Scenario Generation for a 2D Videogame using Logic Programming

Rafael Alcalde Azpiazu

Adviser: Pedro Cabalar Fernández

UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA
Department of Computer Science

Final degree project

A Coruña, 8th August 2018

Table of Contents

Table of Contents

Freeciv

- Turn-based strategy videogame.

Freeciv

- Turn-based strategy videogame.
- Created by students of Aarhus University.

Freeciv

- Turn-based strategy videogame.
- Created by students of Aarhus University.
- Nowadays developed by an open source community.

Game design

- Player controls a group of settlers in 4000 B.C.

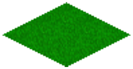
Game design

- Player controls a group of settlers in 4000 B.C.
- 5 ways to end the game:

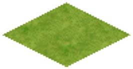
Game design

- Player controls a group of settlers in 4000 B.C.
- 5 ways to end the game:
    - Domination victory.
    - Science victory.
    - Religion victory.
    - Culture victory.
    - Score victory.

Freeciv terrains



- Grassland: Common. Units can move easily.



- Plains: You can create roads on these cells.

- Hills: Units move slowly. +200% defense bonus.

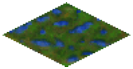
- Forest: +1 production unit. +150% defense bonus.

Freeciv terrains

- Jungle: +4 production units with gems/fruit bonus.

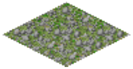
- Mountains: +300% defense bonus.

- Desert: +3 production units with oasis bonus.



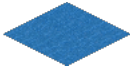
- Swamp: Fast irrigating. +5/9 production units with peat and spice bonus.

Freeciv terrains

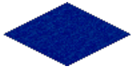


- Tundra: Only create roads.

- Glacier: No units can pass through.



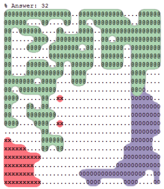
- Sea: All types of boats can pass through.



- Ocean: Only big ships can pass through.

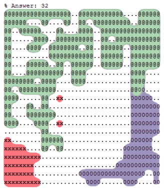
Table of Contents

First approach



- *Answer Set Programming for Procedural Content Generation: A Design Space Approach* [Smith et al, 11]

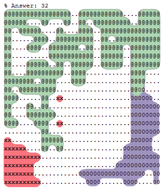
First approach



```
% Answer: 32
@@@@@@@@@@@@@@@@@@..@@@@@@@@@@@@....@@@@@@
@@@@@@....@@....@@..@@..@@@@@@@@@@..@@@@@@
@@..@@@@@@....@@....@@@@..@@@@@@@@@@@@@@@@
@@......@@@@..@@@@@@@@@@..@@..@@@@@@@@@@@@
@@....@@@@..@@@@@@@@..@@..@@@@@@..@@@@@@@@
@@..........@@@@@@@@@@@@..@@@@@@..@@@@@@@@
@@....@@@@@@..@@..@@@@@@..@@@@@@..@@@@@@@@
@@....@@@@@@@@@@..@@@@............@@@@....
@@@@@@@@..@@@@....@@@@............@@@@....
@@..@@@@@@@@@@....................@@00....
@@@@......@@..xx..................000000..
@@....@@..@@......................00000000
@@....@@@@@@@@....................00000000
@@@@....@@@@..xx..................00000000
..........@@......................00000000
xx........@@@@@@..................000000..
xxxxxx....@@..@@................00000000..
xxxxxxxxxx......................0000000000
xxxxxxxx......................000000000000
xxxxxxxx............000000000000000000..00
xxxxxxxxxx............0000......0000....00
```

- *Answer Set Programming for Procedural Content Generation: A Design Space Approach* [Smith et al, 11]
- This approximation creates a single solid island.

First approach



- *Answer Set Programming for Procedural Content Generation: A Design Space Approach* [Smith et al, 11]
- This approximation creates a single solid island.
- But we need more that one island.

First approach

- We created a starting point to generate an island.

First approach

- We created a starting point to generate an island.
- The program expanded these points with adjacency rules.

First approach

- We created a starting point to generate an island.
- The program expanded these points with adjacency rules.
- We added the constrains rules so the adjacent islands wouldn't stick together.

First approach

- We created a starting point to generate an island.
- The program expanded these points with adjacency rules.
- We added the constrains rules so the adjacent islands wouldn't stick together.
- Problem: This approach was inefficient with large maps.

Second approach

- We divided the map in regions.

Second approach

- We divided the map in regions.
- One region is a single island.

Second approach

- We divided the map in regions.
- One region is a single island.
- We used Lua built-in module clingo to call only one region generation.

Second approach

- We divided the map in regions.
- One region is a single island.
- We used Lua built-in module clingo to call only one region generation.
- Once this worked, we generated all the regions with build-in module.

Second approach

- We divided the map in regions.
- One region is a single island.
- We used Lua built-in module clingo to call only one region generation.
- Once this worked, we generated all the regions with build-in module.
- Finally we added the restriction rules to the regions not stick together.

Related work

- Add user restrictions to the generation.

Related work

- Add user restrictions to the generation.
- Add all types of terrain.

Related work

- Add user restrictions to the generation.
- Add all types of terrain.
- Generate player starting points.

Related work

- Add user restrictions to the generation.
- Add all types of terrain.
- Generate player starting points.
- Add a exporter to Freeciv map.

Scenario Generation for a 2D Videogame using Logic Programming

Rafael Alcalde Azpiazu

Adviser: Pedro Cabalar Fernández

UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA
Department of Computer Science

Final degree project

A Coruña, 8th August 2018