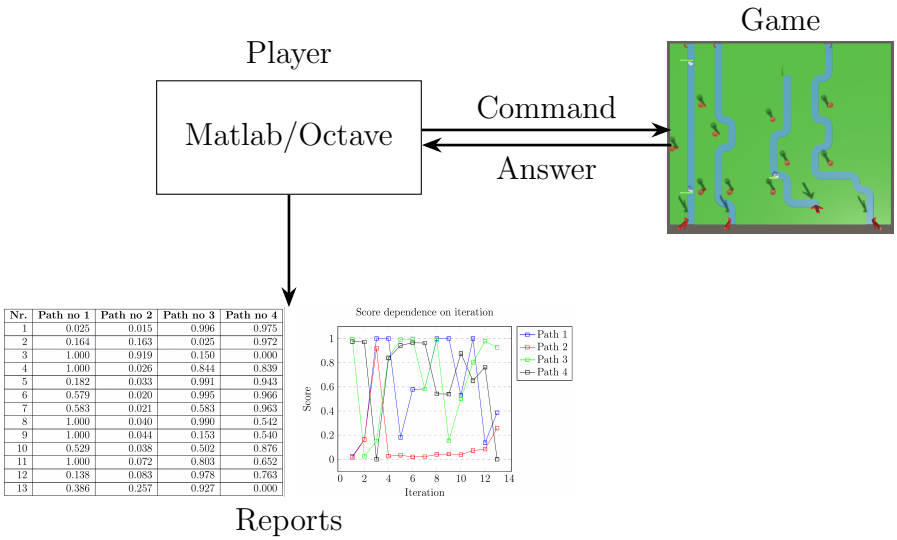
Contents

1 Introduction

The MCDA MEASURE (Multi-Criteria Decision Analysis MEthods Assessment through SimUlation REsearch) software includes a game module using the Unity game engine and a library created in the Matlab/Octave environment. The game module can be controlled using commands in which data is saved in XML format (fig. 1). Responsible for visualizing the game board and controlling its course. By default, the game is configured to listen on port 55001. The default address (127.0.0.1) allows you to connect to the game only from the computer on which it is installed (however, this can be changed).
The library is intended to facilitate the creation of software in Matlab/Octave intended for testing decision support methods. Contains methods for preparing and formatting commands to be sent to the game. It also allows you to decode the answers received from the game. An additional option of the library is the conversion of tables to the tabular Latex environment format and a format that allows them to be read by the tikz package's charting module.

2 Game module

The game is made in the Unity environment. It belongs to the tower defense type. The game board consists of tiles, which are three-dimensional models divided into two categories: ground and road. There is one ground tile and five road tiles (fig. 2).
The figure 3 shows where in the Unity editor interface you should change the tile models (if necessary). The top selection shows where the earth tile model changes. The lower selection shows where the first road tile model was changed. The rest are in WaterStraight, WaterCurve, etc. The WaterBegin, WaterBeginEnd, etc. elements are combinations of the models in Figure 2 with arrows.



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

Figure 1: System architecture

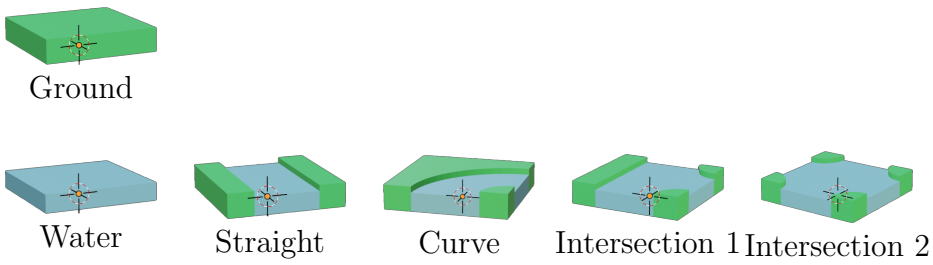


Figure 2: Tile models

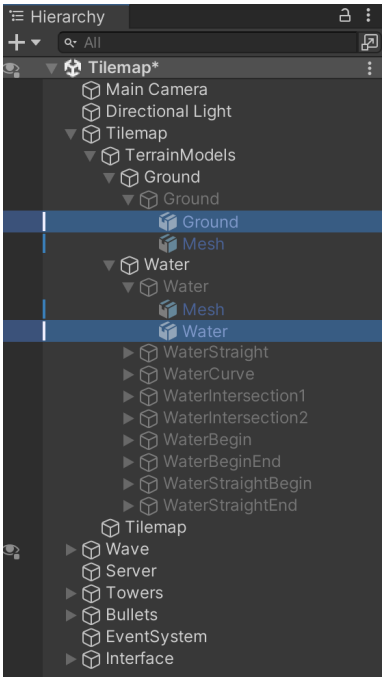


Figure 3: Changing tile models

Similarly, you can change the models of opponents and towers. The location of model changes is shown in Figures 4 and 5.
Figure 6 shows the place where you can change enemy parameters in the Unity editor:

- Speed – movement speed,
- Start health – start health,
- Armor – value subtracted from the damage dealt (causes invulnerability to bullets that have a lower amount of damage than Armour),
- Coins – number of coins needed to create an opponent,
- Destroy Coins – the number of coins that towers receive for killing an opponent,
- Coins To End – the number of coins that opponents receive if the opponent reaches the end point,
- Count – number of available opponents (-1 means unlimited number of opponents).

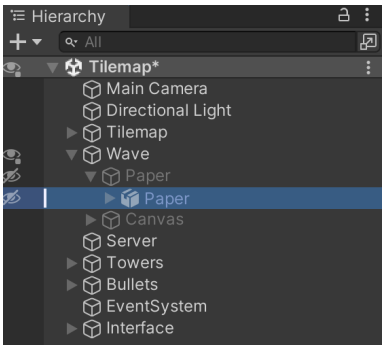


Figure 4: Changing the opponent's model

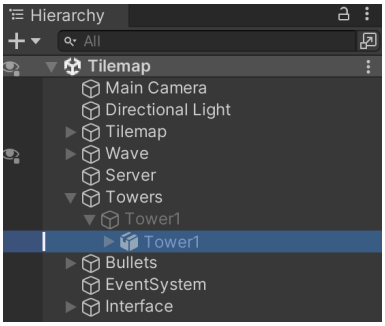


Figure 5: Changing the tower model

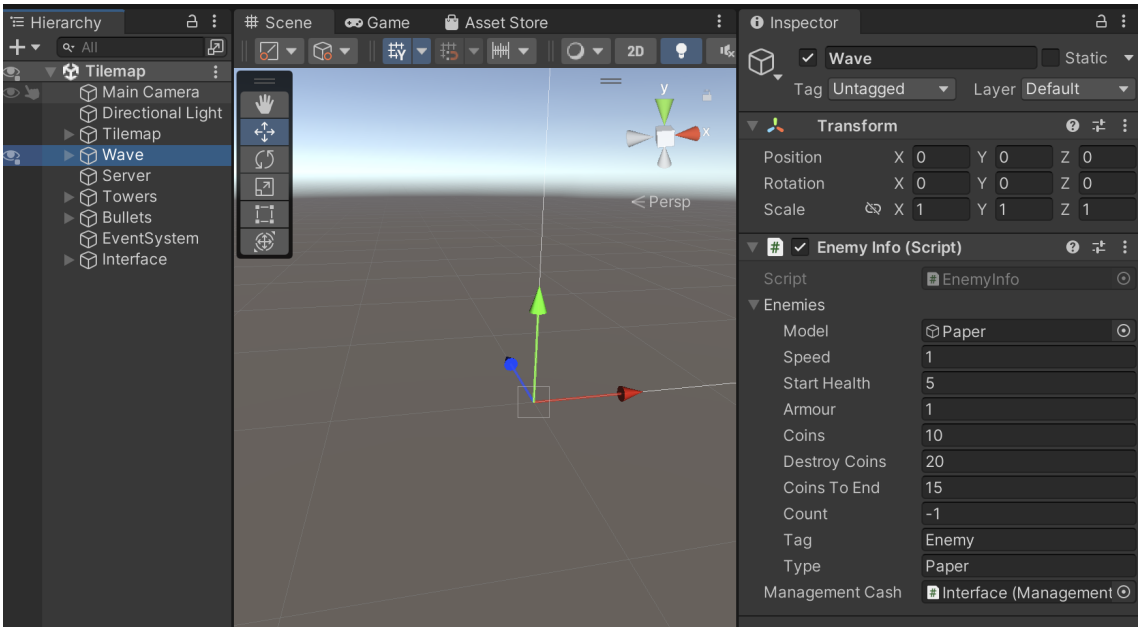


Figure 6: Changing opponents' parameters

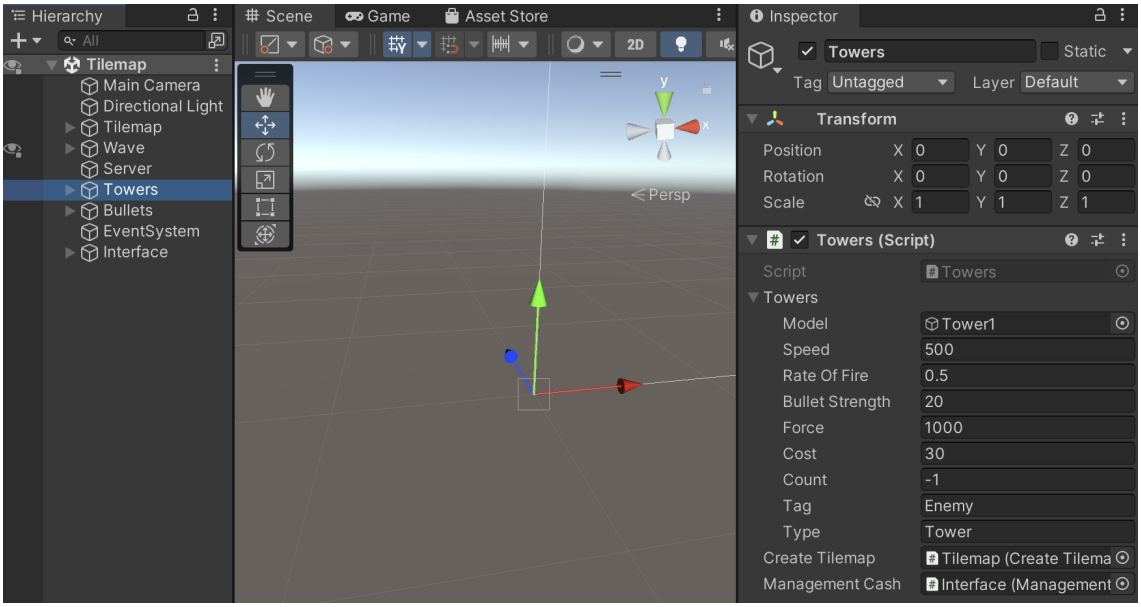


Figure 7: Changing tower parameters

Figure 7 shows the place where you can change tower parameters in the Unity editor:

- Speed – rotation speed,
- Rate From Fire – rate of fire,
- Bullet Strength – number of damage dealt,
- Force – the force of firing the projectile that translates into its range,
- Coins – number of coins needed to create a tower,
- Count – number of available towers (-1 means unlimited number of towers).

Figure 8 shows where you can change the starting number of coins in the Unity editor:

- Start Coins – initial number of tower coins;
- Start Enemy Coins – the initial number of enemy coins.

Figure 9 shows the place where you can change server parameters in the Unity editor:

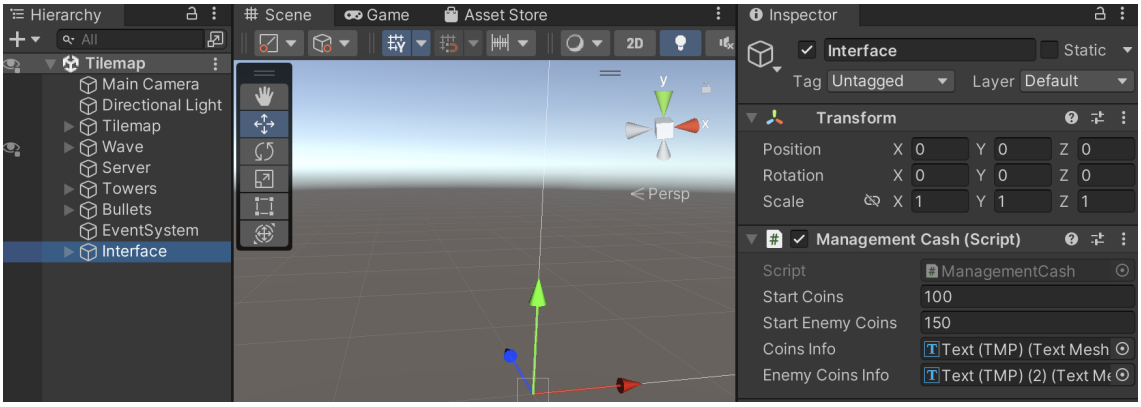


Figure 8: Starting coins

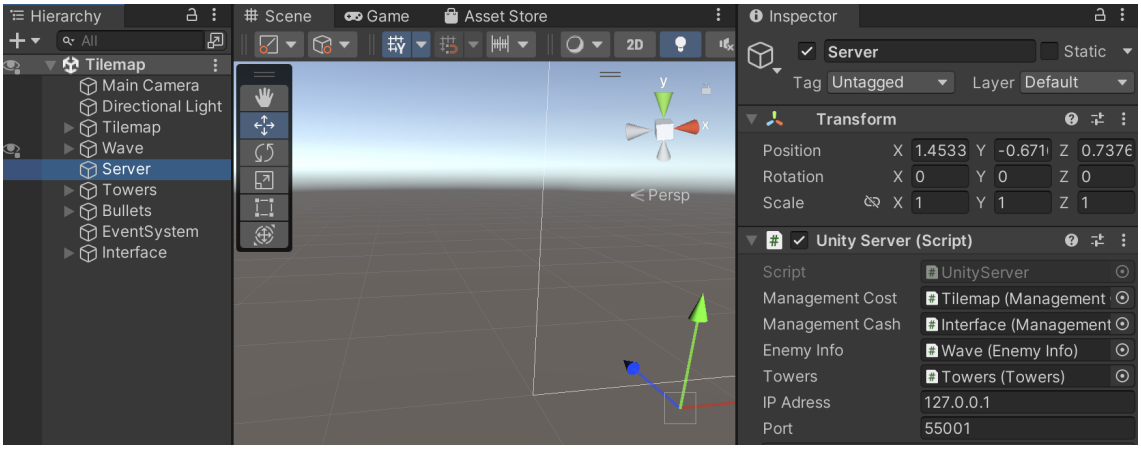


Figure 9: Communication server parameters

- IP Address – server IP address (127.0.0.1 means that the server will only be available to software installed on the same computer as the game);
- Port – port on which the server is listening.

The game board is a set of $n \times m$ tiles with a square base. n and m are the width and height of the board, respectively, expressed in the lengths of the side of the square that is the base of the tile. Sample XML defining the game board:

```
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 <!-- Information that the game board will be further defined. -->
```

```
<Tilemap>
<!-- Table designation. The table contains as many rows and columns as the
     board has rows and columns. -->
  <Table>
    <Row>
<!-- Specify the type of a single tile. The allowed types are Ground and
    Water. Enemies can move on the Water-type tile. -->
      <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>
<!-- Type definition for tile with special properties. End means the end
    of the track. This is the place where the opponents must reach. -->
      <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>
<!-- Type definition for tile with special properties. Begin means the
    beginning of the path. This is the place where enemies appear. -->
      <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>
      <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>
    <Row>
      <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>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>
    </Row>
    <Row>
      <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>Water</Data></Cell>
      <Cell><Data>Water</Data></Cell>
      <Cell><Data type="Begin">Water</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>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>
      <Cell><Data>Ground</Data></Cell>
    </Row>
  </Table>
</Tilemap>
</Data>
```

After sending the data, information is returned in the form of XML. In the case of game board definition, XML data is returned with information about the correctness (or not) of the information sent to the server:

```
<?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 defining opponent parameters:

```
<?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">
<!-- Command for the server. SetEnemies means the requirement to define
    enemy types. -->
<Command name="SetEnemies">
<!-- List of enemy types. -->
  <SetEnemies>
<!-- Enemy type definition. no - type number, count - maximum number of
    opponents (a negative value means any number of opponents), speed -
    speed of movement of opponents, startHealth - amount of initial life,
    armor - degree of resistance to arrows of towers, cost - cost of
    creating and sending an opponent, destroyCoins - the amount the tower
    gets for destroying the opponent, coinsToEnd - the amount the
    opponents get for reaching the end point, type - the name of the
    opponent's type, tag - the name of the object type. -->
      <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>
```

After defining the parameters, information is returned about the correctness of the command execution (same as in the case of tilemap).
XML defining tower parameters:

```
<?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">
<!-- Command for the server. SetTowers means to define the types of towers
    . -->
```

```
<Command name="SetTowers">
<!-- List of tower types. -->
<SetTowers>
<!-- Tower type definition. no - type number, count - maximum number of
    towers (a negative value means any number of towers), speed - the
    speed of rotation of the tower, rateofFire - the rate of fire of the
    tower, force - the force with which the bullet is thrown,
    bulletStrength - the number of wounds inflicted, cost - the cost of
    placing the tower , type - name of the tower type, tag - type of
    object that the tower will attack. -->
<Tower no="0" count="-10" speed="1000" rateofFire="1" force="1000"
    bulletStrength="5" cost="10" type="Tower" tag="Enemy">
</Tower>
    </SetTowers>
</Command>
</Data>
```

After defining the parameters, information is returned about the correctness of the command execution (same as in the case of tilemap).
Creating an opponent and releasing him along the selected path:

```
<?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">
<!-- Command for the server. StartEnemy means the order to create an enemy
     and send him out on a selected path. -->
<Command name="StartEnemy">
<!-- Create an opponent. no - type of opponent. -->
<StartEnemy no="1">
<!-- Specify the starting point. no - point number. -->
      <Begin no="1">
      </Begin>
<!-- Specify an end point. no - point number. -->
      <End no="2">
      </End>
    </StartEnemy>
</Command>
</Data>
```

After creating an opponent, information is returned about the correctness of the command execution (same as in the case of tilemap).
XML to add a new tower:

```
<?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">
<!-- Command for the server. AddTower means an order to create a tower. --
    >
<Command name="AddTower">
```

```
<!-- Adding a tower. no - tower type number, x,y - tower location
    coordinates. -->
  <AddTower no="0" x="2" y="8">
  </AddTower>
</Command>
</Data>
```

After adding a new tower, information is returned about the correctness of the command execution (same as in the case of tilemap).
Request to send track information:

```
<?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">
<!-- Command for the server. GetChoiceOfPathData means an order to send
    path information. -->
<Command name="GetChoiceOfPathData">
</Command>
</Data>
```

Information returned by the server:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="ChoiceOfPathData">
<!-- Track list. beginTileCount - number of starting points, endTileCount
    - number of end points. -->
  <ChoiceOfPath beginTileCount="4" endTileCount="4">
<!-- Path information. cost - the cost of crossing the path, shotAtTiles -
     the number of path tiles shot at, towers - the number of towers along
     the path, sumTowerPlace - the number of free places to place towers (
    so that they fire at the path), hits - the number of bullets that hit
    the opponent, nohits - the number of bullets that they didn't hit the
    opponent. -->
    <Path cost="12" shotAtTiles="3" towers="1" sumTowerPlace="26" hits="0"
        nohits="0">
<!-- Beginning of the path. x, y - coordinates of the beginning of the
    path, no - number of the starting point. -->
      <Begin x="1" y="12" no="0">
<!-- Information about enemies that have entered the path. type - name of
    the enemy type, enemies - number of enemies who entered the path,
    endMeanHealth - average health level of enemies who entered the path.
    -->
        <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
        <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
      </Begin>
<!-- End of track. x, y - coordinates of the beginning of the path, no -
    number of the end point. -->
      <End x="1" y="0" no="0">
<!-- Information about enemies that have passed the path. type - name of
    the enemy type, enemies - number of enemies who reached the end of the
```

```
      path, endMeanHealth - average health level of enemies who reached the
      end of the path. -->
16        <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
17        <Enemy type="Paper" enemies="0" endMeanHealth="0" />
18       </End>
19       <Table />
20     </Path>
21     <Path cost="15" shotAtTiles="3" towers="1" sumTowerPlace="31" hits="0"
           nohits="0">
22       <Begin x="3" y="12" no="1">
23         <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
24         <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
25       </Begin>
26       <End x="4" y="0" no="1">
27         <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
28         <Enemy type="Paper" enemies="0" endMeanHealth="0" />
29       </End>
30       <Table />
31     </Path>
32     <Path cost="13" shotAtTiles="0" towers="0" sumTowerPlace="34" hits="0"
           nohits="0">
33       <Begin x="8" y="10" no="2">
34         <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
35         <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
36       </Begin>
37       <End x="10" y="1" no="2">
38         <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
39         <Enemy type="Paper" enemies="0" endMeanHealth="0" />
40       </End>
41       <Table />
42     </Path>
43     <Path cost="19" shotAtTiles="0" towers="0" sumTowerPlace="38" hits="0"
           nohits="0">
44       <Begin x="11" y="12" no="3">
45         <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
46         <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
47       </Begin>
48       <End x="14" y="0" no="3">
49         <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
50         <Enemy type="Paper" enemies="0" endMeanHealth="0" />
51       </End>
52       <Table />
53     </Path>
54 <!-- Resources available for opponents. -->
55     <Waves cash="150" />
56 <!-- Available funds for the tower manager. -->
57     <Towers cash="90" />
58   </ChoiceOfPath>
59 </Answer>
```

Request for detailed game status information:

```
<?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">
<!-- Command for the server. LevelData means an order to send detailed
    information about the state of the game. -->
<Command name="LevelData">
</Command>
</Data>
```

Information returned by the server:

```
<?xml version="1.0"?>
<Answer xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www
    .w3.org/2001/XMLSchema-instance" title="LevelData">
<!-- List of paths and a collection of information about the game state.
    beginTileCount - number of starting points, endTileCount - number of
    ending points. -->
  <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>
<!-- Table with 3D coordinates of terrain tiles. -->
      <Table>
<!-- The x,y,z coordinates of the terrain tiles. -->
        <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>
    </Path>
    <Path cost="15" shotAtTiles="3" towers="1" sumTowerPlace="31" hits="0"
        nohits="0">
      <Begin x="3" y="12" no="1">
```

```
      <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
      <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
    </Begin>
    <End x="4" y="0" no="1">
      <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
      <Enemy type="Paper" enemies="0" endMeanHealth="0" />
    </End>
    <Table>
      <Element x="3" y="0" z="12" />
      <Element x="3" y="0" z="11" />
      <Element x="3" y="0" z="10" />
      <Element x="3" y="0" z="9" />
      <Element x="3" y="0" z="8" />
      <Element x="3" y="0" z="7" />
      <Element x="4" y="0" z="7" />
      <Element x="4" y="0" z="6" />
      <Element x="4" y="0" z="5" />
      <Element x="3" y="0" z="5" />
      <Element x="3" y="0" z="4" />
      <Element x="3" y="0" z="3" />
      <Element x="3" y="0" z="2" />
      <Element x="3" y="0" z="1" />
      <Element x="4" y="0" z="1" />
      <Element x="4" y="0" z="0" />
    </Table>
  </Path>
  <Path cost="13" shotAtTiles="0" towers="0" sumTowerPlace="34" hits="0"
      nohits="0">
    <Begin x="8" y="10" no="2">
      <Enemy type="Bottle" enemies="0" endMeanHealth="NaN" />
      <Enemy type="Paper" enemies="0" endMeanHealth="NaN" />
    </Begin>
    <End x="10" y="1" no="2">
      <Enemy type="Bottle" enemies="0" endMeanHealth="0" />
      <Enemy type="Paper" enemies="0" endMeanHealth="0" />
    </End>
    <Table>
      <Element x="8" y="0" z="10" />
      <Element x="8" y="0" z="9" />
      <Element x="8" y="0" z="8" />
      <Element x="8" y="0" z="7" />
      <Element x="8" y="0" z="6" />
      <Element x="7" y="0" z="6" />
      <Element x="7" y="0" z="5" />
      <Element x="7" y="0" z="4" />
      <Element x="7" y="0" z="3" />
      <Element x="8" y="0" z="3" />
      <Element x="8" y="0" z="2" />
      <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 <!-- Information about the opponent's type. count - the maximum number of
      opponents (a negative value means any number of opponents), speed -
      the speed of movement of opponents, startHealth - the amount of
      initial life, armor - the degree of resistance to arrows of the tower,
       destroyCoins - the amount the tower receives for destroying an
      opponent, cost - the cost of creating and sending the opponent,
      coinsToEnd - the amount that the opponents receive for reaching the
      end point, no - the opponent's type number, type - the name of the
      opponent's type, tag - the name of the object type. -->
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" />
<!-- Information about the tower type. count - the maximum number of
    towers (a negative value means any number of towers), speed - the
    speed of rotation of the tower, rateofFire - the rate of fire of the
    tower, force - the force with which the bullet is thrown,
    bulletStrength - the number of wounds caused by the bullet, cost - the
     cost of erecting the tower, no - type number, type - name of the
    tower type, tag - type of object that will be attacked by the tower.
    -->
    <Tower count="-10" speed="1000" rateofFire="1" force="1000"
        bulletStrength="5" cost="10" no="0" type="Tower" tag="Enemy" />
    <Waves cash="150" />
    <Towers cash="90" />
  </LevelPath>
</Answer>
```

3 Player module

The library is a set of functions supporting communication with the game and creating tables and charts. These functions work in both Matlab and Octave environments.

SendData
Sending data to the server.

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

Description:

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

Returns the response from the server in xml format.

NumberToName
Replacing numbers representing field types with their names.

```
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.

```
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.

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

Description:

- tilemap – map in the form of an array.

Returns a map saved in xml format.
Example of sending a board creation command to the game:

```
%Server address
IPAddressSend = '127.0.0.1';
%Port on which the server is listening
portSend = 55001;
%Array representing the game board in which the numbers
    represent the types of tiles (1 - earth, 2 - water, which
    is an element of the opponents' path, 3 - the beginning of
    the path, 4 - the end of the path)
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;
           1 2 1 2 1 1 1 1 2 1 2 1 1 1 1 1;
           1 2 1 2 1 1 1 1 2 1 2 1 1 1 1 1;
           1 2 1 2 2 1 1 1 2 1 2 2 1 1 1 1;
           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 %Tile type names. The position in the array corresponds to the
      number in the tilemap array
20 names{1} = 'Ground';
21 names{2} = 'Water';
22 names{3} = 'Begin';
23 names{4} = 'End';
24 %Rotate the board so that the orientation of the board matches
      the orientation of the game board
25 tilemap = rot90(rot90(rot90(tilemap)));
26 % Replace array with arrays of structures with tile names
      instead of numbers
27 tilemapNames = NumberToName(tilemap,names);
28 %Renaming tiles Begin and End to Water. Assign these tiles to
      mark the beginning or end of the path.
29 tilemapNames = ChangeBeginEnd(tilemapNames);
30 %Converting the structure table to text in xml format.
31 txt = TilemapToXML(tilemapNames);
32 %Sending a command to create a new board (Tilemap) and data in
      xml format for the game.
33 SendData(IPAddressSend,portSend,txt,'Tilemap',[]);
```

Char2Code

Replacing character codes with codes used by the machine.

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

Description:

- code – character code.

Returns machine codes.
Example of reading and decoding an xml file:

```
1 %Character conversion
2 code = Char2Code('a');
```

Machine
Machine that divides text into elements and assigns them codes for the final states of the machine.

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

Description:

- data – text,
- t – table of transitions between the states of the machine.

Returns an array of structures containing a text fragment (txt field) and the state assigned to it (state field).
Example of reading and decoding an xml file:

```
%Reading xml file
dataTower = fileread('towers.xml');
%State table definition
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;
%File analysis
result = Machine(dataTower,t);
```

The result variable contains an array of structures containing the text and the state number assigned to it.

ParseXML
Parses text containing xml.

```
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.
Example of reading and decoding an xml file:

```
%Reading xml file
dataTower = fileread('towers.xml');
%xml file conversion
dataTower = ParseXML(dataTower);
```

Contents of the towers.xml file:

```
<?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>
```

The towers.xml file contains the coordinates of the towers and their ordinal numbers. Example of access to this data:

```
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

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

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

Description:

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

Returns a data vector.
Example of reading the x-coordinates of towers as an array:

```
%Reading xml file
dataTower = fileread('towers.xml');
%Xml file conversion
dataTower = ParseXML(dataTower);
%Reading tower x coordinates as an array
x = GetVectorFromCell(dataTower.Answer.TowerCoordinates{1}.
    Element,'x');
```

Contents of the towers.xml file:

```
<?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

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

```
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.
Example of sending a command to the game to create a new type of opponent:

```
%Server address
IPAddressSend = '127.0.0.1';
%Port on which the server is listening
portSend = 55001;
%Creating data in xml format regarding the new enemy type
txt = SetEnemies(-1,2,20,2,30,30,40,'Paper','Enemy');
%Sending a command (Command) to create a new type of enemy (
    SetEnemies)
SendData(IPAddressSend,portSend,txt,'Command','name="SetEnemies
    "');
```

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

```
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.
Example of sending a command to the game to create a new type of tower:

```
%Server address
IPAddressSend = '127.0.0.1';
%Port on which the server is listening
portSend = 55001;
```

```
%Creation of data in xml format regarding the new tower type
txt = SetTowers(-10,1000,1,1000,5,10,'Tower','Enemy');
%Sending a command (Command) to create a new type of tower (
    SetTowers)
SendData(IPAddressSend,portSend,txt,'Command','name="SetTowers"
    ');
```

StartEnemy

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

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

Description:

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

Returns information saved in xml format.
Example of creating an opponent and sending it from start point 1 to end point 3:

```
%Server address
IPAddressSend = '127.0.0.1';
%Port on which the server is listening
portSend = 55001;
%Create data in xml format containing information about the
    start and end point
txt = StartEnemy(1,3);
%Sending a command (Command) to create an enemy and send it
    from the specified starting point to the specified end
    point (StartEnemy)
errorStartEnemy = SendData(IPAddressSend,portSend,txt,'Command'
    ,'name="StartEnemy"');
```

AddTower

Adding a tower.

```
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.
Example of adding tower number 3 to the coordinates $x = 1$, $y = 4$:

```
%Server address
IPAddressSend = '127.0.0.1';
%Port on which the server is listening
portSend = 55001;
%Creating data in xml format containing information about the
    tower number and its coordinates
txt = AddTower(3,1,4);
%Sending a command (Command) to create a tower (AddTower)
errorAddTower = SendData(IPAddressSend,portSend,txt,'Command','
    name="AddTower"');
```

RemoveCriteria
Removal of criteria whose values for all decision variants do not differ from each other.

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

Description:

- E – data table, the columns are the criteria and the rows are the alternatives,
- W – criteria weights,
- PrefDirection – criteria Preference direction (1-max;2-min),
- ind – criteria indexes that have not been deleted.

Returns a table of data, columns are criteria and rows are alternatives.
Example of deleting criteria:

```
%Data table
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];
```

```
%Vector of criteria weights
W=[2 10 8 1 20 10];
%Vector of criteria preference directions: 1-max, 2-min
PrefDirection=[2 2 2 2 1 1];
%The criteria are removed, the data table and the vectors of
    weights and preference directions are updated
[E,W,PrefDirection, ind] = RemoveCriteria(E,W,PrefDirection)
```

TOPSIS
TOPIS function. Calculates alternatives measure values using the TOPSIS method.

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

Description:

- E – data table, the columns are the criteria and the rows are the alternatives,
- W – criteria weights,
- PrefDirection – criteria Preference direction (1-max;2-min),
- p – coefficient.

Returns an array of measure values for decision variants.
Example of calculating a measure value:

```
%Data table
E = [12 3 1;
      15 0 0;
      13 0 0;
      19 0 0];
%Vector of criteria weights
W=[2 10 8];
%Vector of criteria preference directions: 1-max, 2-min
PrefDirection=[2 2 2];
%Obliczenie wartosci miary
score=TOPSIS(E,W,PrefDirection,p);
```

VIKOR
VIKOR function. Calculates alternatives measure values using the VIKOR method.

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

Description:

- E – data table, the columns are the criteria and the rows are the alternatives,
- W – criteria weights,
- PrefDirection – criteria Preference direction (1-max;2-min),
- p – coefficient.

Returns an array of measure values for decision variants.
Example of calculating a measure value:

```
%Data table
E = [12 3 1;
      15 0 0;
      13 0 0;
      19 0 0];
%Vector of criteria weights
W=[2 10 8];
%Vector of criteria preference directions: 1-max, 2-min
PrefDirection=[2 2 2];
%Calculate three types of measure values
[Q,S,R]=VIKOR(E,W,PrefDirection,p);
```

GenerateRanking
Creates a ranking of alternatives..

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

Description:

- v – Vector of alternative performances.

Zwraca pozycje w rankingu.
Przykład utworzenia rankingu:

```
%Example vector
exampleVector = [1.25 2.23 1.25 0.3 1.5 4.5];
%Creating a ranking
ranking = GenerateRanking(exampleVector);
```

GenerateReport
Generating a report with the results of the tested MCDA method.

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

Description:

- fileNamePlot – name of the file to which the plot will be saved,
- fileNameTab – name of the file to which the table will be saved,
- scoreArray – scores (assessments) ot paths,
- funName – name of the tested MCDA method,
- EnemiesToEnd – number of enemies that have reached the end of all paths (main score of the MCDA method),
- EnemiesMeanHealthRatio – average health level of enemies that have reached the end of the paths (additional score of MCDA method),
- TracksCost – total length of the paths selected by the MCDA method (second additional score of MCDA method).

Report generation example:

```
%Array containing example (hypothetical) path selection results
     using the MCDA method in a game of four rounds with two
    paths to choose from
exampleArray = [0.5 0.4;
  0.3 0.4;
  0.7 0.3;
  0.6 0.8];
%Variables containing example (hypothetical) results of the
    MCDA method
exampleEnemiesToEnd = 3;
exampleEnemiesMeanHealthRatio = 0.65;
exampleTracksCost = 56;
%Creating a game report with the results of the MCDA method
GenerateReport('scoreRounds.png','scoreRound.html',exampleArray
    ,'TOPSIS',exampleEnemiesToEnd,exampleEnemiesMeanHealthRatio
    ,exampleTracksCost);
```

Contents of the 'scoreRounds.html' file:

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

Figure 10: Visualization of the generated html page

```
<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>
<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>
```

Figure 10 shows a screenshot of what the generated page looks like. Figure 11 shows the contents of the 'scoreRounds.png' file.

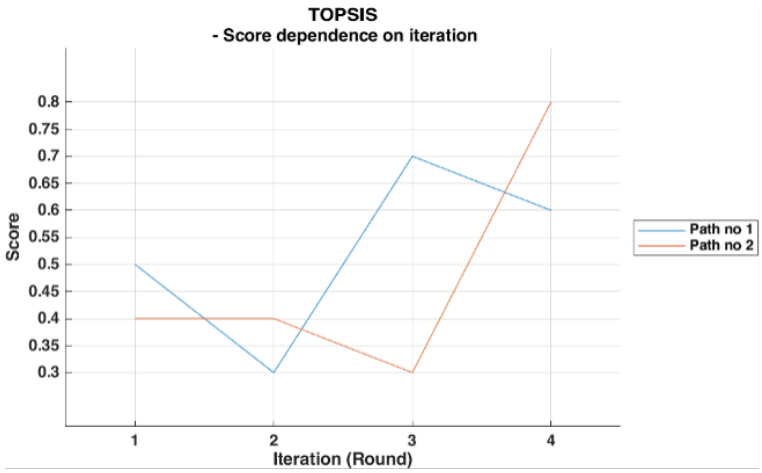


Figure 11: Contents of the 'scoreRounds.png' file

GenerateTabular
Generation of a tabular table.

```
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 that table row descriptions are not bold, and 1 means that they are bold,
- decimalPlaces – number of decimal places.

Example of generating an array:

```
%Array
exampleArray = [1 2;
                3 1;
                5 2;
                2 4];
%Column descriptions
```

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

Table 1: Generated array

```
columnDescriptions={'No','Date 1','Date 2'};
%Line descriptions
rowDescriptions={'1','2','3','4'};
%Creating a file containing the tabular environment
GenerateTabular('array.tex',exampleArray,columnDescriptions,
    rowDescriptions,0,0);
```

Contents of the array.tex file:

```
\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}
```

You can attach an array to a Latex file:

```
\begin{table}
\input{array}
\caption{Generated array}
\end{table}
```

The effect obtained is presented in the array 1.

GenerateTikzData
Generating data files for tikz charts.

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

Description:

- fileName – name of the file to which the array will be saved,

- data – saved array,
- columnDescriptions – column descriptions.

Data generation example:

```
%Array
exampleArray = [1 2;
                3 1;
                5 2;
                2 4];
%Column descriptions
column Descriptions={'No','D1','D2'};
%Creating a file containing data for tikz charts
GenerateTikzData('array.dat',[[1:size(exampleArray,1)]'
    exampleArray],columnDescriptions);
```

Contents of the array.dat file:

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

The array.dat file can be attached to a tikz chart:

```
\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};
```

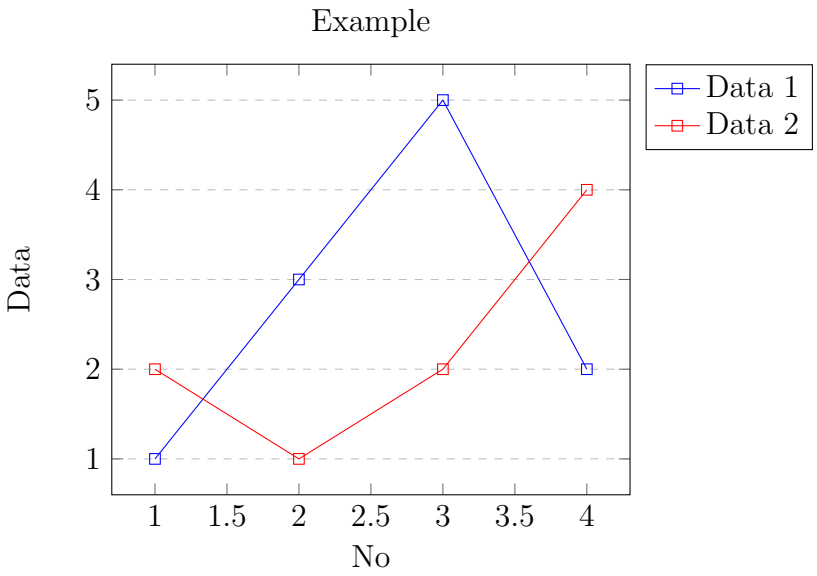


Figure 12: A chart based on data generated by the GenerateTikzData function

```
23
24      \legend{Data 1, Data 2}
25
26  \end{axis}
27  \end{tikzpicture}
```

The effect obtained is shown in the figure 12.