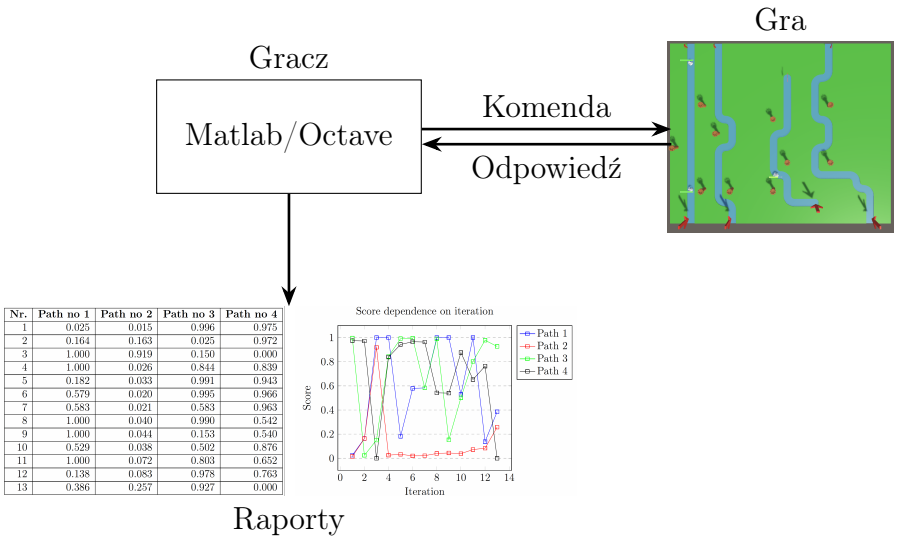
Spis treści

1 Wprowadzenie

W skład oprogramowania MCDA MEASURE (ang. Multi-Criteria Decision Analysis MEthods Assessment through SimUlation REsearch) wchodzi moduł gry wykorzystujący silnik gry Unity oraz biblioteka wykonana w środowisku Matlab/Octave. Moduł gry może być sterowany za pomocą komend w których dane są zapisane formacie XML-a (rys. 1). Odpowiada za wizualizację planszy gry oraz sterowanie jej przebiegiem. Domyślnie gra jest tak skonfigurowana, żeby nasłuchiwała na porcie 55001. Wpisany domyślnie adres (127.0.0.1) umożliwia łączenie się z grą tylko z komputera na którym jest ona zainstalowana (można to jednak zmienić).
Biblioteka ma za zadanie ułatwić tworzyć oprogramowanie w Matlab-ie/Octave przeznaczone do testowania metod wspomagania decyzji. Zawiera metody przygotowujące i formatujące komendy, które mają być przesyłane do gry. Umożliwia także dekodowanie otrzymanych od gry odpowiedzi. Dodatkową możliwością biblioteki jet konwersja tablic do formatu środowiska tabular Latex'a oraz formatu umożliwiającego odczytanie ich przez moduł tworzenia wykresów pakietu tikz.

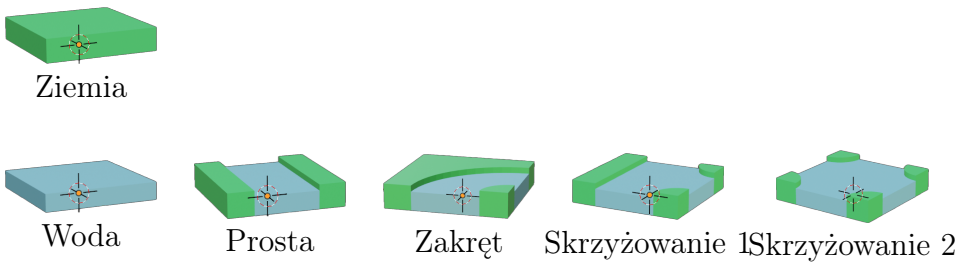
2 Moduł gry

Gra jest wykonana w środowisku Unity. Należy ona do typu tower defense. Plansza gry składa się z kafelków będących trójwymiarowymi modelami dzielącymi się na dwie kategorie: ziemia i droga. Jest jeden kafelek typu ziemia i pięć kafelków typu droga (rys. 2).
Na rysunku 3 pokazano w którym miejscu w interfejsie edytora Unity należy zmienić modele kafelków (o ile to będzie istniała taka potrzeba). Górne zaznaczenie pokazuje miejsce zmiany modelu kafelka typu ziemia. Dolne zaznaczenie pokazuje miejsce zmiany pierwszego modelu kafelka typu droga. Pozostałe znajdują się w WaterStraight, WaterCurve itd. Elementy WaterBegin, WaterBeginEnd itp. są kombinacją modeli z rysunku 2 ze strzałkami.

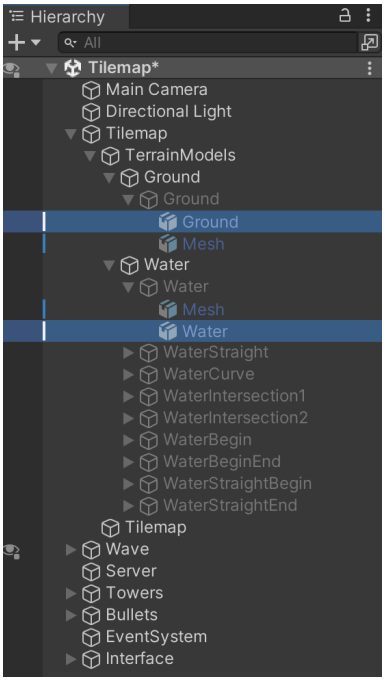


Nr.	Path no 1	Path no 2	Path no 3	Path no 4
1	0.025	0.015	0.996	0.975
2	0.164	0.163	0.025	0.972
3	1.000	0.919	0.150	0.000
4	1.000	0.026	0.844	0.839
5	0.182	0.033	0.991	0.943
6	0.579	0.020	0.995	0.966
7	0.583	0.021	0.583	0.963
8	1.000	0.040	0.990	0.542
9	1.000	0.044	0.153	0.540
10	0.529	0.038	0.502	0.876
11	1.000	0.072	0.803	0.652
12	0.138	0.083	0.978	0.763
13	0.386	0.257	0.927	0.000

Rysunek 1: Architektura systemu



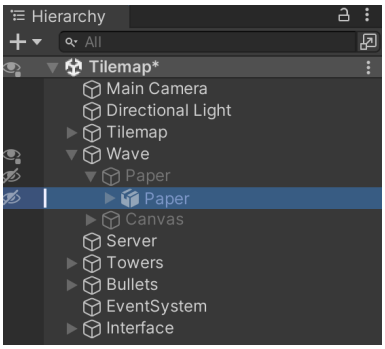
Rysunek 2: Modele kafelków



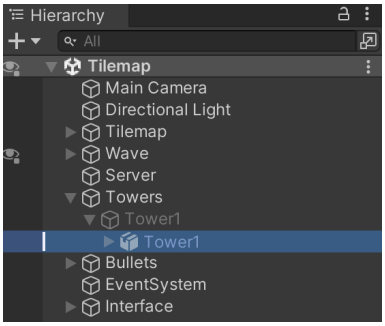
Rysunek 3: Zmiana modeli tiles

Na podobnej zasadzie można zmieniać model przeciwników i wież. Miejsce zmian modeli są pokazane na rysunkach 4 i 5.
Rysunek 6 przedstawia miejsce w którym w edytorze Unity można zmienić parametry przeciwników:

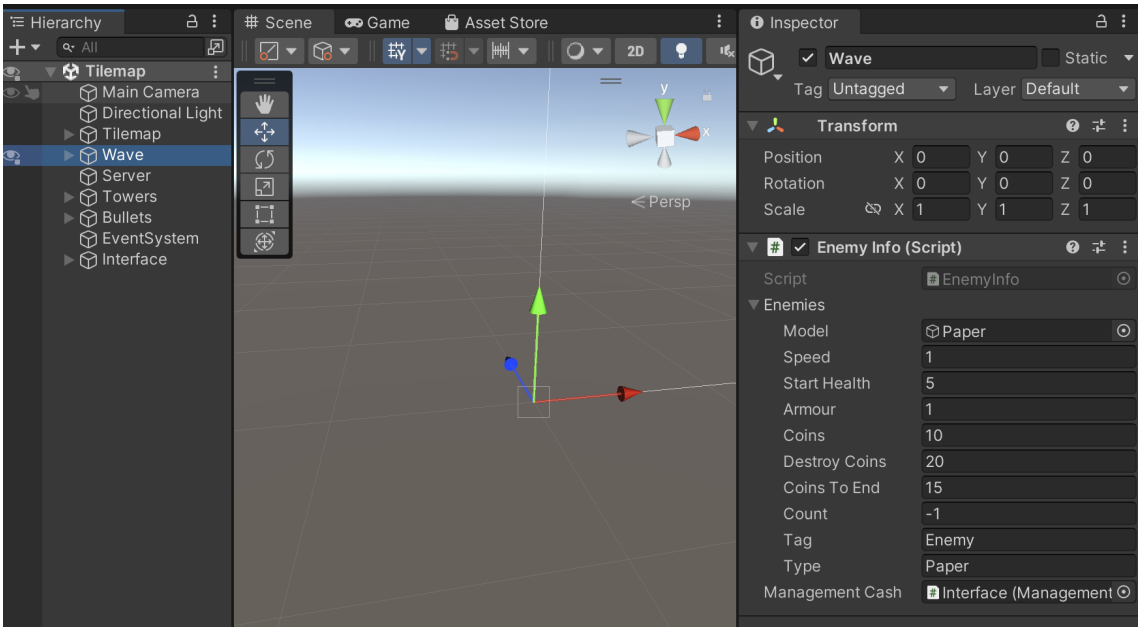
- Speed – prędkość poruszania się,
- Start health – startowe życie,
- Armour – wartość odejmowana od zadanych obrażeń (powoduje niewrażliwość na pociski, które mają niższą liczbę zadawanych obrażeń od Armour),
- Coins – liczba monet potrzebna do utworzenia przeciwnika,
- Destroy Coins – liczba monet jaką otrzymują wieże za zabicie przeciwnika,
- Coins To End – liczba monet jaką otrzymują przeciwnicy jeżeli przeciwnik dotrze do punktu końcowego,
- Count – liczba dostępnych przeciwników (-1 oznacza niegraniczoną liczbę przeciwników).



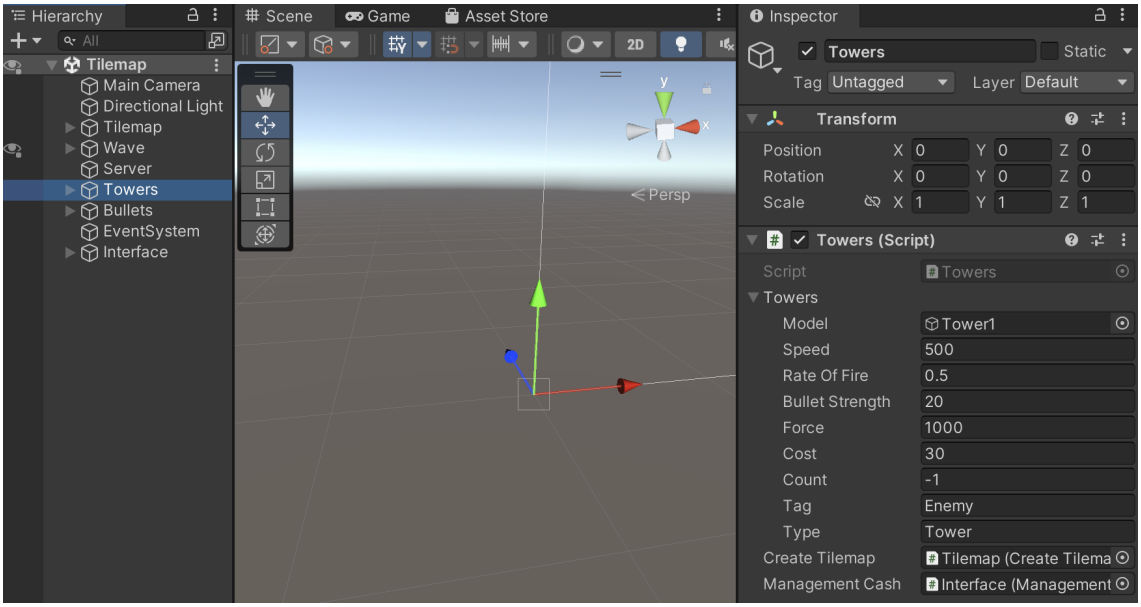
Rysunek 4: Zmiana modelu przeciwnika



Rysunek 5: Zmiana modelu wieży



Rysunek 6: Zmiana parametrów przeciwników



Rysunek 7: Zmiana parametrów wież

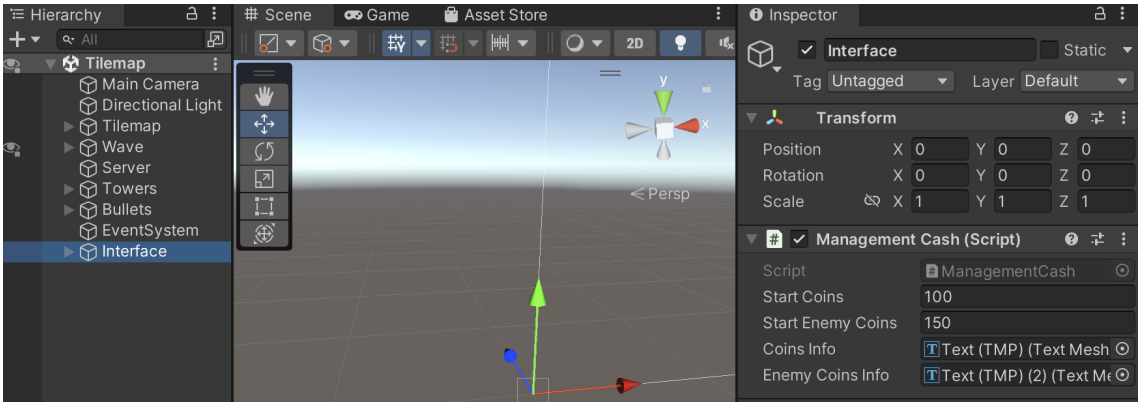
Rysunek 7 przedstawia miejsce w którym w edytorze Unity można zmienić parametry wież:

- Speed – prędkość obrotu,
- Rate Od Fire – szybkostrzelność,
- Bullet Strength – liczba zadawanych obrażeń,
- Force – siła wystrzelenia pocisku przekładająca się na jego zasięg,
- Coins – liczba monet potrzebna do utworzenia wieży,
- Count – liczba dostępnych wież (-1 oznacza niegraniczoną liczbę wież).

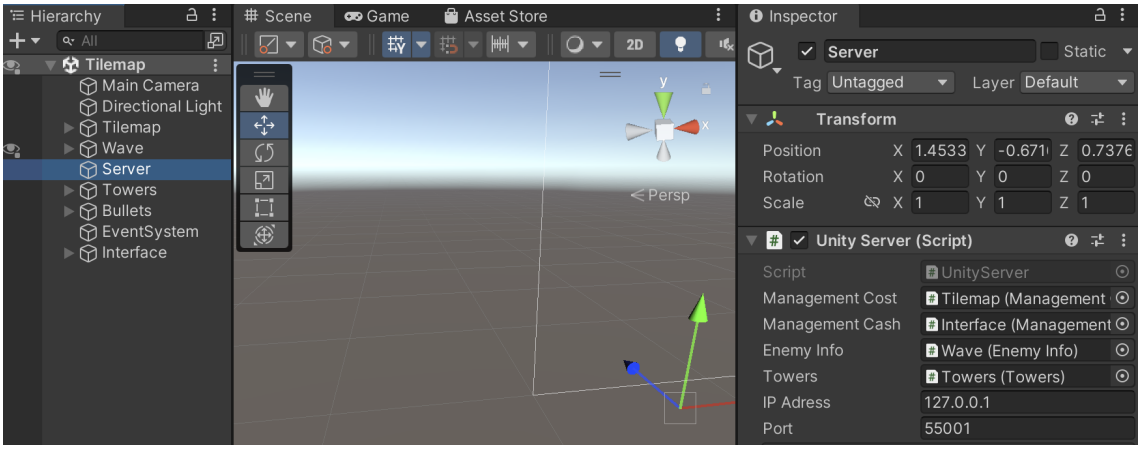
Rysunek 8 przedstawia miejsce w którym w edytorze Unity można zmienić początkową liczbę monet:

- Start Coins – początkowa liczba monet wież;
- Start Enemy Coins – początkowa liczba monet przeciwników.

Rysunek 9 przedstawia miejsce w którym w edytorze Unity można zmienić parametry serwera:



Rysunek 8: Początkowe coins



Rysunek 9: Parametry serwera komunikacyjnego

- IP Adress – adres ip serwera (127.0.0.1 oznacza, że serwer będzie dostępny tylko dla oprogramowania zainstalowanego na tym samym komputerze co gra);
- Port – port na którym nasłuchuje serwer.

Planszę gry stanowi zbiór $n \times m$ kafelków mających podstawę kwadratu. n i m to odpowiednio szerokość i wysokość planszy wyrażona w długościach boku kwadratu będącego podstawą kafelka. Przykładowy XML definiujący planszę gry:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
      http://www.w3.org/2001/XMLSchema">
3 <!--Informacja o tym, ze dalej bedzie definiowana plansza gry.-->
```

```
<Tilemap>
<!-- Oznaczenie tabeli. Tabela zawiera tyle wierszy i kolumn ile wierszy i kolumn posiada plansza. -->
  <Table>
    <Row>
<!-- Okreslenie rodzaju pojedynczego kafelka. Dopuszczalne typy to Ground i Water. Po kafleku typu Water moga poruszac sie przeciwnicy. -->
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
<!-- Okreslenie typu dla kafelka o specjalnych wlasnosciach. End oznacza koniec sciezki. Jest to miejsce, do ktorego maja dotrzec przeciwnicy. -->
      <Cell><Data type="End">Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
<!-- Okreslenie typu dla kafelka o specjalnych wlasnosciach. Begin oznacza poczatek sciezki. Jest to miejsce w ktorym pojawiaja sie przeciwnicy. -->
      <Cell><Data type="Begin">Water</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
```

```
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data type="Begin">Water</Data></Cell>
    </Row>
    <Row>
      <Cell><Data type="End">Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
```

```
96        <Cell><Data>Ground</Data></Cell>
97        <Cell><Data>Ground</Data></Cell>
98        <Cell><Data>Ground</Data></Cell>
99      </Row>
100     <Row>
101       <Cell><Data>Ground</Data></Cell>
102       <Cell><Data>Ground</Data></Cell>
103       <Cell><Data>Ground</Data></Cell>
104       <Cell><Data>Ground</Data></Cell>
105       <Cell><Data>Ground</Data></Cell>
106       <Cell><Data>Ground</Data></Cell>
107       <Cell><Data>Ground</Data></Cell>
108       <Cell><Data>Ground</Data></Cell>
109       <Cell><Data>Ground</Data></Cell>
110       <Cell><Data>Ground</Data></Cell>
111       <Cell><Data>Ground</Data></Cell>
112       <Cell><Data>Ground</Data></Cell>
113       <Cell><Data>Ground</Data></Cell>
114     </Row>
115     <Row>
116       <Cell><Data>Ground</Data></Cell>
117       <Cell><Data>Ground</Data></Cell>
118       <Cell><Data>Ground</Data></Cell>
119       <Cell><Data>Water</Data></Cell>
120       <Cell><Data>Water</Data></Cell>
121       <Cell><Data>Water</Data></Cell>
122       <Cell><Data>Water</Data></Cell>
123       <Cell><Data>Ground</Data></Cell>
124       <Cell><Data>Ground</Data></Cell>
125       <Cell><Data>Ground</Data></Cell>
126       <Cell><Data>Ground</Data></Cell>
127       <Cell><Data>Ground</Data></Cell>
128       <Cell><Data>Ground</Data></Cell>
129     </Row>
130     <Row>
131       <Cell><Data>Ground</Data></Cell>
132       <Cell><Data>Water</Data></Cell>
133       <Cell><Data>Water</Data></Cell>
134       <Cell><Data>Water</Data></Cell>
135       <Cell><Data>Ground</Data></Cell>
136       <Cell><Data>Ground</Data></Cell>
137       <Cell><Data>Water</Data></Cell>
138       <Cell><Data>Water</Data></Cell>
139       <Cell><Data>Water</Data></Cell>
140       <Cell><Data>Water</Data></Cell>
141       <Cell><Data type="Begin">Water</Data></Cell>
142       <Cell><Data>Ground</Data></Cell>
143       <Cell><Data>Ground</Data></Cell>
144     </Row>
```

```
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data type="End">Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data type="Begin">Water</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
```

```
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data type="End">Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
    <Row>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
```

```
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
      <Cell><Data>Ground</Data></Cell>
    </Row>
  </Table>
</Tilemap>
</Data>
```

Po wysłaniu danych, zwracana jest informacja w postaci XML. W przypadku definicji planszy gry zwracane są dane XML z informacją o poprawności (lub nie) informacji przekazanej do serwera:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="Ok" />
```

XML definiujący parametry przeciwników:

```
<?xml version="1.0" encoding="utf-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema">
<!-- Komenda dla serwera. SetEnemies oznacza nakaz zdefiniowania typow
    przeciwnikow. -->
<Command name="SetEnemies">
<!-- Lista typow przeciwnikow. -->
  <SetEnemies>
<!-- Definicja typu przeciwnika. no - numer typu, count - maksymalna
    liczba przeciwnikow (wartosc ujemna oznacza dowolna liczbe
    przeciwnikow), speed - szybkosc przemieszczania sie przeciwnikow,
    startHealth - ilosc poczatkowego zycia, armour - stopien odpornosci na
     strzaly wiez, cost - koszt stworzenia i wyslania przeciwnika,
    destroyCoins - kwota jaka dostaja wieze za zniszczenie przeciwnika,
    coinsToEnd - kwota jaka dostaja przeciwnicy za dotarcie przeciwnika do
     punktu koncowego, type - nazwa typu przeciwnika, tag - nazwa rodzaju
    obiektu. -->
      <Enemy no="1" count="-1" speed="2" startHealth="20" armour="2" cost=
          "30" destroyCoins="30" coinsToEnd="40" type="Paper" tag="Enemy">
      </Enemy>
    </SetEnemies>
</Command>
</Data>
```

Po zdefiniowaniu parametrów zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).
XML definiujący parametry wież:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema">
<!-- Komenda dla serwera. SetTowers oznacza nakaz zdefiniowania typow wiez
    . -->
<Command name="SetTowers">
<!-- Lista typow wiez. -->
<SetTowers>
<!-- Definicja typu wiezy. no - numer typu, count - maksymalna liczba wiez
     (wartosc ujemna oznacza dowolna liczbe wiez), speed - szybkosc
    rotacji wiezy, rateofFire - szybkostrzelnosc wiezy, force - sila z
    jaka wyrzucany jest pocisk, bulletStrength - ilosc zadanych ran, cost
    - koszt postawienia wiezy, type - nazwa typu wiezy, tag - rodzaj
    obiektu ktory atakowac bedzie wieza. -->
<Tower no="0" count="-10" speed="1000" rateofFire="1" force="1000"
    bulletStrength="5" cost="10" type="Tower" tag="Enemy">
</Tower>
    </SetTowers>
</Command>
</Data>
```

Po definiowaniu parametrów zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).
Utworzenie przeciwnika i wypuszczenie go wybraną ścieżką:

```
<?xml version="1.0" encoding="utf-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema">
<!-- Komenda dla serwera. StartEnemy oznacza nakaz utworzenia przeciwnika
    i wypuszczenia go wybrana sciezka. -->
<Command name="StartEnemy">
<!-- Utworzenie przeciwnika. no - typ przeciwnika. -->
<StartEnemy no="1">
<!-- Okreslenie punktu startowego. no - numer punktu. -->
      <Begin no="1">
      </Begin>
<!-- Okreslenie punktu koncowego. no - numer punktu. -->
      <End no="2">
      </End>
    </StartEnemy>
</Command>
</Data>
```

Po utworzeniu przeciwnika zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).
XML powodujący dodanie nowej wieży:

```
<?xml version="1.0" encoding="utf-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema">
<!-- Komenda dla serwera. AddTower oznacza nakaz utworzenia wiezy. -->
```

```
<Command name="AddTower">
<!-- Dodanie wiezy. no - numer typu wiezy, x,y - wspolrzedne polozenia
    wiezy. -->
  <AddTower no="0" x="2" y="8">
  </AddTower>
</Command>
</Data>
```

Po dodaniu nowej wieży zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).
Żądanie przesłania informacji o ścieżkach:

```
<?xml version="1.0" encoding="utf-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema">
<!-- Komenda dla serwera. GetChoiceOfPathData oznacza nakaz przeslania
    informacji o sciezkach. -->
<Command name="GetChoiceOfPathData">
</Command>
</Data>
```

Zwracana przez serwer informacja:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="ChoiceOfPathData">
<!-- Lista sciezek. beginTileCount - liczba punktow poczatkowych,
    endTileCount - liczba punktow koncowych. -->
  <ChoiceOfPath beginTileCount="4" endTileCount="4">
<!-- Informacje o sciezce. cost - koszt przejscia sciezki, shotAtTiles -
    liczba ostrzeliwanych kafelkow sciezki, towers - liczba wiez przy
    sciezce, sumTowerPlace - liczba wolnych miejsc do postawienia wiez (
    tak aby ostrzeliwaly sciezke), hits - liczba pociskow ktore trafily w
    przeciwnika, nohits - liczba pociskow, ktore nie trafily w przeciwnika
    . -->
    <Path cost="12" shotAtTiles="3" towers="1" sumTowerPlace="26" hits="0"
        nohits="0">
<!-- Poczatek sciezki. x, y - wspolrzedne poczatku sciezki, no - numer
    punktu poczatkowego. -->
      <Begin x="1" y="12" no="0">
<!-- Poczatek sciezki. x, y - wspolrzedne poczatku sciezki, no - numer
    punktu poczatkowego. -->
<!-- Informacja o przeciwnikach, ktorzy weszli sciezke. type - nazwa typu
    przeciwnika, enemies - liczba przeciwnikow, ktorzy weszli na sciezke,
    endMeanHealth - sredni poziom zycia przeciwnikow, ktorzy wchodza na
    sciezke. -->
        <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
        <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
      </Begin>
<!-- Koniec sciezki. x, y - wspolrzedne poczatku sciezki, no - numer
    punktu koncowego. -->
```

```
15        <End x="1" y="0" no="0">
16  <!-- Informacja o przeciwnikach, ktorzy przeszli sciezke. type - nazwa
        typu przeciwnika, enemies - liczba przeciwnikow, ktorzy dotarli do
        konca sciezki, endMeanHealth - sredni poziom zycia przeciwnikow,
        ktorzy dotarli do konca sciezki. -->
17          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
18          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
19        </End>
20        <Table />
21      </Path>
22      <Path cost="15" shotAtTiles="3" towers="1" sumTowerPlace="31" hits="0"
            nohits="0">
23        <Begin x="3" y="12" no="1">
24          <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
25          <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
26        </Begin>
27        <End x="4" y="0" no="1">
28          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
29          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
30        </End>
31        <Table />
32      </Path>
33      <Path cost="13" shotAtTiles="0" towers="0" sumTowerPlace="34" hits="0"
            nohits="0">
34        <Begin x="8" y="10" no="2">
35          <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
36          <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
37        </Begin>
38        <End x="10" y="1" no="2">
39          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
40          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
41        </End>
42        <Table />
43      </Path>
44      <Path cost="19" shotAtTiles="0" towers="0" sumTowerPlace="38" hits="0"
            nohits="0">
45        <Begin x="11" y="12" no="3">
46          <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
47          <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
48        </Begin>
49        <End x="14" y="0" no="3">
50          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
51          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
52        </End>
53        <Table />
54      </Path>
55  <!-- Dostepne srodki dla przeciwnikow. -->
56      <Waves cash="150" />
57  <!-- Dostepne srodki dla menadzera wiez. -->
```

```
    <Towers cash="90" />
  </ChoiceOfPath>
</Answer>
```

Żądanie przesłania szczegółowych informacji o stanie gry:

```
<?xml version="1.0" encoding="utf-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="
    http://www.w3.org/2001/XMLSchema">
<!-- Komenda dla serwera. LevelData oznacza nakaz przeslania szczegolowych
     informacji o stanie gry. -->
<Command name="LevelData">
</Command>
</Data>
```

Zwracana przez serwer informacja:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="LevelData">
<!-- Lista sciezek oraz zbior informacji o stanie gry. beginTileCount -
    liczba punktow poczatkowych, endTileCount - liczba punktow koncowych.
    -->
  <LevelPath beginTileCount="4" endTileCount="4">
    <Path cost="12" shotAtTiles="3" towers="1" sumTowerPlace="26" hits="0"
         nohits="0">
      <Begin x="1" y="12" no="0">
        <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
        <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
      </Begin>
      <End x="1" y="0" no="0">
        <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
        <Enemy type="Paper" enemies="0" endMeanHealth="0" />
      </End>
<!-- Tabela ze wspolrzednymi 3D kafelek terenu. -->
      <Table>
<!-- Wspolrzedne x,y,z kafelek terenu. -->
        <Element x="1" y="0" z="12" />
        <Element x="1" y="0" z="11" />
        <Element x="1" y="0" z="10" />
        <Element x="1" y="0" z="9" />
        <Element x="1" y="0" z="8" />
        <Element x="1" y="0" z="7" />
        <Element x="1" y="0" z="6" />
        <Element x="1" y="0" z="5" />
        <Element x="1" y="0" z="4" />
        <Element x="1" y="0" z="3" />
        <Element x="1" y="0" z="2" />
        <Element x="1" y="0" z="1" />
        <Element x="1" y="0" z="0" />
      </Table>
```

```
31      </Path>
32      <Path cost="15" shotAtTiles="3" towers="1" sumTowerPlace="31" hits="0"
            nohits="0">
33        <Begin x="3" y="12" no="1">
34          <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
35          <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
36        </Begin>
37        <End x="4" y="0" no="1">
38          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
39          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
40        </End>
41        <Table>
42          <Element x="3" y="0" z="12" />
43          <Element x="3" y="0" z="11" />
44          <Element x="3" y="0" z="10" />
45          <Element x="3" y="0" z="9" />
46          <Element x="3" y="0" z="8" />
47          <Element x="3" y="0" z="7" />
48          <Element x="4" y="0" z="7" />
49          <Element x="4" y="0" z="6" />
50          <Element x="4" y="0" z="5" />
51          <Element x="3" y="0" z="5" />
52          <Element x="3" y="0" z="4" />
53          <Element x="3" y="0" z="3" />
54          <Element x="3" y="0" z="2" />
55          <Element x="3" y="0" z="1" />
56          <Element x="4" y="0" z="1" />
57          <Element x="4" y="0" z="0" />
58        </Table>
59      </Path>
60      <Path cost="13" shotAtTiles="0" towers="0" sumTowerPlace="34" hits="0"
            nohits="0">
61        <Begin x="8" y="10" no="2">
62          <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
63          <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
64        </Begin>
65        <End x="10" y="1" no="2">
66          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
67          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
68        </End>
69        <Table>
70          <Element x="8" y="0" z="10" />
71          <Element x="8" y="0" z="9" />
72          <Element x="8" y="0" z="8" />
73          <Element x="8" y="0" z="7" />
74          <Element x="8" y="0" z="6" />
75          <Element x="7" y="0" z="6" />
76          <Element x="7" y="0" z="5" />
77          <Element x="7" y="0" z="4" />
```

```
78          <Element x="7" y="0" z="3" />
79          <Element x="8" y="0" z="3" />
80          <Element x="8" y="0" z="2" />
81          <Element x="8" y="0" z="1" />
82          <Element x="9" y="0" z="1" />
83          <Element x="10" y="0" z="1" />
84        </Table>
85      </Path>
86      <Path cost="19" shotAtTiles="0" towers="0" sumTowerPlace="38" hits="0"
            nohits="0">
87        <Begin x="11" y="12" no="3">
88          <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
89          <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
90        </Begin>
91        <End x="14" y="0" no="3">
92          <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
93          <Enemy type="Paper" enemies="0" endMeanHealth="0" />
94        </End>
95        <Table>
96          <Element x="11" y="0" z="12" />
97          <Element x="11" y="0" z="11" />
98          <Element x="11" y="0" z="10" />
99          <Element x="10" y="0" z="10" />
100         <Element x="10" y="0" z="9" />
101         <Element x="10" y="0" z="8" />
102         <Element x="10" y="0" z="7" />
103         <Element x="11" y="0" z="7" />
104         <Element x="11" y="0" z="6" />
105         <Element x="11" y="0" z="5" />
106         <Element x="10" y="0" z="5" />
107         <Element x="10" y="0" z="4" />
108         <Element x="10" y="0" z="3" />
109         <Element x="11" y="0" z="3" />
110         <Element x="12" y="0" z="3" />
111         <Element x="13" y="0" z="3" />
112         <Element x="13" y="0" z="2" />
113         <Element x="14" y="0" z="2" />
114         <Element x="14" y="0" z="1" />
115         <Element x="14" y="0" z="0" />
116       </Table>
117     </Path>
118 <!-- Informacje o typie przeciwnika. count - maksymalna liczba
      przeciwnikow (wartosc ujemna oznacza dowolna liczbe przeciwnikow),
      speed - szybkosc przemieszczania sie przeciwnikow, startHealth - ilosc
       poczatkowego zycia, armour - stopien odpornosci na strzaly wiez,
      destroyCoins - kwota jaka dosteja wieze za zniszczenie przeciwnika,
      cost - koszt stworzenia i wyslania przeciwnika, coinsToEnd - kwota
      jaka dostaja przeciwnicy za dotarcie przeciwnika do punktu koncowego,
      no - numer typu przeciwnika, type - nazwa typu przeciwnika, tag -
```

```
    nazwa rodzaju obiektu. -->
119     <Enemy count="-1" speed="1" startHealth="5" armour="1" destroyCoins="
            20" cost="10" coinsToEnd="15" no="0" type="Bottle" tag="Enemy" />
120     <Enemy count="-1" speed="2" startHealth="20" armour="2" destroyCoins="
            30" cost="30" coinsToEnd="40" no="1" type="Paper" tag="Enemy" />
121 <!-- Informacje o typie wiezy. count - maksymalna liczba wiez (wartosc
    ujemna oznacza dowolna liczbe wiez), speed - szybkosc obrotu wiezy,
    rateofFire - szybkostrzelnosc wiezy, force - sila z jaka wyrzucany
    jest pocisk, bulletStrength - ilosc zadanych ran przez pocisk, cost -
    koszt postawienia wiezy, no - numer typu, type - nazwa typu wiezy, tag
     - rodzaj obiektu ktory bedzie atakowac wieza. -->
122     <Tower count="-10" speed="1000" rateofFire="1" force="1000"
            bulletStrength="5" cost="10" no="0" type="Tower" tag="Enemy" />
123     <Waves cash="150" />
124     <Towers cash="90" />
125   </LevelPath>
126 </Answer>
```

3 Moduł gracza

Biblioteka stanowi zbiór funkcji wspomagających komunikację z grą oraz tworzenie tabel i wykresów. Funkcje te działają zarówno w środowisku Matlab jak i Octave.

SendData
Wysłanie danych na serwer.

```
1 txt = SendData(IPAddressSend,portSend,data,name, args)
```

Opis:

- IPAddressSend – adres ip serwera,
- portSend – port serwera,
- data – data packet sent to the server,
- name – określa sposób interpretowania danych,
- args – argumenty związane z informacjami sterującymi.

Zwraca odpowiedź serwera w formacie XML.

NumberToName
Zastępuje liczby reprezentującą typy pól ich nazwami.

```
result = NumberToName(array, names)
```

Opis:

- array – tabela zawierająca informacje o mapie,
- names – nazwy pól mapy.

Zwraca tabelę zawierającą informacje o mapie.

ChangeBeginEnd
Oznaczanie początków i końców ścieżek.

```
result = ChangeBeginEnd(array)
```

Opis:

- array – tabela zawierająca informacje o mapie.

Zwraca tabelę zawierającą informacje o mapie.

TilemapToXML
Konwersja mapy z tablicy do formatu XML.

```
txt = TilemapToXML(tilemap)
```

Opis:

- tilemap – mapa w postaci tablicy.

Zwraca mapę zapisaną w formacie xml.
Przykład przesłania do gry polecenia utworzenia planszy:

```
%Adres serwera
IPAddressSend = '127.0.0.1';
%Port na ktorym nasluchuje serwer
portSend = 55001;
%Tablica reprezentujaca plansze gry w ktorej liczby
    reprezentuja typy kafelkow (1 - ziemia, 2 - woda bedaca
    elementem sciezki ruchu przeciwnikow, 3 - poczatek sciezki,
     4 - koniec sciezki)
tilemap = [1 3 1 3 1 1 1 1 1 1 1 3 1 1 1 1;
           1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 1;
           1 2 1 2 1 1 1 1 3 1 2 2 1 1 1 1;
```

```
9            1 2 1 2 1 1 1 1 2 1 2 1 1 1 1 1;
10           1 2 1 2 1 1 1 1 2 1 2 1 1 1 1 1;
11           1 2 1 2 2 1 1 1 2 1 2 2 1 1 1 1;
12           1 2 1 1 2 1 1 2 2 1 1 2 1 1 1 1;
13           1 2 1 2 2 1 1 2 1 1 2 2 1 1 1 1;
14           1 2 1 2 1 1 1 2 1 1 2 1 1 1 1 1;
15           1 2 1 2 1 1 1 2 2 1 2 2 2 2 1 1;
16           1 2 1 2 1 1 1 1 2 1 1 1 1 2 2 1;
17           1 2 1 2 2 1 1 1 2 2 4 1 1 1 2 1;
18           1 4 1 1 4 1 1 1 1 1 1 1 1 1 4 1];
19 %Nazwy typow kafelkow. Pozycja w tablicy odpowiada numerowi z
     tablicy tilemap
20 names{1} = 'Ground';
21 names{2} = 'Water';
22 names{3} = 'Begin';
23 names{4} = 'End';
24 %Obrot tablicy tak aby orientacja tablicy odpowiadala
     orientacji planszy w grze
25 tilemap = rot90(rot90(rot90(tilemap)));
26 % Zamiana tablicy na tablice struktur z nazwami kafekow zamiast
      numerow
27 tilemapNames = NumberToName(tilemap,names);
28 %Zamiana nazw kafelkow Begin i End na Water. Przypisanie tym
     kafelkom oznaczenia poczatku lub konca sciezki.
29 tilemapNames = ChangeBeginEnd(tilemapNames);
30 %Zamiana tablicy sturktur na tekst w formacie xml.
31 txt = TilemapToXML(tilemapNames);
32 %Wyslanie polecenia utworzenia nowej planszy (Tilemap) oraz
     danych w formacie xml do gry.
33 SendData(IPAddressSend,portSend,txt,'Tilemap',[]);
```

Char2Code

Zastępowanie kodów znaków kodami używanymi przez maszynę.

```
1 machineCode = Char2Code(code)
```

Opis:

- code – kod znaku.

Zwraca kody automatu.
Przykład odczytu i dekodowania pliku xml:

```
%Konwersja znaku
code = Char2Code('a');
```

Machine
Automat dzielący tekst na elementy i przypisujący im kody końcowych stanów maszyny.

```
result = Machine(data,t)
```

Opis:

- data – tekst,
- t – tabela przejść pomiędzy stanami automatu.

Zwraca tablicę struktur zawierającą fragment tekstu (pole txt) i przypisany do niego stan (pole stanu).
Przykład odczytu i dekodowania pliku xml:

```
%Odczyt pliku xml
dataTower = fileread('towers.xml');
%Definicja tablicy stanow
t = zeros(11,24);
t(1,1) = 1;t(2,1) = 17;t(3,1) = 2;t(4,1) = 8;t(5,1) = 12;t(6,1)
    = 4;t(8,1) = 6;t(9,1) = 15;t(10,1) = 22;
t(:,2) = 3;t(5,2) = 10;t(10,2) = 20;
t(1,4) = 5;t(2,4) = 5;t(4,4) = 5;t(5,4) = 5;t(6,4) = 4;t(7,4) =
    4;t(9,4) = 5;
t(:,6) = 6;t(8,6) = 7;
t(:,7) = 19;
t(:,8) = 9;
t(:,10) = 11;
t(4,12) = 13;
t(:,13) = 14;
t(:,15) = 16;
t(:,17) = 18;
t(:,20) = 21;
t(4,22) = 23;
t(:,23) = 24;
%Analiza pliku
result = Machine(dataTower,t);
```

W zmiennej result znajduje się tablica struktur zawierająca tekst i numer stanu mu przypisany.

ParseXML

Analizuje tekst zawierający plik XML.

```
result = ParseXML(data)
```

Opis:

- data – tablica tekstowa zawierająca dane w formacie XML.

Zwraca tablicę struktur, których struktura odzwierciedla strukturę danych XML, nazwami pól są nazwy elementów XML.
Przykład odczytania i zdekodowania pliku xml:

```
%Odczytanie pliku xml
dataTower = fileread('towers.xml');
%Konwersja pliku xml
dataTower = ParseXML(dataTower);
```

Zawartość pliku towers.xml:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="LevelData">
  <TowerCoordinates>
    <Element x="2" y="8" no="0" />
    <Element x="8" y="4" no="2" />
  </TowerCoordinates>
</Answer>
```

Plik towers.xml zawiera współrzędne wież i ich numery porządkowe. Przykład dostępu do tych danych:

```
x=dataTower.Answer.TowerCoordinates{1}.Element{i}.x;
y=dataTower.Answer.TowerCoordinates{1}.Element{i}.y;
no=dataTower.Answer.TowerCoordinates{1}.Element{i}.no;
```

GetVectorFromCell

Odczyt wektora danych z wybranego pola tablicy struktury.

```
res = GetVectorFromCell(data, field)
```

Opis:

- data – tablica struktur,
- field – odczytywane pola struktury.

Zwraca wektor danych.
Przykład odczytania współrzędnych x wież jako tablicy:

```
%Odczytanie pliku xml
dataTower = fileread('towers.xml');
%Konwersja pliku xml
dataTower = ParseXML(dataTower);
%Odczytanie wspolrzednych x wiez jako tablicy
x = GetVectorFromCell(dataTower.Answer.TowerCoordinates{1}.
    Element,'x');
```

Zawartość pliku towers.xml:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="LevelData">
  <TowerCoordinates>
    <Element x="2" y="8" no="0" />
    <Element x="8" y="4" no="2" />
  </TowerCoordinates>
</Answer>
```

SetEnemies

Tworzenie informacji w formie XML o konkretnym typie przeciwnika.

```
txt = SetEnemies(count,speed,startHealth,armour,cost,
    destroyCoins,coinsToEnd,type,tag)
```

Opis:

- count – maksymalna liczba przeciwników,
- speed – prędkość przeciwnika,
- startHealth – początkowa wartość życia przeciwnika,
- armour – zbroja wroga (odporność na kule),
- cost – koszt stworzenia i wysłania wroga,
- destroyCoins – zysk dla zarządcy wież za zestrzelenie wroga,
- coinsToEnd – zysk dla menadżera przeciwników, jeśli dotrze on do końca ścieżki,
- type – typ przeciwnika,
- tag – nazwa typu obiektu.

Zwraca informacje zapisane w formacie xml.
Przykład przesłania do gry polecenia utworzenia nowego typu przeciwnika:

```
%Adres serwera
IPAddressSend = '127.0.0.1';
%Port na ktorym nasluchuje serwer
portSend = 55001;
%Utworzenie danych w formacie xml dotyczacych nowego typu
    przeciwnika
txt = SetEnemies(-1,2,20,2,30,30,40,'Paper','Enemy');
%Wyslanie polecenia (Command) utworzenia nowego typu
    przeciwnika (SetEnemies)
SendData(IPAddressSend,portSend,txt,'Command','name="SetEnemies
    "');
```

SetTowers
Tworzenie informacji w formie XML o konkretnym typie wieży

```
txt = SetTowers(count,speed,rateOfFire,force,bulletStrength,
    cost,type,tag)
```

Opis:

- count – maksymalna liczba wież,
- speed – prędkość obrotu wież,
- rateOfFire – szybkostrzelność wież,
- force – siła ognia wieży (określa zasięg),
- bulletStrength – siła pocisku wieży (wpływa na liczbę ran zadanych wrogowi),
- cost – koszt stworzenia wieży,
- type – typ wieży,
- tag – typ obiektu, który zaatakuje wieża.

Zwraca informacje zapisane w formacie xml.
Przykład przesłania do gry polecenia utworzenia nowego typu wieży:

```
%Adres serwera
IPAddressSend = '127.0.0.1';
%Port na ktorym nasluchuje serwer
portSend = 55001;
%Utworzenie danych w formacie xml dotyczacych nowego typu wiezy
```

```
txt = SetTowers(-10,1000,1,1000,5,10,'Tower','Enemy');
%Wyslanie polecenia (Command) utworzenia nowego typu wiezy (SetTowers)
SendData(IPAddressSend,portSend,txt,'Command','name="SetTowers"');
```

StartEnemy

Tworzenie przeciwnika i wysyłanie go wybraną ścieżką.

```
txt = StartEnemy(beginNo,endNo)
```

Description:

- beginNo – numer punktu początkowego,
- endNo – numer punktu końcowego.

Zwraca informacje zapisane w formacie xml.
Przykład utworzenia przeciwnika i wysłania go z punktu startowego 1 do punktu końcowego 3:

```
%Adres serwera
IPAddressSend = '127.0.0.1';
%Port na ktorym nasluchuje serwer
portSend = 55001;
%Utworzenie danych w formacie xml zawierajacych informacje o punkcie startowym i koncowym
txt = StartEnemy(1,3);
%Wyslanie polecenia (Command) utworzenia przeciwnika i wyslania go od wskazanego punktu startowego do wskazanego punktu koncowego (StartEnemy)
errorStartEnemy = SendData(IPAddressSend,portSend,txt,'Command','name="StartEnemy"');
```

AddTower

Dodanie wieży.

```
txt = AddTower(noTower,x,y)
```

Opis:

- noTower – numer wieży,

- x – współrzędna x wieży,
- y – współrzędna y wieży.

Zwraca informacje zapisane w formacie xml.
Przykład dodania wieży o numerze 3 w miejsce o współrzędnych $x = 1$, $y = 4$:

```
%Adres serwera
IPAddressSend = '127.0.0.1';
%Port na ktorym nasluchuje serwer
portSend = 55001;
%Utworzenie danych w formacie xml zawierajacych informacje o
    numerze wiezy i jej wspolrzednych
txt = AddTower(3,1,4);
%Wyslanie polecenia (Command) utworzenia wiezy (AddTower)
errorAddTower = SendData(IPAddressSend,portSend,txt,'Command','
    name="AddTower"');
```

RemoveCriteria
Usunięcie kryteriów, których wartości dla wszystkich wariantów decyzyjnych nie różnią się od siebie.

```
[E,W,PrefDirection, ind] = RemoveCriteria(E,W,PrefDirection)
```

Opis:

- E – tabela danych, kolumny są kryteriami, a wiersze alternatywami,
- W – wagi kryteriów,
- PrefDirection – kryteria Kierunek preferencji (1-max;2-min),
- ind – indeksy kryteriów, które nie zostały usunięte.

Zwraca tabelę danych, kolumny są kryteriami, a wiersze alternatywami.
Przykład usuwania kryteriów:

```
%Tablica danych
E =[12 3 1 13 1 1;
      15 0 0 13 1 1;
      13 0 0 13 1 1;
      19 0 0 13 1 1];
%Wektor wag kryteriow
W=[2 10 8 1 20 10];
```

```
%Wektor kierunkow preferencji kryteriow: 1-max, 2-min
PrefDirection=[2 2 2 2 1 1];
%Usuniecie kryteriow, aktualizowana jest tablica danych oraz
   wektory wag i kierunkow preferencji
[E,W,PrefDirection, ind] = RemoveCriteria(E,W,PrefDirection)
```

TOPSIS
Funkcja TOPIS. Oblicza wartości miary alternatyw metodą TOPSIS.

```
S=TOPSIS(E,W,PrefDirection,p)
```

Opis:

- E – tabela danych, kolumny są kryteriami, a wiersze alternatywami,
- W – wagi kryteriów,
- PrefDirection – kierunek preferencji kryteriów (1-max;2-min),
- p – współczynnik.

Zwraca tablicę wartości miar dla wariantów decyzji.
Przykład obliczania wartości miary:

```
%Tablica danych
E = [12 3 1;
      15 0 0;
      13 0 0;
      19 0 0];
%Wektor wag kryteriow
W=[2 10 8];
%Wektor kierunkow preferencji kryteriow: 1-max, 2-min
PrefDirection=[2 2 2];
%Obliczanie wartosci miary
score=TOPSIS(E,W,PrefDirection,p);
```

VIKOR
Funkcja VIKOR. Oblicza wartości miary alternatyw metodą VIKOR.

```
[Q,S,R]=VIKOR(E,W,PrefDirection,q)
```

Opis:

- E – tabela danych, kolumny są kryteriami, a wiersze alternatywami,

- W – wagi kryteriów,
- PrefDirection – kierunek preferencji kryteriów (1-max;2-min),
- p – współczynnik.

Returns an array of measure values for decision variants.
Przykład obliczenia wartości miary:

```
%Tablica danych
E = [12 3 1;
      15 0 0;
      13 0 0;
      19 0 0];
%Wektor wag kryteriow
W=[2 10 8];
%Wektor kierunkow preferencji kryteriow: 1-max, 2-min
PrefDirection=[2 2 2];
%Obliczenie trzech typow wartosci miary
[Q,S,R]=VIKOR(E,W,PrefDirection,p);
```

GenerateRanking
Tworzy ranking alternatyw.

```
ranking = GenerateRanking(v)
```

Opis:

- v – Wektor ocen alternatyw.

Returns ranking positions.
Example of creating a ranking:

```
%Przykladowy wektor
exampleVector = [1.25 2.23 1.25 0.3 1.5 4.5];
%Utworzenie rankingu
ranking = GenerateRanking(exampleVector);
```

GenerateReport
Generowanie raportu z wynikami testowanej metody MCDA.

```
ranking = GenerateReport(fileNamePlot,fileNameTab,scoreArray,
    funName,EnemiesToEnd,EnemiesMeanHealthRatio,TracksCost);
```

Description:

- fileNamePlot – nazwa pliku, do którego zostanie zapisany wykres,
- fileNameTab – nazwa pliku, do którego zostanie zapisana tabela,
- scoreArray – ocenia (oceny) ścieżki,
- funName – nazwa testowanej metody MCDA,
- EnemiesToEnd – liczba wrogów, którzy dotarli do końca wszystkich ścieżek (główny wynik metody MCDA),
- EnemiesMeanHealthRatio – średni poziom zdrowia wrogów, którzy dotarli do końca ścieżek (dodatkowy wynik metody MCDA),
- TracksCost – całkowita długość ścieżek wybranych metodą MCDA (drugi dodatkowy wynik metody MCDA).

Przykład generowania raportu:

```
%Tablica zawierajaca przykladowe (hipotetyczne) wyniki wyboru sciezki metoda MCDA w grze skladajacej sie z czterech rund z dwiema sciezkami do wyboru
exampleArray = [0.5 0.4;
  0.3 0.4;
  0.7 0.3;
  0.6 0.8];
%Zmienne zawierajace przykladowe (hipotetyczne) wyniki metody MCDA
exampleEnemiesToEnd = 3;
exampleEnemiesMeanHealthRatio = 0.65;
exampleTracksCost = 56;
%Tworzenie raportu z gry z wynikami metody MCDA
GenerateReport('scoreRounds.png','scoreRound.html',exampleArray,'TOPSIS',exampleEnemiesToEnd,exampleEnemiesMeanHealthRatio,exampleTracksCost);
```

Zawartość pliku 'scoreRounds.html':

```
<html>
<body>
<p>MCDA Method: TOPSIS</p>
<p>Enemies to end: 3</p>
<p>Enemies mean health: 0.650000</p>
<p>Tracks cost: 56</p>
<table>
<tr>
```

MCDA Method: TOPSIS

Enemies to end: 3

Enemies mean health: 0.650000

Tracks cost: 56

No.	Path 1	Path 2
1	0.5000	0.4000
2	0.3000	0.4000
3	0.7000	0.3000
4	0.6000	0.8000

Rysunek 10: Wizualizacja wygenerowanej strony html

```
<th>No.</th><th>Path 1</th><th>Path 2</th>
</tr>
<tr>
<td>1</td><td>0.5000</td><td>0.4000</td>
</tr>
<tr>
<td>2</td><td>0.3000</td><td>0.4000</td>
</tr>
<tr>
<td>3</td><td>0.7000</td><td>0.3000</td>
</tr>
<tr>
<td>4</td><td>0.6000</td><td>0.8000</td>
</tr>
</table>
<style>
table {border: 1px solid; border-collapse: collapse;}
th {border: 1px solid; padding-left: 10px; padding-right: 10px;}
td {border: 1px solid; padding-left: 10px; padding-right: 10px;}
</style>
</body>
</html>
```

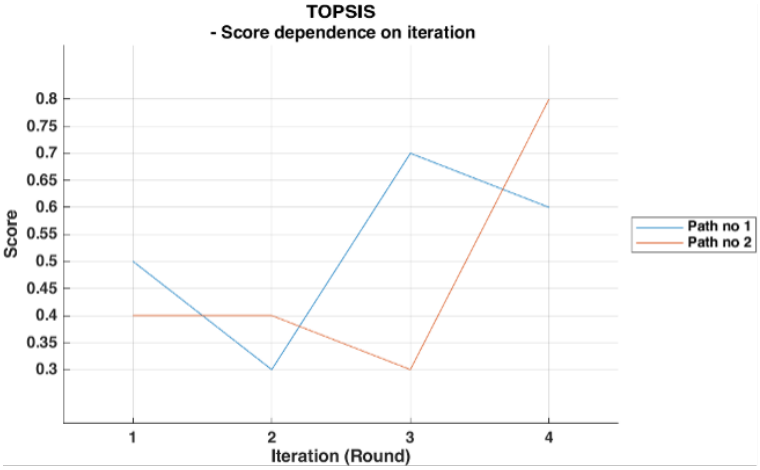
Rysunek 10 przedstawia zrzut ekranu przedstawiający wygląd wygenerowanej strony.
Rysunek 11 przedstawia zawartość pliku 'scoreRounds.png'.

GenerateTabular

Generowanie tabeli typu tabular.

```
GenerateTabular(fileName,data,columnDescriptions,
    rowDescriptions,rowsBold,decimalPlaces)
```

Opis:



Rysunek 11: Zawartość pliku 'scoreRounds.png'

- fileName – nazwa pliku, do którego zostanie zapisana tablica,
- data – zapisana tablica,
- columnDescriptions – opisy kolumn,
- rowDescriptions – opisy wierszy, pusta tablica([]) oznacza brak opisów,
- rowsBold – 0 oznacza, że opisy wierszy tabeli nie są pogrubione, a 1 oznacza że są pogrubione,
- decimalPlaces – liczba miejsc po przecinku.

Przykład generowania tablicy:

```
%Tablica
exampleArray = [1 2;
                3 1;
                5 2;
                2 4];
%Opisy kolumn
columnDescriptions={'No','Data 1','Data 2'};
%Opisy wierszy
rowDescriptions={'1','2','3','4'};
%Utworzenie pliku zawierajacego srodowisko tablular
GenerateTabular('array.tex',exampleArray,columnDescriptions,
   rowDescriptions,0,0);
```

Zawartość pliku array.tex:

No	Data 1	Data 2
1	1	2
2	3	1
3	5	2
4	2	4

Tablica 1: Wygenerowana tablica

```
\begin{tabular}{|r|r|r|}
  \hline
  \textbf{No}& \textbf{Data 1}& \textbf{Data 2}\\
  \hline
  1& 1& 2\\
  \hline
  2& 3& 1\\
  \hline
  3& 5& 2\\
  \hline
  4& 2& 4\\
  \hline
\end{tabular}
```

Tablicę można dołączyć do pliku Latex-a:

```
\begin{table}
\input{array}
\caption{Wygenerowana tablica}
\end{table}
```

Uzyskany efekt przedstawia tablica 1.

GenerateTikzData
Generowanie plików danych dla wykresów Tikz.

```
GenerateTikzData(fileName,data,columnDescriptions)
```

Opis:

- fileName – nazwa pliku, do którego zostanie zapisana tablica,
- data – zapisywana tablica,
- columnDescriptions – opisy kolumn.

Przykład generowania danych:

```
%Tablica
exampleArray = [1 2;
```

```
                3 1;
                5 2;
                2 4];
%Opisy kolumn
columnDescriptions={'No','D1','D2'};
%Utworzenie pliku zawierajacego dane dla wykresow tikz
GenerateTikzData('array.dat',[[1:size(exampleArray,1)]'
    exampleArray],columnDescriptions);
```

Zawartość pliku array.dat:

```
No D1 D2
1 1 2
2 3 1
3 5 2
4 2 4
```

Plik array.dat można dołączyć do wykresu tikz-a:

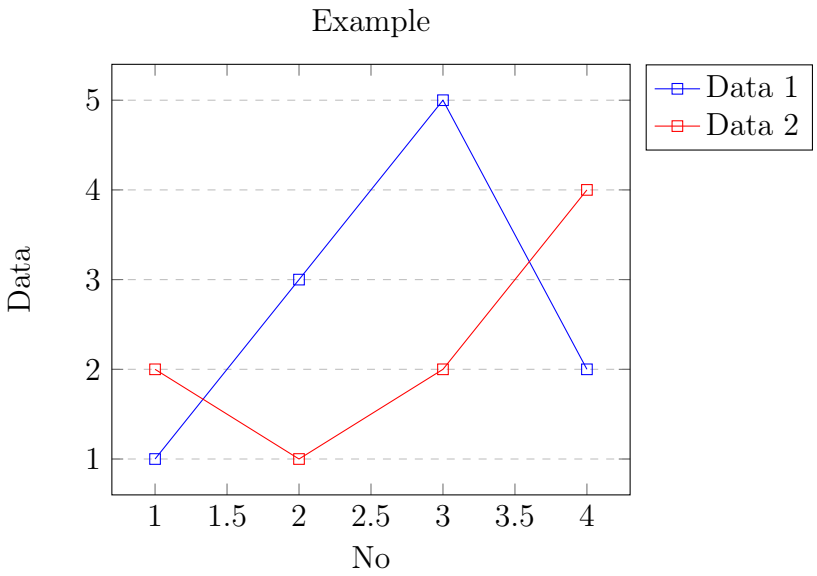
```
\begin{tikzpicture}
\begin{axis}[
    title={Example},
    xlabel={No},
    ylabel={Data},
    legend pos=outer north east,
    ymajorgrids=true,
    grid style=dashed,
]

\addplot[
    color=blue,
    mark=square
    ]
    table[x=No,y=D1]
    {fig/array.dat};
\addplot[
    color=red,
    mark=square
    ]
    table[x=No,y=D2]
    {fig/array.dat};

    \legend{Data 1, Data 2}

\end{axis}
\end{tikzpicture}
```

Uzyskany efekt przedstawia rysunek 12.



Rysunek 12: Wykres na podstawie danych wygenerowanych przez funkcję GenerateTikzData