Marta Głowacka 160158
Mateusz Namysł 160198

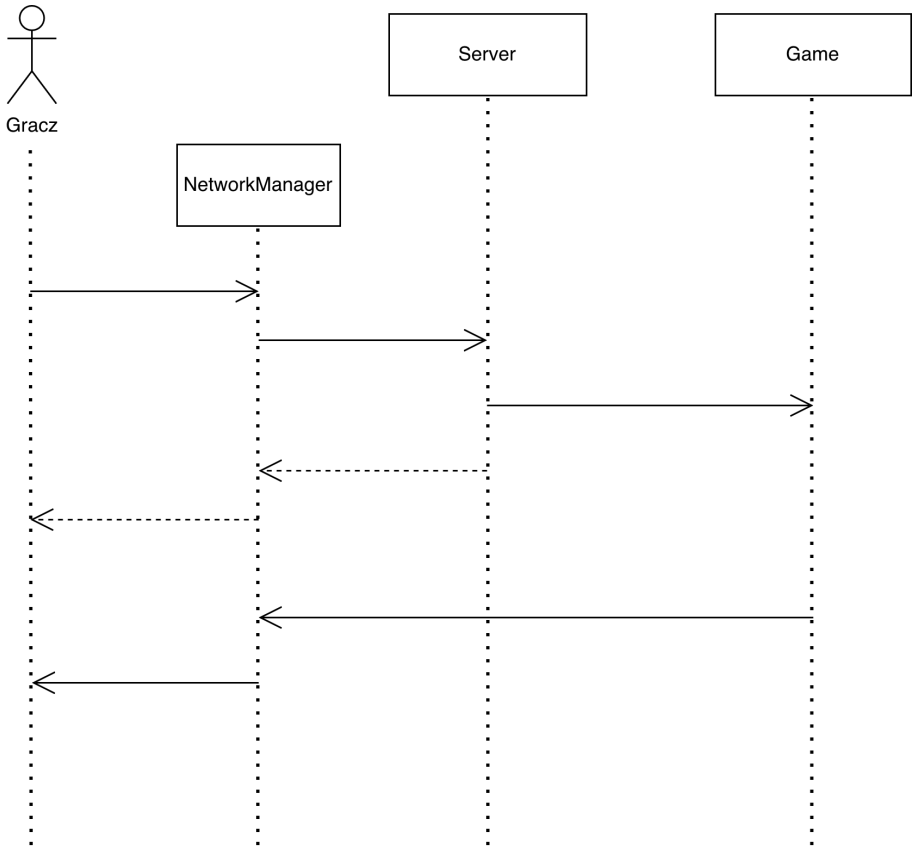
Sprawozdanie sk2

Klon gry Dixit

1. Opis projektu.

Nasz projekt jest implementacją gry karcianej Dixit, którą wykonaliśmy w C++. Do przygotowania GUI klienta sieciowego wykorzystaliśmy bibliotekę SFML. Zdjęcia kart pochodzą z oryginalnej gry. Celem gry jest zdobycie trzydziestu punktów. Na początku rozgrywki każdy z graczy dostaje 6 losowo wybranych kart. Następnie pierwszy gracz, który dołączył do pokoju zostaje narratorem - jako pierwszy wybiera swoją kartę oraz podaje słowo, które mu się z nią kojarzy. Następnie wszyscy pozostali gracze, na podstawie wskazówki otrzymanej od narratora wybierają spośród spośród swoich kart decydują, która najlepiej do niej pasuje. Kiedy wszyscy gracze zatwierdzają swój wybór następuje faza głosowania - wszyscy poza bajarzem wybierają, która karta została przez niego wybrana. Po oddaniu wszystkich głosów następuje podliczenie punktów.
Taki cykl powtarza się do momentu osiągnięcia warunków zakończenia rozgrywki. W przypadku rozłączenia się gracza gra toczy się do końca rundy- ostatnia runda nie jest punktowana.

2. Opis komunikacji pomiędzy serwerem i klientem.



Serwer działa na porcie 1100. Wiadomości wysyłane do serwera odbierane są przez network manager, z którego serwer za pomocą poll pobiera listę eventów do obsłużenia. Jeżeli jest to zdarzenie, na które odpowiedzieć ma bezpośrednio serwer, wysyła on wiadomośc przy pomocy network managera, jeżeli wydarzenie ma obsłużyć gra to przekazuje wiadomość do gry i to ona odpowiada poprzez network manager.

3. Podsumowanie

Kluczowymi elementami naszego projektu są rodzaje wysyłanych wiadomości i network manager. Pozwalają one na poprawną komunikację pomiędzy wszystkimi elementami systemu. Po stronie klienckiej ważną rolę pełni Texture Manager, który zapewnia poprawne wczytanie kart.

Fakt, że po połączeniu się z serwerem klient dostaje unikalne PlayerId, które pozwala na identyfikację, czy i w której grze jest gracz sprawia, że jeżeli nowy gracz dołączy na tym samym deskryptorze, co gracz, który zakończył połączenie zostanie poprawnie obsłużony.

Zagadnieniem, które sprawiło nam największą trudność była serializacja i deserializacja wiadomości. Ustalenie jednolitego formatu komunikacji było zadaniem nietrywialnym, ponieważ w zależności od konkretnej fazy gry wysyłane komunikaty różnią się od siebie, a zarówno serwer jak i klient muszą wiedzieć jakie akcje wykonać w danym momencie. Doprowadziło to do stworzenia uniwersalnego typu Message, który jest podstawą konkretnych komunikatów, które w zależności od typu posiadają inny payload, a funkcje odpowiednio serializer i deserializer pozwalają na wysyłanie i odczytywanie ich przez serwer i użytkowników końcowych.