Uniwersytet Wrocławski

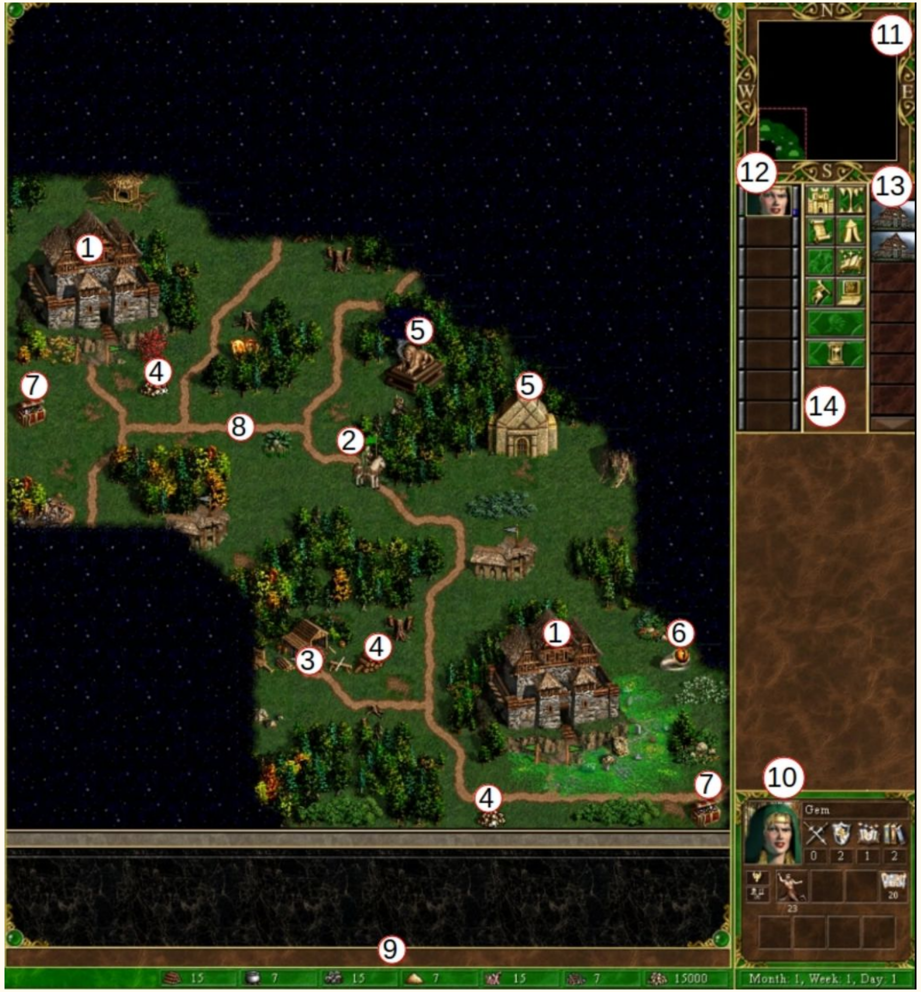
Designing a Template-Based Map Generator for Heroes of Might & Magic III

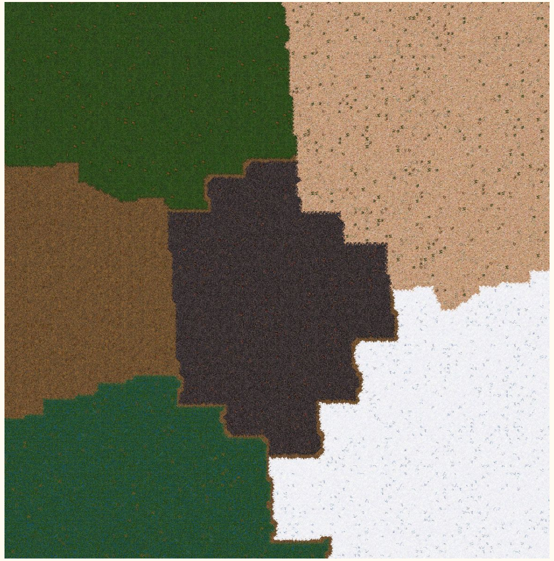
Authors: Dawid Skowronek, Grzegorz Kodrzycki
Supervisors: dr Jakub Kowalski, mgr inż. Radosław Miernik

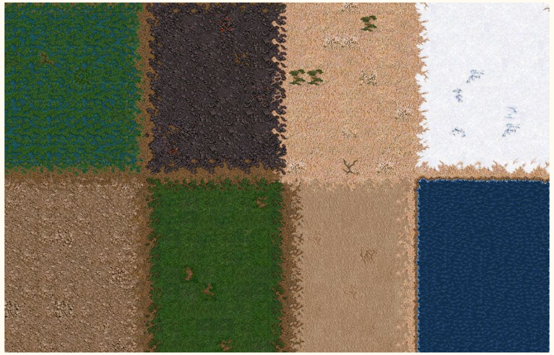
Aim of the thesis

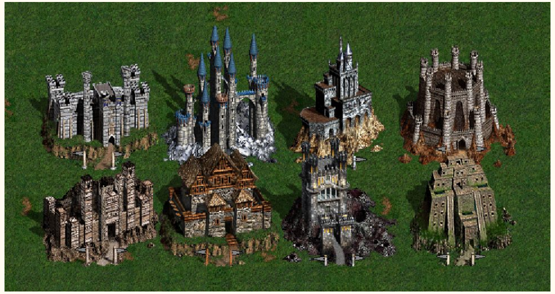
The aim of this thesis was to create a fully functional map generator for Heroes of Might & Magic III. Additionally, the thesis presents a possible solution for testing maps with the available AI built into the game.

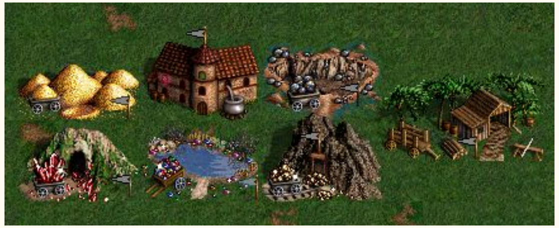
GUI of HOMM3

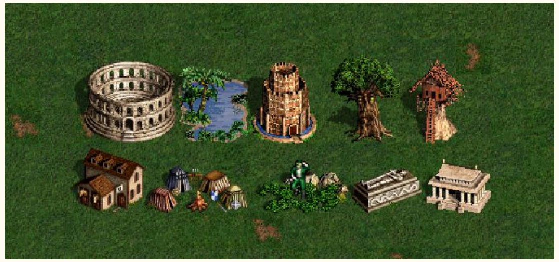

1
2
3
4
5
6
7
8
9
10
11
12
13
14
N
S
W
E
Gem
0 2 1 2
15 7 15 7 15 7 15000
Month: 1, Week: 1, Day: 1

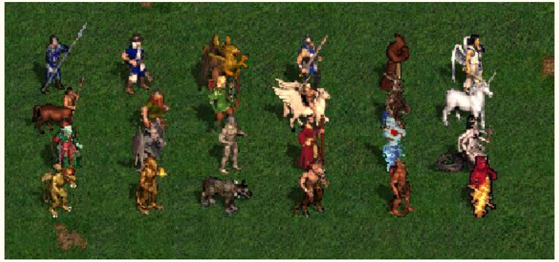

zones

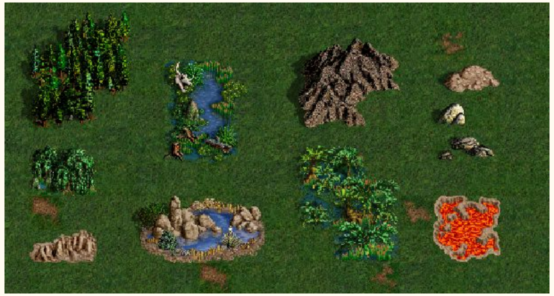

terrains

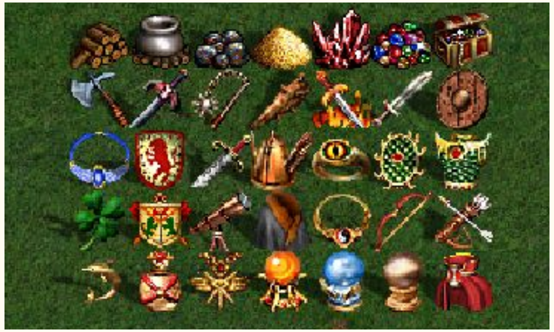

towns


mines


special buildings


monsters


obstacles


collectibles

Gus Smedstad

- **Zone Placement:** Circles, arbitrary grid, “shaking box”, water.
- **Zone Polishing:** Voronoi, fractal randomization.
- **Obstacles:** Square states, “reverse building”, reachability, landmarks task.
- **Buildings:** Density, placement order.
- **Connections:** Borders, junction zones, water connections, special connections.
- **Guards:** Rating and item value, alignment, number of guards.
- **Treasures:** Classification, dwellings, variations, blocks.
- **Aesthetics:** Obstacle pools, rivers for water mills etc.

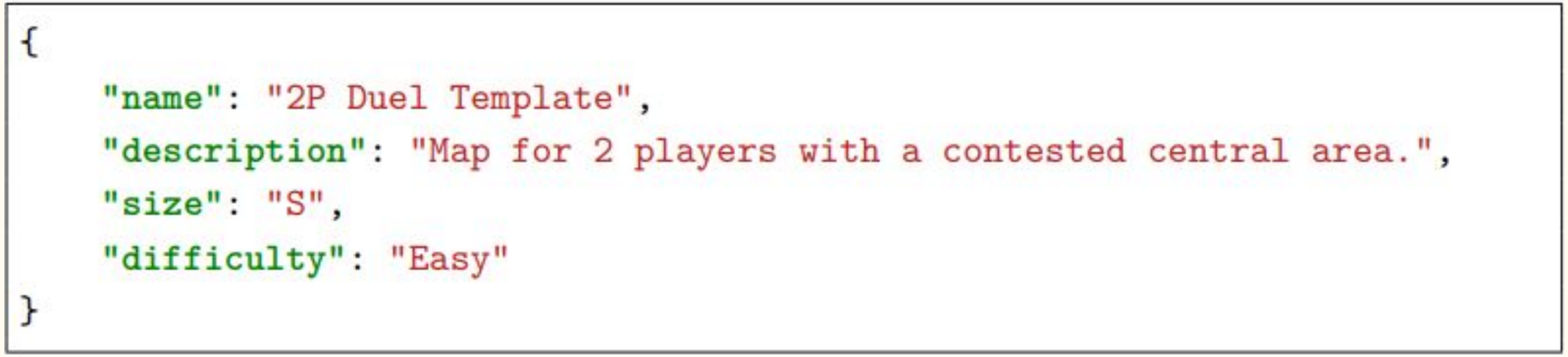
VCMI

- **Fundamentals:** Undirected graph, water zone.
- **Modifiers:** Phased filling, dependency management, topological order.
- **Zone Placement:** Dijkstra, Fruchterman-Reingold, Penrose tiling.
- **Connections:** Guards, roads, gates, water handling.
- **Fractalization:** Free paths, junction exclusion.
- **Treasures:** Distance-based placement, value ranges.
- **Obstacles:** Blocking, biomes, space filling.
- **Template:** example VCMI template

Template description

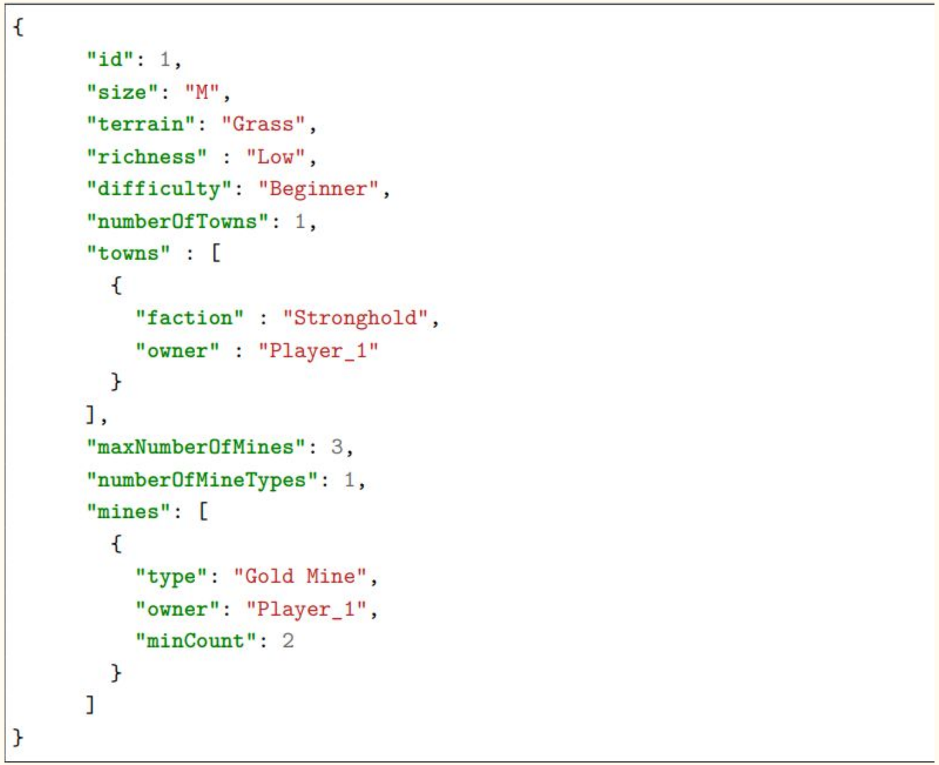
The template is in JSON format, allowing for easy readability and modification. the template is divided into smaller components.

General map information



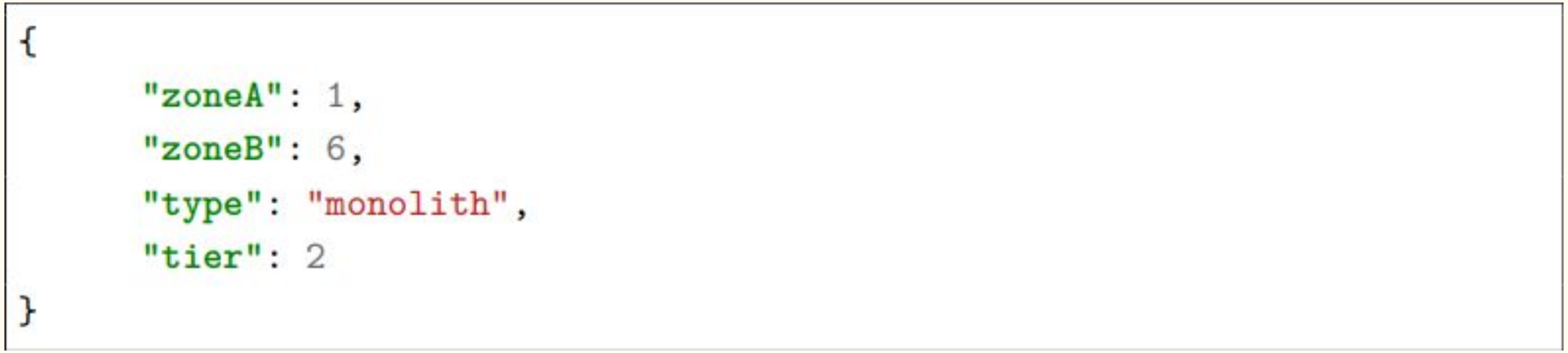
```json
{
    "name": "2P Duel Template",
    "description": "Map for 2 players with a contested central area.",
    "size": "S",
    "difficulty": "Easy"
}
```

Zone information



```
{
      "id": 1,
      "size": "M",
      "terrain": "Grass",
      "richness" : "Low",
      "difficulty": "Beginner",
      "numberOfTowns": 1,
      "towns" : [
        {
          "faction" : "Stronghold",
          "owner" : "Player_1"
        }
      ],
      "maxNumberOfMines": 3,
      "numberOfMineTypes": 1,
      "mines": [
        {
          "type": "Gold Mine",
          "owner": "Player_1",
          "minCount": 2
        }
      ]
}
```

Connection information



```
{
        "zoneA": 1,
        "zoneB": 6,
        "type": "monolith",
        "tier": 2
}
```

Map Generation

Map generation is quite a complex task to tackle. Dividing it into smaller parts allows for code modularization, making the work a bit easier.

Map Generation

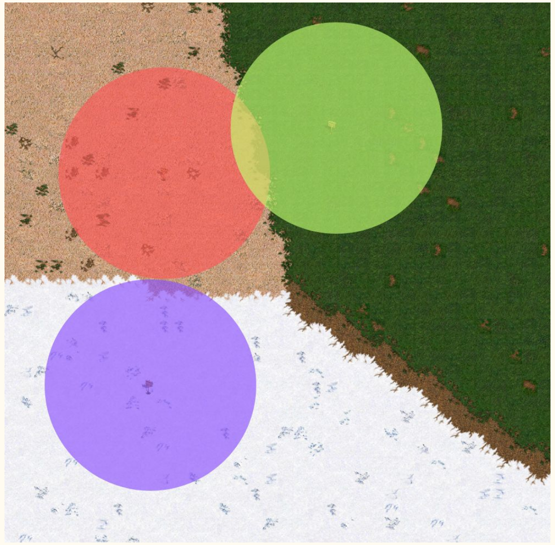
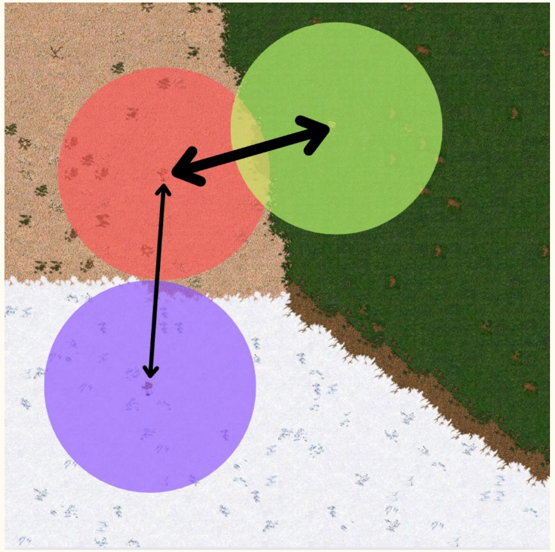
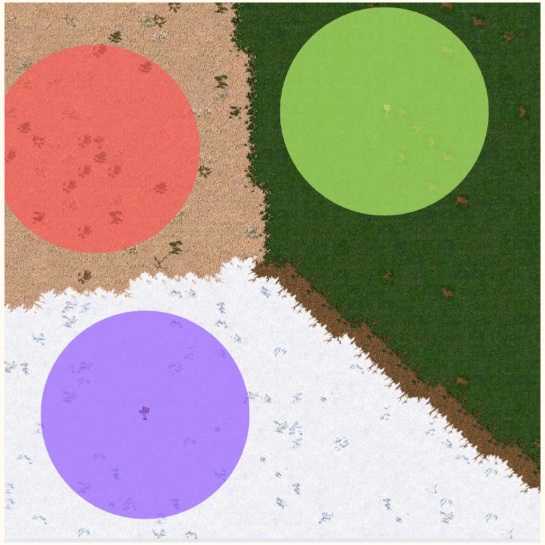
- Zone generation
- Town placement
- Border and Connection generation
- Object placement
- Road placement
- Guard placement
- Noise placement

Map Generation

- **Zone generation**
- Town placement
- Border and Connection generation
- Object placement
- Road placement
- Guard placement
- Noise placement

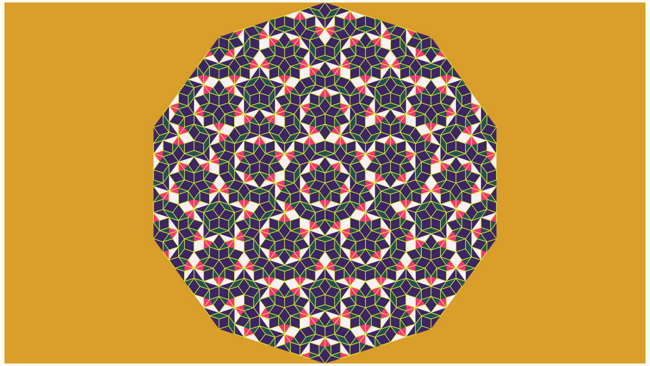
Zone generation

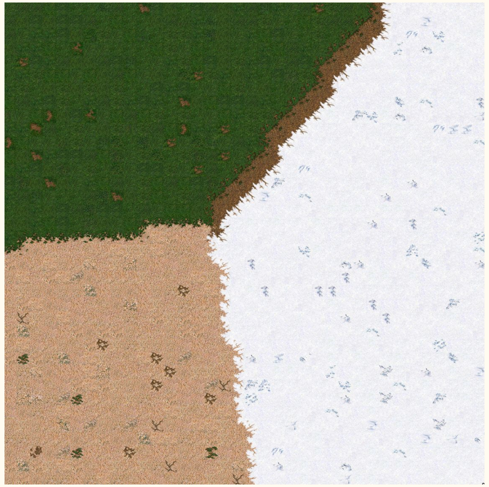
First, we need to find N such $N^2 \geq \#\text{Zones}$. Then, we create an $N \times N$ grid. Starting from the first zone, we place it on a random grid edge. Finally, we place zones in each cell while maximizing their distances.

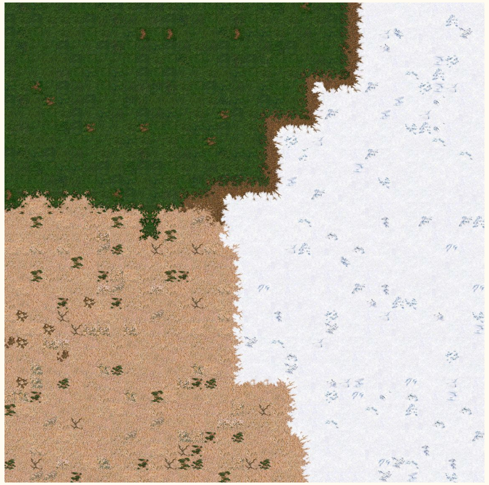
Zone generation

The final step is to assign the corresponding terrain type to each tile. In the first version, we assigned terrain from the closest zone. Another approach is to generate Penrose Tiling and assign vertices from this tiling to the nearest zone (therefore, the desired terrain type), then assign each pixel to the closest vertex.





without Penrose



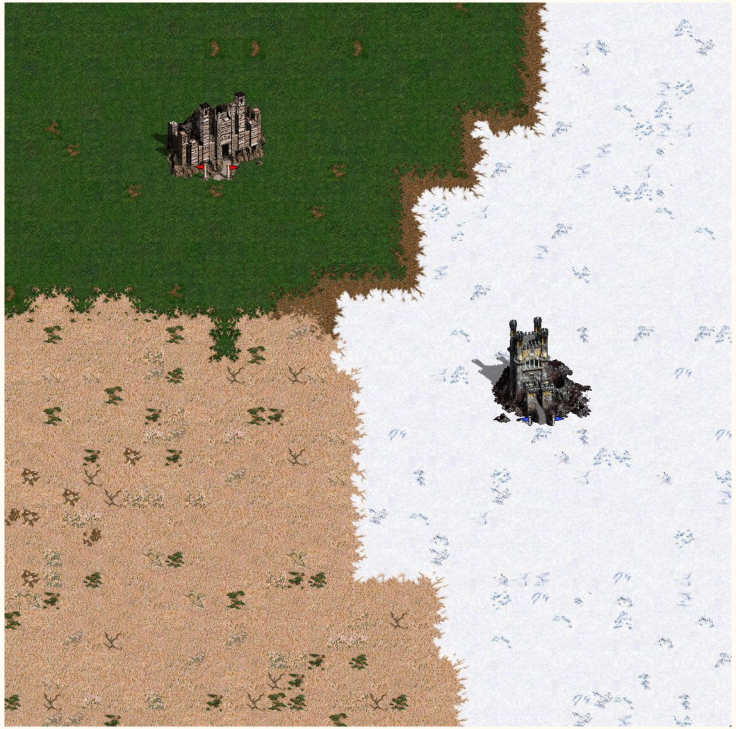
with Penrose

Map Generation

- Zone generation
- **Town placement**
- Border and Connection generation
- Object placement
- Road placement
- Guard placement
- Noise placement

Town placement

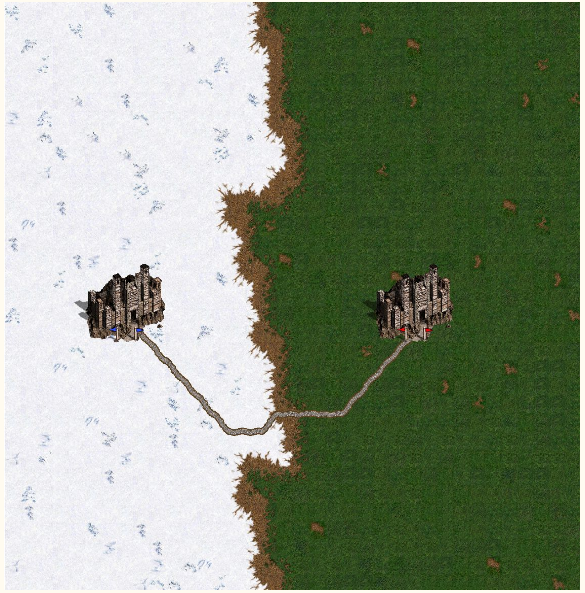
The position for each of them is set to the center of mass of its respective zone. It is crucial to place towns early in the process, as subsequent steps heavily depend on towns' positions.

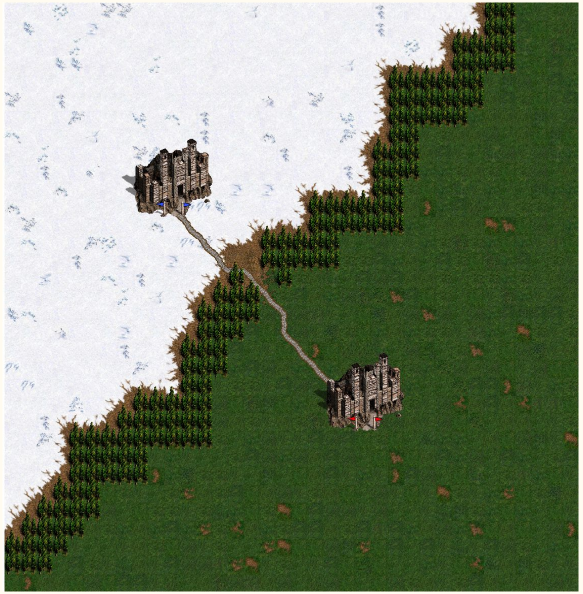


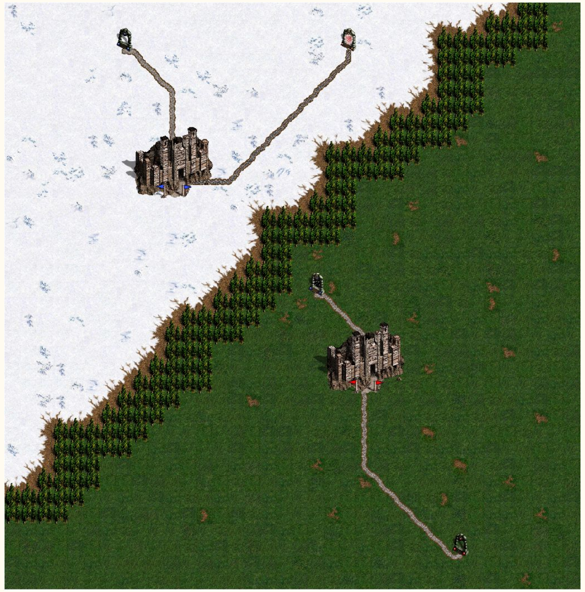
Map Generation

- Zone generation
- Town placement
- **Border and Connection generation**
- Object placement
- Road placement
- Guard placement
- Noise placement

Border and connection of zones generation


wide


narrow

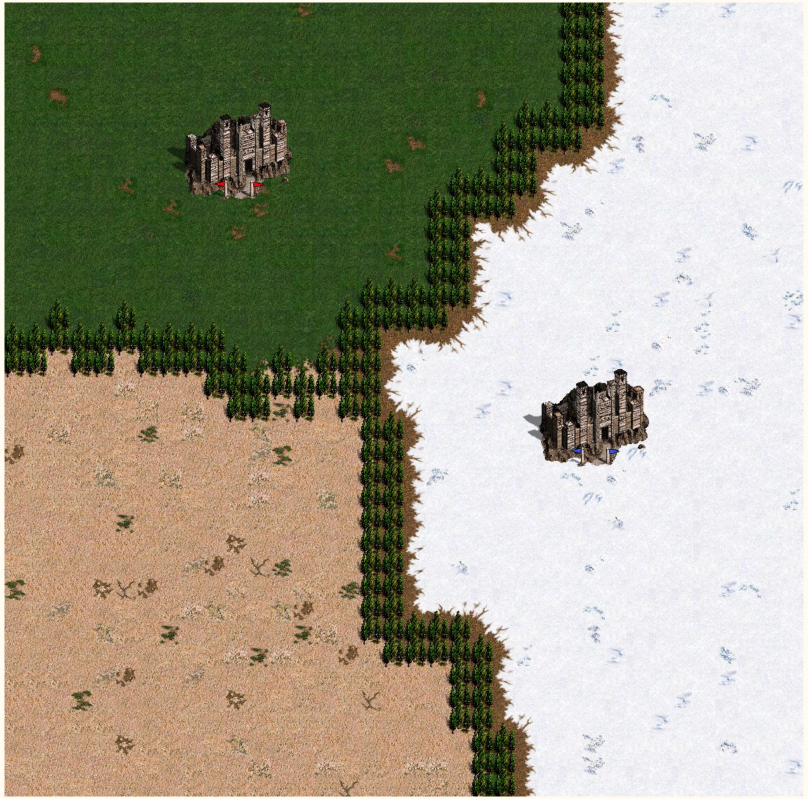

monolith

Border and connection of zones generation

Generating borders and finding connections is done in a few steps:

1. Determining zone borders
2. Finding connection points
3. Setting wide connections

Border and connection of zones generation



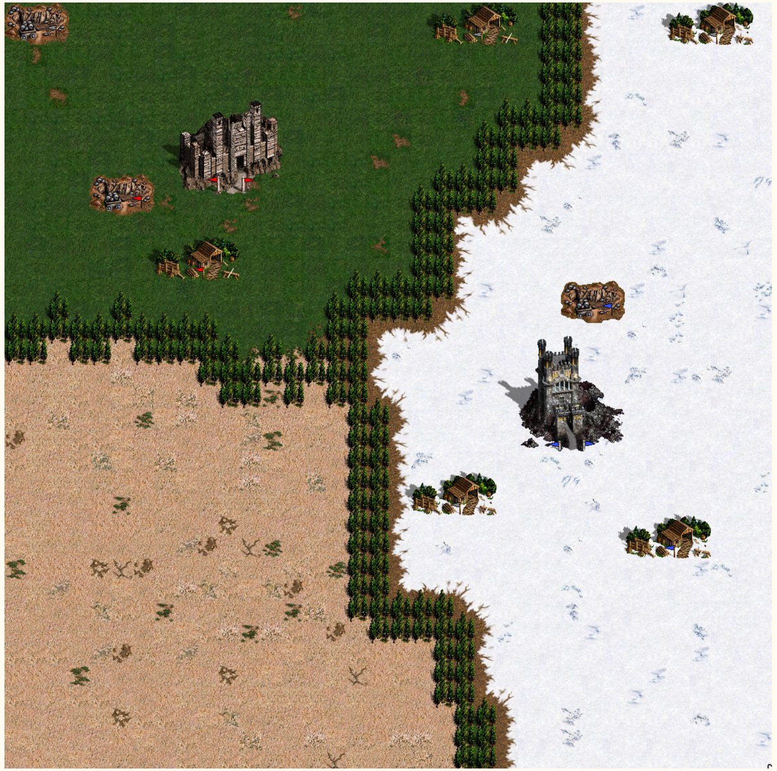
Map Generation

- Zone generation
- Town placement
- Border and Connection generation
- **Object placement**
- Road placement
- Guard placement
- Noise placement

Object placement

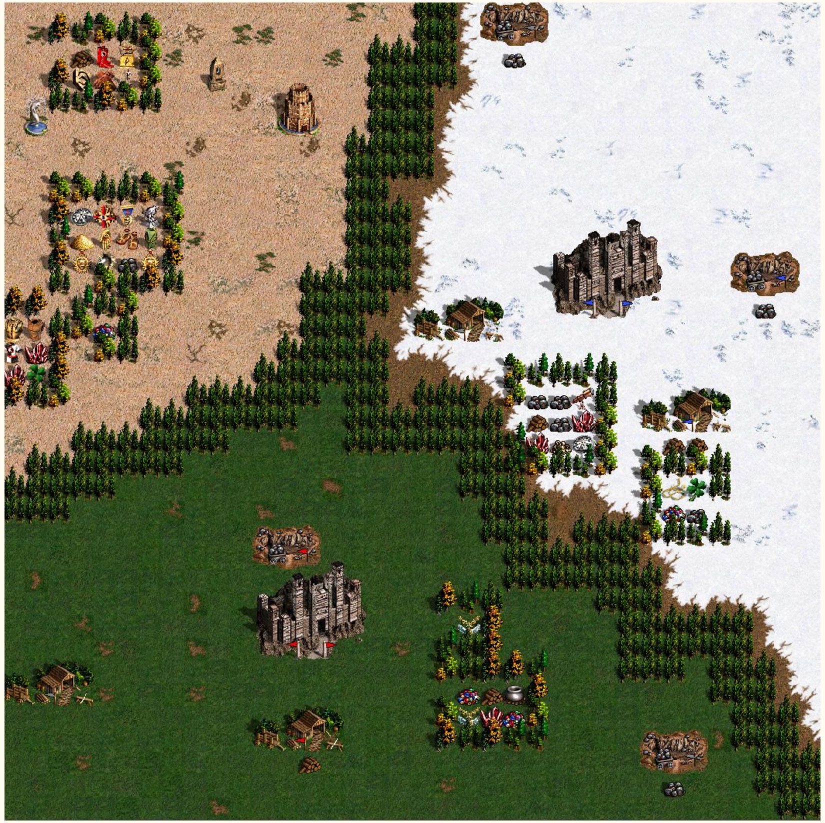
We classify mines, collectibles, and special buildings as objects. We begin with the placement of mines, followed by collectibles and special buildings.

1. Mines placement
   a. Placing essential mines
   b. Placing the minimum count of specified mines
   c. Placing random mines to hit *maxNumberOfMines*



Object placement

2. Treasures placement
    a. Resources near mines
    b. Treasure block
    c. Treasure buildings



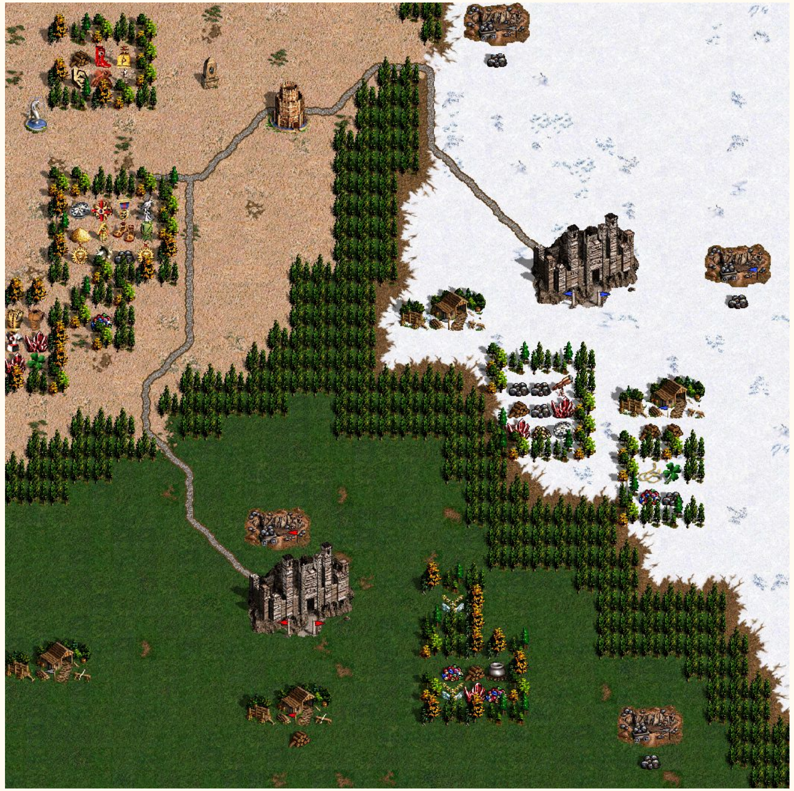
Map Generation

- Zone generation
- Town placement
- Border and Connection generation
- Object placement
- **Road placement**
- Guard placement
- Noise placement

Road placement

We run Dijkstra’s algorithm again, ensuring that we will step on free tiles. We may choose ones just next to objects, but this is not obligatory.

Once we generate the entire path, we unmark necessary tiles if they were marked as borders. Additionally, if the path is not from a town, we find a tile surrounded by obstacles on any axis and designate it as a guardian tile, where we will place a guard later in the guard placement phase.



Map Generation

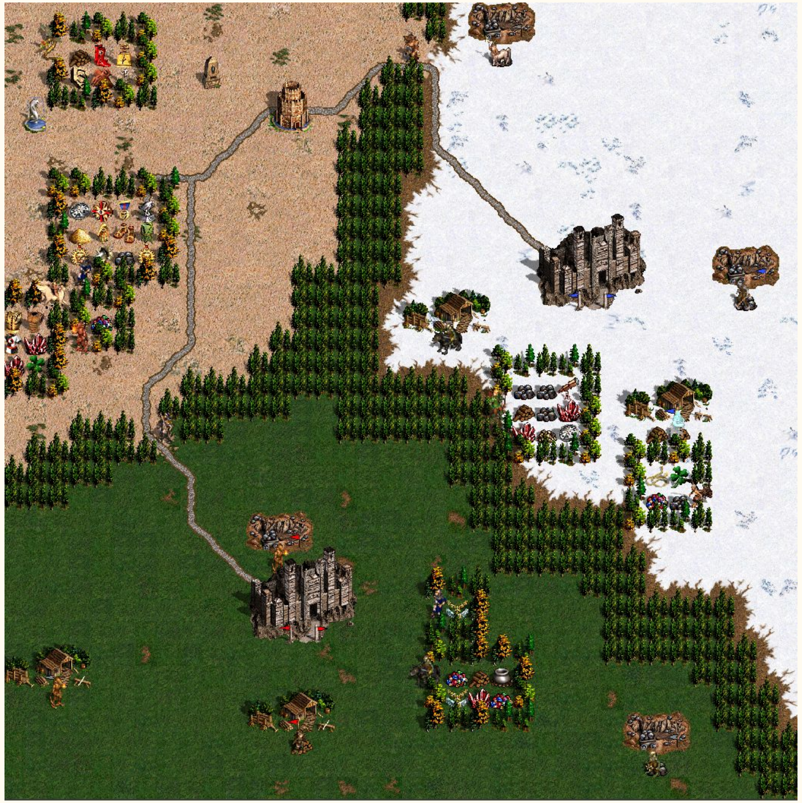
- Zone generation
- Town placement
- Border and Connection generation
- Object placement
- Road placement
- **Guard placement**
- Noise placement

Guard placement

In previous steps, we marked tiles that should have a guard on them. We consider two distinct types of guard:

- Gate guards
- Treasures/mines guards

Each type has a different difficulty scale, determining creatures' level, quantity, disposition, and whether they can flee or grow in numbers.



Map Generation

- Zone generation
- Town placement
- Border and Connection generation
- Object placement
- Road placement
- Guard placement
- **Noise placement**

Noise placement

We generate Perlin Noise on an $N \times N$ grid. We are generating noise in the $[-1, 1]$ range. To get the binary output, we are taking only values which are greater than 0.

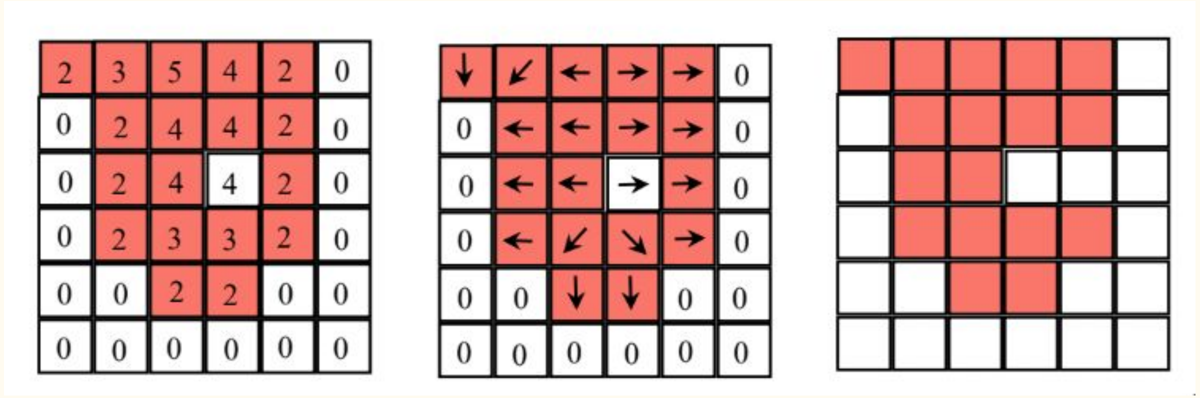
The next step is to apply this noise to free tiles. Then, we mark tiles that must remain reachable (e.g., roads, the bottom line of buildings).

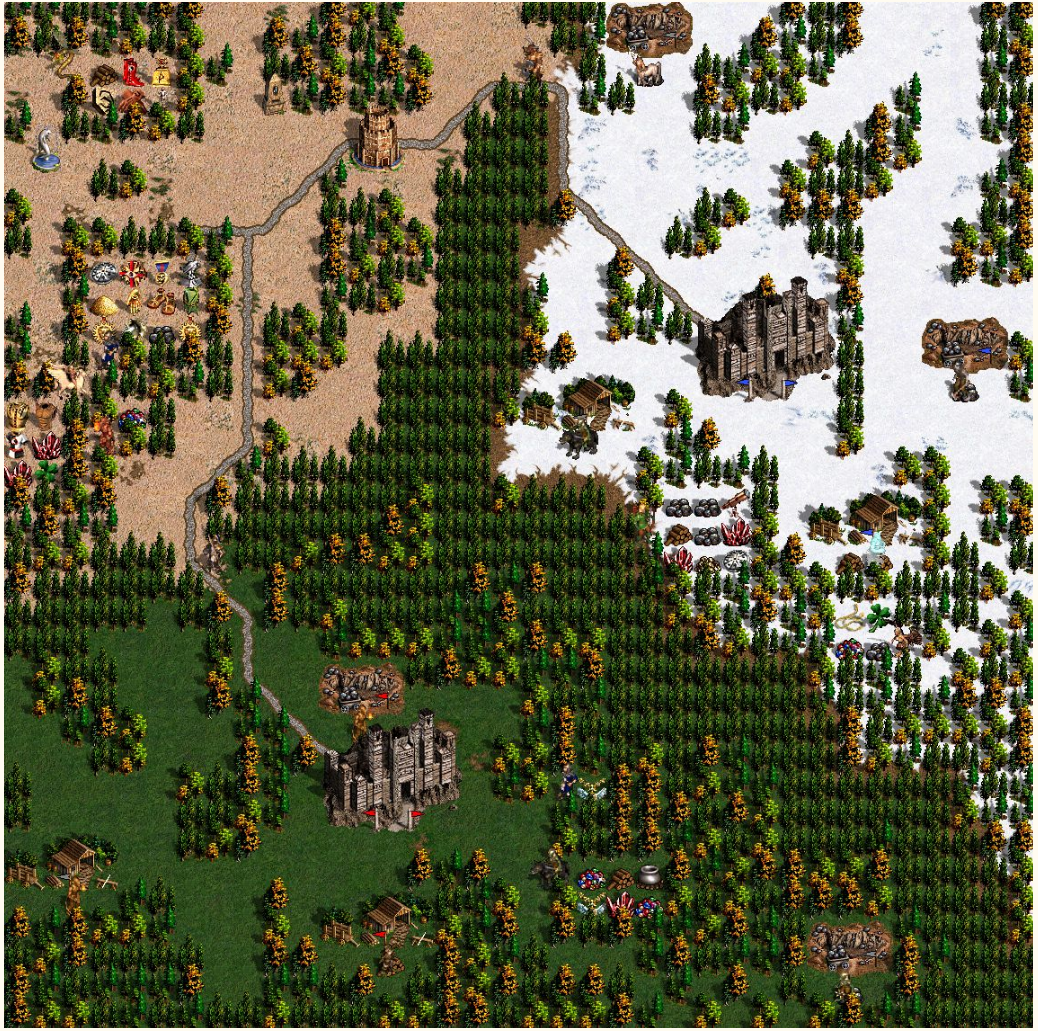
Noise placement

The next step is running Dijkstra’s algorithm to calculate a minimal number of obstacles from each town to be in a specific tile using the following rules:

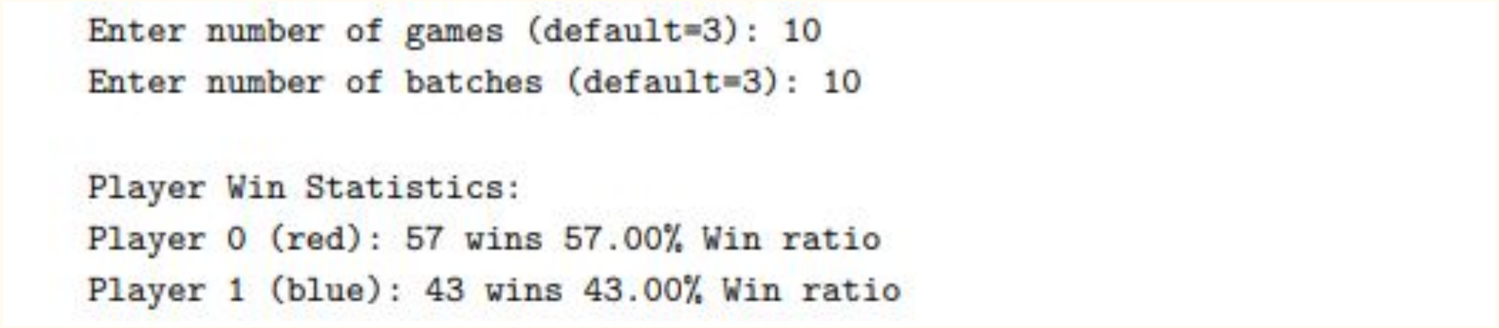
- if the tile is free: maintain the number from the previous tile,
- if the tile is occupied: treat it as an obstacle,
- if the tile was free and now is occupied: add weights after moving to it (1 for orthogonal movement, 2 for diagonal movement).

Noise placement





Fairness of the Map



```
Enter number of games (default=3): 10
Enter number of batches (default=3): 10

Player Win Statistics:
Player 0 (red): 57 wins 57.00% Win ratio
Player 1 (blue): 43 wins 43.00% Win ratio
```

Conclusion

We successfully created the generator, but we believe the topic is much more complex and that several other issues could be addressed.

What is missing in our generator:

- underground levels
- water zones

What we could improve:

- placement of objects
- change how we control economy of zone