Instrukcja użytkownika - Pang

Wymagana wersja JDK:

-java 15.x.x

Cel gry:

Celem gry jest unikanie odbijających się piłek. Zadaniem gracza jest zniszczenie widocznych piłek.

Opis:

Gra składa się z poziomów różniących się ilością piłek. Gracz ma możliwość wyboru poziomu na jakim zagra.
Bohater może chodzić lewo-prawo, skakać i strzelać.
Bohater może wchodzić w interakcje z wrogami poprzez:

1. *zderzanie się z przeciwnikami (przy zderzeniu gracz traci punkty hp, piłka odbija się od gracza pod kątem 90 stopni)*
2. *zabieranie życia przeciwnikom poprzez strzelanie do nich*

Bohater ma ograniczoną liczbę żyć i amunicji.
Trafienie piłki w gracza powoduje utratę określonej ilości życia (zależne od poziomu i typu piłki)
Po skończonej rozgrywce program zapisuje wynik gracza do listy najlepszych wyników.
W trakcie gry można zmieniać rozmiar okna. Pole gry jest skalowane do aktualnego rozmiaru okna.
Gracz ma możliwość zapauzowania gry w dowolnym momencie.
Gra posiada możliwość łączenia się z serwerem i grania na mapach dostępnych online, a także pobierania plików konfiguracyjnych i list najlepszych wyników

Sterowanie:

w - skok
a - ruch w lewo
d - ruch w prawo
k - strzał
p - pauza

Funkcjonalność aplikacji:

- *menu startowe*
- *wybór planszy*
- *możliwość pauzy w trakcie rozgrywki*
- *zapisywanie wyników*
- *menu ustawień*
- *łączenie z serwerem i granie w trybie “online”*

Elementy dodatkowe:

- *bohater ma możliwość strzelania*
- *skok bohatera*
- *wybór dowolnego poziomu trudności*
- *listy najlepszych wyników dla każdej mapy*
- *możliwość odtwarzania muzyki*
- *różne rodzaje przeciwników*

Szczegółowe zasady gry:

- *Na planszy znajduje się bohater oraz piłki.*
- *Piłki poruszają się w pionie i poziomie z różnymi prędkościami, z zachowaniem zasad fizycznych, jednak bez uwzględniania strat energii.*
- *Piłki pojawiają się na mapie w losowych miejscach na górze mapy, w czasie określonym w pliku konfiguracyjnym.*
- *Gracz ma sterować bohaterem tak, aby nie trafiła go żadna piłka.*
- *Gracz może poruszać się w lewo, w prawo, oraz podskakiwać, a także strzelać do piłek.*
- *Piłki mają trzy rodzaje("mała", "duża", "mega")*
- *Aby rozbić małą piłkę gracz musi w nią trafić 1 raz.*
- *Aby rozbić dużą piłkę gracz musi w nią trafić 3 razy.*
- *Aby rozbić mega piłkę gracz musi w nią trafić 7 razy.*
- *Bohater:*
    - *na start 10 punktów życia*
    - *możliwość strzelania*
    - *możliwość poruszania się*

Zasady punktacji:

- *Za zniszczenie małej piłki gracz otrzymuje 1 pkt*
- *Za zniszczenie dużej piłki gracz otrzymuje 5 pkt*
- *Za zniszczenie mega piłki gracz otrzymuje 10 punktów*
- *Za przejście poziomu gracz otrzymuje dodatkową liczbę punktów, która wynika ze wzoru:*

W=ceil(10[(0.028k^6-0.4k^2+2.3k-2.55)+0.4(p-2.5)+0.8*(s-3)]+29)*

,gdzie: k - poziom trudności (łatwy -> k=1, trudny -> k=3) p - pozostałe hp (1-10) s - numer poziomu(1-3)

Rodzaje przeciwników:

Rodzaj	Mała piłka	Duża piłka	Mega piłka
życie	1	3	7
obrażenia	1	1.5	3

Kompilacja:

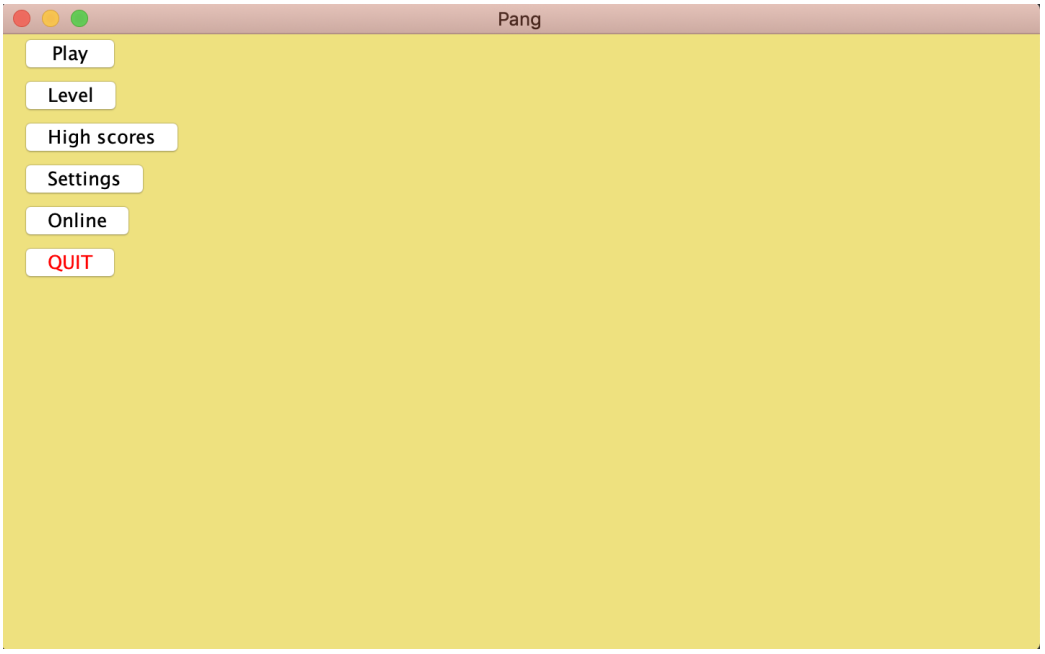
- *Mac/Linux:*
  - *komenda: compilePangLinuxMac.sh*
- *Windows:*
  - *komenda: compilePangWindows.sh*

Uruchamianie:

- *Mac/Linux/Windows:*
  - *komenda runPang.sh*

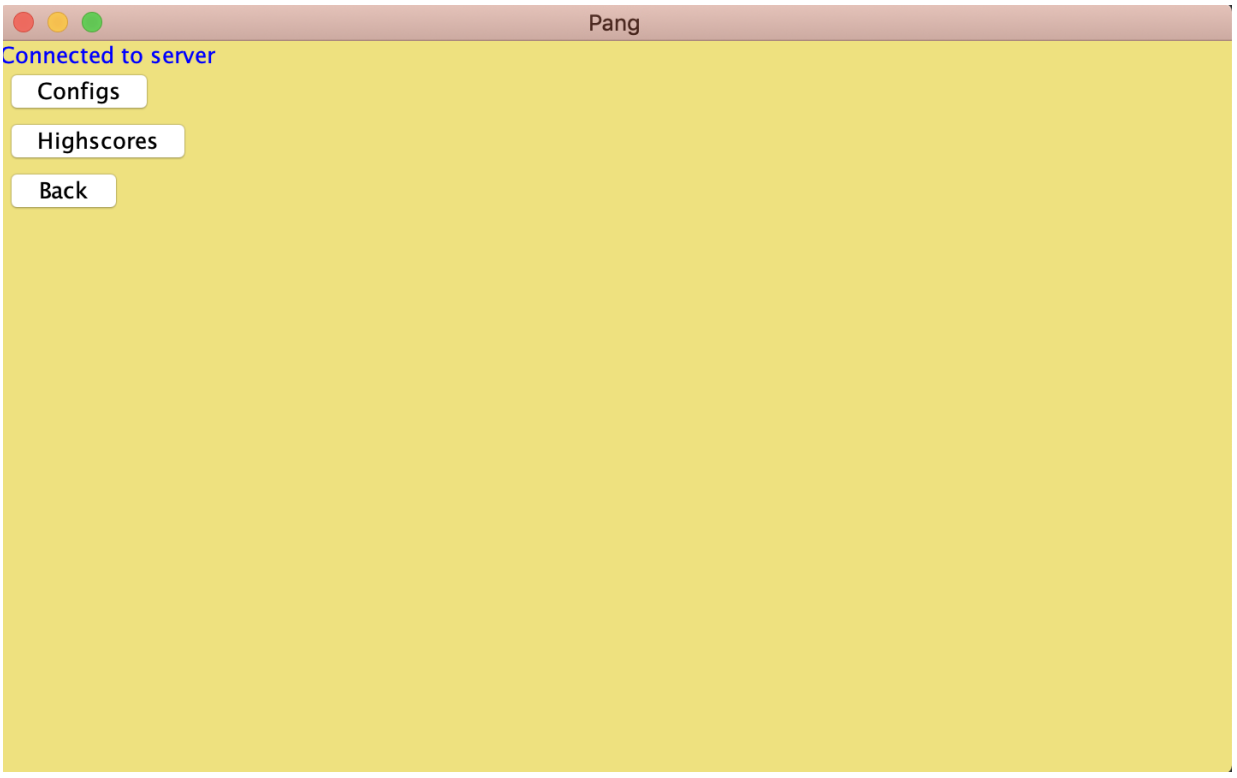
Zrzuty ekranu z aplikacji:

a) *ekran startowy aplikacji:*

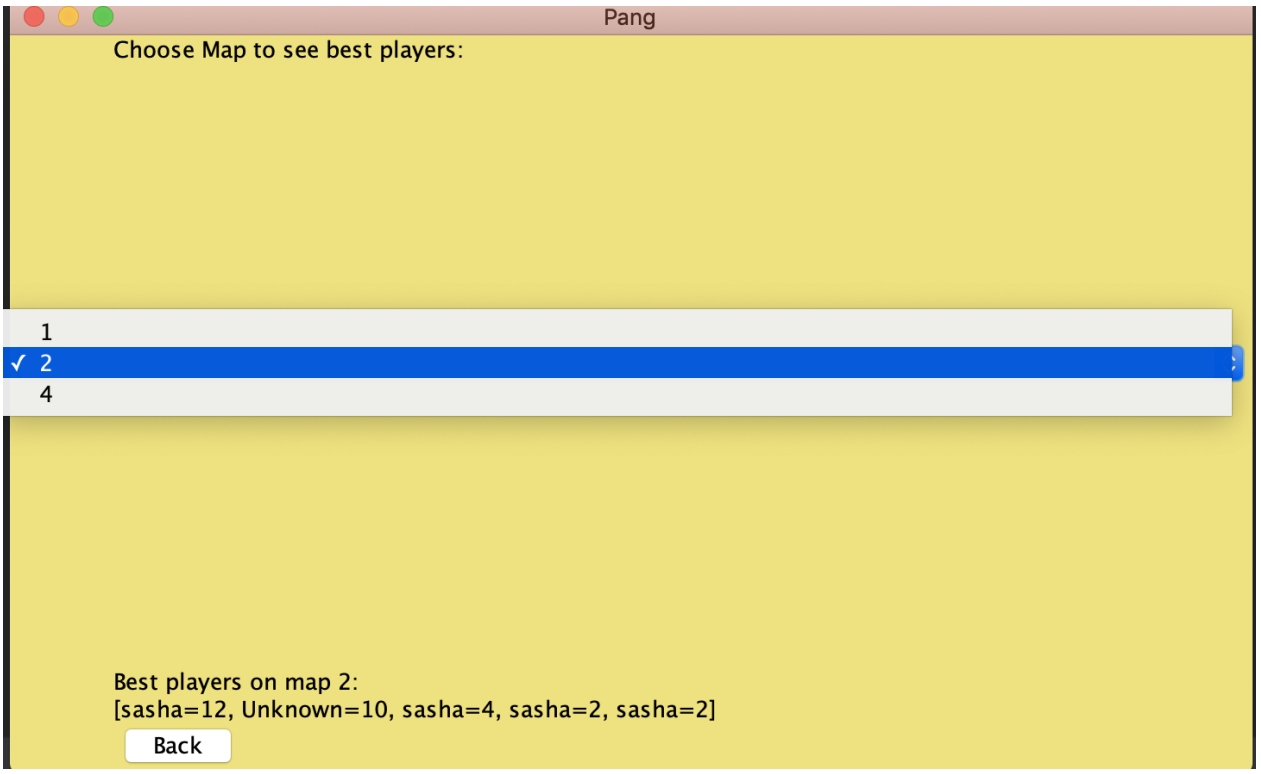


b) *ekran ustawień:*

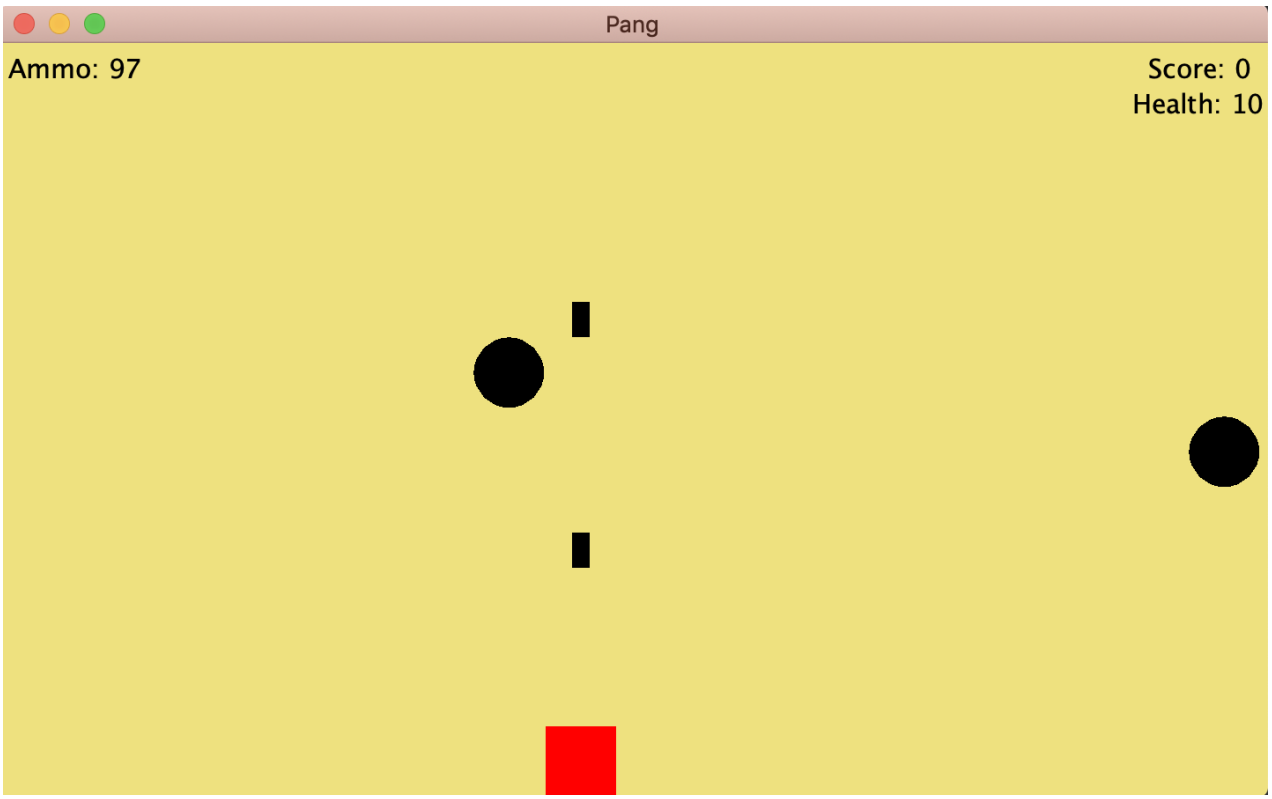
c) *ekran trybu sieciowego:*

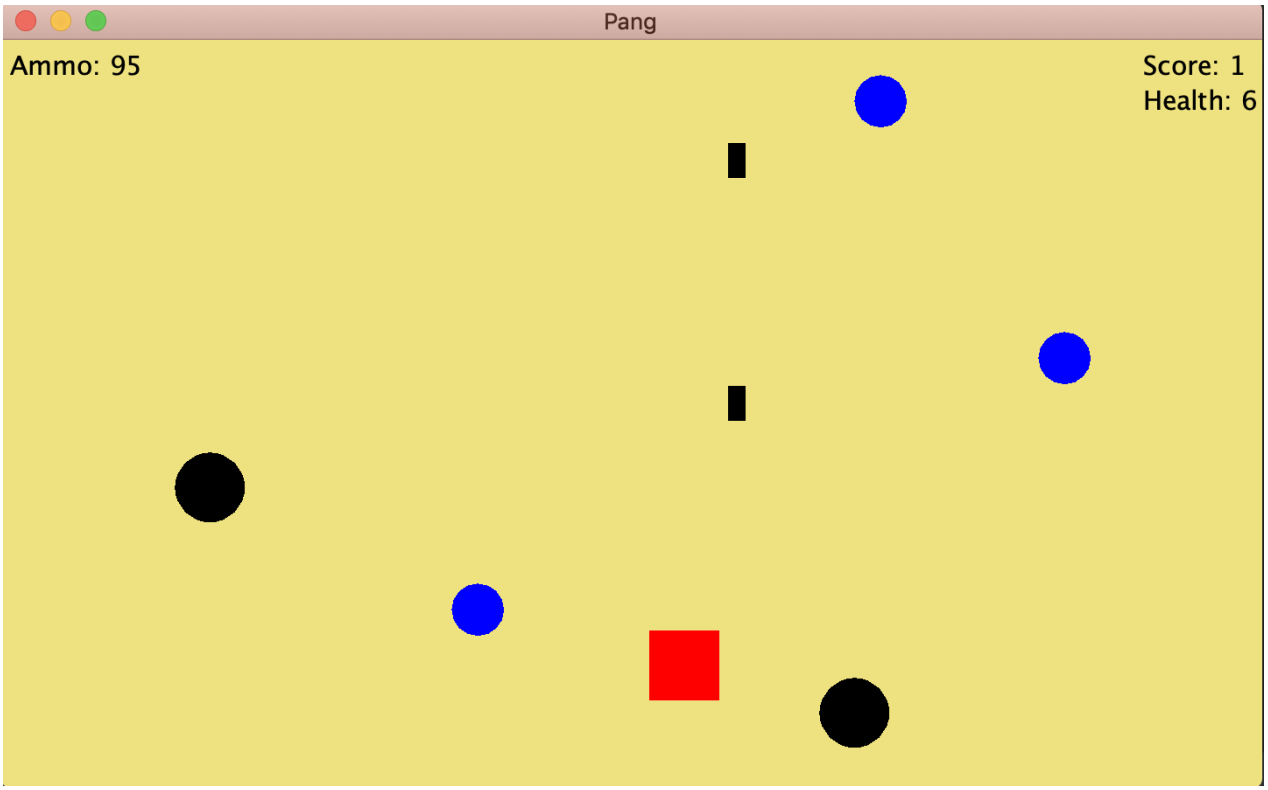


d) *ekran najlepszych wyników:*



e) *ekran rozgrywki:*

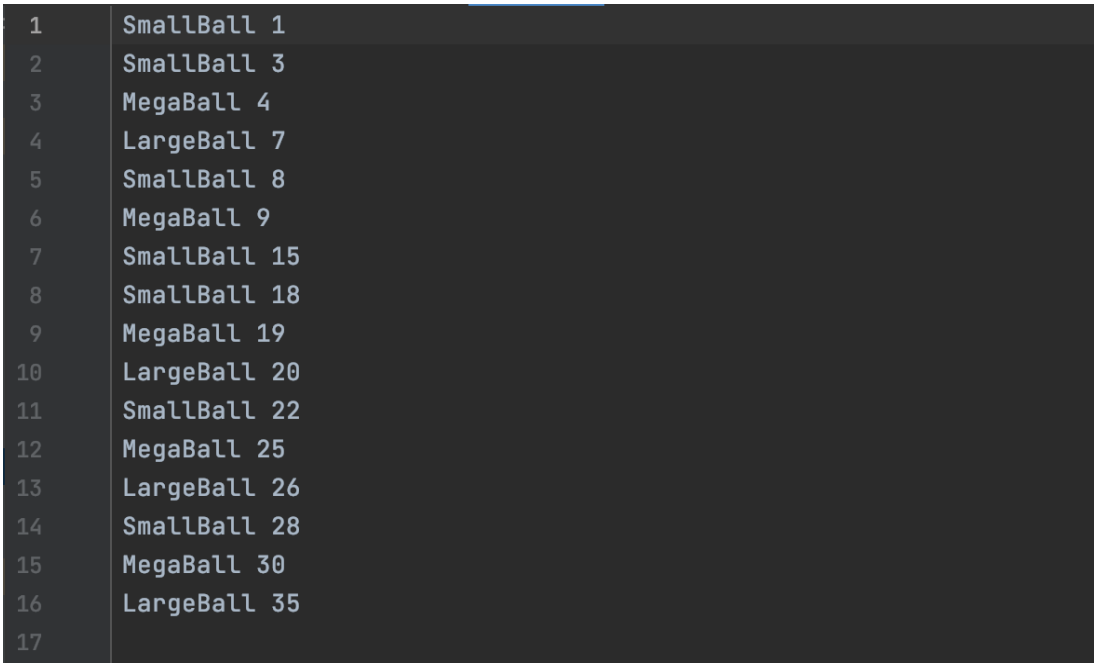

Pang
Ammo: 97
Score: 0
Health: 10


Pang
Ammo: 95
Score: 1
Health: 6

Pliki konfiguracyjne:

Definicje poziomów trudności / configi / mapy / listy najlepszych wyników są przechowywane w plikach konfiguracyjnych.

a) *Zrzut ekranu przykładowej mapy: (1.txt)*



```
SmallBall 1
SmallBall 3
MegaBall 4
LargeBall 7
SmallBall 8
MegaBall 9
SmallBall 15
SmallBall 18
MegaBall 19
LargeBall 20
SmallBall 22
MegaBall 25
LargeBall 26
SmallBall 28
MegaBall 30
LargeBall 35
```

SmallBall / MegaBall / LargeBall - rodzaje przeciwników
1 / 3 /4 /7 / 8 / 9 … - czas po jakim dany przeciwnik pojawi się na mapie (w ms)

b) *Zrzut ekranu pokazujący zapisane wyniki graczy na przykładowej mapie:*

```
sasha 2
sasha 2
sasha 2
sasha 4
sasha 12
Unknown 10
ja 2
ja 22
ja 33
ja 727227
```

sasha, sasha … - nicki użytkowników
2, 2 … - wynik danego użytkownika

c) *Zrzuty ekranu z pliku konfiguracyjnego gry:*

```
Pang
    difficulty=2
    defaultXFrameSize=960
    defaultYFrameSize=540
Player
    health=10
    damage=1.5
    speed=2
    height=40
    width=40
    ammunition=100
    gravityForce=2
    gravityLimit=0
    posX=0
    posY=0
    motionVectorX=0
    motionVectorY=0
    motionVectorBlanking=50
    score=0
PlayerCoolDown
    attack=1000
    move=5
    gravity=10
    movingVector=20
    jumping=2000
```

```
Bullet
    width=10
    height=20
World
    worldCapacity=100
    maxSpawnCount=50
SmallBall
    health=1
    damage=1
    speed=1.12
    height=30
    width=30
    posX=0
    posY=0
    score=1
SmallBallCoolDown
    attack=1000
    move=30
    bounceOff=600
LargeBall
    health=2
    damage=1.5
    speed=1.12
    height=40
    width=40
    posX=30
    posY=30
    score=3
```

```
LargeBallCoolDown
    attack=1000
    move=30
    bounceOff=600
MegaBall
    health=3
    damage=3
    speed=1.12
    height=50
    width=50
    posX=1
    posY=1
    score=5
MegaBallCoolDown
    attack=1000
    move=30
    bounceOff=600
Keyboard
    a=-10
    A=-10
    d=10
    D=10
    w=-10
    W=-10
    k=-1
    K=-1

```

Znaczenie poszczególnych pól:

1.Pang - config dla gry
2. difficulty=2 - poziom trudności
3. defaultXFrameSize=960 - domyślny rozmiar ramki w płaszczyźnie X
4. defaultYFrameSize=540 - domyślny rozmiar ramki w płaszczyźnie Y
5.Player - config dla gracza
6. health=10 - punkty *życia*
7. damage=1.5 - zadawane obrażenia
8. speed=2 -współczynnik szybkości poruszania się
9. height=40 - wysokość gracza
10. width=40 - szerokość gracza
11. ammunition=100 - ilość amunicji na start
12. gravityForce=2 - siła grawitacji
13. gravityLimit=0 - poziom, do którego działa grawitacja
14. posX=0 - pozycja w płaszczyźnie X
15. posY=0 - pozycja w płaszczyźnie Y
16. motionVectorX=0 - wektor ruchu w płaszczyźnie X
17. motionVectorY=0 - wektor ruchu w płaszczyźnie Y
18. motionVectorBlanking=50 - wygaszanie wektora ruchu
19. score=0 - startowy wynik gracza
20.PlayerCooldown - config dla cooldownu playera
21. attack=1000 - opóźnienie dla strzelania
22. move=5 - opóźnienie przy poruszaniu się
23. gravity=10 - cooldown grawitacji
24. movingVector=20 - cooldown na poruszanie się
25. jumping=2000 - cooldown na skok
26.Bullet - config dla naboju
27. width=10 - wysokość
28. height=20 - szerokość
29.World - config dla świata
30. worldCapacity=100 - maksymalna ilość przeciwników na mapie
31. maxSpawnCount=50 - maksymalna liczba przeciwników widocznych w jednym momencie na ekranie
32.SmallBall - config dla małej piłki
33. health=1 - punkty *życia*
34. damage=1 - zadawane obrażenia graczowi
35. speed=1.12 - szybkość przemieszczania się
36. height=30 - wysokość piłki
37. width=30 - szerokość piłki
38. posX=0 - startowa pozycja w płaszczyźnie X
39. posY=0 - startowa pozycja w płaszczyźnie Y
40. score=1 - punkty jakie otrzyma player po zniszczeniu piłki
41.SmallBallCoolDown - config dla cooldownu małej piłki
42. attack=1000 - czas po jakim piłka może ponownie zadać obrażenia(ms)
43. move=30 - czas w ms, po jakim piłka zmienia swoją pozycję
44. bounceOff=600 - opóźnienie możliwości odbijania się od ścian

45.LargeBall - *config dla dużej piłki*
46. health=2 - punkty *życia*
47. damage=1.5 - *zadawane obrażenia graczowi*
48. speed=1.12 - *szybkość przemieszczania się*
49. height=40 - *wysokość*
50. width=40 - *szerokość piłki*
51. posX=30 - *startowa pozycja w płaszczyźnie X*
52. posY=30 - *startowa pozycja w płaszczyźnie Y*
53. score=3 - *punkty jakie otrzyma player po zniszczeniu piłki*
54.LargeBallCoolDown - *config dla cooldownu dużej piłki*
55. attack=1000 - *czas po jakim piłka może ponownie zadać obrażenia(ms)*
56. move=30 - *czas w ms, po jakim piłka zmienia swoją pozycję*
57. bounceOff=600 - *opóźnienie możliwości odbijania się od ścian*
58.MegaBall - *config dla mega piłki*
59. health=3 - punkty *życia*
60. damage=3 - *zadawane obrażenia graczowi*
61. speed=1.12 - *szybkość przemieszczania się*
62. height=50 - *wysokość piłki*
63. width=50 - *szerokość piłki*
64. posX=1 - *startowa pozycja w płaszczyźnie X*
65. posY=1 - *startowa pozycja w płaszczyźnie Y*
66. score=5 - *punkty jakie otrzyma player po zniszczeniu piłki*
67.MegaBallCoolDown - config dla cooldownu mega piłki
68. attack=1000 - czas po jakim piłka może ponownie zadać obrażenia(ms)
69. move=30 - czas w ms, po jakim piłka zmienia swoją pozycję
70. bounceOff=600 - opóźnienie możliwości odbijania się od ścian
71.Keyboard - config dla klawiatury
72. a=-10 - szybkość poruszania się w lewo
73. A=-10 - szybkość poruszania się w lewo (CAPS LOCK ON)
74. d=10 - szybkość poruszania się w prawo
75. D=10 - szybkość poruszania się w prawo (CAPS LOCK ON)
76. w=-10 - wysokość skoku
77. W=-10 - wysokość skoku (CAPS LOCK ON)
78. k=-1 - ilość amunicji jaką traci gracz przy strzale
79. K=-1 - ilość amunicji jaką traci gracz przy strzale (CAPS LOCK ON)

Funkcjonalność sieciowa:

Projekt zawiera serwer sieciowy, który przechowuje i udostępnia klientowi informacje tj. definicje poziomów trudności, listy najlepszych wyników, pliki konfiguracyjne
Serwer jest w stanie obsłużyć dowolną liczbę klientów. Dla każdego klienta jest tworzony nowy wątek.
Serwer jest niezależną aplikacją posiadająca swoją bazę danych.

Kompilacja serwera:

- *Mac/Linux:*
    - *komenda compileServerLinuxMac.sh*

- *Windows:*
    - *komenda compileServerWindows.sh*

Uruchamianie serwera:

- *Mac/Linux/Windows:*
    - *komenda runServer.sh*

<u>Protokół sieciowy:</u>

Typ protokołu: Binarnie/Tekstowy

- Klient wysyła żądania w formie tekstowej
- Serwer odsyła odpowiedzi w formie binarnej

- *Klient wysyła żądanie o listy najlepszych wyników:*
  C: getHighScores\n => S

  Wiadomość jaką otrzyma serwer:
  a) *From client: getHighScores*

- *Serwer wysyła do klienta odpowiedź z listą najlepszych wyników*
  (int)(int)(byte[])(int)(byte[]) … (int)(byte[])
  S: replyForHighScores ***(int numberOfHighscores) (int fileNameByteLength1) (byte[] fileNameData1) (int fileBytelength1) (byte[] fileData1) …... (int fileNameByteLengthX) (byte[] fileNameDataX) (int fileBytelengthX) (byte[] fileDataX)****\n =>C*

  gdzie:
  int numberOfHighscores *- liczba dostępnych list z najlepszymi wynikami, do których przesłane zostaną dane. Dzięki temu klient wie ile razy ma odczytać dane wysłane z serwera*

  int fileNameByteLength *- informacja o rozmiarze nazwy mapy, dla której przesłane zostaną listy najlepszych wyników w bajtach*
  Dzięki temu klient wie, kiedy ma przestać odbierać dane.
  byte[] fileNameData *- nazwa mapy, dla której dostępny jest wynik np (“Level1.txt”)*
  int fileBytelength *- inforamcja o rozmiarze pliku z listą najlepszych wyników, który zostanie przesłany. Dzięki temu klient wie, kiedy ma przestać odbierać dane.*
  byte[] fileData *- Zawartość danego pliku z najlepszymi wynikami*

  Przykładowe dane otrzymane przez klienta:
  a) *From server: replyForHighScores 2 7 Level1.txt 23 Asia 21 Marek 32 Jacek 17 8 Level22.txt 12 Jan 5 Kuba 12*

- *Klient wysyła żądanie o wysłanie dostępnych map:*
  C: getMaps\n =>S

Przykład wiadomości jaką otrzyma serwer:

a) *From client: getMaps*

Opis działania: Na podstawie tej wiadomości serwer sprawdza ilość dostępnych map w swojej bazie danych

- *Serwer odpowiada klientowi przesyłając dane do wszystkich posiadanych map (int)(int)(byte[])(int)(byte[]) … (int)(byte[])*
  *S:replyForMaps**(int numberOfMaps) (int fileNameByteLength1) (byte[] fileNameData1) (int fileBytelength1) (byte[] fileData1) …... (int fileNameByteLengthX) (byte[] fileNameDataX) (int fileBytelengthX) (byte[] fileDataX)**\n =>C*

gdzie:
***int numberOfMaps** - liczba dostępnych map, do których przesłane zostaną dane. Dzięki temu klient wie ile razy ma odczytać dane wysłane z serwera*

***int fileNameByteLength** - informacja o rozmiarze nazwy mapy w bajtach*
Dzięki temu klient wie, kiedy ma przestać odbierać dane.
***byte[] fileNameData** - nazwa mapy, np (“Level54.txt”)*
***int fileBytelength** - inforamcja o rozmiarze mapy, która zostanie przesłana. Dzięki temu klient wie, kiedy ma przestać odbierać dane.*
***byte[] fileData** - Zawartość danej mapy*

Przykładowe dane otrzymane przez klienta:

b) *From server: replyForMaps 2 7 Level1.txt 10 SmallBall 2 MegaBall 5 SmallBall 8 7 Level2.txt 14 SmallBall 3 SmallBall 4 SmallBall 5 LargeBall 7*

- *Klient wysyła żądanie o listę configów:*

C: getConfigs\n=>S

Przykład wiadomości jaką otrzyma serwer:

a) *From client: getConfigs*

- *Serwer odpowiada klientowi przesyłając dostępne configi (int)(int)(byte[])(int)(byte[]) … (int)(byte[])*
  *S: replyConfigs **(int numberOfConfigs) (int fileNameByteLength1) (byte[] fileNameData1) (int fileBytelength1) (byte[] fileData1) …... (int fileNameByteLengthX) (byte[] fileNameDataX) (int fileBytelengthX) (byte[] fileDataX)**\n =>C*

gdzie:
***int numberOfConfigs** - liczba dostępnych configów, do których przesłane zostaną dane. Dzięki temu klient wie ile razy ma odczytać dane wysłane z serwera*

***int fileNameByteLength** - informacja o rozmiarze nazwy configu w bajtach*
Dzięki temu klient wie, kiedy ma przestać odbierać dane.
***byte[] fileNameData** - nazwa configu, np (“serverConfig.txt”)*

int fileBytelength *- inforamcja o rozmiarze configu, który zostanie przesłany. Dzięki temu klient wie, kiedy ma przestać odbierać dane.*
byte[] fileData *- Zawartość danego configu*

Przykład wiadomości jaką otrzyma klient:

a) *From server: sreplyForConfig 1 13 serverConfig.txt 621 Pang difficulty=2 defaultXFrameSize=960 defaultYFrameSize=540*
Player health=5 damage=50 ...

- *Po zakończeniu rozgrywki klient wysyła żądanie o zapis nicku gracza i jego wyniku uzyskanego na konkretnej mapie (String nick) (String mapName) (int score): Parametry są rozdzielone spacją*
C: saveScore ***nick mapName score****\n => S*

gdzie:
mapName - nazwa mapy, dla której gracz uzyskał dany wynik
nick - nick gracza podany na początku gry (Gdy gracz nie wpisał swojego nicku, klient prześle domyślny nick - ”Unknown”)
score - wynik gracza po skończeniu gry

Przykład wiadomości jaką otrzyma serwer:

a) *From client: savePlayerScore* ***Janek level5 312***

Opis działania: Serwer otrzymuje dane, które zapisuje w swojej bazie danych.

<u>Opis działania serwera i klienta:</u>

Zasada działania protokołu:

- serwer tworzy socket i czeka na połączenie od klienta
- klient inicjuje połączenie z serwerem (adres –> zapisany w plikach konfiguracyjnych klienta, port -> dynamicznie przydzielany w momencie odpalania aplikacji / zapisany w configu)
- serwer akceptuje połączenie od klienta
- aby móc obsłużyć nowych klientów serwer tworzy nowy socket

Działanie klienta:

- po odpaleniu gry użytkownik może połączyć się z serwerem, gdy ten jest online
- klient łączy się z serwerem i w zależności od wyboru użytkownika wysyła żądanie o: mapy / listę najlepszych wyników
- gdy gra się zakończy, a użytkownik jest połączony z serwerem to klient wysyła żądanie do serwera, aby ten zapisał wynik gracza do swojej bazy danych

Działanie serwera:

- serwer jest w stanie obsłużyć dowolną liczbę klientów

- serwer przechowuje lisy najlepszych wyników dla danych poziomów, a także nowe poziomy
- adres serwera jest zapisany w aplikacji klienta
- łączenie poszczególnych klientów -> wielowątkowość (aby serwer mógł obsługiwać wielu klientów jednocześnie i oni nie czekali na odpowiedź w nieskończoność)
-serwer przechowuje dane na temat najlepszych wyników dla każdej mapy

Łączenie klienta z serwerem:

- klient może zainicjować połączenie z serwerem
- jest określony maksymalny czas na zainicjowanie połączenia:
- *jeśli zostanie przekroczony to użytkownik dostaje informację o braku połączenia i gra w trybie offline, używając tylko map lokalnych i własnych list wyników (tylko z rozgrywek na danym komputerze)*
- *jeśli użytkownik połączy się z serwerem w określonym czasie to wysyła do niego prośbę o przesłanie map oraz list najlepszych wyników do każdej mapy (gramy na mapach serwerowych)*

- po skończeniu gry klient przesyła do serwera wynik i nick użytkownika, żeby serwer zapisał go w swojej bazie danych

Przesyłanie informacji:

a) klient -> serwer (klient wysyła informacje do serwera)

- po przejściu w tryb online klient wysyła prośbę do serwera o przesłanie danych na temat dostępnej ilości map i list najlepszych wyników
- po zakończeniu gry, klient wysyła żądanie do serwera, aby ten zapisał nick użytkownika i jego wynik

b) serwer ->klient (serwer wysyła informacje do klienta)

- serwer przesyła mapy / wyniki na prośbę klienta
- po każdym wejściu w listę „highscores" następuje przesłanie aktualnych list wyników z serwera do klienta

Wczytywanie map z serwera:

- po przesłaniu danych na temat ilości map użytkownik wybiera, na której mapie chce zagrać
- po kliknięciu na daną mapę klient wysyła żądanie do serwera, aby ten przesłał mu dane tej mapy (przeciwnicy + czas ich spawn'u)
- po wczytaniu mapy użytkownik jest przenoszony do panelu z grą, gdzie może pograć na wczytanej wcześniej mapie

Wczytywanie list najlepszych wyników z serwera:

-lista wyników jest pobierana po każdym kliknięciu na wynik danej mapy, w celu pobrania najbardziej aktualnych danych

- aplikacja klienta wczytuje dane i wyświetla na ekranie kilka najlepszych wyników dla wybranej mapy