie jest to osobny wątek dla chatu, który działa niezależnie w konsoli.
2. Window – odpowiada za sprawy związane z oknem programu jak jego wielkość, parametry ale również zdarzenia (eventy) jak wciśnięcie guzika myszy lub klawisza na klawiaturze
3. Graphics – pozwala na rysowanie na ekranie, wykorzystano go do importu tekstur, stworzenia sprite’ów, rysowanie bohaterów planszy itd.
4. Network – odpowiada za najważniejszy punkt projektu czyli obsługę sieci, dzięki niemu można łatwo stworzyć sockety, nawiązać połączenie, wysyłać i odbierać pakiety.

Dodatkowo wykorzystano takie biblioteki jak: string do napisów, cstdio do konwersji int na string, jak się okazało w trakcie budowy projektu taka możliwość została domyślnie dodana dopiero w standardzie C++ 11, iostream do wczytywania i wyświetlania danych na ekran, tu głównie chodzi o chat ale też kreację bohatera.

1. **Sposób realizacji tematu:**

Głównym tematem jest sieć więc to na niej skupię się w tym punkcie. Jak wspominałem wcześniej wykorzystałem tu bibliotekę SFML a konkretnie w tym przypadku moduł Network. Zawiera on wiele narzędzi do łatwego połączenia urządzeń opartego o sockety. Umożliwia zarówno połączenie przez tcp jak i udp, ja wybrałem tą pierwszą opcję by mieć gwarancję, że pakiet dotarł do odbiorcy. Poniżej omówię przykładowy wycinek kodu, reszta ruchu sieciowego w moim projekcie działa analogicznie:

*//stworzenie nowego socketu do gry*

*TcpSocket gsck;*

*if(serwer==true){*

*TcpListener listener;*

*listener.listen(2001);*

*listener.accept(gsck);*

*}*

*else*

*gsck.connect(IPADDRESS, 2001);*

*gsck.setBlocking(false);*

Na początku tworzę nowy socket który działa na protokole tcp, następnie w zależności od tego czy jestem klientem ( po prostu łącze się do tego socketu, podając ip i port) lub jeśli jestem serwerem tworze nowy listener który nasłuchuje na podanym porcie czy zostały przesłane jakieś pakiety, jeśli tak odbiera je.

*//przechowuje pozycje gracza psp serwer pcp klient*

*Vector2f psp, pcp;*

Zmienne przechowujące pozycję gracza, Vector2f jest kontenerem SFML więc nie muszę się martwić o różny rozmiar np. int na różnych systemach operacyjnych. Jednak i w takim przypadku jest rozwiązanie tj. stosowanie zmiennych z określonym rozmiarem np. Int32 zamiast int.

*//zczytuje pozycje serwera*

*psp=sprite1.getPosition();*

Do wcześniej omówionej zmiennej przypisuję aktualną pozycję gracza, w dalszej części kodu porównuję ją z aktualną jeśli się różni wysłam ją do drugiego komputera co umożliwia widok u niego naszej pozycji na planszy.

*//wyslanie i odbieranie pakietow*

*Packet pck;*

*if(psp!=sprite1.getPosition()){*

*pck<<sprite1.getPosition().x<<sprite1.getPosition().y;*

*gsck.send(pck);*

*}*

*gsck.receive(pck);*

*if(pck>>pcp.x>>pcp.y){*

*sprite2.setPosition(pcp);*

*}*

Tworzę obiekt klasy pakiet, do którego ‘wkładam’ obecne współrzędne mojej postaci tj. x,y. Wykorzystując wcześniej stworzony socket tj. ‘gsck’ wysyłam przy pomocy wbudowanej w SFML metody send do drugiego komputera. Dalej w tym miejscu odbieram pakiet od drugiego komputera o jego pozycji na mapie po ‘wypakowaniu’ pakietu zmieniam ją przy pomocy sprite.setPosition.

*/////////////////////*

*///chat w innnym watku*

*/////////////////////*

*Thread thread0 = (&mainChat);*

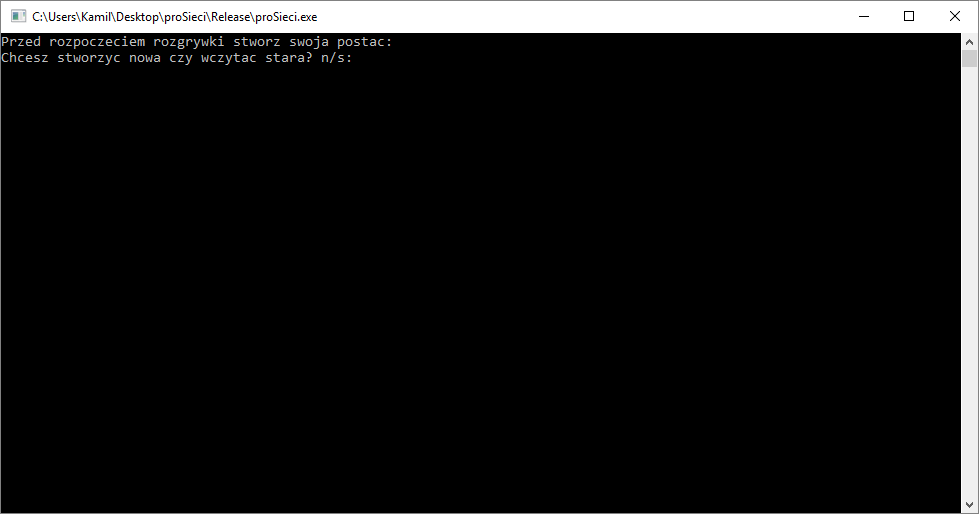
*thread0.launch();*

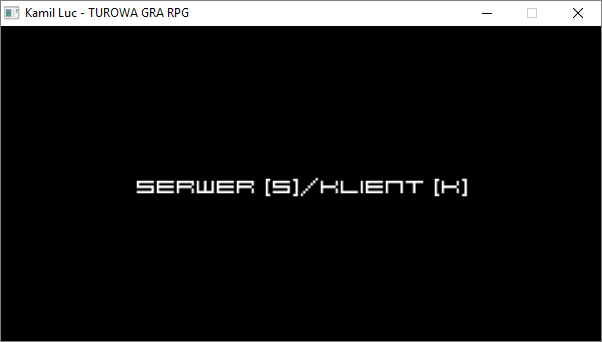
Tutaj krótkie i proste zastosowanie wątków z modułu System, po prostu tworzę nowy wątek który za argument przyjmuje referencję do funkcji obsługującej cały chat (została opisana wyżej). Polecenie x.launch(); odpala wątek x. Całość działa równolegle w oknie konsoli.

1. **Opis działania projektu:**

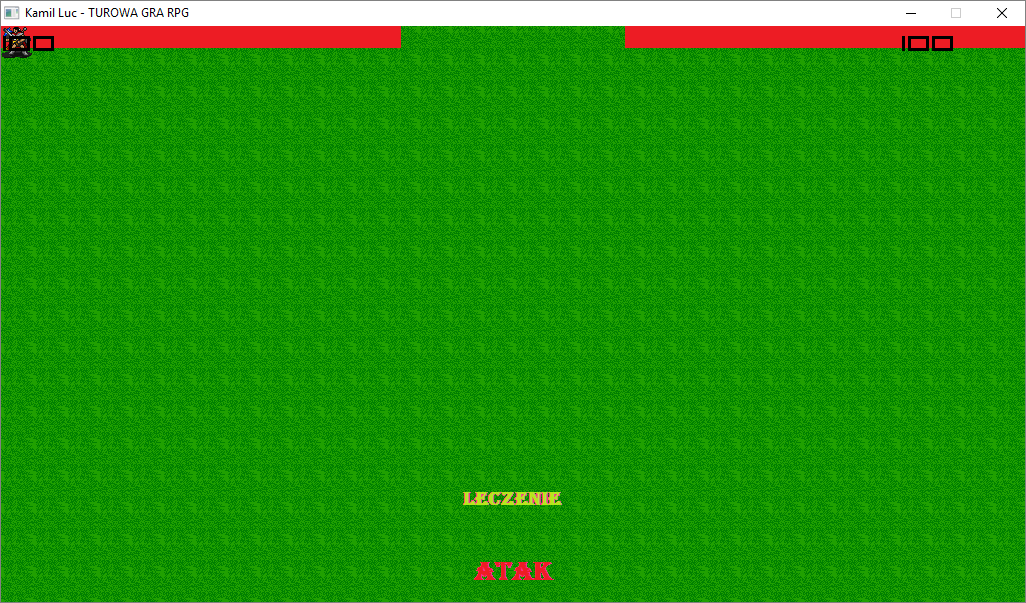
Na początku kopiujemy pliki z folderu Release czyli proSieci.exe oraz folder Images na dwa docelowe komputery lub ewentualnie jeden w przypadku działania na localhost. Po włączeniu aplikacji jako pierwsze pojawia się okno z zapytaniem o to czy chcemy stworzyć nowego bohatera w tym przypadku uruchomi się kreator postaci w konsoli, gdy ta zostanie utworzona zostaje zapisana do pliku, drugą opcją jest wczytanie danych poprzedniej. Po wykonaniu tych czynności uruchamia się okno z pytaniem czy chcemy być klientem czy serwerem, wybieramy z klawiatury ‘k’ dla tego pierwszego lub ‘s’ dla drugiego. Pierwszy powinien połączyć się serwer by to klient mógł dołączyć następnie do gry. Po wybraniu tego kim jesteśmy w przypadku serwera jesteśmy już gotowi a w przypadku klienta pojawi się nowe zapytanie o podanie numeru IP z których chcemy się połączyć po jego wpisaniu i zaakceptowaniu (klawisz enter) rozpoczyna się gra. Każdy z graczy ma swoją postać, może się nią poruszać klawiszami W S A D oraz wykonywać akcję wciskając lewym guzikiem myszy butony podpisane atak i leczenie na ekranie. Celem gry jest zbicie punktów życia przeciwnika do zera przy utrzymaniu swoich powyżej tej wartości. Działanie gry zostało przetestowane na różnych komputerach i konfiguracjach. Należy tu zauważyć jeden fakt, gdy gramy na localhoscie mamy do dyspozycji jedna klawiaturę więc akcję ‘czyta’ tylko to okno które jest aktywne w danym momencie. By wyłączyć aplikację klikamy w czerwony X w oknie konsoli, nie gry.

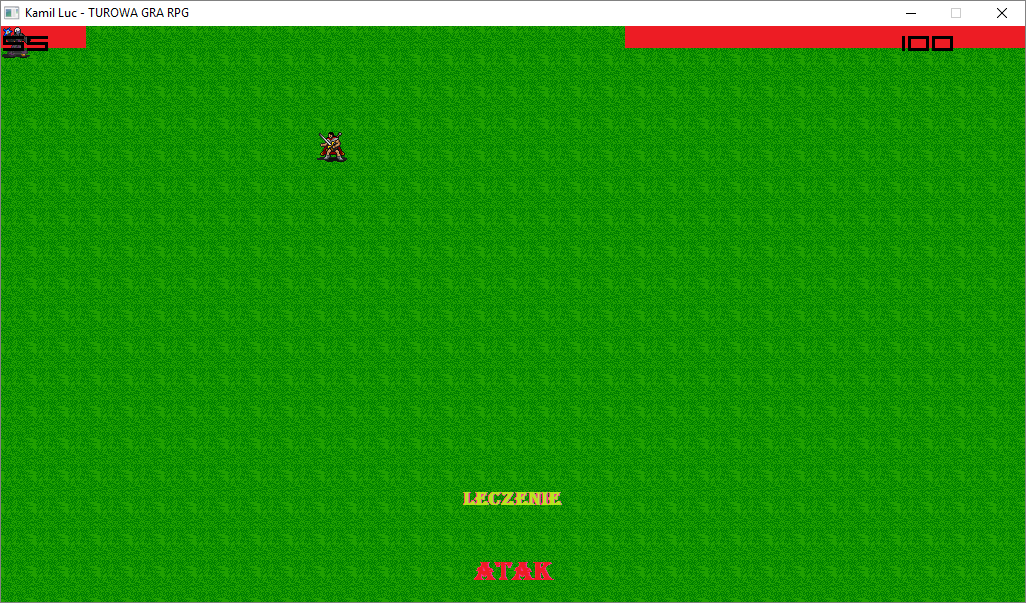
1. **Przykładowe zrzuty ekranu:**

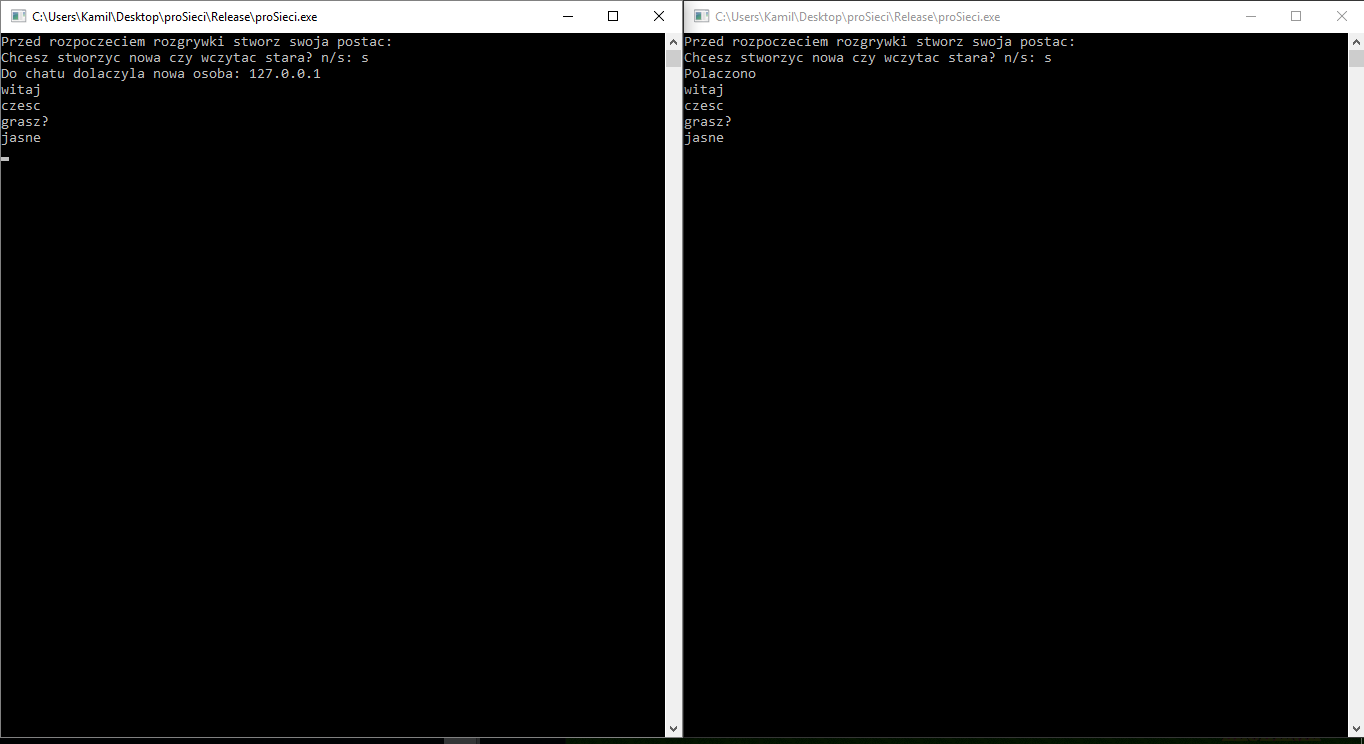
(Kreator postaci)

(Wybór tego kim jesteśmy serwer/klient)

(Pytanie o IP serwera, domyślnie wpisany localhost, można kasować znaki klawiszem backspace)

(Początek rozgrywki, widok planszy)

(Gra w trakcie, ubytek życia u 1 gracza)

****(Okna chatu oraz wycinek dialogu, okno chatu jest cały czas otwarte w tle,

równolegle do okna rozgrywki)