

Blastback
Raport z przebiegu projektu

16 stycznia 2019

Aspekt sieciowy

- 1.1 Przyjęte założenia
- 1.2 Symulacja po stronie klienta
- 1.3 Rodzaje wiadomości
- 1.4 Serializacja

Grafika i dźwięk

Interfejs użytkownika

3.1 NiftyGUI

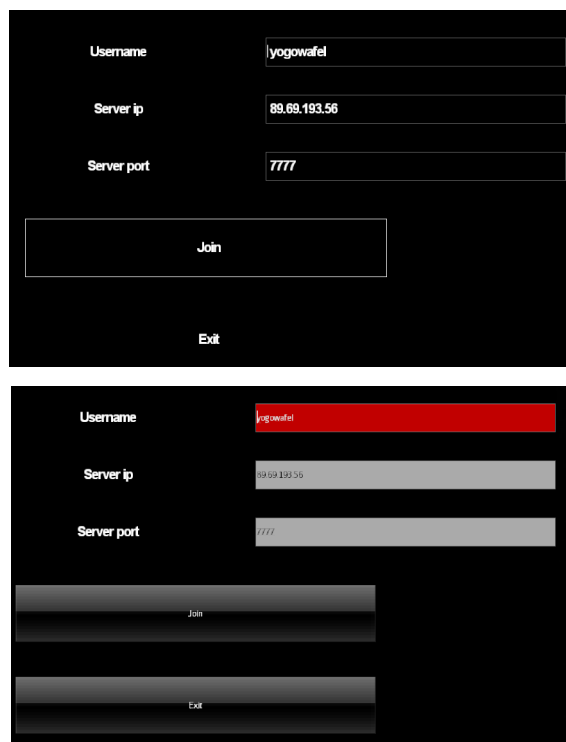
Interfejs graficzny w naszej aplikacji jest renderowany przy użyciu **NiftyGUI**. Jest to oprogramowanie typu open source, dostępne pod linkiem <https://github.com/nifty-gui/nifty-gui>. Wykorzystuje bibliotekę graficzną **OpenGL**. Rozwijając aplikację w **jMonkeyEngine** mieliśmy do wyboru jeszcze 3 inne rozwiązania: **Swing**, **Lemur**, **tonegodGUI** oraz powszechnie stosowany. Niestety dobrze nam znane pierwsze rozwiązanie nie jest dobrze zintegrowane z silnikiem, a drugie okazało się być nierozwijane już od 3 lat: <https://github.com/meltzow/tonegodgui>. Wybraliśmy **NiftyGUI** zamiast **Lemur**, ponieważ możliwość definiowania układu kontrolki oraz ich styli w xml wydawała nam się najlepszym rozwiązaniem.

3.2 Układ elementów na ekranie

NiftyGUI jest oprogramowaniem, w którym do projektowania układu elementów można wykorzystać język znaczników xml. Alternatywnie można robić to też przy użyciu kodu jawnego. Uznaliśmy jednak, że xml będzie bardziej przejrzystym rozwiązaniem. Definicje układów wszystkich ekranów wyświetlanych użytkownikowi znajdują się w `assets/Interface/Screens/screens.xml`.

3.3 Stylowanie kontrolki

Aby odpowiednio wystylować wygląd kontrolki pod nasze potrzeby, należało przeciągać ich domyślne style. Nasze definicje styli zostały również zapisane w pliku xml i znajdują się w `assets/Interface/Styles/styles.xml`. Powyżej - wygląd głównego menu gry z przeciążonymi definicjami stylów, poniżej - ich domyślne odpowiedniki.



Podsumowanie

4.1 Propozycje rozwoju

4.2 Wnioski po pracy z silnikiem