Katarzyna, Jaromirska 155910

Łukasz, Grobelny 155898

Sprawozdanie sk2

Kalambury

1. Opis projektu (0.5 strony)

* Jako projekt stworzyliśmy popularną grę „Kalambury” w wersji komputerowej. Gracz ma możliwość utworzenia lobby poprzez podanie nazwy lobby, nazwy użytkownika w ramach pokoju oraz opcjonalnie hasła. Alternatywnie gracz może dołączyć do już istniejącego lobby.   
  W tym celu gracz podaje, przed dołączeniem do lobby, wybraną nazwę użytkownika oraz hasło, jeżeli takowe jest wymagane. Osoba, która stworzyła dane lobby może w każdej chwili rozpocząć rozgrywkę. Rozgrywka jest odwzorowaniem gry w kalambury: jedna osoba rysuje, reszta zgaduje. Osoba rysująca jest kolejno wybierana na podstawie kolejności dołączenia do lobby. Zadaniem osoby rysującej jest przedstawienie na białym płótnie wybranego, z pośród trzech podanych przez serwer, słowa używając dostępnych kolorów. Osoby zgadujące starają się odgadnąć hasło poprzez udostępniony im czat. Runda kończy się, gdy któryś z graczy odgadnie hasło.
* Do postawienia serwera, jak również stworzenia aplikacji klienta, posłużyliśmy się językiem C++. W celu zbudowania interfejsu graficznego gracza posłużyliśmy się popularną biblioteką SDL (Simple DirectMedia Library).

2. Opis komunikacji pomiędzy serwerem i klientem (0.5 strony, może być schemat/rysunek)

Komunikacja pomiędzy serwerem a klientem odbywa się na zasadzie prośba - odpowiedź (request - reply). Klient przesyła prośbę udostępnienia danych przez serwer, na co ten odpowiada odpowiednim zestawem danych, bądź informacją o powodzeniu/niepowodzeniu operacji (np. gdy lobby o podanej nazwie już istnieje). Jedynymi odstępstwami od tej reguły są przypadki udostępnienia przez klienta informacji o zmianie na płótnie, próby odgadnięcia hasła oraz wyboru hasła, w których to przypadkach serwer nie odpowiada, a jedynie dokonuje odpowiednich zmian w ramach pokoju.

Serwer oczekuje na strukturę Message, z której odczytuje informacje na temat typu wiadomości oraz opcjonalny rozmiar kolejnej wiadomości, w przypadku gdy klient udostępnia serwerowi dodatkowe informacje. Dostępne typy wiadomości zebrane są w enumeratorach MessageToServer oraz MessageToClient. Typy wiadomości są podstawą, na której serwer oraz klient wykonuje odpowiednie operacje. Dla przykładu, klient pragnący stworzyć lobby wpisuje nazwę pokoju, swoją nazwę użytkownika oraz opcjonalnie hasło do pokoju. Następnie poprzez potwierdzenie przyciskiem „CONFIRM” wysyła serwerowi chęć utworzenia lobby (struktura Message z typem wiadomości „CREATE\_LOBBY”) oraz podane informacje jako osobną wiadomość (struktura „ConnectInfo”). Nazwa pokoju, gracza oraz hasło posiadają wymagane do spełnienia ograniczenia. Nazwa lobby oraz gracza nie może być pusta. Dodatkowo nazwa lobby musi być unikalna. Jeśli klient nie spełni tych wymagań, serwer nie utworzy lobby, zwróci w zamiast tego wiadomość informującą o typie porażki (struktura Message z odpowiednim typem). Jeżeli dane spełniają wszystkie wymagania, serwer utworzy lobby, przypisze użytkownika do tego lobby i wyśle wiadomość potwierdzającą pomyślne utworzenie lobby (struktura Message z typem „CONFIRM\_CONNECT”). Klient po poprawnym wysłaniu danych oczekuje na odpowiedź (strukturę Message). Po odczytaniu typu zwróconej wiadomości dokonuje odpowiednich zmian.

3. Sposób uruchomienia

Aby uruchomić serwer oraz gracza należy skorzystać z systemu operacyjnego Linux.

* Serwer:

W celu uruchomienia serwera pobierz plik server.zip z zakładki Releases, następnie rozpakuj projekt oraz uruchom server.exe. Alternatywnie, aby samodzielnie skompilować serwer należy pobrać kod źródłowy z gałęzi „main”, a konkretniej foldery „global” oraz „server”. Będąc na poziomie „…/server” w wierszu polecenia należy użyć komendy:

g++ -std=c++17 -g src/\*cpp ../global/src/\*cpp -Wall -o server.exe

Następnie można uruchomić serwer przy pomocy pliku server.exe

* Klient/Gracz:

W celu uruchomienia gracza należy zaopatrzyć się w biblioteki:

* libsdl2-dev
* libsdl2-ttf-dev
* libsdl2-image-dev
* libsdl2-gfx-dev

(Możliwe do pobrania przy pomocy polecenia: apt-get install libsdl2-dev libsdl2-ttf-dev libsdl2-image-dev libsdl2-gfx-dev)

Proces uruchomienia klienta jest analogiczny do serwera, jednakże podczas uruchomienia klienta należy podać adres oraz port serwera jako argumenty wywołania. Należy pobrać plik client.zip z zakładki Releases, rozpakować go oraz uruchomić plik client.exe. Można także samodzielnie skompilować gracza pobierając foldery „global” oraz „client” z gałęzi „main”. Na poziomie „…/client” we wierszu polecenia należy wywołać komendę:

g++ -std=c++17 -g src/\*.cpp ../global/src/\*cpp -Wall -o client.exe -lSDL2 -lSDL2main -lSDL2\_ttf -lSDL2\_image -lSDL2\_gfx

Następnie można uruchomić gracza przy pomocy utworzonego pliku client.exe.

4. Podsumowanie (0.5-1 strona)

* Najważniejsze informacje o implementacji:

Połączenie klienta z serwerem jest połączeniem TCP. Aplikacja klienta została napisana w C++ z użyciem biblioteki SDL. Główne zadania aplikacji, takie jak wyświetlanie, obsługa wejść i logika gry obsługiwane są w odrębnych klasach zarządzających, które połączone są w jeden działający program w klasie GameManager. Projekt posiada wiele klas opisujących każdy z elementów widocznych w aplikacji, np. płótno, pole do pisania tekstu, przyciski itd., wszystkie dziedziczce z jednej klasy Component, a te elementy które reagują na wejścia dziedziczą z klasy Interactable (rozwinięcia klasy Component). Serwer podobnie do aplikacji klienta korzysta z pomocniczych klas zarządzających. Struktury te zarządzają połączonymi klientami, istniejącymi pokojami oraz hasłami do gry. Obie implementacje korzystają z jednoznacznie ustalonych struktur do przesyłania wiadomości.

* Problemy w implementacji, wpływ na pracę:

Głównym czynnikiem wpływającym na naszą pracę był czas dostępny na wykonanie projektu. Nadchodzący termin oddania projektu oraz inne projekty nad którymi pracę musielismy podjąć sprawiły, że niektóre elementy implementacji nie zostały dobrze przemyślane. Z pewnością istnieje możliwość poprawy kodu w celu osiągnięcia lepszej czytelności, bądź także lepszej wydajności.