**BlockBall**

Game that will allow you to play in a versus mode, as well as against an opponent which will be a computer with different difficulty levels. BlockBall was created in C++ with Allegro 5 library.

**Game Rules**

Each hit block captured by a player's ball gives him 100 points to his score and increases the ball's speed multiplier, however, if the opponent hits the wrong ball in his square, then 150 points are taken away. Each life is worth 1000 points, if the player loses one life then 1000 points go to the opponent's account, but if the player manages to keep them then 1000 points go to his account during the game summary.

**Control Keys**

1. Menu:
   * Up Arrow – go up
   * Down Arrow - go down
   * Enter - Accept
2. First Player:
   * Up Arrow – start ball
   * Left Arrow – go left
   * Right Arrow - go right
3. Second Player:
   * S – start ball
   * A – go left
   * D – go right
4. Pause:
   * Escape – enter pause
   * Space – continue game
   * Enter – quit game
5. Game:
   * O – load game
   * P – save game

**Documentation**

The documentation was generated using Doxygen version 1.9.1. It has been written in html format, a separate folder has been separated in the project for the content of the documentation and a separate folder for the source files of the Doxygen program.To run the documentation, you can install the doxygen program and upload the project to it. However, the simplest way is to go to html folder and open index.html like this: **Blockball > html > index.html**

**Installation**

The simplest way to run game is to open *BlockBall.exe* file. You can install that game on your computer. Installer is in installer folder **intaller > Setup Files > BlockBall.exe**

**Tech Stack**

* C++
* Allegro 5+

**Authors**

* @kondi171
* @kwypych4

**Panzers1916**

A game based on Tank 1990 game released on pegasus. This project is a little bit different, because there is on co-operation, instead players fight each other.

**How to run?**

First of all install a Java Runtime Environment (JRE) or Java Development Kit (JDK) in version above or equal 12! After that run Panzers 1916®.jar file.

Download:

[Java Runtime Environment](https://www.java.com/pl/download/manual.jsp)

[Java Development Kit](https://www.oracle.com/pl/java/technologies/downloads/)

**ATTENTION!** Full screen does not work on some screens! Epilepsy warning!

**Game Rules**

The board has been divided into 26 x 13 tiles, excluding black border. Players and their bullets can only move around fixed grid. For each player was reserved part of screen where is stats are displayed screen where his stats and player name are displayed.

There are 4 types of blocks on the Board:

* Brick - A static destructible unit that can be obtained for 100 points for each hit and destruction,
* Stone - Static and indestructible unit,
* Water - A unit that only bullets can pass through,
* Empty - A block that players can move freely on.

Due to the Grid template, the player can only move in 4 directions, no indirect route possible. After firing the maximum number of bullets allowed in a burst (set to two), begins a shot restriction - on the board can be a maximum of two shoots per player, when player reach limit, the next time he can fire only when there is a bullet collision and the bullets will disappear from the board. Each player has 3 lives, each life taken from the opponent awards 1000 points to the player who managed to hit the opponent, and takes the life of the unit hit. For every hit destructible bricks, the player gets 100 points, which diversify the game and decide which player was better in a given game. Every tank can fire two bullets in one burst. When the life of one of the players reaches zero - the game is over.

**Control Keys**

1. Menu:
   * Up Arrow – go up
   * Down Arrow - go down
   * Enter - Accept
2. First Player:
   * W – go up
   * S – go down
   * A – go left
   * D – go right
   * Space - shoot
3. Second Player:
   * Up Arrow – go up
   * Down Arrow - go down
   * Left Arrow – go left
   * Right Arrow - go right
   * K – shoot

**Documentation**

Whole documentation is in **documentation** folder after running **index.html** file.

**Tech Stack**

* Java
* Swing
* AWT

**Authors**

* @kondi171
* @kwypych4

**MultiLines**

Single Page Application implemented with React.js. Ready to use in local game. Multiplayer is not implemented yet.

**How to run?**

Visit deployment: <https://multilines.netlify.app/>

**Game Rules**

Game is played in 5 rounds. Survive as long as you can, the longer you are on the board, the more points you get. Dodge walls, own trails and other players. Smash your enemies, blocking their way. The player with the most points wins. Winner take everything!

**Control Keys**

1. Red player:

* W – go up
* S – go down
* A – go left
* D – go right

1. Green player:

* Up Arrow – go up
* Down Arrow – go down
* Left Arrow – go left
* Right Arrow – go right

1. Blue player:

* I – go up
* K – go down
* J – go left
* L – go right

1. Yellow player:

* Num8 – go up
* Num5 – go down
* Num4 – go left
* Num6 – go right

**Tech Stack**

* HTML
* SCSS
* JavaScript
* React.js

**Authors**

* @kondi171

**CzasNaMasaz**

Website made commisioned by consumer for advertising his services in the field of massages. Website was implemented in two versions. One of them is with own mailing system implemented in PHP and the second one is with external mailing system – **formspree.io** which integrate mail with website form.

**How to run?**

To run website with **formspree.io** mailing system - Visit deployment: <https://czas-na-masaz.netlify.app/>

To run website with own mailing system:

* Download localhost server hosting like **XAMPP** and run **Apache server** and **MySQL**
* Create in **phpMyAdmin** database *czasnamasaz*, next import to this database czasnamasaz.sql file which is in **db** folder. File contains structure of tables
* Go to *\_phpVersion* folder and paste content to main folder replacing *index.html* to *index.php*
* All project paste to **htdocs** folder in **xampp** folder
* Db is empty if you want to login type **localhost/php/adminlog.php**, before it create login and password in *phpMyAdmin* in *admin* table

**IMPORTANT!** Website is no more used by consumer, all website is under development.

**Tech Stack**

* HTML
* CSS
* JavaScript
* PHP
* MySQL

**Authors**

* @kondi171

**LazyTaste**

An engineering application as an intermediary system between restaurant and customer. System enabling ordering meals online. Customer system is supported by Artificial Intelligence, which will recommend gastronomic meals based on collected data. System was implemented on JavaScript engine supported by React.js framework for visual layer and business logic. Storage and segregation are supported by no-relational database – MongoDB. Implementation of backend is based on Node.js framework which support Javascript syntax.

**How to run?**

Visit deployment: <https://lazytaste-frontend.onrender.com>

API:

Opinions database: <https://lazytaste-backend.onrender.com/API/opinions>

Customers database: <https://lazytaste-backend.onrender.com/API/customers>

Restaurants database: <https://lazytaste-backend.onrender.com/API/restaurants>

Lazy Assistant database: <https://lazytaste-backend.onrender.com/API/lazy-assistant>

**IMPORTANT!** Due to the free hosting package, the speed of data exchange between frontend and backend is poor, patience is recommended :)

**Artificial Neural Network**

The client system was supported by the implementation of the Artificial Neural Network - Long Short Term Memory (LSTM),. It is a recursive type of neural network, which, unlike standard neural networks, has the possibility of feedback. The main advantage of this type of network is the ability to process both single data and sequences, including character strings, which are transferred to the training data, on the basis of which the result is predicted.

**Documentation**

Documentation is an integral part of the engineering work protected by copyright. Documentation will be available only for individual request.

**Tech Stack**

**Client:**

* HTML
* SCSS
* JavaScript
* React.js

**Server:**

* Node.js
* Express.js
* Mongoose
* MongoDB
* Brain.js

**Authors**

* @kondi171

**BlockBall**

Gra umożliwiająca granie w trybie versus, a także przeciwko zaprogramowanemu komputerowi w różnych poziomach trudności. BlockBall został zaimplementowany w języku C++ z biblioteką Allegro 5.

**Zasady Gry**

Każde uderzenie bloku przez piłkę gracza poszerza sumę punktów o 100, a także zwiększa mnożnik prędkości piłki, jednakże jeżeli przeciwnik uderzy swoją piłką o paletkę gracza, to zabierane mu jest 150 punktów. Każde życie warte jest 1000 punktów, jeżeli gracz straci życie to taka suma jest zabierana z wyniku gracza, jednak jeżeli uda mu się zachować dodatkowe życia, to 1000 punktów jest mnożone przez ilość pozostałych żyć i doliczane do wyniku końcowego. Utrata wszystkich żyć przez jednego z graczy wiąże się z końcem gry.

**Sterowanie**

1. Menu:
   * Strzałka w górę – przesuń kursor w górę
   * Strzałka w dół – przesuń kursor w dół
   * Enter - Akceptuj
2. Gracz pierwszy:
   * Strzałka w górę – wypuść piłkę
   * Strzałka w lewo – przesuń paletkę w lewo
   * Strzałka w prawo – przesuń paletkę w prawo
3. Gracz drugi:
   * S – wypuść piłkę
   * A – przesuń paletkę w lewo
   * D – przesuń paletkę w prawo
4. Pauza:
   * Escape – włącz pauzę
   * Space – kontunuuj grę
   * Enter – wyjdź z gry
5. Gra:
   * O – załaduj poprzedni stan gry
   * P – zapisz stan gry

**Dokumentacja**

Dokumentacja została wygenerowana z pomocą programu Doxygen w wersji 1.9.1. Została zapisana w formacie html w oddzielnym folderze, w którym znajdują się niezbędne pliki potrzebne do uruchomienia dokumentacji. Aby uruchomić dokumentacje można zainstalować program Doxygen i wgrać do niego projekt, najprościej jednak jest otworzyć folder html i otworzenie pliku index.html: **Blockball > html > index.html**

**Instalacja**

Najłatwiejszym sposobem uruchomienia projektu jest otwarcie pliku BlockBall.exe. Grę można jednak zainstalować, w tym celu należy podążać tą ścieżką: **intaller > Setup Files > BlockBall.exe**

**Stos Technologiczny**

* C++
* Allegro 5+

**Autorzy**

* @kondi171
* @kwypych4

**Panzers1916**

Gra wzorowana na klasycznej grze wydanej na Pegasusa – Tank 1990. Projekt ten różni się od pierwowzoru ponieważ nie ma w niej kooperacji, zamiast tego gracze prowadzą ze sobą rozgrywkę w trybie versus.

**Jak uruchomić?**

Najpierw należy zainstalować maszynę wirtualną Javy (JRE) lub zestaw narzędzi programistycznych dla języka Java (JDK) w wersji powyżej 12! Następnie należy uruchomić plik Panzers 1916®.jar

Pobierz:

[Java Runtime Environment](https://www.java.com/pl/download/manual.jsp)

[Java Development Kit](https://www.oracle.com/pl/java/technologies/downloads/)

**Uwaga!** Pełny eran nie funkcjonuje poprawnie na niektórych ekranach! Ostrzeżenie o epilepsji!

**Zasady Gry**

Plansza została podzielona na kafelki 26 x 13, wykluczając czarne obramowanie. gracze oraz ich pociski, mogą poruszać się jedynie po ustalonej siatce. Dla każdego gracza zostało przydzielone miejsce na ekranie gdzie wyświetlają się jego statystyki oraz nazwa gracza.

Na Planszy znajdują się 4 rodzaje bloków:

* Cegła – Statyczna do zniszczenia jednostka za którą można dostać 100 punktów za każde jej trafienie i zniszczenie,
* Kamień – Statyczna i niezniszczalna jednostka,
* Woda – Jednostka przez którą może przelecieć pocisk, natomiast gracz już nie,
* Pusta – Blok po którym gracze, mogą poruszać się swobodnie.

Dzięki Siatkowemu szablonowi, gracz może poruszać się tylko w 4 kierunkach, brak możliwości drogi pośredniej. Po wystrzeleniu maksymalnej dozwolonej ilości pocisków w serii (ustawiona na 2), następuje ograniczenie strzału, na planszy mogą znajdować się maksymalnie 2 pociski jednego gracza, gdy osiągniemy limit, następny raz będziemy mogli wystrzeli dopiero gdy nastąpi kolizja pocisku i pocisk zniknie z planszy.

**Sterowanie**

1. Menu:
   * Strzałka w górę – poruszanie w górę
   * Strzałka w dół – poruszanie w dół
   * Enter - Akceptuj

2. Gracz pierwszy:

* + W – poruszanie w górę
  + S – poruszanie w dół
  + A – poruszanie w lewo
  + D – poruszanie w prawo
  + Spacja – Strzał

3. Gracz drugi:

* + Strzałka w górę – poruszanie w górę
  + Strzałka w dół – poruszanie w dół
  + Strzałka w lewo – poruszanie w lewo
  + Strzałka w prawo - poruszanie w prawo
  + K – strzał

**Dokumentacja**

Cała dokumentacja znajduje się w folderze **documentation** uruchamiając plik **index.html**

**Stos Technologiczny**

* Java
* Swing
* AWT

**Autorzy**

* @kondi171
* @kwypych4

**MultiLines**

Aplikacja typu Single Page Application zaimplementowana w technologiach webowych. Gotowa do uruchomienia w trybie lokalnym. Tryb online nie został jeszcze zaimplementowany.

**Jak uruchomić?**

Odwiedź witrynę: <https://multilines.netlify.app/>

**Zasady Gry**

Gra rozgrywa się przez 5 rund. Przetrwaj tak długo jak potrafisz, im dłużej znajdujesz się na planszy tym więcej punktów dostajesz. Unikaj ścian oraz szlaków, zarówno własnych jak i przeciwników. Niszcz wrogów blokując im drogę. Gracz z największą ilością punktów wygrywa. Wygrywający bierze wszystko!

**Sterowanie**

1. Red player:
   * W – poruszanie w górę
   * S – poruszanie w dół
   * A – poruszanie w lewo
   * D – poruszanie w prawo
2. Green player:
   * Strzałka w górę – poruszanie w górę
   * Strzałka w dół – poruszanie w dół
   * Strzałka w lewo – poruszanie w lewo
   * Strzałka w prawo - poruszanie w prawo
3. Blue player:

* I – poruszanie w górę
* K – poruszanie w dół
* J – poruszanie w lewo
* L – poruszanie w prawo

1. Yellow player:

* Num8 – poruszanie w górę
* Num5 – poruszanie w dół
* Num4 – poruszanie w lewo
* Num6 – poruszanie w prawo

**Stos Technologiczny**

* HTML
* SCSS
* JavaScript
* React.js

**Autorzy**

* @kondi171

**CzasNaMasaz**

Strona internetowa wykonana na zlecenie Marcina Kasińskiego do ogłaszania jego usług w zakresie wykonywania masaży. Witryna została zaimplementowana w dwóch wersjach. Pierwsza z nich ma zaimplementowany własny system mailingowy napisany w języku PHP, a drugi wariant ma zaimplementowany zewnętrzny sytem mailingowy – **formspree.io**, który integruje skrzynkę mailową z formularzem na stronie.

**Jak uruchomić?**

Aby uruchomić stronę internetową z systemem mailingowym **formspree.io** – odwiedź witrynę: <https://czas-na-masaz.netlify.app/>

Aby uruchomić witryne z własnym systemem mailingowym należy:

* Zainstalować hostujący serwer lokalny np. XAMPP i uruchomić w nim Apache server oraz MySQL
* Stworzyć w phpMyAdmin bazę danych o nazwie *czasnamasaz*, następnie zaimportować bazę danych z pliku *czasnamasaz.sql*, który znajduje się w folderze **db**. Plik ten zawiera struktury tabel
* Otwórz folder *\_phpVersion* skopiuj i wklej zawartość folderu do folderu głównego, zamieniając przy tym *index.html* na *index.php*
* Cały projekt wklej do folderu **htdocs** w folderze **xampp**
* Baza danych jest pusta, jeżeli chcesz się zalogować wpisz w URL: **localhost/php/adminlog.php**, przed tym jednak stwórz swój login i hasło w *phpMyAdmin* w tabeli *admin*

**WAŻNE!** Strona internetowa nie jest już używana przez klienta. Cała witryna jest na etapie developerskim.

**Stos Technologiczny**

* HTML
* CSS
* JavaScript
* PHP
* MySQL

**Autorzy**

* @kondi171

**LazyTaste**

Aplikacja inżynierska jako system pośredniczący pomiędzy restauracją a klientem, umożliwiający zamawianie posiłków online. System kliencki wspierany jest przez Sztuczną Inteligencję, która za zadanie ma rekomendowanie posiłków gastronomicznych na podstawie zebranych danych z zamówień klienta. System został zaimplementowany w języku JavaScript wraz z biblioteką React.js dla warstwy wizualnej oraz logiki biznesowej. Za przechowywanie oraz segregację danych odpowiada nierelacyjna baza danych – MongoDB. Do obsługi bazy danych wykorzystana została backendowa implementacja JavaScript’u – Node.js.

**Jak uruchomić?**

Odwiedź witrynę: <https://lazytaste-frontend.onrender.com>

API:

Baza opinii: <https://lazytaste-backend.onrender.com/API/opinions>

Baza klientów: <https://lazytaste-backend.onrender.com/API/customers>

Baza restauracji: <https://lazytaste-backend.onrender.com/API/restaurants>

Baza leniwego asystenta: <https://lazytaste-backend.onrender.com/API/lazy-assistant>

**WAŻNE!** Z powodu darmowego pakietu hostującego, prędkość wymiany danych pomiędzy frontendem a backendem jest uboga, zalecana cierpliwość :)

**Sztuczna Sieć Neuronowa**

System kliencki został wsparty o implementacje Sztucznej Sieci Neuronowej – Long Short Term Memory (LSTM) czyli długoterminowa pamięć krótkoterminowa. Jest to rekurencyjny typ sieci neuronowej, który w odróżnieniu od standardowych sieci neronowych ma możliwość sprzężenia zwrotnego. Główną zaletą tego typu sieci jest możliwość przetwarzania zarówno pojedynczych danych jak i sekwencji w tym ciągów znaków, które są przekazywane do danych uczących, na podstawie których jest przewidywany wynik.

**Dokumentacja**

Dokumentacja stanowi nieodłączną część pracy inżynierskiej chroniącej przez prawa autorskie. Dokumentacja zostanie udostępniona na indywidualną prośbę.

**Stos Technologiczny**

**Client:**

* HTML
* SCSS
* JavaScript
* React.js

**Server:**

* Node.js
* Express.js
* Mongoose
* MongoDB
* Brain.js

**Autorzy**

* @kondi171