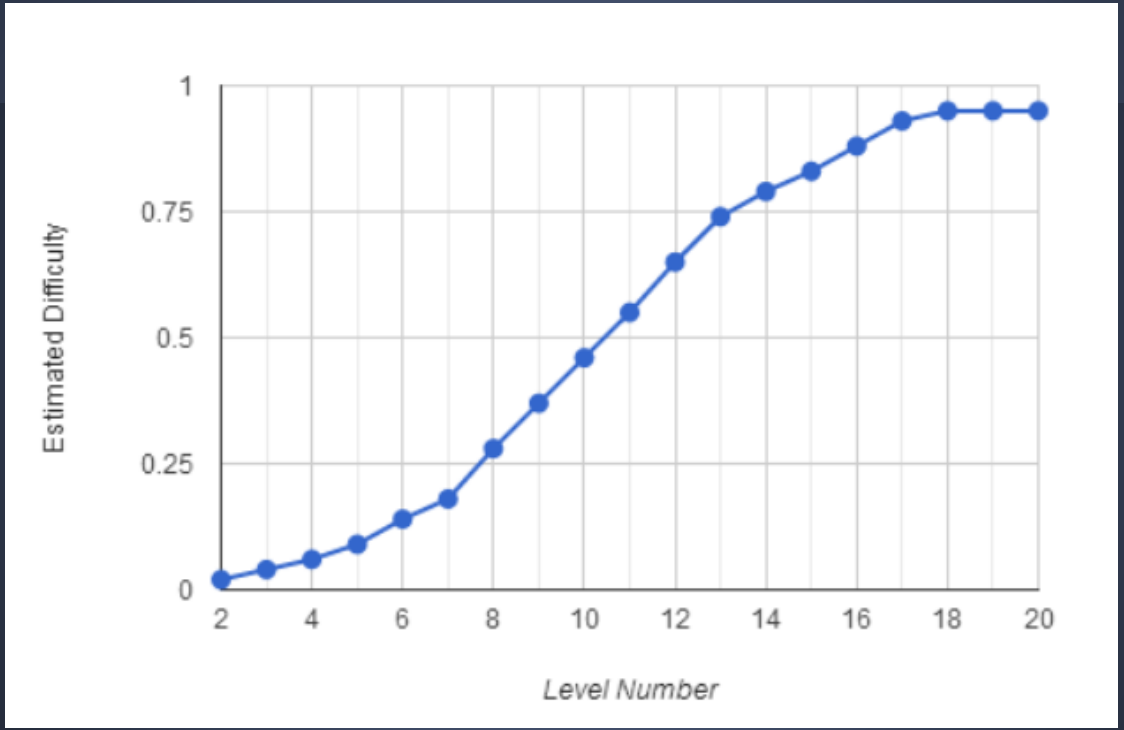
AR4: Turn-Based Strategy Game

Pedram Amirkhalili, 20254413, CSE

Project Purpose

2 major issues in TBT's:

1. Time - Solved via Random Generation
2. Difficulty Level - Can be solved via a Genetic Algorithm



The “ideal” difficulty curve

Objectives

1. Randomly Generated Levels
2. Core Mechanics
3. Genetic Algorithm
4. Gameplay Features

Random Generation

A Level consists of 3 components:

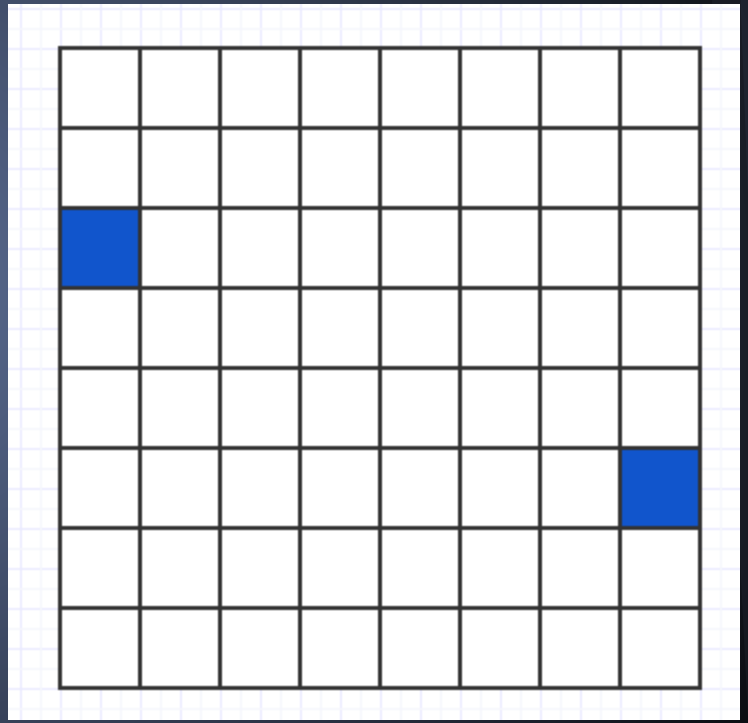
1. Map
2. Units
3. AI

Map

- Generated in stages
- Specific ordering
- Validated upon completion
- Base of Plains tiles

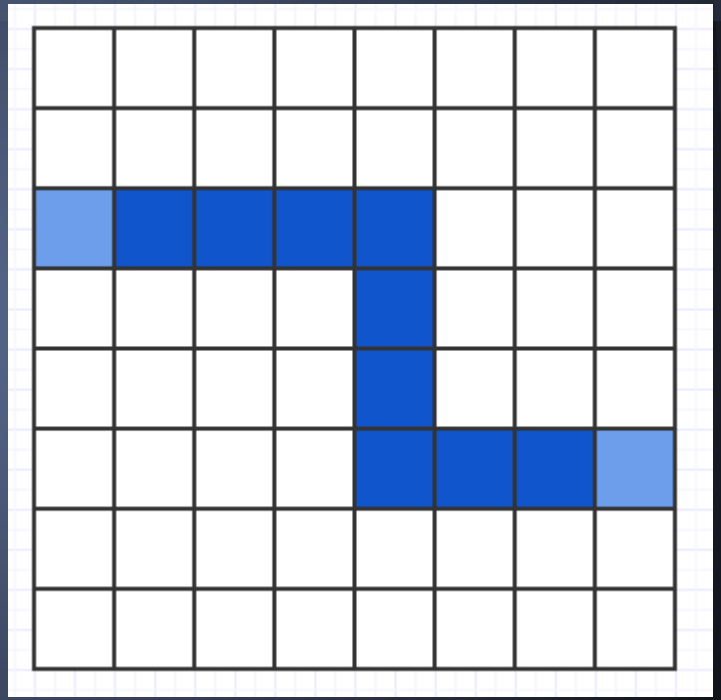
Map - Rivers

- Placed next to make it easier
- Algorithm works to create less “rigid” rivers
- Starts by selecting 2 edge tiles as endpoints



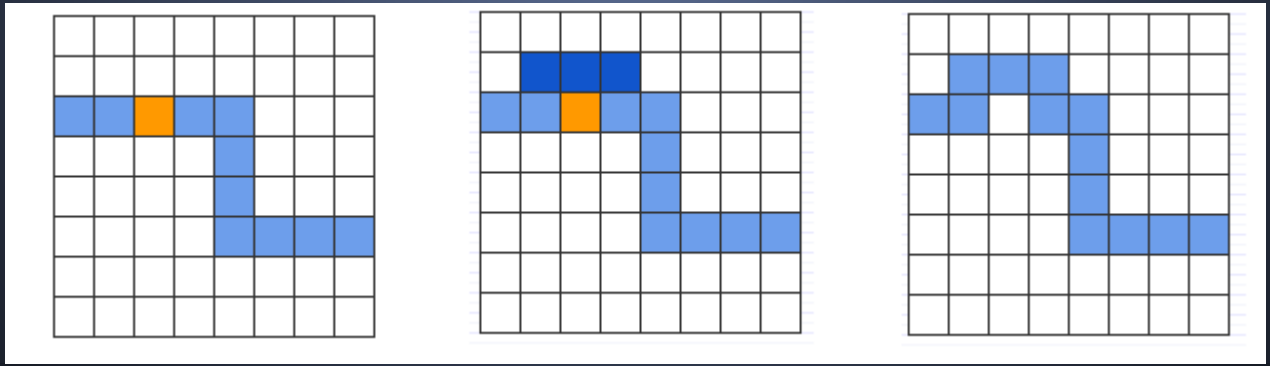
Map - Rivers

- Connect the endpoints
- Still very “rigid”



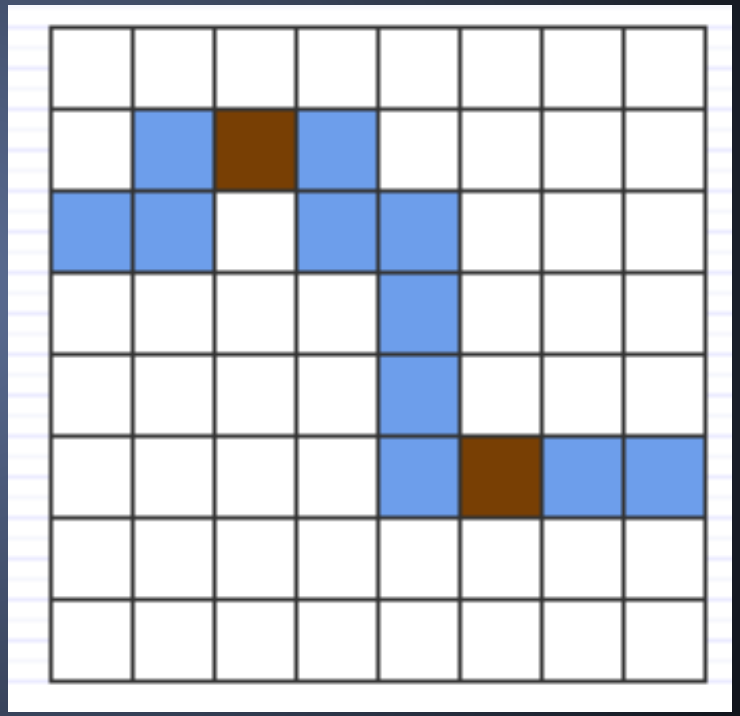
Map - River

- Try to bend the river
- Select a point on a straight edge
- Push in/out, adjust river
- Repeat recursively



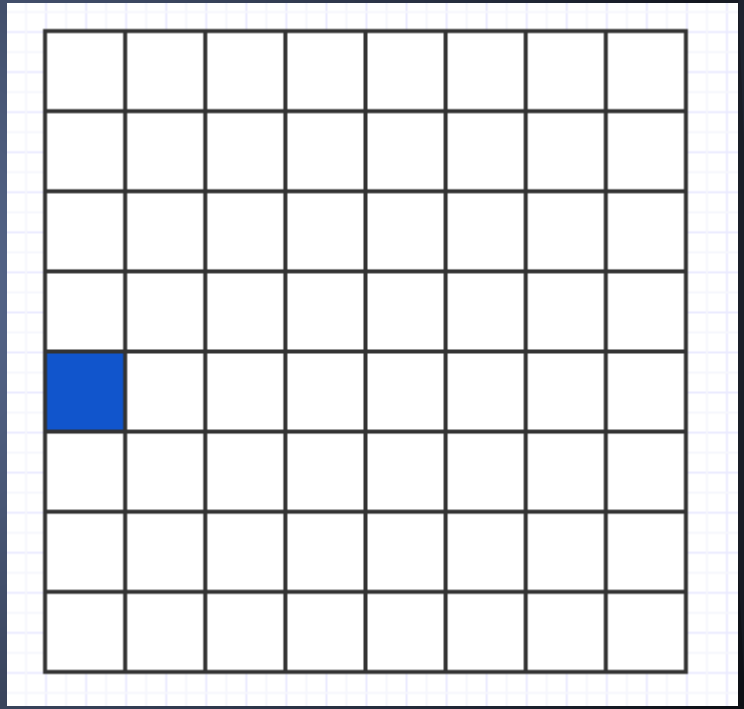
Map - Bridges

- Having just placed a River
- Bridges are placed along with it
- 15% chance that a River tile will be a Bridge tile during generation



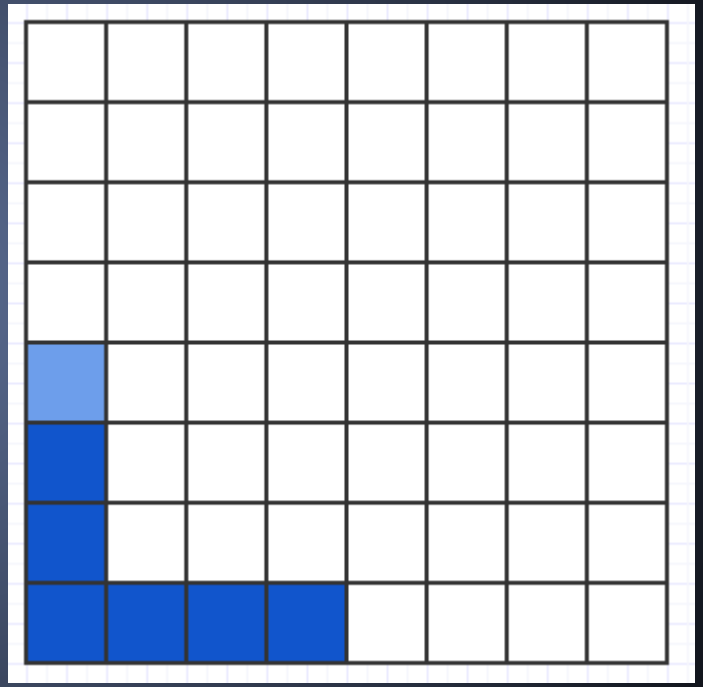
Map - Sea

- Placed after River, allowed to overwrite
- Start by selecting an edge tile as start



Map - Sea

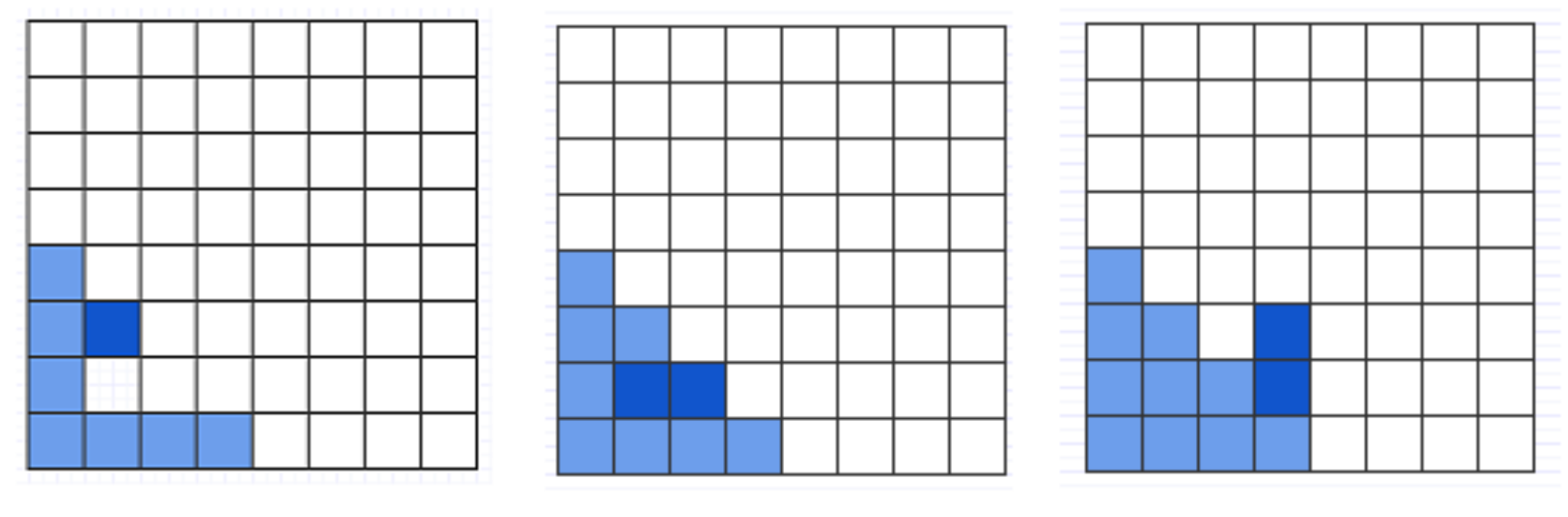
- Assign 25% of Sea tiles as base
- Draw along edge(s) to create base



Map - Sea

For each base tile:

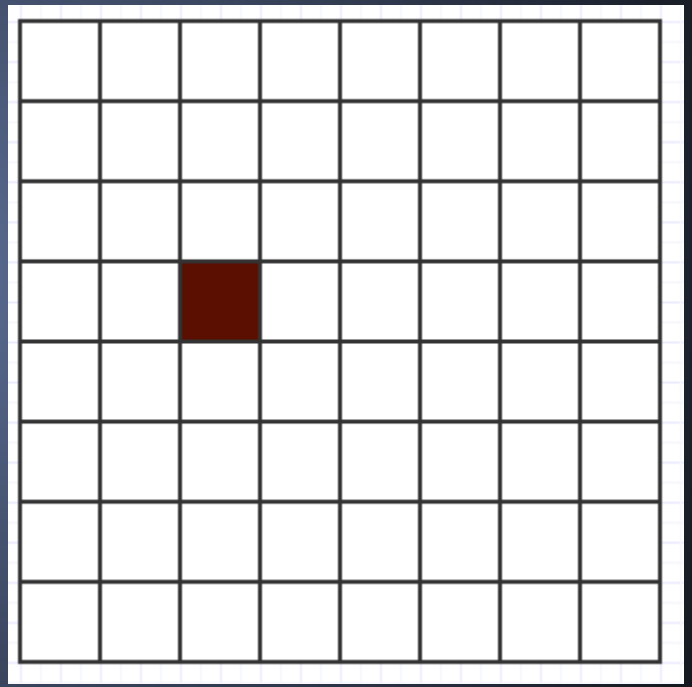
- Draw towards center of the map
- Amount drawn between 1 - 8
- Based on previous tile amount, increase, decrease, stay the same
- Probability of the type of change based on previous tile



Final Process being applied to the base

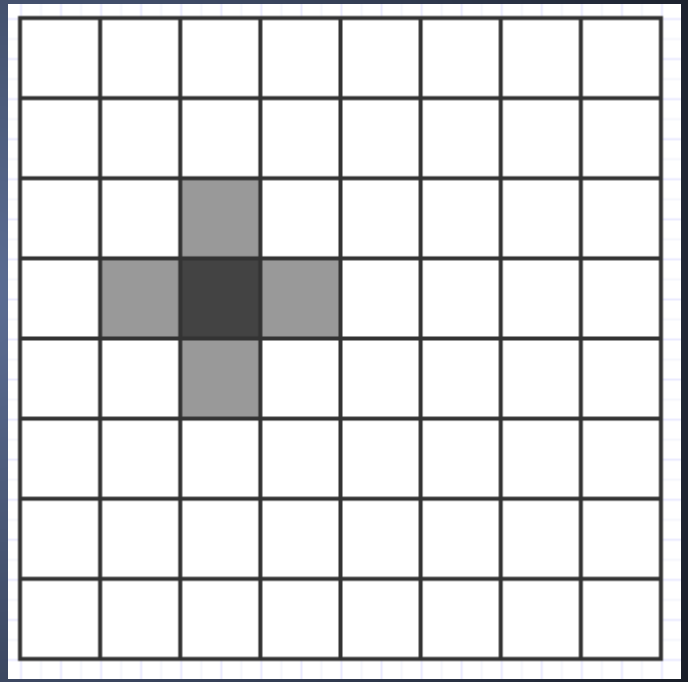
Map - Mountains

- Can only overwrite Plains tiles
- Aims to create “Clusters”
- Starts by randomly selecting a Plains tile as the start



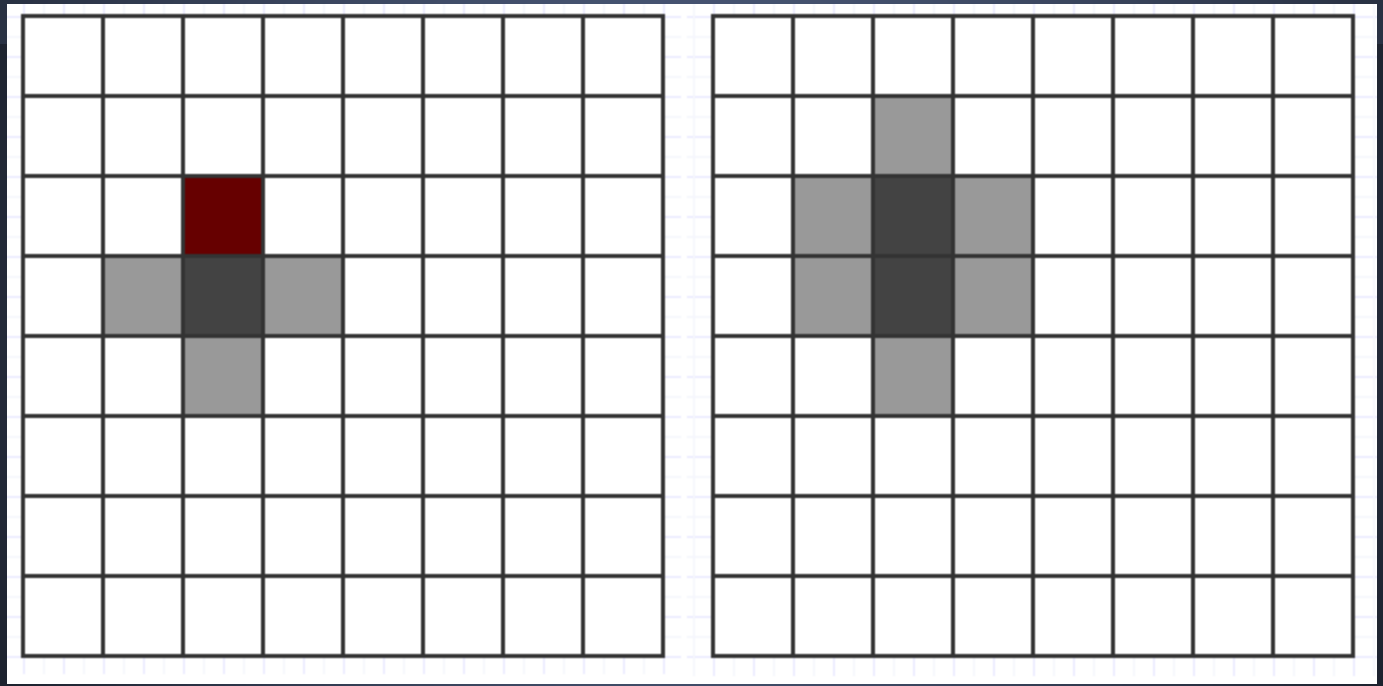
Map - Mountains

- Add the adjacent tiles to a pool of potential new locations



Map - Mountains

- Perform check if new cluster should be started.
- If not select a tile from candidates
- If Plains add as Mountain, if not reselect
- Add new adjacent tiles

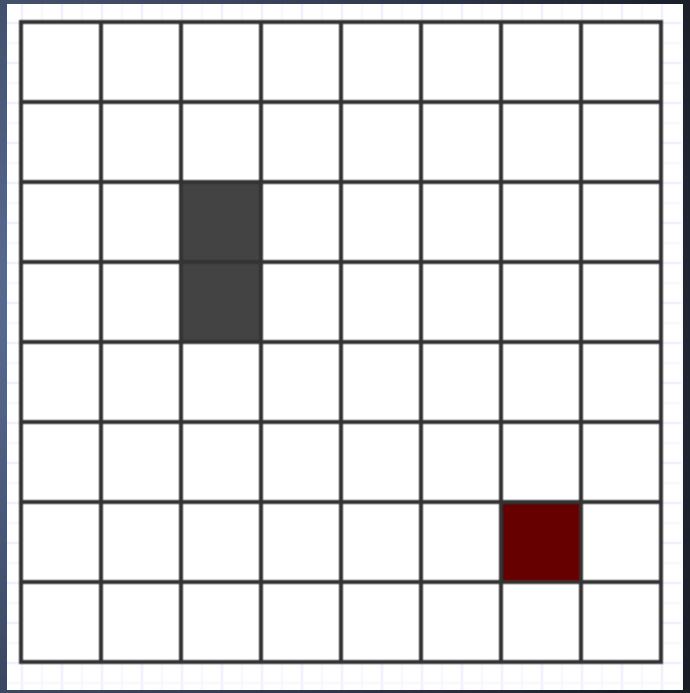


Process if new cluster check fails (don’t need a new cluster)

Map - Mountains

If a new cluster should be started:

- Clear pool of candidates
- Select a new start location



Map - Woods and Forts

- Placed last, least significance
- Simply randomly selecting tiles.
- If they are Plains they are replaced

Map - Validity Condition 1

Move from any crossable tile to another crossable tile

- Tested via a graph representation
- Connected graphs fulfill conditions
- Exhaustive test, uses Breadth-First Search

Map - Validity Condition 2

Rivers flow continuously

- Check each River tile
- Ensure that the adjacent River tiles are correct

Units

- Randomly selecting from a set of classes
  - Axeman, Acolyte, Archer, Shaman, Defender, Swordsman, Cavalier
  - Lord is player specific, generated separately
- Created at level 1
- Base stats are adjusted
- Levelled up if required

AI

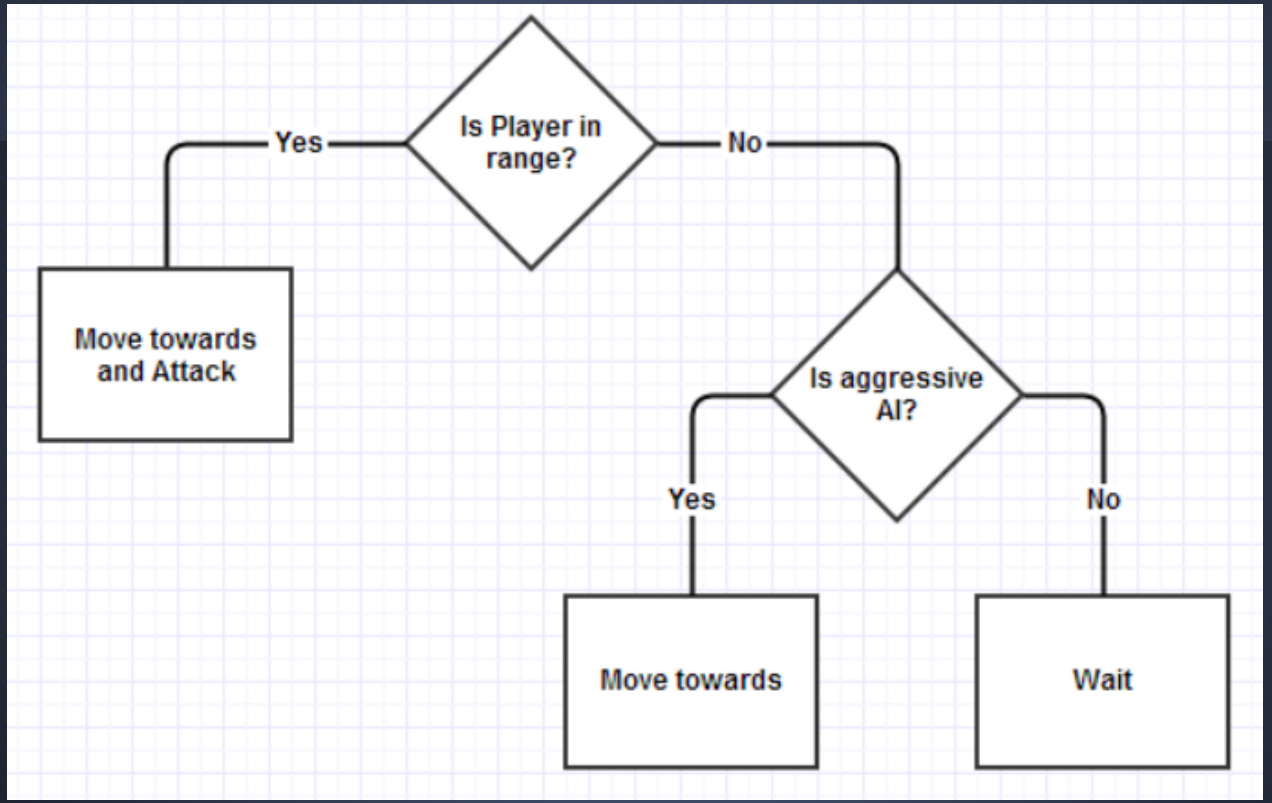
Two different types:

1. Passive
2. Aggressive

As well as a Ruthless Modifier

AI

- Dijkstra's Algorithm used for path finding
- Shaman class has unique AI
  - Hunts injured allies
  - Heals them



Flow diagram for the AI

AI - Ruthless

- Priority ordering to attack selection
  - Lord
  - Shaman
  - Archer, Acolyte
  - Swordsman, Axeman, Cavalier
  - Defender

Genetic Algorithm

Needed to control:

- Random Generation
- Difficulty

Works on a population of 10 levels

Fitness Function

A fitness function was made to try and evaluate levels

$$f(x) = MapDifficulty+$$
$$((PlayerUnitsScore * PlayerSkillScore) - (AIUnitsScore * AISkillScore))$$

Upon evaluating all levels, check if one fits requirements

Fitness Function - Map

Map is rated based on choke points:

$$m(x) = \sum_{cP=0}^{n} ChokePointScore \text{ (cP)}$$

$$ChokePointScore = \text{closer to player} \begin{cases} \text{true} = \text{positive} \\ \text{false} = \text{negative} \end{cases}$$

cP is a individual chokePoint

Fitness Function - Forces

Player/AI's forces are rated on the strength of the units:

$$pU(x) = \sum_{u=0}^{n} UnitScore\ (\mathrm{u})$$

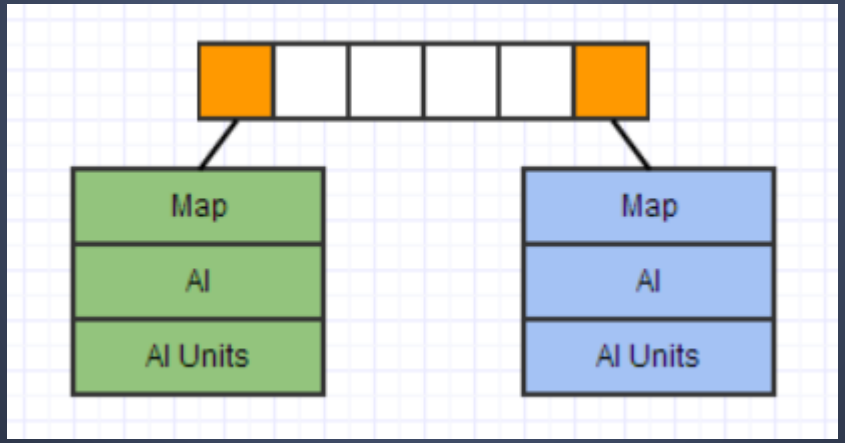
$UnitScore$ = rating based on the level and value of the unit (level * value)

u is a individual unit

This total is then multiplied by a skill level

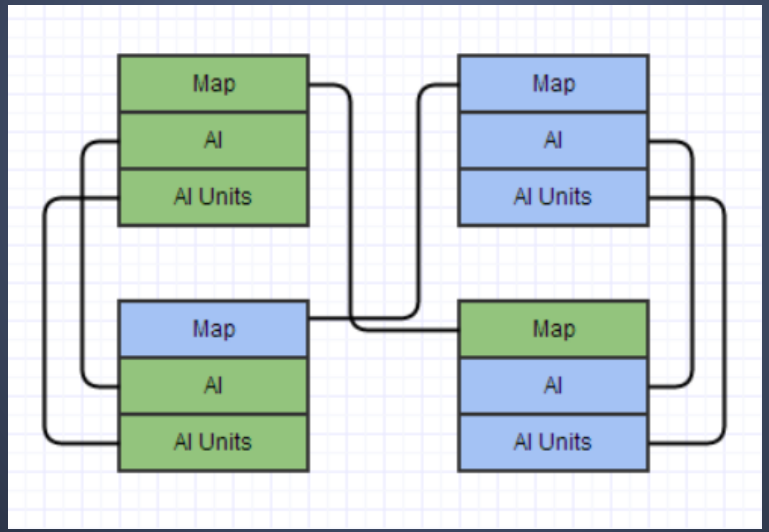
Selection

First and last of the population are selected



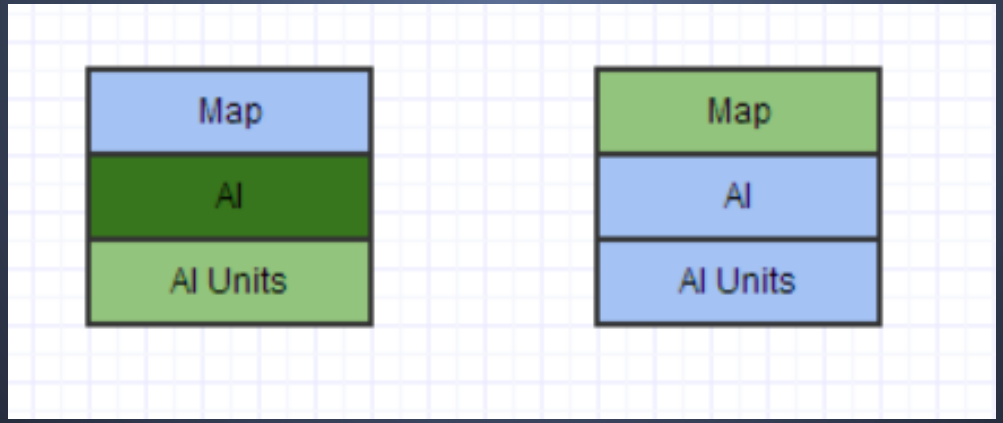
Crossover

Locus (crossover point) is selected



Mutation

Some aspect is potentially changed or regenerated



Testing

- Unit Testing
- System Testing
- Acceptance Testing
  - Player’s asked to answer question

Evaluation

- Majority of Objectives achieved
- Player’s liked the concept
  - Felt execution was hit and miss

Future Improvements:

- Full implementation of Map evaluation
- More intelligent breeding
- Better AIs

Demo

Thank you, any questions?