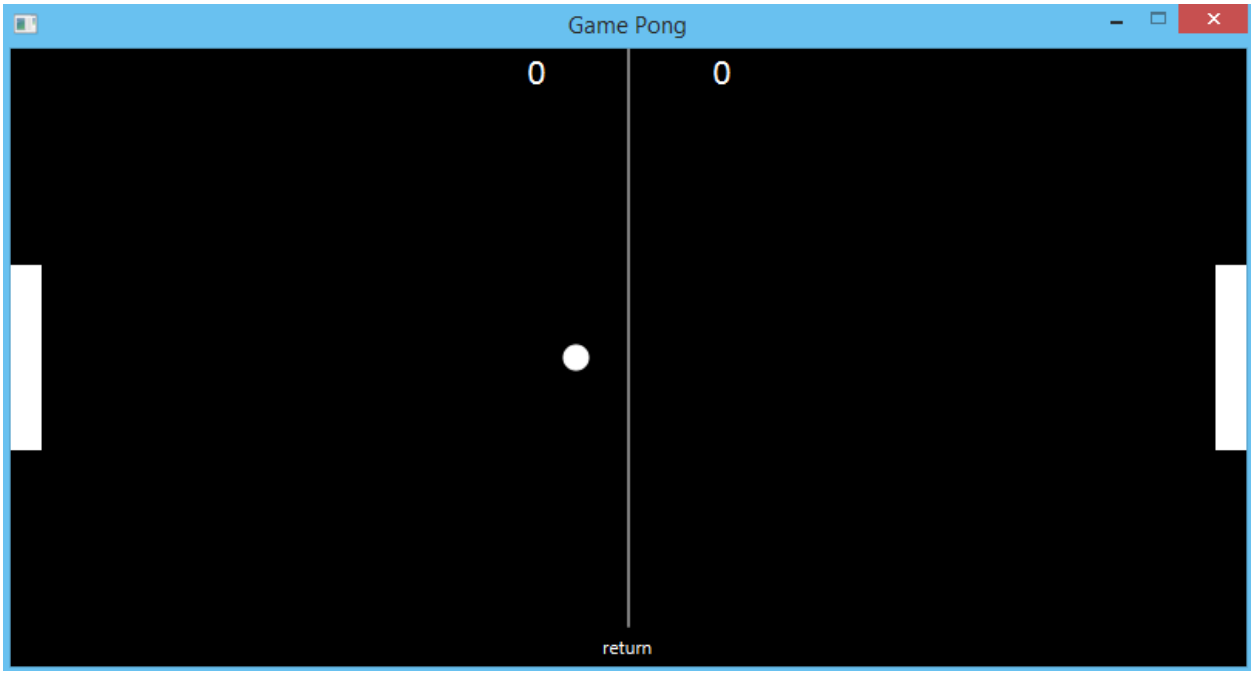
Pong – gra zręcznościowa

Twórcy: Kamil Kolmus i Tomasz Kuźnar

1 Wstęp

Gra Pong jest popularną grą zręcznościową polegającą na odbijaniu piłeczki od swoich paletek. Z racji prostoty piłeczką jest kulka a paletki to prostokąty poruszające się w górę i w dół(Rysunek 1). Gra była już na nowo tworzona przez wielu w przeróżnych postaciach. Naszą interpretację przedstawiamy w bardzo klasycznej formie. Całość jest napisana w języku Java w technologii *JavaFX*. Podstawą zabawy ma być pojedynek 1 na 1 między graczami na jednym komputerze, jak i poprzez połączenie internetowe. Łączność sieciową naszej aplikacji zapewni framework *Netty.io*.



Rysunek 1 – Widok z gry Pong

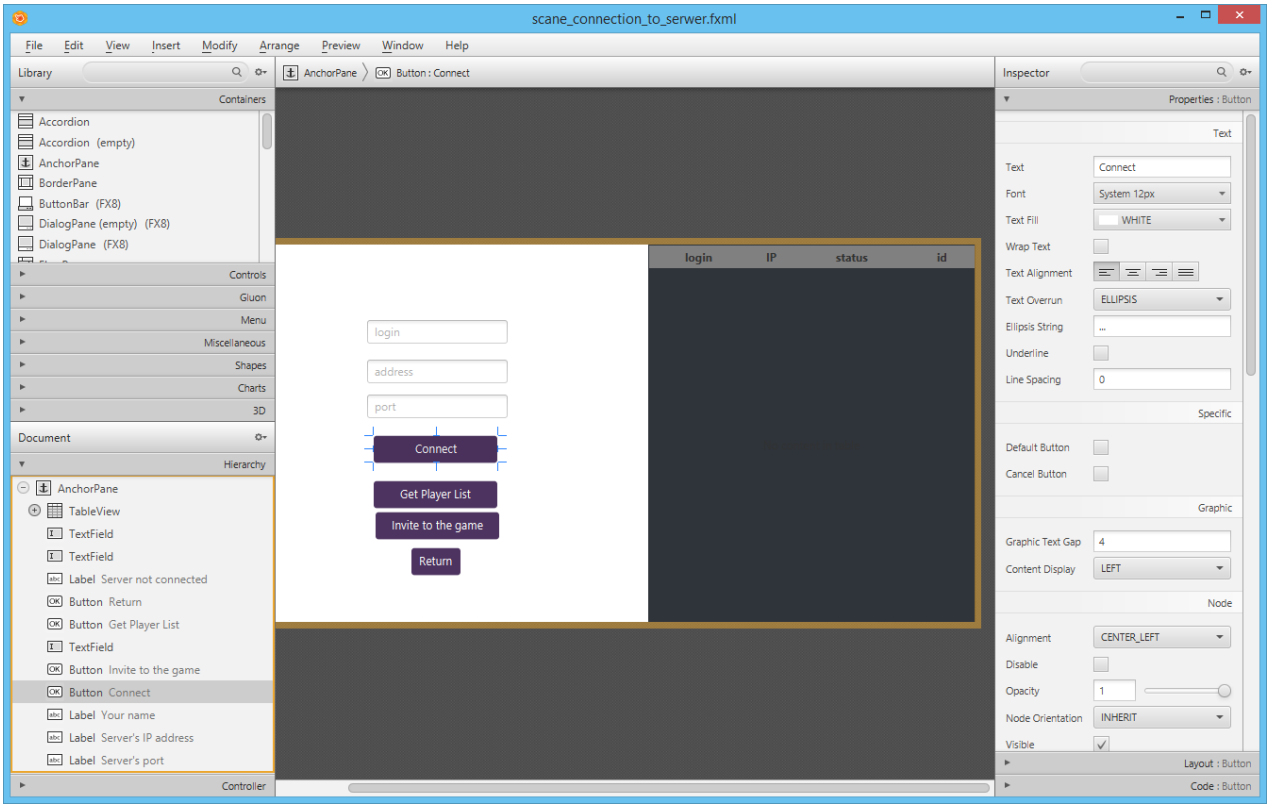
1.1 JavaFX i Scene Builder i pozostałe używane oprogramowanie

JavaFX jest platformą zaprojektowaną do tworzenia aplikacji internetowych. Sprawdza się także przy tworzeniu aplikacji desktopowych. Rozwiązanie to miało zastąpić bibliotekę GUI *Swing* dla Javy SE. Przy wykorzystaniu plików XML wraz z aplikacjami, takimi jak *Scene Builder* jest się w stanie tworzyć bardzo szybko estetyczne aplikacje. Kwestią gustu jest to czy programista zaprojektuje swoje GUI przy użyciu plików FXML czy klasycznie wykorzysta do tego kod Javy.

Scene Builder jest aplikacją *Drag'n'Drop(*Rysunek 2), dzięki której możemy w łatwy i szybki sposób utworzyć interfejs graficzny dla naszej aplikacji. Generuje ona pliki FXML, które wraz z Kontrolerem w języku java tworzą to co zobaczymy na ekranie komputra. Nic nie stoi na przeszkodzie aby tworzyć i edytować pliki XML samemu za pomocą edytora tekstowego. W przypadku małych zmian jest to na pewno szybsza metoda. Jednak, ciągły widok na tworzoną aplikację polepsza tempo w jakim ją tworzymy. Nasza aplikacja wykorzystuje dwie metody tworzenia GUI(kod w javie i pliki XML). Wybór sposobu był tylko i wyłącznie podyktowany decyzją piszącego.

Głównym narzędziem wykorzystywanym przy pisaniu kodu było IDE *IntelliJ IDEA*. Oprogramowanie firmy *JetBrains* zapewnia komfort przy pisaniu kodu, jak i organizacji projektu.

Narzędziem używanym do ułatwienia współpracy między obydwoma członkami zespołu był system kontroli wersji Git wraz z platformą *Github*, na którym umieszczona jest najnowsza wersja gry wraz z krótkim opisem.

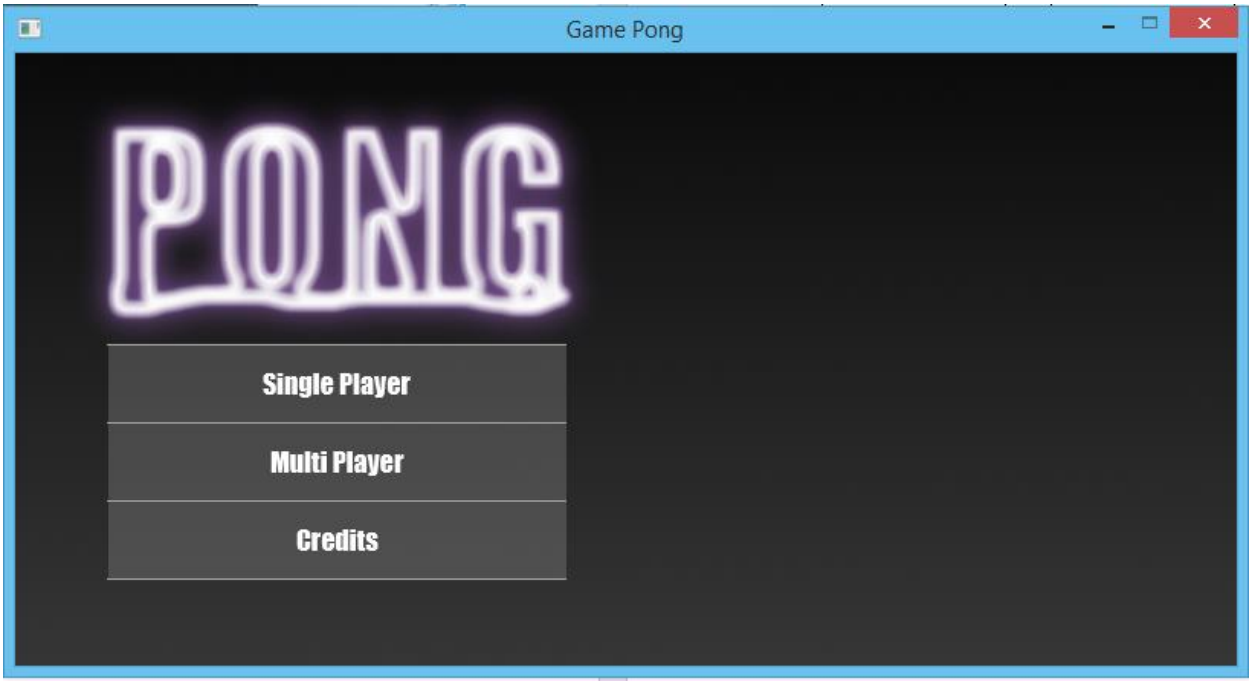


Rysunek 2 - Okno aplikacji Scene Builder

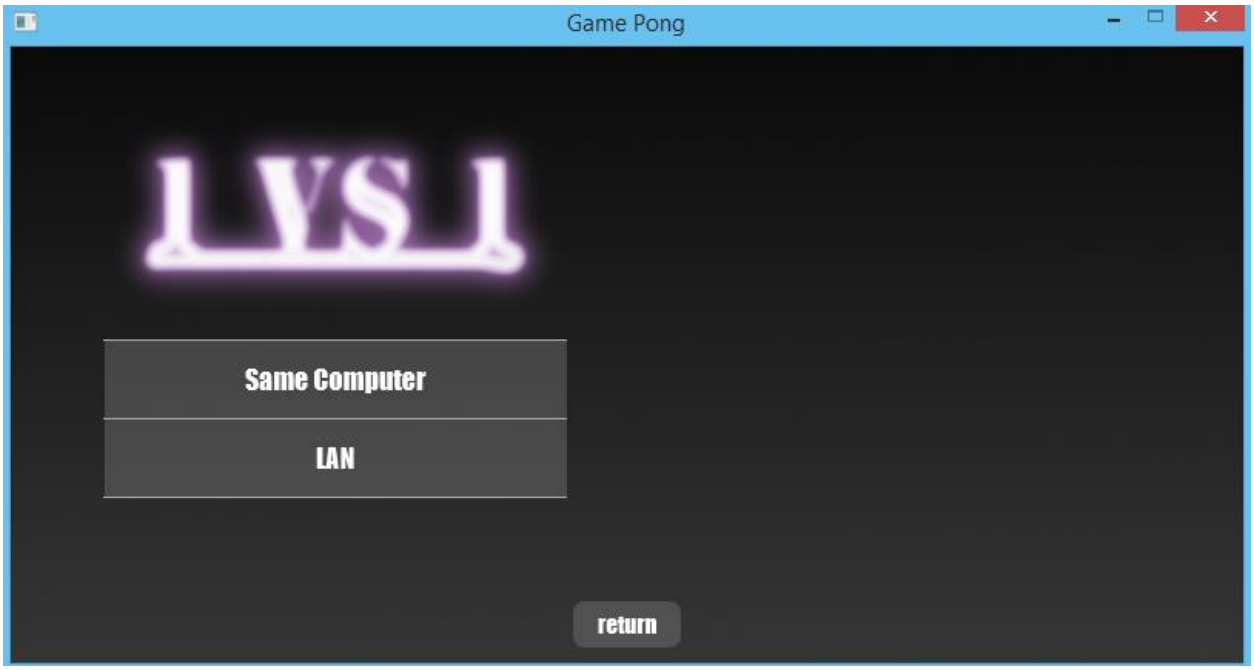
2 Opis Aplikacji

Uruchamiając grę Pong rozpoczynamy od standardowego menu głównego, które zawiera trzy przyciski i logo gry(Rysunek 3). Przycisk *Single Player* przenosi nas do widoku gry(Rysunek 1). Widok jest taki sam dla wszystkich trybów gry. W grze dla pojedynczego gracza zmierzymy się z botem. Wybór opcji *Multi Player* przenosi nas do kolejnego menu z wyborem(Rysunek 4): *Same Computer*, w którym zmierzymy się z żywym przeciwnikiem grając na jednej klawiaturze.

Gracz na lewo steruje klawiszami A i Z a gracz na prawo K i M. Opcja LAN przenosi nas do widoku w którym nawiązujemy połączenie z serwerem(Rysunek 5).

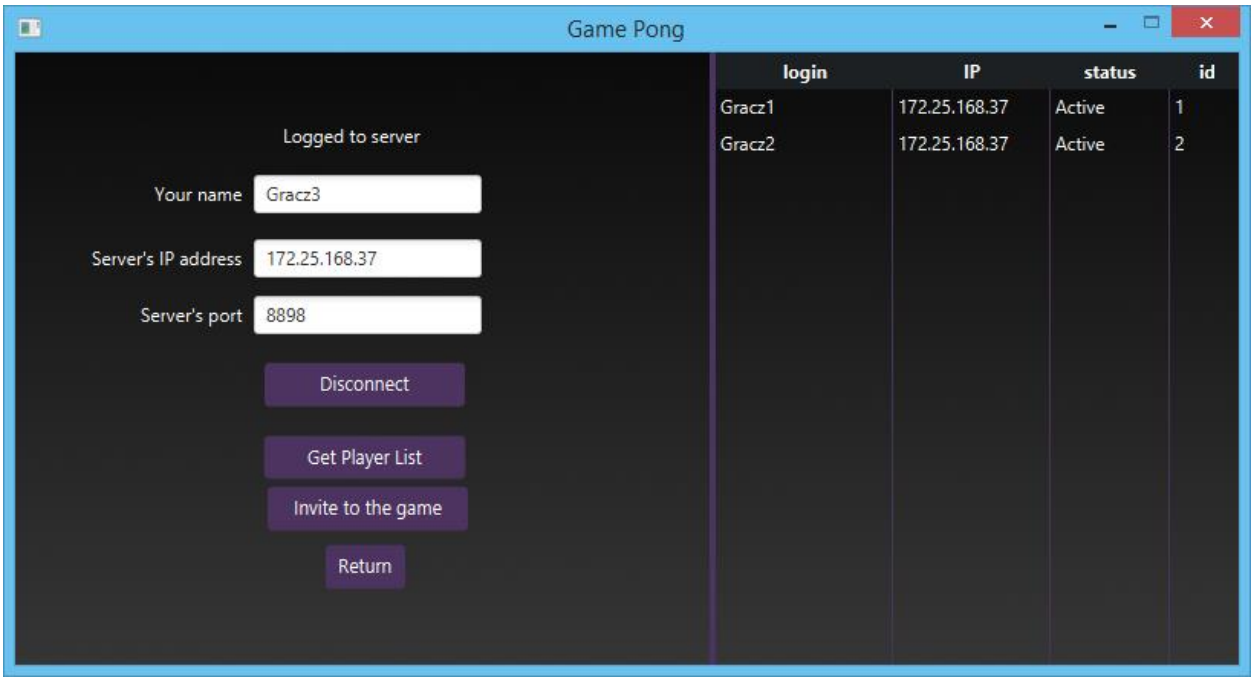


Rysunek 3 - Menu główne gry



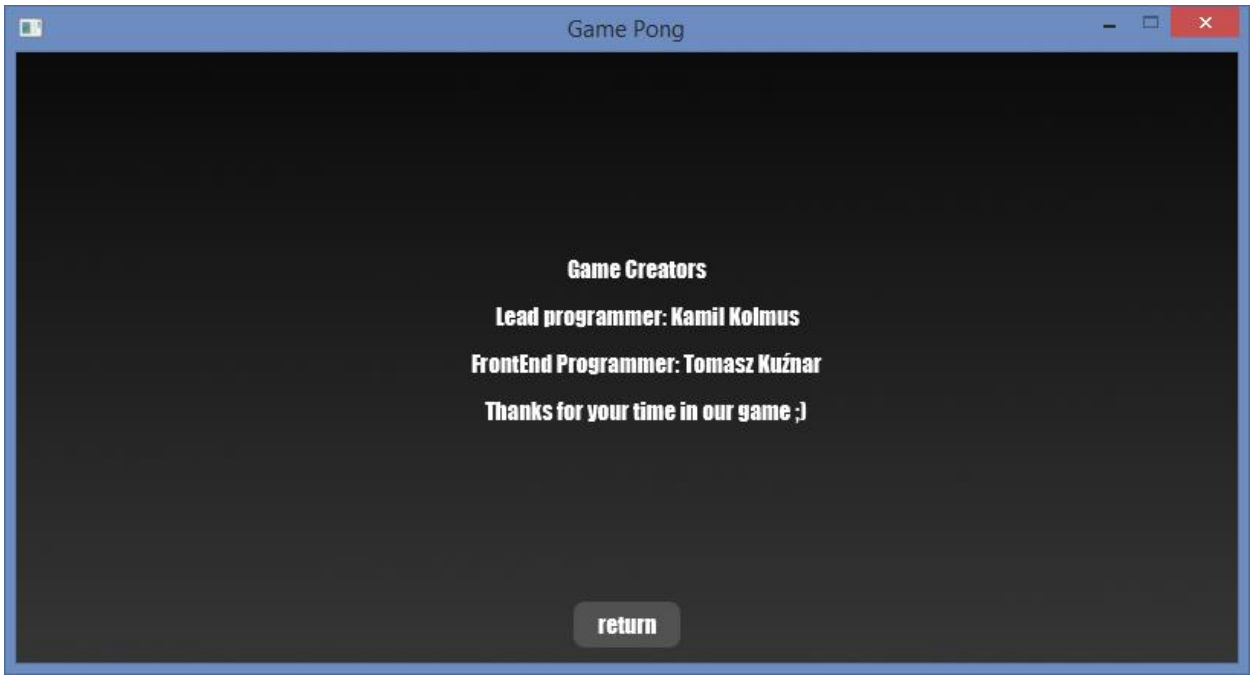
Rysunek 4 - Menu trybu dla wielu graczy

Aby połączyć się z serwerem musimy wpisać swoją unikalną nazwę! Drugie i trzecie pole to odpowiednio adres i numer portu serwera. W przypadku gdy to nasz komputer pełni rolę serwera możemy połączyć się wpisując adres swojego interfejsu sieciowego. Po prawej stronie widzimy tabelkę z zalogowanymi graczami. W przypadku jak na rysunku 5. nie ma żadnych zalogowanych graczy. Na rysunku 5. widzimy sytuację, w której do serwera jest połączonych trzech graczy. W takiej sytuacji możemy wybrać odpowiedniego gracza i użyć przycisku *Invite to the game*. Gdy drugi gracz zaakceptuje twoje zaproszenie oboje przejdziecie do widoku gry.

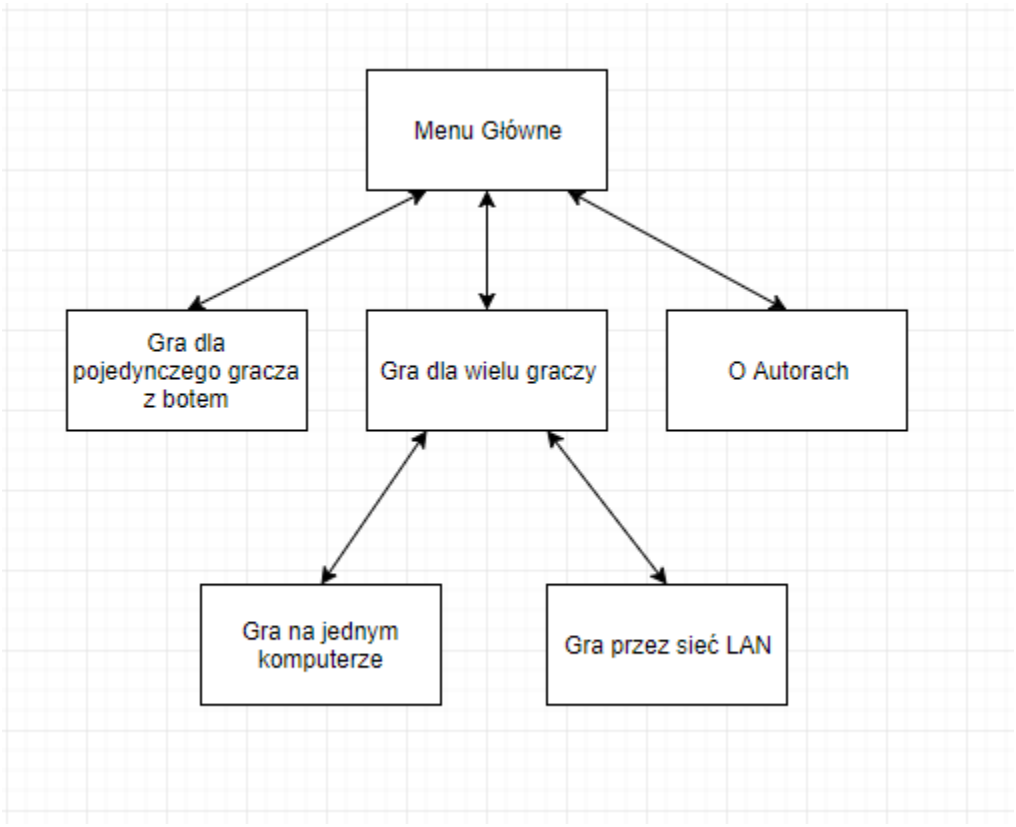


Rysunek 5 - Sytuacja gdy do serwera jest zalogowanych jest trzech graczy

Ostatnim punktem w menu głównym jest przycisk O Autorach tzw. *Credits*. Strukturę przeskakiwania między scenami przedstawia Rysunek 7.



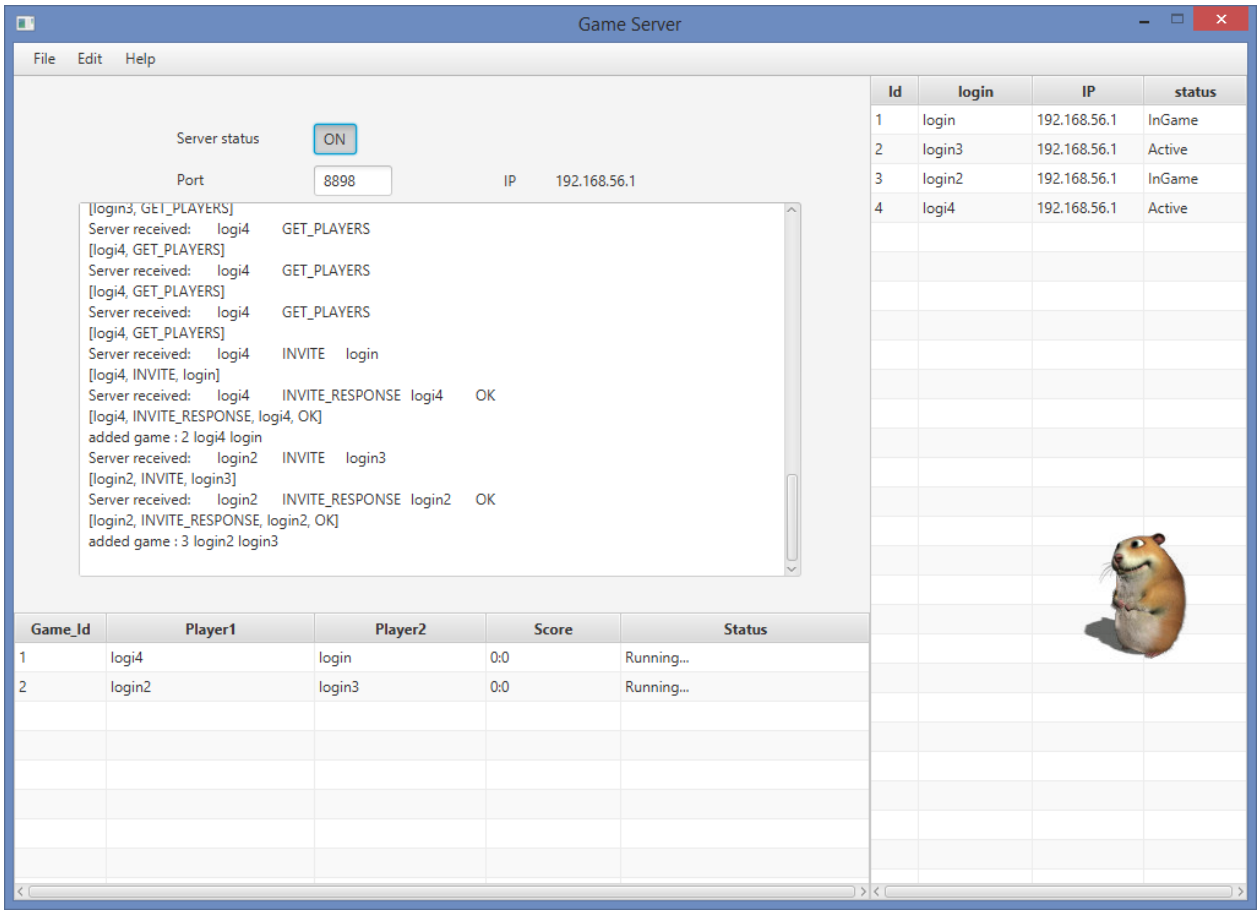
Rysunek 6 Widok Credits



Rysunek 7 - Struktura aplikacji gry

2.1 Aplikacja Serwera

Aplikacja Serwera jest jedno widokową aplikacją (Rysunek 8) służącą do zestawiania połączenia miedzy graczami którzy chcieli by rozegrać grę online. Gracze mogą zalogować się z aplikacji klienta podając unikalną nazwę użytkownika, IP serwera oraz port na których serwer obsługuje połączenia. Aplikacja ta umożliwia zestawienie wielu instancji gry miedzy zalogowanymi graczami. Lista graczy znajduje się w tabeli z prawej strony okna, natomiast lista zestawionych gier w tabeli na dole okna. W polu *Text Area* możliwe jest podejrzenie ruchu sieciowego miedzy Serwerem a Klientami.

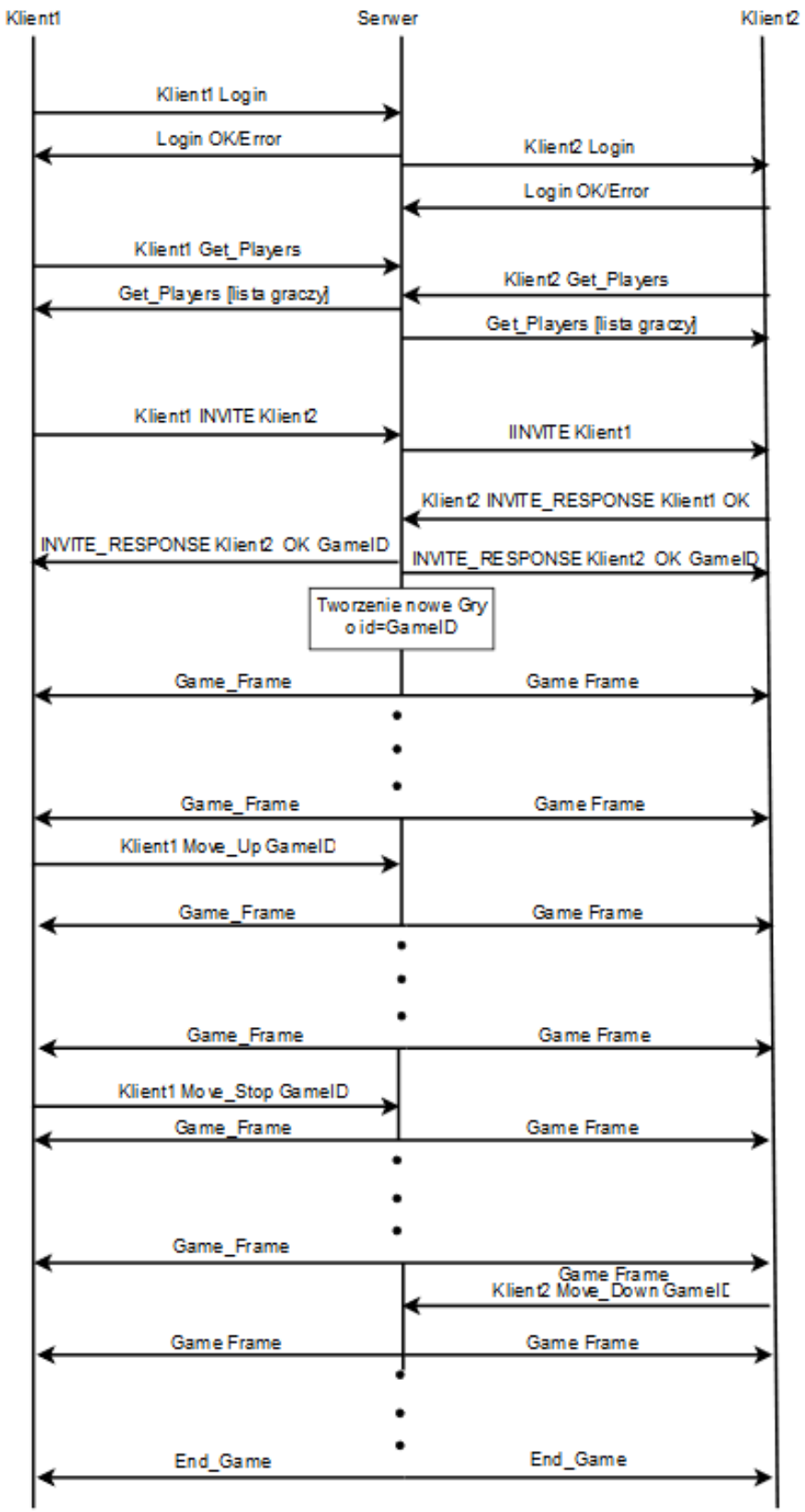


Rysunek 8 Widok aplikacji serwera

2.2 Netty.io – implementacja gry sieciowej

Zastosowanie biblioteki Netty pozwoliło na implementację sieciowej rozgrywki w naszej aplikacji. Framework Netty został zaprojektowany dla developerów Javy w celu ułatwienia i przyśpieszenia prac nad aplikacjami klient-serwer. W naszym przypadku serwer jest sercem komunikacji w strukturze wielu klientów – serwer.

Rysunek 9 przedstawia sytuację w której w której zestawiana jest gra między dwoma graczami. W pierwszej kolejności gracze logują się na serwerze swoją unikalną nazwą użytkownika. Następnie pobierają z serwera listę dostępnych graczy (Najnowsza aktualizacja rozsyła broadcastowo listę graczy w przypadku: zalogowani nowego gracza, wylogowania gracza oraz zmiany statusu gracza. Status informuje o tym czy dany gracz jest aktualnie aktywny "Active" czy też w trakcie rozgrywki "In Game"). W kolejnym kroku Klient1 wysyła zapytanie o grę do Klienta2 . Klient2 może zaakceptować lub odrzucić prośbę zestawie gry. W tym przypadku Klient2 akceptuje połączenie i tworzona jest nowa gra o unikalnym ID. Następnie gracze otrzymują cykliczną ramkę nazwaną *Game_Frame* (wysyłaną z częstotliwością 60 Hz) która zawiera pozycje piłki i graczy oraz aktualny wynik gry. Gracze mogą aktualizować swoje pozycje wysyłając Ramki nazwane odpowiednio Move_Up, Move_Down oraz Move_Stop. Gra kończy się ramką *End_Game* po której gracze wracają do widoku *Gry w sieci LAN* zgodnie ze z rysunkiem 7.



Rysunek 9 - Graf przepływu komunikatów

2.3 Silnik gry

Pod pojęciem "*Silnik Gry*" kryje się fragment kodu odpowiedzialny za generowanie liczb które opowiadają pozycji graczy, pozycji piłki oraz aktualnego wyniku gry. Silnik gry udostępnia dwie funkcje *movePlayer_1* i *movePlayer_2* które odpowiedzialne są za ruch graczy oraz interfejs składający sie z czterech funkcji które zwracają pozycje piłki, pozycje graczy oraz wynik rozgrywki. Przykład implementacji interfejsu po stronie Aplikacji Klienta oraz funkcje odpowiedzialne za ruch graczy przedstawione są na rysunku10.

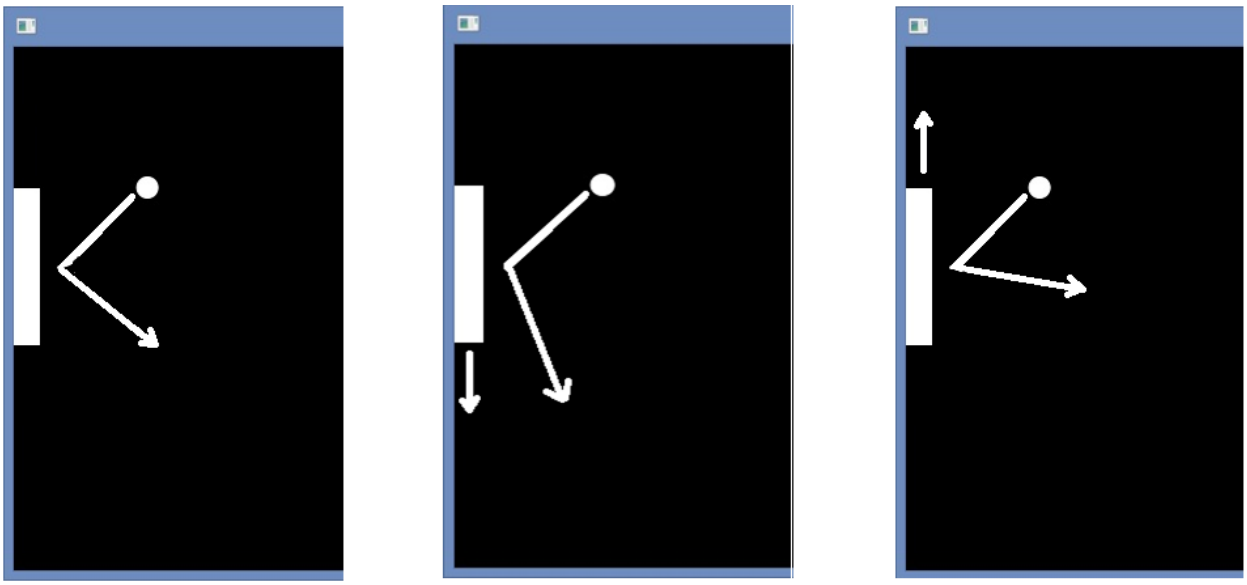
```
@Override
public void ballPosition(int x, int y) {
    ball.setLayoutX(x);
    ball.setLayoutY(y);
}

@Override
public void positionPlayer1(int y) { player_1.setLayoutY(y); }

@Override
public void positionPlayer2(int y) { player_2.setLayoutY(y); }

@Override
public void gameScore(int player1Score, int player2Score) {
    labelPlayerScore.setText(""+player1Score);
    labelBotScore.setText(""+player2Score);

}
```

```
void movePlayer_1(PLAYER_MOVE move) {

    switch (move) {
        case MOVE_UP:

            player1_isMoving=PLAYER_MOVE.MOVE_UP;

            break;
        case MOVE_DOWN:
            player1_isMoving=PLAYER_MOVE.MOVE_DOWN;

            break;
        case MOVE_STOP:
            player1_isMoving=PLAYER_MOVE.MOVE_STOP;
        default:
    }
}


void movePlayer_2(PLAYER_MOVE move) {

    if (botControl == false) {
        switch (move) {
            case MOVE_UP:
                player2_isMoving=PLAYER_MOVE.MOVE_UP;

                break;
            case MOVE_DOWN:
                player2_isMoving=PLAYER_MOVE.MOVE_DOWN;
```

Rysunek 10 Implementacji interfejsu oraz funkcje o sterowania graczami Silnika Gry

Gracze mogą wpływać na pozycje swojej *platformy* jak oraz na ruch piłki zgodnie z rysunkiem 11. Gracz może nadać dodatkową prędkości piłce w kierunku y jeśli w momencie odbicia porusza się porusza się w tą samą stronę co piłka .W przeciwnym wypadku gdy porusza się w przeciwnym kierunku może odebrać jej tą prędkość lub nawet zmienić jej kierunek na przeciwny. Jednak nie wpływa to na wypadkową prędkość piłki która zwiększa się o pewną

wartość po każdym odbiciu lub pozostaje stała po osiągnięciu prędkości maksymalnej.



Rysunek 11 Zmiana kierunku ruchu piłki

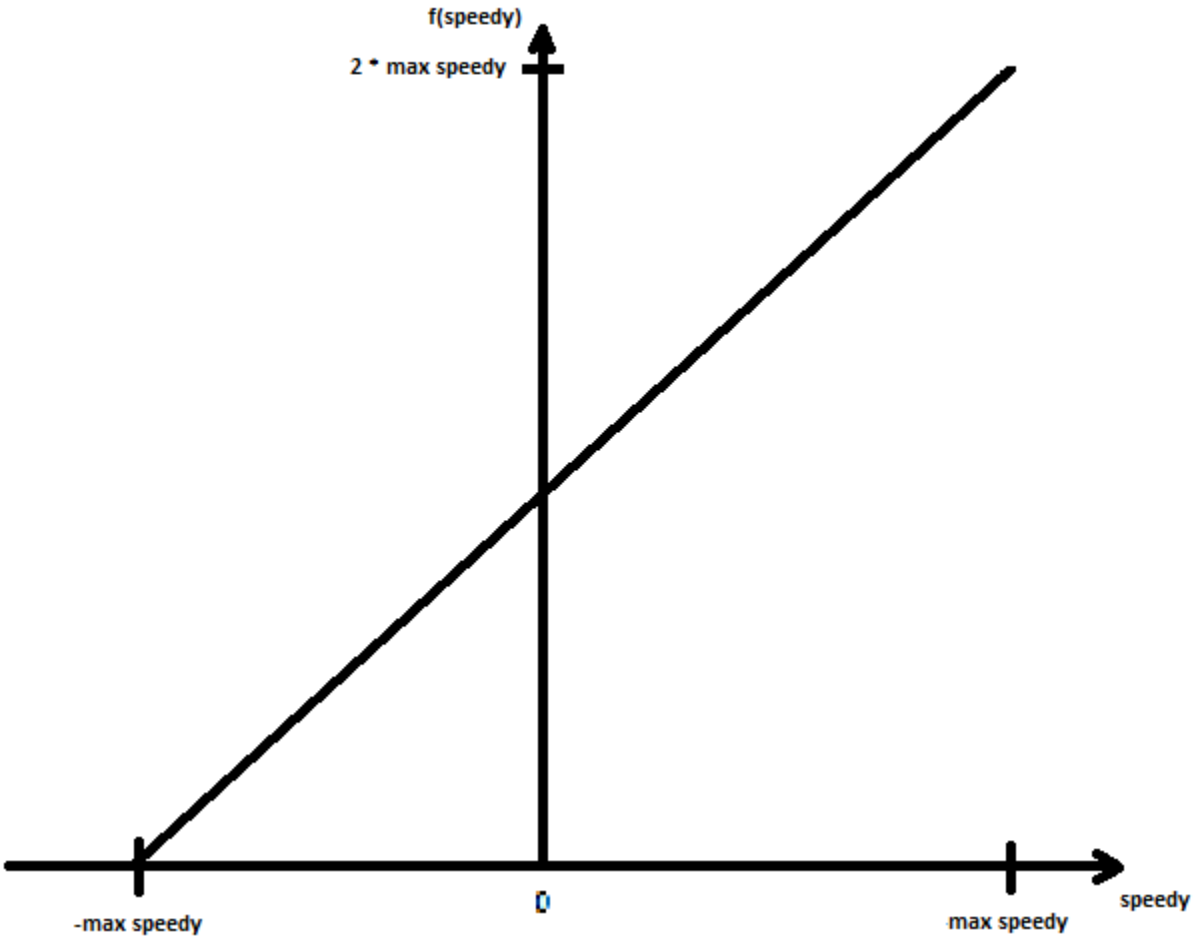
Realizacja obliczenia nowych składowych prędkości piłki została zamieszczona na listingu 1.

```
if(player1_move== PlayerMove.MOVE_UP){

        speedy -= ((refl_engle_norm )/(speedy_max_norm))*speedy+(refl_engle_norm *speed);
        speedx = Math.sqrt(speed*speed- speedy * speedy);


}else if(player1_move== PlayerMove.MOVE_DOWN){

        speedy += -((refl_engle_norm )/( speedy_max_norm))*speedy+(refl_engle_norm *speed);
        speedx = Math.sqrt(speed*speed- speedy * speedy);


}
```

List 1 Obliczanie nowych składowych prędkości piłki

Do wartości prędkości piłki w kierunku y (pionowo) dodatkowa lub odejmowana jest składowa która obliczana jest na podstawie funkcji (rysunek 12) opisanej wzorem (1) gdzie *maxSpeedy* to aktualna wypadkowa prędkość piłki a *speedy* to aktualna prędkość piłki w kierunku *y*. Dodatkowo wprowadzone zostały 2 modyfikatory które umożliwiają ograniczenie maksymalnej prędkości piłki w kierunku *y* i zmniejszenie lub zwiększenie kąta odbicia piłki.

$$f(speedy) = \frac{\text{reflAngleNorm}}{\text{maxSpeedyNorm}} * speedy + \text{reflAngleNorm} * \text{speed} \qquad (1)$$



Rysunek 12 Funkcja do obliczania nowej wartości stałej modyfikującej prędkość piłki w kierunku y

3 Funkcjonalność

Jak już wcześniej zostało wspomniane gra umożliwia rozgrywkę onlinową jak i offlinową . Gra onlinows testowa była na dwóch komputerach (na jednym został utworzony wirtualny router Wi-Fi oraz uruchomiony serwer gry). Gra na daną chwile umożliwia uruchomienie kilku klientów na jednym komputerze i zalogowanie się pod różnymi nazwami użytkownika, dzięki czemu mogliśmy zestawić kilka gier miedzy dwoma komputerami. Przy zestawieniu trzech gier nie było widać żadnych problemów jeśli chodzi o płynność gry na oby komputerach. Przy

zestawianiu czwartego połączenia pojawiły się drobne zacięcia gry co było spowodowane tym że nie wszystkie ramki gry nie docierały do klienta.

4 Niezrealizowane zadania

Zakładana początkowa funkcjonalność gry została całkowicie zrealizowana.

5 Zakres Odpowiedzialności

Za wygląd aplikacji klienta(*front-end*) oraz zawarte animacje odpowiedzialny był inż.Tomasz Kuźniar, Za Aplikacje Serwera, aplikację klienta (*back-end*), implementacje protokołu oraz silnik gry odpowiedzialny był inż.Kamil Kolmus.