

Spis treści

1	Introduction	1
2	Game	1
3	Biblioteka	19

1 Introduction

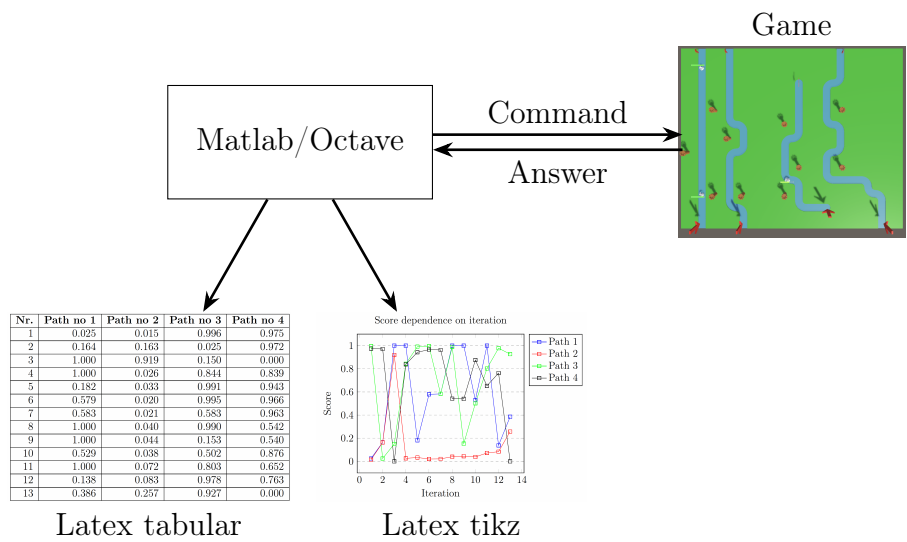
W skład oprogramowania GTest wchodzi moduł gry wykorzystujący silnik gry Unity oraz biblioteka wykonana w środowisku Matlab/Octave. Moduł gry może być sterowany za pomocą komend w których dane są zapisane formacie XML-a (fig. 1). Odpowiada za wizualizację planszy gry oraz sterowanie jej przebiegiem. 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 jest konwersja tablic do formatu środowiska tabular Latex'a oraz formatu umożliwiającego odczytanie ich przez moduł tworzenia wykresów pakietu tikz.

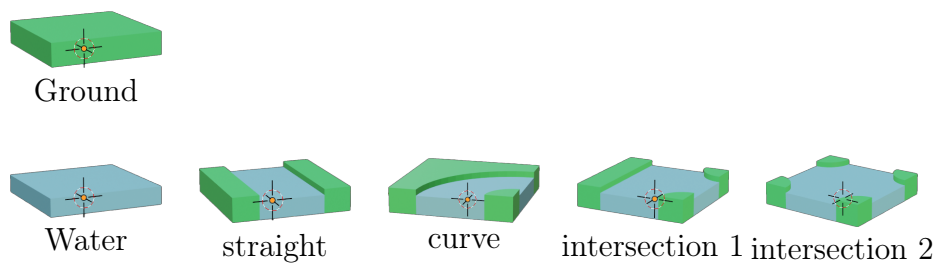
2 Game

Gra jest wykonana w środowisku Unity. Należy ona do typu tower defense. Plansza gry składa się z tiles będących trójwymiarowymi modelami dzielącymi się na dwie kategorie: ziemia i droga. Jest jeden tile typu ziemia i pięć tiles typu droga (fig. 2).

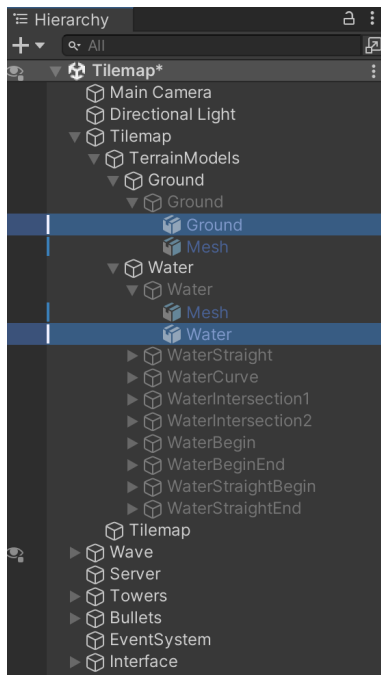
Na rysunku 3 pokazano w którym miejscu w interfejsie edytora Unity należy zmienić modele tiles (o ile to będzie istniała taka potrzeba). Górne zaznaczenie pokazuje miejsce zmiany modelu tile typu ziemia. Dolne zaznaczenie pokazuje miejsce zmiany pierwszego modelu tile typu droga. Pozostałe znajdują się w WaterStraight, WaterCurve itd. Elementy WaterBegin, WaterBeginEnd itp. są kombinacją modeli z rysunku 2 ze strzałkami.



Rysunek 1: Architektura systemu



Rysunek 2: Modele tiles

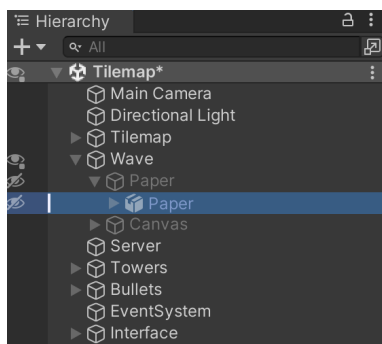


Rysunek 3: Zmiana modeli tiles

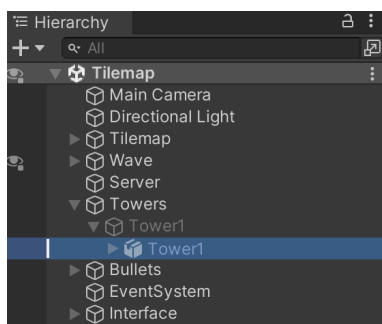
Na podobnej zasadzie można zmieniać model przeciwników i wież. Miejsce zmian modeli są pokazane na rysunkach 4 i 5.

Rysunek 6 przedstawia miejsce w którym w edytorze Unity można zmienić parametry przeciwników:

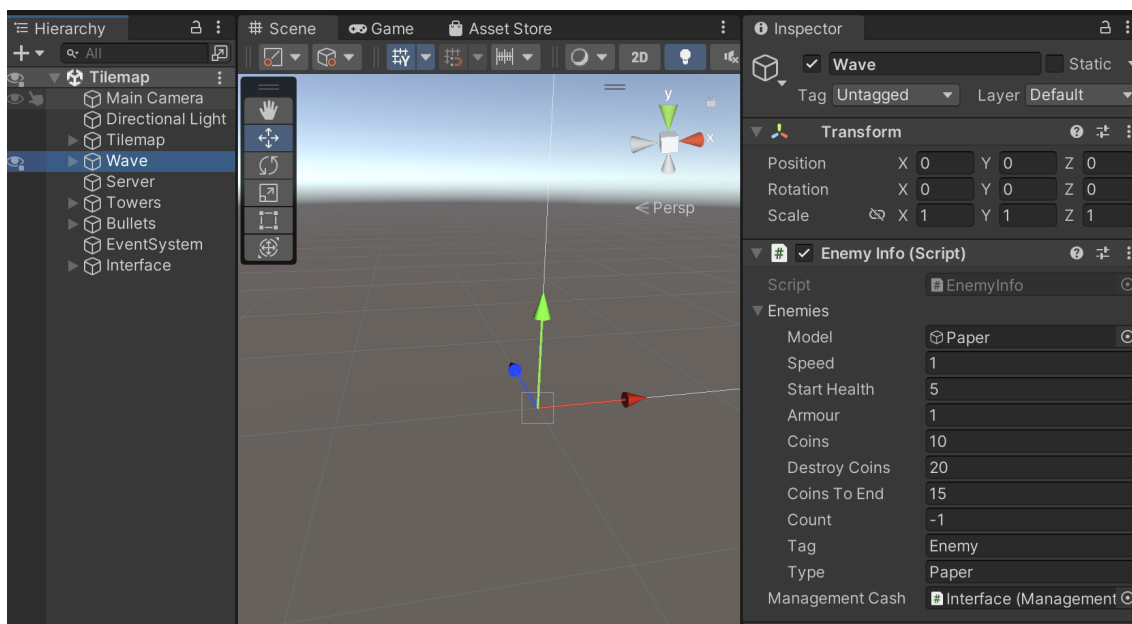
- Speed – prędkość poruszania się,
- Start health – start health,
- Armour – wartość odejmowana od zadanych obrażeń (powoduje niewrażliwość na pociski, które mają niższą liczbę zadawanych obrażeń od Armour),
- Coins – liczba coins potrzebna do utworzenia przeciwnika,
- Destroy Coins – liczba coins jaką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



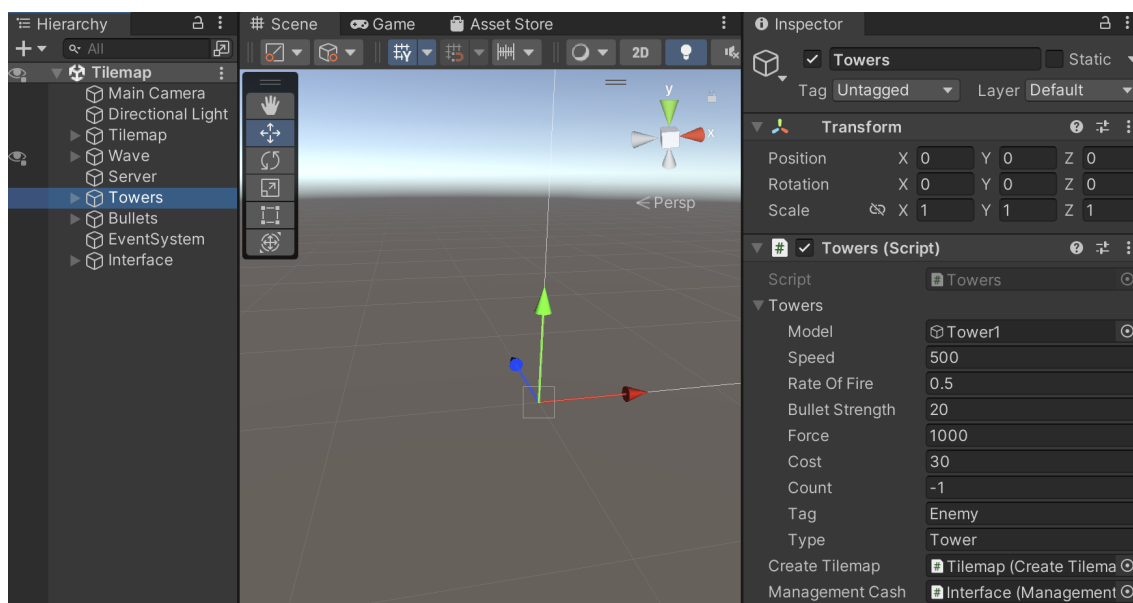
Rysunek 4: Zmiana modelu przeciwnika



Rysunek 5: Zmiana modelu wieży



Rysunek 6: Zmiana parametrów przeciwników



Rysunek 7: Zmiana parametrów wież

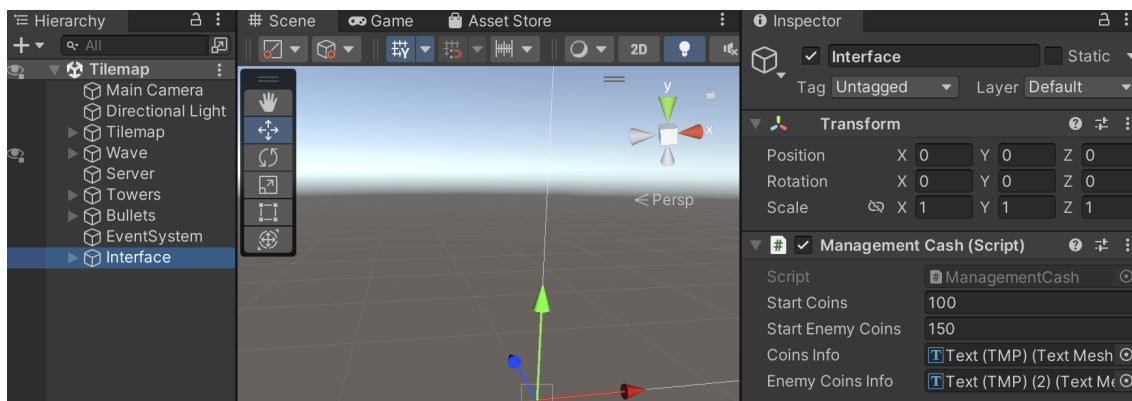
Rysunek 7 przedstawia miejsce w którym w edytorze Unity można zmienić parametry wież:

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

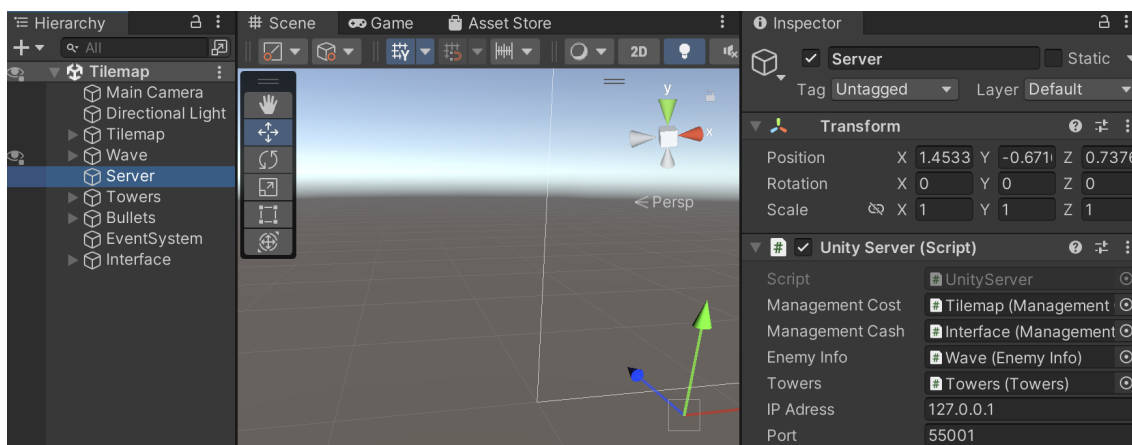
Rysunek 8 przedstawia miejsce w którym w edytorze Unity można zmienić początkową liczbę coins:

- Start Coins – początkowa liczba coins wież;
- Start Enemy Coins – początkowa liczba coins 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 $n \times m$ tiles mających podstawę kwadratu. n i m to odpowiednio szerokość i wysokość planszy wyrażona w długościach boku kwadratu będącego podstawą tile. 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, że dalej będzie definiowana plansza gry.-->

```

```

4 <Tilemap>
5 <!-- Oznaczenie tabeli. Tabela zawiera tyle wierszy i kolumn ile wierszy i
   kolumn posiada plansza. -->
6 <Table>
7   <Row>
8 <!-- Okreslenie rodzaju pojedynczego prostopadloscianu. Dopuszczalne typy
   to Ground i Water. Po prostopadloscianie typu Water moga poruszac sie
   przeciwnicy. -->
9     <Cell><Data>Ground</Data></Cell>
10    <Cell><Data>Ground</Data></Cell>
11    <Cell><Data>Ground</Data></Cell>
12    <Cell><Data>Ground</Data></Cell>
13    <Cell><Data>Ground</Data></Cell>
14    <Cell><Data>Ground</Data></Cell>
15    <Cell><Data>Ground</Data></Cell>
16    <Cell><Data>Ground</Data></Cell>
17    <Cell><Data>Ground</Data></Cell>
18    <Cell><Data>Ground</Data></Cell>
19    <Cell><Data>Ground</Data></Cell>
20    <Cell><Data>Ground</Data></Cell>
21    <Cell><Data>Ground</Data></Cell>
22  </Row>
23  <Row>
24 <!-- Okreslenie typu dla prostopadloscianow o specjalnych wlasnoscich.
   End oznacza koniec sciezki. Jest to miejsce, do ktorego maja dotrzec
   przeciwnicy. -->
25    <Cell><Data type="End">Water</Data></Cell>
26    <Cell><Data>Water</Data></Cell>
27    <Cell><Data>Water</Data></Cell>
28    <Cell><Data>Water</Data></Cell>
29    <Cell><Data>Water</Data></Cell>
30    <Cell><Data>Water</Data></Cell>
31    <Cell><Data>Water</Data></Cell>
32    <Cell><Data>Water</Data></Cell>
33    <Cell><Data>Water</Data></Cell>
34    <Cell><Data>Water</Data></Cell>
35    <Cell><Data>Water</Data></Cell>
36    <Cell><Data>Water</Data></Cell>
37 <!-- Okreslenie typu dla prostopadloscianow o specjalnych wlasnoscich.
   Begin oznacza poczatek sciezki. Jest to miejsce w ktorym pojawiaja sie
   przeciwnicy. -->
38    <Cell><Data type="Begin">Water</Data></Cell>
39  </Row>
40  <Row>
41    <Cell><Data>Ground</Data></Cell>
42    <Cell><Data>Ground</Data></Cell>
43    <Cell><Data>Ground</Data></Cell>
44    <Cell><Data>Ground</Data></Cell>
45    <Cell><Data>Ground</Data></Cell>

```



```

144 </Row>
145 <Row>
146   <Cell><Data>Ground</Data></Cell>
147   <Cell><Data>Water</Data></Cell>
148   <Cell><Data>Ground</Data></Cell>
149   <Cell><Data>Ground</Data></Cell>
150   <Cell><Data>Ground</Data></Cell>
151   <Cell><Data>Ground</Data></Cell>
152   <Cell><Data>Ground</Data></Cell>
153   <Cell><Data>Ground</Data></Cell>
154   <Cell><Data>Ground</Data></Cell>
155   <Cell><Data>Ground</Data></Cell>
156   <Cell><Data>Ground</Data></Cell>
157   <Cell><Data>Ground</Data></Cell>
158   <Cell><Data>Ground</Data></Cell>
159 </Row>
160 <Row>
161   <Cell><Data>Ground</Data></Cell>
162   <Cell><Data type="End">Water</Data></Cell>
163   <Cell><Data>Ground</Data></Cell>
164   <Cell><Data>Water</Data></Cell>
165   <Cell><Data>Water</Data></Cell>
166   <Cell><Data>Water</Data></Cell>
167   <Cell><Data>Ground</Data></Cell>
168   <Cell><Data>Water</Data></Cell>
169   <Cell><Data>Water</Data></Cell>
170   <Cell><Data>Water</Data></Cell>
171   <Cell><Data>Water</Data></Cell>
172   <Cell><Data>Ground</Data></Cell>
173   <Cell><Data>Ground</Data></Cell>
174 </Row>
175 <Row>
176   <Cell><Data>Ground</Data></Cell>
177   <Cell><Data>Ground</Data></Cell>
178   <Cell><Data>Ground</Data></Cell>
179   <Cell><Data>Water</Data></Cell>
180   <Cell><Data>Ground</Data></Cell>
181   <Cell><Data>Water</Data></Cell>
182   <Cell><Data>Water</Data></Cell>
183   <Cell><Data>Water</Data></Cell>
184   <Cell><Data>Ground</Data></Cell>
185   <Cell><Data>Ground</Data></Cell>
186   <Cell><Data>Water</Data></Cell>
187   <Cell><Data>Water</Data></Cell>
188   <Cell><Data type="Begin">Water</Data></Cell>
189 </Row>
190 <Row>
191   <Cell><Data>Ground</Data></Cell>
192   <Cell><Data>Ground</Data></Cell>

```



```

242     <Cell><Data>Ground</Data></Cell>
243     <Cell><Data>Ground</Data></Cell>
244     <Cell><Data>Ground</Data></Cell>
245     <Cell><Data>Ground</Data></Cell>
246     <Cell><Data>Ground</Data></Cell>
247     <Cell><Data>Ground</Data></Cell>
248     <Cell><Data>Ground</Data></Cell>
249 </Row>
250 </Table>
251 </Tilemap>
252 </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:

```

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

```

XML definiujący parametry przeciwników:

```

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 <!-- Komenda dla serwera. SetEnemies oznacza nakaz zdefiniowania typow
  przeciwnikow. -->
4 <Command name="SetEnemies">
5 <!-- Lista typow przeciwnikow. -->
6   <SetEnemies>
7 <!-- Definicja typu przeciwnika. no - numer typu, count - maksymalna
  liczba przeciwnikow (wartosc ujemna oznacza dowolna liczbe
  przeciwnikow), speed - szybkość przemieszczania się przeciwnikow,
  startHealth - ilość początkowego życia, armour - stopień odporności na
  strzały wież, cost - koszt stworzenia i wysłania przeciwnika,
  destroyCoins - kwota jaka dostają wieże za zniszczenie przeciwnika,
  coinsToEnd - kwota jaka dostają przeciwnicy za dotarcie przeciwnika do
  punktu końcowego, type - nazwa typu przeciwnika, tag - nazwa rodzaju
  obiektu. -->
8     <Enemy no="1" count="-1" speed="2" startHealth="20" armour="2" cost=
       "30" destroyCoins="30" coinsToEnd="40" type="Paper" tag="Enemy">
9     </Enemy>
10   </SetEnemies>
11 </Command>
12 </Data>

```

Po zdefiniowaniu parametrów zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).

XML definiujący parametry wież:

```

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 <!-- Komenda dla serwera. SetTowers oznacza nakaz zdefiniowania typow wiez
  . -->
4 <Command name="SetTowers">
5 <!-- Lista typow wiez. -->
6 <SetTowers>
7 <!-- Definicja typu wiezy. no - numer typu, count - maksymalna liczba wiez
  (wartosc ujemna oznacza dowolna liczbe wiez), speed - szybkoosc
  rotacji wiezy, rateofFire - szybkostrzelność 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. -->
8 <Tower no="0" count="-10" speed="1000" rateofFire="1" force="1000"
  bulletStrength="5" cost="10" type="Tower" tag="Enemy">
9 </Tower>
10 </SetTowers>
11 </Command>
12 </Data>

```

Po definiowaniu parametrów zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).

Utworzenie przeciwnika i wypuszczenie go wybraną ścieżką:

```

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 <!-- Komenda dla serwera. StartEnemy oznacza nakaz utworzenia przeciwnika
  i wypuszczenia go wybrana sciezka. -->
4 <Command name="StartEnemy">
5 <!-- Utworzenie przeciwnika. no - typ przeciwnika. -->
6 <StartEnemy no="1">
7 <!-- Okreslenie punktu startowego. no - numer punktu. -->
8   <Begin no="1">
9     </Begin>
10 <!-- Okreslenie punktu koncowego. no - numer punktu. -->
11   <End no="2">
12     </End>
13   </StartEnemy>
14 </Command>
15 </Data>

```

Po utworzeniu przeciwnika zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).

XML powodujący dodanie nowej wieży:

```

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 <!-- Komenda dla serwera. AddTower oznacza nakaz utworzenia wiezy. -->

```

```

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

```

Po dodaniu nowej wieży zwracana jest informacja o poprawności wykonania komendy (analogiczna jak w przypadku tilemap).

Żądanie przesłania informacji o ścieżkach:

```

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 <!-- Komenda dla serwera. GetChoiceOfPathData oznacza nakaz przeslania
   informacji o sciezках. -->
4 <Command name="GetChoiceOfPathData">
5 </Command>
6 </Data>

```

Zwracana przez serwer informacja:

```

1 <?xml version="1.0"?>
2 <Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
   .w3.org/2001/XMLSchema-instance" title="ChoiceOfPathData">
3 <!-- Lista sciezek. beginTileCount - liczba punktow poczatkowych,
   endTileCount - liczba punktow koncowych. -->
4 <ChoiceOfPath beginTileCount="4" endTileCount="4">
5 <!-- Informacje o sciezce. cost - koszt przejścia 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
   . -->
6 <Path cost="12" shotAtTiles="3" towers="1" sumTowerPlace="26" hits="0"
   nohits="0">
7 <!-- Początek sciezki. x, y - wspolrzedne poczatku sciezki, no - numer
   punktu poczatkowego. -->
8 <Begin x="1" y="12" no="0">
9 <!-- Początek sciezki. x, y - wspolrzedne poczatku sciezki, no - numer
   punktu poczatkowego. -->
10 <!-- Informacja o przeciwnikach, ktorzy weszli sciezke. type - nazwa typu
   przeciwnika, enemies - liczba przeciwnikow, ktorzy weszli na sciezke,
   endMeanHealth - sredni poziom zycia przeciwnikow, ktorzy wchodzi na
   sciezke. -->
11 <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
12 <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
13 </Begin>
14 <!-- 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. -->

```

```

58     <Towers cash="90" />
59   </ChoiceOfPath>
60 </Answer>

```

Żądanie przesłania szczegółowych informacji o stanie 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 <!-- Komenda dla serwera. LevelData oznacza nakaz przesłania szczegolowych
  informacji o stanie gry. -->
4 <Command name="LevelData">
5 </Command>
6 </Data>

```

Zwracana przez serwer informacja:

```

1 <?xml version="1.0"?>
2 <Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
  .w3.org/2001/XMLSchema-instance" title="LevelData">
3 <!-- Lista sciezek oraz zbior informacji o stanie gry. beginTileCount -
  liczba punktow poczatkowych, endTileCount - liczba punktow koncowych.
  -->
4 <LevelPath beginTileCount="4" endTileCount="4">
5   <Path cost="12" shotAtTiles="3" towers="1" sumTowerPlace="26" hits="0"
     nohits="0">
6     <Begin x="1" y="12" no="0">
7       <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
8       <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
9     </Begin>
10    <End x="1" y="0" no="0">
11      <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
12      <Enemy type="Paper" enemies="0" endMeanHealth="0" />
13    </End>
14 <!-- Tabela ze wspolrzednymi 3D kafelek terenu. -->
15 <Table>
16 <!-- Wspolrzedne x,y,z kafelek terenu. -->
17   <Element x="1" y="0" z="12" />
18   <Element x="1" y="0" z="11" />
19   <Element x="1" y="0" z="10" />
20   <Element x="1" y="0" z="9" />
21   <Element x="1" y="0" z="8" />
22   <Element x="1" y="0" z="7" />
23   <Element x="1" y="0" z="6" />
24   <Element x="1" y="0" z="5" />
25   <Element x="1" y="0" z="4" />
26   <Element x="1" y="0" z="3" />
27   <Element x="1" y="0" z="2" />
28   <Element x="1" y="0" z="1" />
29   <Element x="1" y="0" z="0" />
30 </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 - szybkość przemieszczania się przeciwnikow, startHealth - ilość
    początkowego życia, armour - stopień odporności na strzały wież,
    destroyCoins - kwota jaka dostęja wieze za zniszczenie przeciwnika,
    cost - koszt stworzenia i wysłania przeciwnika, coinsToEnd - kwota
    jaka dostają przeciwnicy za dotarcie przeciwnika do punktu końcowego,
    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 - szybkoosc obrotu wiezy,
    rateofFire - szybkostrzelność 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 Biblioteka

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

SendData

Sending data to the server.

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

Description:

- IPAddressSend – server ip address,
- portSend – server port,
- data – data packet sent to the server,
- name – control information sent to the server,
- args – arguments related to control information.

Returns the response from the server in xml format.

NumberToName

Zamienia tablicę dwuwymiarową w tablicę struktur. Replacing numbers representing field types with their names.

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

Description:

- array – a table containing information about the map,
- names – map field names.

Returns a table containing information about the map.

ChangeBeginEnd

Marking the beginnings and ends of paths.

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

Description:

- array – a table containing information about the map.

Returns a table containing information about the map.

TilemapToXML

Map conversion from an array to xml format.

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

Description:

- tilemap – map in the form of an array.

Returns a map saved in xml format.

Przykład przesłania do gry polecenia utworzenia planszy:

```
1 %Adres serwera
2 IPAddressSend = '127.0.0.1';
3 %Port na którym nasluchuje serwer
4 portSend = 55001;
5 %Tablica reprezentujaca plansze gry w ktorej liczby
  reprezentuja typy tiles (1 - ziemia, 2 - woda bedaca
  elementem sciezki ruchu przeciwnikow, 3 - poczatek sciezki,
  4 - koniec sciezki)
6 tilemap = [1 3 1 3 1 1 1 1 1 1 1 3 1 1 1 1;
7           1 2 1 2 1 1 1 1 1 1 1 2 1 1 1 1;
```

```

8           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 tiles. 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 tiles zamiast
    numerow
27 tilemapNames = NumberToName(tilemap,names);
28 %Zamiana nazw tiles Begin i End na Water. Przypisanie tym tiles
    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',[]);

```

ParseXML

Parses text containing xml.

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

Description:

- data – a text array containing data in xml format.

Returns an array of structures whose structure reflects the structure of the XML data, the field names are the names of the XML elements.

Przykład odczytania i zdekodowania pliku xml:

```
1 %Odczytanie pliku xml
2 dataTower = fileread('towers.xml');
3 %Konwersja pliku xml
4 dataTower = ParseXML(dataTower);
```

Zawartość pliku towers.xml:

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

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

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

GetVectorFromCell

Reading a data vector from a selected field of the structure array.

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

Description:

- data – structure array,
- field – read structure fields.

Returns a data vector.

Przykład odczytania współrzędnych x wież jako tablicy:

```
1 %Odczytanie pliku xml
2 dataTower = fileread('towers.xml');
3 %Konwersja pliku xml
4 dataTower = ParseXML(dataTower);
5 %Odczytanie współrzędnych x wież jako tablicy
6 x = GetVectorFromCell(dataTower.Answer.TowerCoordinates{1}.
  Element, 'x');
```

Zawartość pliku towers.xml:

```

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

```

SetEnemies

Creating information in the form of XML about a specific type of opponent.

```

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

```

Description:

- count – maximum number of opponents,
- speed – opponent's speed,
- startHealth – opponent's starting life value,
- armour – enemy's armor (bullet resistance),
- cost – the cost of creating and sending an enemy,
- destroyCoins – profit for the tower manager for shooting down an enemy,
- coinsToEnd – gain for the opponent's manager if he reaches the end of the path,
- type – opponent type,
- tag – name of the object type.

Returns information saved in xml format.

Przykład przesłania do gry polecenia utworzenia nowego typu przeciwnika:

```

1 %Adres serwera
2 IPAddressSend = '127.0.0.1';
3 %Port na którym nasluchuje serwer
4 portSend = 55001;
5 %Utworzenie danych w formacie xml dotyczacych nowego typu
  przeciwnika

```

```

6 txt = SetEnemies(-1,2,20,2,30,30,40, 'Paper', 'Enemy');
7 %Wyslanie polecenia (Command) utworzenia nowego typu
  przeciwnika (SetEnemies)
8 SendData(IPAddressSend,portSend,txt, 'Command', 'name="SetEnemies
  "')';

```

SetTowers

Creating information in the form of XML about a specific type of tower.

```

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

```

Description:

- count – maximum number of towers,
- speed – the rotation speed of the towers,
- rateOfFire – rate of fire towers,
- force – turret firing power (determines range),
- bulletStrength – turret projectile strength (affects the number of wounds dealt to the enemy),
- cost – cost of creating a tower,
- type – tower type,
- tag – the type of object that the tower will attack.

Returns information saved in xml format.

Przykład przesłania do gry polecenia utworzenia nowego typu wieży:

```

1 %Adres serwera
2 IPAddressSend = '127.0.0.1';
3 %Port na którym nasluchuje serwer
4 portSend = 55001;
5 %Utworzenie danych w formacie xml dotyczacych nowego typu wiezy
6 txt = SetTowers(-10,1000,1,1000,5,10, 'Tower', 'Enemy');
7 %Wyslanie polecenia (Command) utworzenia nowego typu wiezy (
  SetTowers)
8 SendData(IPAddressSend,portSend,txt, 'Command', 'name="SetTowers"
  ');

```


StartEnemy

Creating an opponent and sending him out along a selected path.

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

Description:

- beginNo – starting point number,
- endNo – endpoint number.

Returns information saved in xml format.

Przykład utworzenia przeciwnika i wysłania go z punktu startowego 1 do punktu końcowego 3:

```
1 %Adres serwera
2 IPAddressSend = '127.0.0.1';
3 %Port na którym nasluchuje serwer
4 portSend = 55001;
5 %Utworzenie danych w formacie xml zawierajacych informacje o
  punkcie startowym i koncowym
6 txt = StartEnemy(1,3);
7 %Wyslanie polecenia (Command) utworzenia przeciwnika i wyslania
  go od wskazanego punktu startowego do wskazanego punktu
  koncowego (StartEnemy)
8 errorStartEnemy = SendData(IPAddressSend,portSend,txt,'Command'
  , 'name="StartEnemy"');
```

AddTower

Adding a tower.

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

Description:

- noTower – tower number,
- x – x coordinate of the tower,
- y – y coordinate of the tower.

Returns information saved in xml format.

Przykład dodania wieży o numerze 3 w miejsce o współrzędnych $x = 1$, $y = 4$:

```

1 %Adres serwera
2 IPAddressSend = '127.0.0.1';
3 %Port na którym nasluchuje serwer
4 portSend = 55001;
5 %Utworzenie danych w formacie xml zawierajacych informacje o
   numerze wiezy i jej wspolrzednych
6 txt = AddTower(3,1,4);
7 %Wyslanie polecenia (Command) utworzenia wiezy (AddTower)
8 errorAddTower = SendData(IPAddressSend,portSend,txt,'Command','
   name="AddTower"');

```

GenerateTabular

Generation of a tabular table.

```

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

```

Description:

- fileName – name of the file to which the array will be saved,
- data – saved array,
- columnDescriptions – column descriptions,
- rowDescriptions – row descriptions, empty array([]) means no descriptions,
- rowsBold – 0 means line descriptions are bold and 1 means bold,
- decimalPlaces – number of decimal places.

Przykład generowania tablicy:

```

1 %Tablica
2 exampleArray = [1 2;
3                 3 1;
4                 5 2;
5                 2 4];
6 %Opisy kolumn
7 columnDescriptions={'No','Data 1','Data 2'};
8 %Opisy wierszy
9 rowDescriptions={'1','2','3','4'};
10 %Utworzenie pliku zawierajacego srodowisko tablular
11 GenerateTabular('array.tex',exampleArray,columnDescriptions,
   rowDescriptions,0,0);

```

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

Tablica 1: Wygenerowana tablica

Zawartość pliku array.tex:

```

1 \begin{tabular}{|r|r|r|}
2   \hline
3   \textbf{No}& \textbf{Data 1}& \textbf{Data 2}\\
4   \hline
5   1& 1& 2\\
6   \hline
7   2& 3& 1\\
8   \hline
9   3& 5& 2\\
10  \hline
11  4& 2& 4\\
12  \hline
13 \end{tabular}

```

Tablicę można dołączyć do pliku Latex-a:

```

1 \begin{table}
2   \input{array}
3   \caption{Wygenerowana tablica}
4 \end{table}

```

Uzyskany efekt przedstawia tablica 1.

GenerateTikzData

Generating data files for tikz charts.

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

Description:

- fileName – name of the file to which the array will be saved,
- data – saved array,
- columnDescriptions – column descriptions.

Przykład generowania danych:

```

1 %Tablica
2 exampleArray = [1 2;
3                 3 1;
4                 5 2;
5                 2 4];
6 %Opisy kolumn
7 columnDescriptions={'No','D1','D2'};
8 %Utworzenie pliku zawierajacego dane dla wykresow tikz
9 GenerateTikzData('array.dat',[1:size(exampleArray,1)] '
    exampleArray],columnDescriptions);

```

Zawartość pliku array.dat:

```

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

```

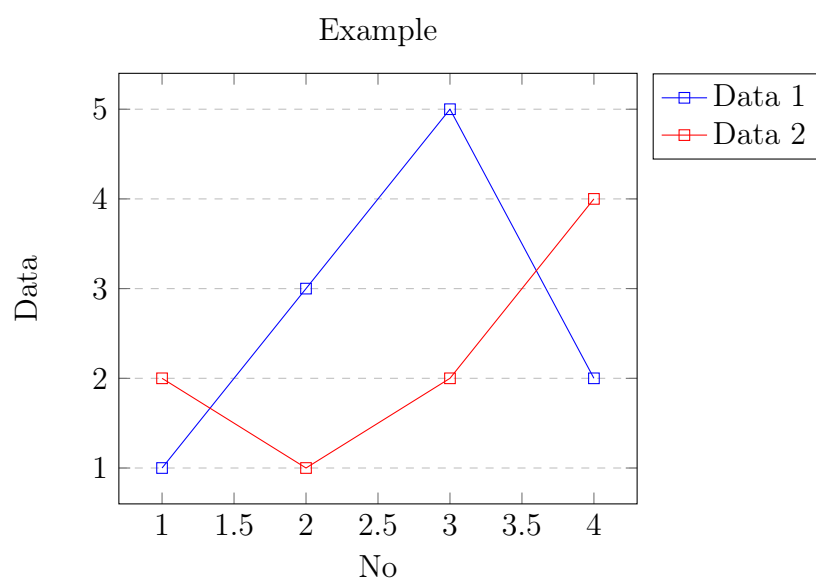
Plik array.dat można dołączyć do wykresu tikz-a:

```

1 \begin{tikzpicture}
2 \begin{axis}[
3     title={Example},
4     xlabel={No},
5     ylabel={Data},
6     legend pos=outer north east,
7     ymajorgrids=true,
8     grid style=dashed,
9 ]
10
11 \addplot[
12     color=blue,
13     mark=square
14 ]
15     table[x=No,y=D1]
16     {fig/array.dat};
17 \addplot[
18     color=red,
19     mark=square
20 ]
21     table[x=No,y=D2]
22     {fig/array.dat};
23
24 \legend{Data 1, Data 2}
25
26 \end{axis}
27 \end{tikzpicture}

```

Uzyskany efekt przedstawia rysunek 10.



Rysunek 10: Wykres na podstawie danych wygenerowanych przez funkcję GenerateTikzData