# GAME DOCUMENTATION

*G.I.G. 'Rising Star' 2020 Competition Entry*

by Daniel Paul

# Contents

## Premise & Design

In response to the brief from Grads in Games, I have proposed an idea that would allow the project to be adapted into a fully functioning game. First and foremost, the player is envisioned to be overseeing the game – playing 'god' – by instructing their army to build and destroy objects on the screen. An isometric perspective was chosen to represent the player's view as it seemed to fit well with the art-style and presentation.

There were many influences for the project. Most notably being 'Stick War' (see Figure 1), Rollercoaster Tycoon 2 (see Figure 2) and 'Age of War' (see Figure 3), as these seemed to represent the strategic top-down experience that would be ideally replicated in this project.


*Figure 1 Stick War: Legacy (https://www.amazon.in/Max-Games-Studios-Stick-War/dp/B01BMTPWC8)*

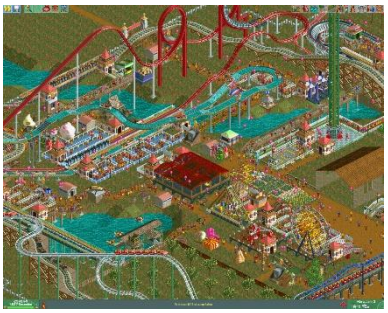

*Figure 2 Rollercoaster Tycoon 2 (https://store.steampowered.com/app/285330/RollerCoaster_Tycoon_2_Triple_Thrill_Pack/)*


*Figure 3 Age of War (https://play.google.com/store/apps/details?id=com.maxgames.ageofwar1&hl=en)*

A great emphasis was put on the procedural techniques used within the project. It was decided to utilise the midpoint displacement algorithm I had created previously to allow for generation of rock clusters. I chose to opt out of using randomly placed obstacles as using a more pseudo-random approach would be more realistic and easier to understand. For instance, having the obstacles spawn in groups would allow for representing rocky landscapes, mountains etc.

Plans were created to allow the player to click on one of their units to control its movements and functions. This gives the player more control over the entire battle. For instance, if there is a specific rock that blocks a quick path to the enemies, it would be useful to tell a forager to go to it and forage it (therefore opening up the path).

Initially there was two unit types designed – a Forager and a Warrior. The Forager would cost the least of the two and can be spawned to then go to the nearest rock, mine the resource and return to base with the earnings. Your base would also generate 10 cash every 10 seconds to balance the game with a potentially simple AI. The Warrior can be spawned at a higher cost to then seek and destroy enemy units until all are dead. Once this is achieved and a path to the enemy base is found, the Warrior then sets out to destroy it and win the game!

Through creation of this dynamic, it is believed to create simple yet enjoyable enough gameplay and by having each level procedurally generated, this adds greater replay-ability to the overall game.

For basic controls refer to Figure 4 or the how to play screen in-game.

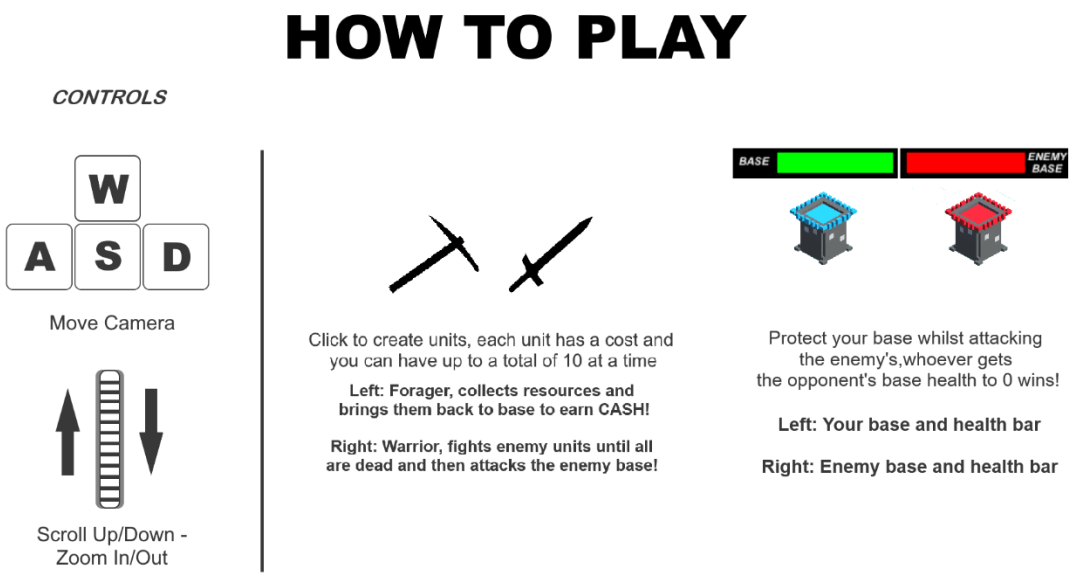


*Figure 4 How to play screen*

# Implementation

Implementation of the main game loop discovered many unique ways to create the experienced designed prior. For instance, the adaption of using the midpoint displacement algorithm allowed more flexibility in choosing how the world is created.

Midpoint displacement worked as a 2d array of floats by applying these 2 steps:

1. Get corner values of the grid and find centre by calculating the average values of these 4 points

2. Calculate midpoints of sides and average out

Visualising this would essentially create a realistic-looking terrain with mountainous peaks and troughs. By taking the values at specific heights, it could then calculate the 'randomness' of the actual game map.

As the algorithm is pseudo-random, it will be different each time but will look similar in many ways. To reduce the possibility of a game map generating with very little rocks, it was decided to have an external helper function to check how much %-wise the inaccessible tiles took up the grid. Through doing this, it was able to then repeat the algorithm until it produced a result that was both more realistic and fairer for the player to have better odds of resources spawning closer to them.

To add to this, rivers were also randomly generated horizontally or vertically to also add more obstacles to the world. Rivers are used to provide a static inaccessible obstacle to navigate around that can't be changed (unlike a rock).

Another major aspect of the implementation was how the 'AI' enemy would spawn its units. This was done very simply (however could be improved significantly in future) by having a random chance to try to create a forager or warrior every 5 seconds. It will favour creating a forager on the first try as it will have enough cash to create it and it will allow the enemy to generate more cash.

A final major aspect to mention is the camera. An isometric image cannot be viewed using a perspective camera unless it was changed to accommodate an orthographic view. Changing the in-game camera to an orthographic perspective allowed the creation of a convincing isometric view which the camera can navigate. The movements were adjusted to adapt to the orthographic view so that it could still move left, right, up and down like a normal perspective view.

# Evaluation

There were many problems I had faced whilst implementing the design. The most notable challenge I faced was incorporating the character controlling system. This allowed the player to click on one of their units and control it directly to forage, attack etc. Due to the timeframe and complexity of having both autonomous and manual control, this was decided to be removed to focus on autonomous simulation of battles.

Another problem faced was the implementation of tile-forager interaction. This created an issue where foragers would converge on a single rock or would just break if the rock was surrounded by water. These issues were resolved by implementing a tracker for checking if a rock was being foraged by someone, a list was then implemented to track all the rocks which can be then edited when one of these impossible tiles is found. This list is then used in tile choosing for the forager until a tile was found.

Currently, the only noticeable bug in the game is the rugged warrior vs warrior interaction. Due to the grid provided, the current tile position saved could be inaccurate (e.g. character is about to leave it etc.) so the warriors would travel to the position they 'think' the other warrior is. This results in some chasing back and forth between each until they both converge and fight. I believe this could be addressed by handling each warrior interaction as a 1v1 scenario (e.g. find a point near them to battle and get both in position etc.).

The resulting game produced has fulfilled most of the design laid out prior and the implementation has addressed the approaches required to create the functionality. There are some aspects that I would have preferred to have added – if given more time.

These include:

- Implementing audio for all aspects of the game (e.g. in-game music)
- More units that have more complex functions
    - For instance, a builder that can create fortifications etc.
- Pre-defined world types
    - This could include a snowy world with slippery ice, or a swamp-like level where small rivers are very common etc.
- More sophisticated river generation (e.g. utilising the pathfinding algorithm to create more realistic river curvature etc.)
- Controllable characters that you can click on and gives bonuses for what they do (e.g. faster foraging if controlling a forager unit etc.)

## Summary

In summary, I feel the project was a success as the design, implementation and result created an enjoyable strategy game. By combining procedural techniques such as midpoint displacement, it allows each level to be approached in different ways and has the potential to expand to become more complex and engaging. Whilst I faced many challenges, I feel the final product reflects the brief and design well, incorporating many of the original features in unique ways.

**Note: All game art not provided by Grads in Games was either created by me or in the public domain!**

A screenshot of a typical level in the final game is shown in Figure 5:

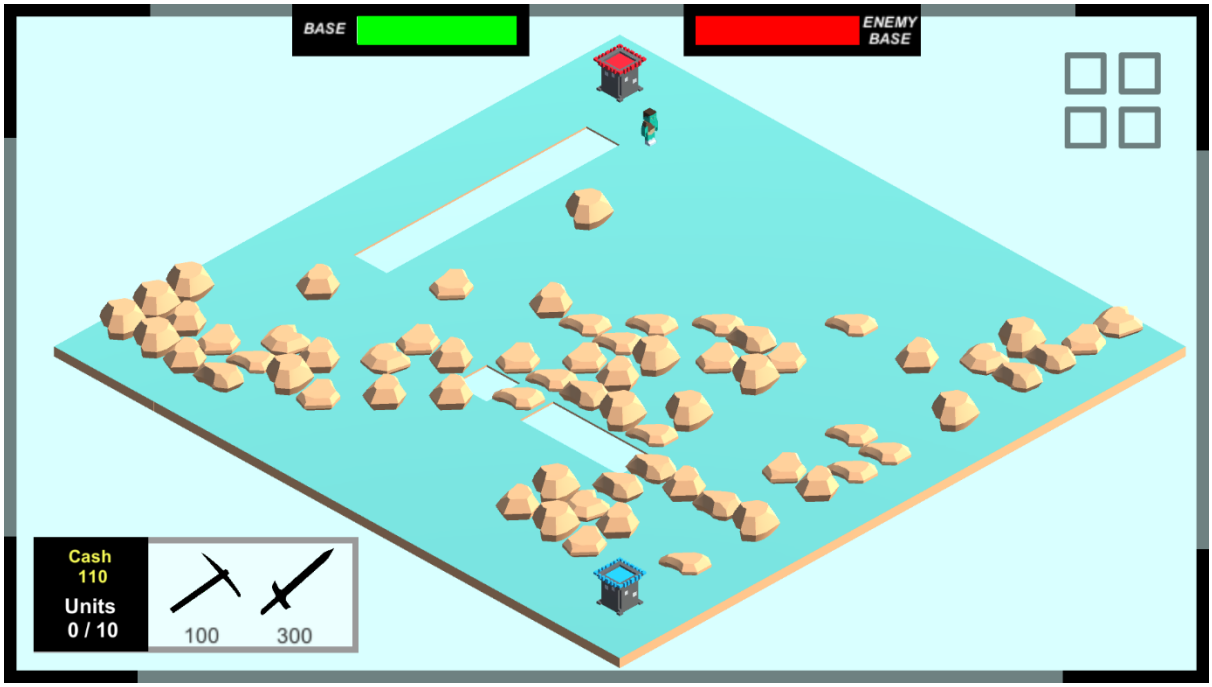


*Figure 5 Screenshot of a level generated in the final game*