# Project for BUas – Concise Development Log

## Preface

This is the concise, cleaner and more comprehensible version of the development log. This one was made with the expectation to be viewed and read by other people, while the other log is mainly for me to personally describe my experience in as much detail as possible. However, it ended up being extremely long, so I of course don’t expect anyone to read it fully.

## Design concept- In summary:

With the theme of ants 3 main ideas played around in my head;

* Some sort of roguelike/survival game using random map generation.
* A puzzle game where a character would shoot ants from a gun to form terrain made out of little ants, which they would use to solve puzzles and platform through levels
* A strategy game, based on Into the Breach, where you control alien ants which invade different planets, akin to the tyranids from Warhammer 40k but more ant-like, where you would face off against an AI in strategy based battles.

All three of these ideas incorporated something interesting which would be difficult to tackle; random level generation, physics and a way to create and delete platforms which the player could interact with, and an AI to play against in a strategy game. While I liked all of these, the strategy game ended up being the most appealing, which is what I went with.

After some thinking, I ended up with the following in scope goals, out of scope goals and difficulties I might face during development.

### In-scope goals:

-Isometric turn-based strategy game, where you compete against an AI controlled enemy.  
-You control various units, different fighters, scouts, and a queen which can produce more units (up to a cap)  
-Different mission types: destroy cities, capture objectives, destroy all enemy units, etc

### Out-of-scope goals:

-Different types of enemy factions, which all have different powers and pose different challenges to the player  
-Upgrades to the player’s units, or different units the player can select. (different types of ants, long ranged ants vs melee ants, etc)  
-Randomly generated challenge, where you face off against a randomly generated map of enemies for rewards.

### Expected Difficulties

-Usage of C++. I’m personally more used to higher-level programming languages, such as javascript or java, so adopting C++ is going to be a bit of a struggle.  
-Implementing the enemy AI. I have never really tackled AI before (or at least not written one myself), so this will be quite difficult. I’m expecting this to be my biggest hurdle, but we’ll see.

## Development log

|  |  |  |
| --- | --- | --- |
| 06/12/2020 | Done: | -Planned the concept;  Ant based strategy game  -Started development  -Preplanned my approach (asked around about general hand built C++ game programming)  -Wrote tile rendering function  -Made sprites for the tiles |
| Reflection: | I felt like this was a pretty good start to it all; I’ve used c++ before, although not a lot so my experience is limited. I’ve also worked on a game before, so I assume I’ll quickly get used to everything. |
| Sources: | -Video on isometric tilemaps which I based my system off of: <https://www.youtube.com/watch?v=ukkbNKTgf5U&t=348s>  -References for learning C++/SDL2 I used: <https://lazyfoo.net/tutorials/SDL/index.php> <https://www.youtube.com/watch?v=QQzAHcojEKg&list=PLhfAbcv9cehhkG7ZQK0nfIGJC_C-wSLrx> <https://wiki.libsdl.org/APIByCategory>  Link to the Fast Track Part 11 about tiles where I copied and modified some code from: <https://www.3dgep.com/cpp-fast-track-11-tiles/> |
| 08/12/2020 | Done: | -Wrote ‘coords’ class as a pair of 2 integers  (so I wouldn’t have to convert every time I used the prebuilt vec2)  -Wrote screen to tilemap coordinate conversion function  -Wrote code to render the first iteration of the tilemap  -Wrote code to detect which tile the player was hovering over with the cursor  This code also draws a marker on the selected spot |
| Reflection: | -The coords class ended up being largely redundant due to the template’s existing vec2 object, although I kept the coords object around seeing as vec2 stores its values in floats rather than integers, which I used in coords. The conversion would slow down the program a lot, so I reasoned the coords were more useful. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 11/12/2020 | Done: | -Started writing unit class to store data about the units |
| Reflection: | -I still largely remembered how objects were generally written in C, so this didn’t take long at all. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 12/12/2020 | Done: | -Finished writing unit class  -Changed rendering function used to render the tilemap to also be able to render units  -Created an array of 3 units for the player to control |
| Reflection: | -Header files were originally kind of tricky for me. In hindsight, I don’t really know why they were so tricky, but that’s just the learning progress I guess. |
| Sources: | -Sources used to learn about header files: <https://stackoverflow.com/questions/25274312/is-it-a-good-practice-to-define-c-functions-inside-header-files> <https://stackoverflow.com/questions/16313174/c-cannot-i-assign-values-to-a-variable-in-header-file> <https://stackoverflow.com/questions/38942013/declaring-variables-in-header-files-c/38942057> |
| 14/12/2020 | Done: | -Added on Click effect;  When the a unit is clicked, they move 1 tile on the x axis  -Mapped out how the game would play   * Player starts their turn in a move-phase, where they get to move their units anywhere on the field. * The player moves their units by first clicking on the unit he wants to move. * The player then selects a tile the unit should move to, depending on whether the unit can actually move there. The units themselves will function kind of like chess pieces, with different movesets, e.g. some can move straight forwards, others can only move diagonally, etc. For now, the units are only going to be movable straight forwards in one of 4 directions. * Once the desired tile is clicked, the unit moves there by increments, as if to play a walking animation. * Once the unit arrived at its destination, it can no longer move any more. * Once all units have moved, the player enters an attacking phase and can attack with their units. * This attacking phase will follow the same principal; the player clicks their units, and then selects a direction for their attack to take place in. The units or objectives in the selected tiles then take X amount of damage. * Then the enemy takes their turn.   -Implemented the turnState variable, which would handle the turns  -Added button to transition between the turns |
| Reflection: | -This was more so a work session about planning rather than actually programming, although I did manage to implement simply clicking on units. Admittedly, the way I did it wasn’t clean, simply because of a lack of understanding about method initialization, but I managed to get it working. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 16/12/2020 | Done: | -Started working on proper unit movement  Requirements for movement;   * Movement would happen in steps; you click the unit, then click a destination, and the unit should move to that destination * The player shouldn’t be able to move his units to a spot occupied by another unit * The units should move over time, e.g.: 4 fields per second * The units shouldn’t be able to move when another unit is moving * The units should have a max range to travel to, which should be indicated on the map when clicked on them. * The units shouldn’t be able to move through walls or other units   -Implemented units moving over time |
| Reflection: | -I was really happy when I got this working, the units moved pretty smoothly to their selected destination and everything seemed to be working fine, although implementing collision might be a bit hard using this method. |
| Sources: | -No sources were used. |
| 22/12/2020 | Done: | -Implemented maximum movement range for units  -Added a distance map to log the distances between objects and an array for the world layout  -Added a max range on movement based on unit speed using the distance map  -Implemented indicator for where unit can move to, modified movement |
| Reflection: | -This was pretty simple after already having implemented the movement, and calculating distance was a neat refresher on my Pythagoras’ theorem. |
| Sources: | -No sources were used. |
| 24/12/2020 | Done: | -Planned out movement |
| Reflection: | -I admittedly didn’t do much today, I just researched how I would implement movement collision with terrain and decided to put it off for a while as it seemed a little too difficult at the time. Instead I decided to work on the attack phase. |
| Sources: | -Links to sources regarding dynamic lighting, which is the same idea as how I wanted the movement to work at this point; only, for my game, it’s infinitely simpler to implement. (I ended up not using this method)  <https://www.reddit.com/r/gamedev/comments/bpqqdg/2d_dynamic_point_light_c/> <https://www.youtube.com/watch?v=6B1IA_Tera4> |

|  |  |  |
| --- | --- | --- |
| 26/12/2020 | Done: | -Added enemy units  -Added attack phase during turns  -Added three different attack patterns and implemented visual indicators on the map  -Implemented attacking in the direction of the cursor  -Added attack cancel button |
| Reflection: | -The attack function gave a lot of bugs at first, but I ended up making it work pretty decently, although the method I used wouldn’t allow for any attacking other than straight in front of the unit, so Area of Effect attack were out of the question. The simple workaround to this issue was to not have any Area of Effect attacks in the game. 😉 |
| Sources: | -Source to the stackoverflow page about returning 2D arrays: <https://stackoverflow.com/questions/8617683/return-a-2d-array-from-a-function>  -Youtube video about passing 2D arrays: <https://www.youtube.com/watch?v=QEKmS221MtM>  -Stackoverflow page about setting an entire array to 0: <https://stackoverflow.com/questions/5636070/zero-an-array-in-c-code> |
| 28/12/2020 | Done: | -Cleaned code and added various variables to a gameState class |
| Reflection: | -I added a lot of variables which make more sense to be in the gameState class to the gameState class, rather than have them be global variables. At this point, I added the layout array, the unit placement array, and distance map to the gameState object. |
| Sources: | -Source for how multidimensional arrays work in C/C++ ( in regards to the gameState class, which would use a multidimensional array to store the level layout) <https://stackoverflow.com/questions/1946830/multidimensional-variable-size-array-in-c> |
| 14/01/2021 | Done: | -Updated unit movement code to accommodate for both friendly and enemy units |
| Reflection: | -To point out the elephant in the room; it’s been a while since the last update. I’ve been busy with both school and work, so I haven’t had much time. But, throughout many small updates during this time, I managed to optimize the movement code to accommodate for both friendly and enemy units. |
| Sources: | -Reference for how pointers work in C++  <https://www.youtube.com/watch?v=iChalAKXffs> |

|  |  |  |
| --- | --- | --- |
| 01/02/2021 | Done: | -Modified unit draw code  -Started development on blocking movement to spaces behind walls |
|  | Reflection: | -Again, it’s been a long time due to school and work obligations, but I’m looking to get to working at a normal pace again soon. Developing code for collision was rough; I never seemed to get it working correctly, so I pretty much gave up for the day in the end. I attempted to calculate the line behind the wall tiles by getting the normalized direction vector between the unit and the wall tile, and clearing out the possible tiles behind this. This would however only cover a thin line, and covering the rest ended up being really hard. |
|  | Sources: | -Reference used for vector normalization  <https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-vectors/a/vector-magnitude-normalization>  -Reference for float conversion from integer division  <https://stackoverflow.com/questions/16221776/why-dividing-two-integers-doesnt-get-a-float> |

|  |  |  |
| --- | --- | --- |
| 06/02/2021 | Done: | -Modified code for blocking movement to spaces behind walls  -Reworked entire movement code, now units move 1 square at a time  -Added water and objective tiles  -Added unit death  -Added 3 unit types with different forms of movement (Grounded, naval, aerial)  - Modified unit interaction with terrain based on movement style |
| Reflection: | -I’m really proud of this new movement method, it simultaneously removed any math I had to do, gave the player more control over the movement, and made everything feel simpler. |
| Sources: | -Reference for dynamically sized arrays in C++  <https://www.geeksforgeeks.org/variable-length-arrays-in-c-and-c/>  <http://www.cplusplus.com/forum/beginner/1601/> |

|  |  |  |
| --- | --- | --- |
| 08/02/2021 | Done: | -Added unit direction for spriting  -Changed gamestate constructor |
| Reflection: |  |
| Sources: | -Reference for passing parameters in a constructor  <https://stackoverflow.com/questions/9426932/how-do-i-pass-an-array-to-a-constructor>  <https://stackoverflow.com/questions/9802341/constructor-and-initialization-of-custom-classes-objects>  <https://en.cppreference.com/w/cpp/language/constructor>  -Reference used for the change in the constructor of the gameState object, regarding the usage of pointers as constructor parameters.  <https://www.youtube.com/watch?v=iChalAKXffs> |
| 09/02/2021 | Done: | -Modified gamestate object  -Modified the gamestate constructor  -Changed attributes to private and added accessor/mutator methods  -Added comparator method to gamestate  -Changed the tile size to a global variable rather than one in the gamestate |
| Reflection: | -At this point I feel like I should have done a lot more research into how I wanted to set up the gameState object. I’ve changed a lot of aspects of it, which has required me to rewrite a lot of code (method calls, etc) a lot. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 12/02/2021 | Done: | -Modified the gamestate constructor  -Created array of levels  -Toyed with the idea of storing levels in files  -Made temporary level changing controls on keybinds  -Made list of things left to add;   * AI * Spritework * UI |
| Reflection: | -Again, I modified the gameState constructor, and I feel like it’s going to change even more in the future. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 13/02/2021 | Done: | -Added backtrack function to movement which stores previous moves |
|  | Reflection: | -Implementing this was actually surprisingly easy. |
|  | Sources: | -No sources were used. |
| 14/02/2021 | Done: | -Mapped out the AI’s logic   * Find target based on parameters * Calculate a position in which the AI is attacking the target * Find path to position * Move to position   -Created character designs for potential world villains |
| Reflection: | -Not much happened today, I made a map of how the AI should act and drew some characters which would have dialogue. I was really excited for the characters I ended up drawing however. |
| Sources: | -No sources were used. |
| 15/02/2021 | Done: | -Started development of AI targeting selection according to certain parameters |
|  | Reflection: | -The targeting for the AI was incredibly obnoxious to work on; I kept getting so many bugs, all the time, which was really infuriating to work with, even at the end of the day there were a bunch of bugs I had that wouldn’t go away. Mostly simple bugs, like the units suddenly going a random direction, or the units targeting wrong units, but annoying bugs nonetheless, which kept coming back. |
|  | Sources: | -Sources used for reference in regards to dynamically sized arrays  <https://www.geeksforgeeks.org/variable-length-arrays-in-c-and-c/>  <https://stackoverflow.com/questions/15013077/arrayn-vs-array10-initializing-array-with-variable-vs-real-number>  <https://stackoverflow.com/questions/13431567/how-to-make-an-array-with-a-dynamic-size-general-usage-of-dynamic-arrays-maybe> |

|  |  |  |
| --- | --- | --- |
| 16/02/2021 | Done: | -Removed distanceMap, which logged the distance of each tile for a selected unit, and threatenedplaces, which logged all places a unit could attack, out of the gamestate class |
| Reflection: | -Yet another update to the gameState class, but considering these 2 arrays weren’t dependent on which level, rather they were calculated and replaced constantly, so it made more sense for them to be stored only once, rather than for every gameState object. |
| Sources: | -No sources were used. |
| 18/02/2021 | Done: | -Created function which finds all tiles a unit could possibly attack on the field for every single position it could move to, and save it in a 2D array.  -Modified function which checks if a unit can move to a certain location |
| Reflection: | -This actually came pretty easily considering the fact that it’s decently complicated, which I’m happy with. |
| Sources: | -No sources were used. |
| 19/02/2021 | Done: | -Modified attacking function to only store the coordinates which could potentially attack a unit or objective  -Modified the function which finds all possible tiles a unit could move to in order to attack another unit  -Updated distance calculation |
| Reflection: | -There was one annoying technicality with the distance calculation; due to the way I wanted this to work, I couldn’t use Pythagoras to calculate distance as I did previously, as it wouldn’t accurately translate (to simplify, when using Pythagoras, the edges of the border of the maximum movement distance were more rounded than they should have been), so I implemented a pretty lengthy workaround. |
| Sources: | -No sources were used. |
| 21/02/2021 | Done: | -Updated distance calculation again  -Modified the function which finds all possible tiles a unit could move to in order to attack another unit  -Added start of a code to calculate a path to move to the target |
| Reflection: | -This was essentially just a small fix on the distance calculation, alongside with a start to the pathfinding code. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 22/02/2021 | Done: | -Started pathfinding algorithm, however it didn’t actually store the correct path yet |
| Reflection: | -At this point the AI is able to properly find a path to its target, but the path is already calculated (which I was able to confirm by logging the coordinates to console). This was actually decently straight forwards to implement, especially after I found the video which detailed some examples of pathfinding algorithms. |
| Sources: | -Source used as reference for the algorithm which I ended up using  <https://youtu.be/KiCBXu4P-2Y> |

|  |  |  |
| --- | --- | --- |
| 23/02/2021 | Done: | -Finished the pathfinding algorithm, adding code to store the actual path  -Added part of code which removes the squares in AIPmoves which the AI, calculated through its movement and using the new distance calculation  -Made it so the AI can’t walk through walls anymore, rather paths around it |
| Reflection: | -Adding the code that saves the correct path to the existing pathfinding algorithm code was very much a complicated process. It took me quite a while to figure out how to even handle it, but I ended up making it work by storing every single path it calculates up until the moment it finds the correct path, at which point it deletes all of the paths stored except for the correct one, which it saves. Additionally, I added the AI’s ability to detect walls and actually path around them. |
| Sources: | -Sources used for referencing how vectors work, in regards to saving the path the AI should take.  <https://stackoverflow.com/questions/42249303/how-can-i-push-back-data-in-a-2d-vector-of-type-int>  <https://stackoverflow.com/questions/41325425/multidimensional-vectors-in-c/41325471>  <http://www.cplusplus.com/reference/vector/vector/>  <https://www.cplusplus.com/forum/general/197176/> |
| 26/02/2021 | Done: | -Put the actual AI together in 1 function which governs the AI   1. The AI finds a target 2. The AI looks which tiles the player can move to and targets the closest one 3. The AI calculates a path to this tile |
| Reflection: | -I knew putting it together wouldn’t be easy and would cause a load of bugs, which it ended up doing. But, I eventually ironed most of them out and by this point the AI was (mostly) working. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 28/02/2021 | Done: | -Finished AI and made it activate on enemy turns |
| Reflection: | -I did some final bugfixing and implemented the AI’s activation on enemy turns by checking the turnState variable in the game loop, and activating this every second on enemy turns. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 01/03/2021 | Done: | -Changed the tile size and updated all the graphics |
| Reflection: | -Small update to the graphics, which did lead to some funny visuals when trying to change the size because of the textures bugging out. |
| Sources: | -No sources were used. |
| 12/03/2021 | Done: | -Added menu  -Added variable which checks if player is in game or not  -Made some attributes of objects public rather than private (variables which were previously set to private but had both accessors and mutators, making them being private redundant) |
| Reflection: | -First things first; I haven’t been able to work too much the last few weeks due to exams, regrettably. That being said though; the menu was pretty easy to implement, although I’m sure there’s not a more optimal way of implementing it that isn’t just adding a giant if statement around the game loop which shows checks whether the player is in game and shows the menu if they aren’t. I assume there is, but I couldn’t, at the time, think of one. However, what was more annoying to implement was changing a lot of private attributes which had both normal get and set methods to public fields. I had to change around a lot of code. |
| Sources: | -Paper I read regarding the conventions surrounding private and public attributes and accessor and mutator methods. (I read some other sources on this, but this was the main one)  <http://www.idinews.com/quasiClass.pdf> |

|  |  |  |
| --- | --- | --- |
| 16/03/2021 | Done: | -Added buttons to select level and world  -Added level initialization in the Init function |
| Reflection: | -Not much to say, honestly; I attempted to load in the levels through reading the variables from a file, after which I’d convert them into variables. This ended up not working at all, so I opted to pretty much leave it at what it was for now. |
| Sources: | -Sources about string manipulation, which I used in the first methods of initializing the levels, but ended up changing.  <https://stackoverflow.com/questions/14265581/parse-split-a-string-in-c-using-string-delimiter-standard-c>  <https://www.fluentcpp.com/2017/04/21/how-to-split-a-string-in-c/>  <https://www.techiedelight.com/split-string-cpp-using-delimiter/>  -Source for parsing strings to integers in C  <https://stackoverflow.com/questions/4442658/c-parse-int-from-string> |

|  |  |  |
| --- | --- | --- |
| 17/03/2021 | Done: | -Added images of planets to world select buttons |
| Reflection: | -I didn’t do much today aside from adding little planet visuals. I first tried to do pixel art, but my pixel art skills left some things to be desired. Also, considering the visuals of the game aren’t that important in the context we’re making it, I opted to instead use some images of planets I found on google. |
| Sources: | -References to the planet textures used.  <https://commons.wikimedia.org/wiki/File:Populated_Planet.png>  <https://pngimg.com/image/61161>  <https://pngio.com/images/png-a44054.html>  <https://commons.wikimedia.org/wiki/File:Lava_planet.png>  <https://blenderartists.org/t/space-background/660743> |
| 20/03/2021 | Done: | -Added dialogue screen  -Added textures for menu items (e.g.: buttons) |
| Reflection: | -I drew up a bunch of characters and made a dialogue menu which I’d use to explain how the game works. I tried adding a custom font, but found that either the template has no built in function for adding fonts or just doesn’t have the capabilities of adding fonts, so I just chose to use the default writing method which does come with the template. |
|  | Sources: | -Pixel art template which I used for the UI.  <https://free-game-assets.itch.io/free-tds-game-ui-pixel-art> |

|  |  |  |
| --- | --- | --- |
| 22/03/2021 | Done: | -Changed all rendering to get more fps, as the drawscaled function that came with the template reduced the fps a lot. |
| Reflection: | -Using fraps really showed me some performance issues, which all originated from the drawScaled method, so I changed around a lot of drawing functions to just using the regular drawing method. Yes, this would mean you can’t rescale the window properly, but seeing as that wasn’t implemented anyways I don’t think this is too big of a problem, especially if it saves me upwards of 30fps. |
| Sources: | -Sources used when attempting to use memcpy to render the sprites, which ended up not working out.  <https://stackoverflow.com/questions/31673359/understanding-the-copy-done-by-memcpy>  <https://stackoverflow.com/questions/56514669/understanding-why-it-doesnt-copy-correctly-using-memcpy>  <https://stackoverflow.com/questions/9688454/cropping-an-area-from-bitmapdata-with-c-sharp/9691388>  <https://www.gamedev.net/forums/topic/694264-directx-9-with-c-how-can-i-draw-an-image-transparent/> |

|  |  |  |
| --- | --- | --- |
| 03/04/2021 | Done: | -Updated AI so it doesn’t get stuck behind walls anymore  -Updated level generation, levels are now stored in files |
| Reflection: | -Testing around with designing a few levels with more complicated wall structures, the AI seemed to suddenly get stuck on walls again. I discovered this was because the target which was set with the targeting code gave a target which was out of reach for the unit. Normally, the code would then generate a path to the closest square the unit can reach, but it didn’t do that so I had to go in and fix this. It ended up being taking a bit longer than I would’ve wanted it to, but it worked out in the end. |
| Sources: | -Sources referencing JSON, which I initially attempted to use to save the files.  <https://www.w3schools.com/js/js_json_arrays.asp>  <https://www.youtube.com/watch?v=P2LdhPpqSiU> |
| 05/04/2021 | Done: | -Updated level generation again, optimizing and simplifying level loading |
| Reflection: | -I was at a bit of a loss with this one, especially when deciding what file format would be best applicable for what I was using. Eventually I ended up just using txt files, but I think the way I did it is actually pretty clever; it scans the file for numbers arranged in a 9x9 grid (the size of the levels). Based on these numbers, it generates a grid out of different tiles. However, when it reads something else than a number, it generates either a unit or an object, after which I added the number to signify what the unit was standing on. |
| Sources: | -No sources were used. |

|  |  |  |
| --- | --- | --- |
| 06/04/2021 | Done: | -Scrapped dialogue screens  -Added health bars for units and objectives  -Added end of turn effects  -Made level layouts for all levels  -Changed textures for enemy ants  -Commented all code |
| Reflection: | -I deemed the dialogue concept a bit unnecessary and decided to scrap it completely at this point. It’s a shame, because I drew up some art for it, but I just thought I’d be a bit overkill. Additionally, I cleaned up the code a lot, commented everything, and added health bars to the units. At this point I’m basically done with the game. The only thing I could see myself still adding is animations for attacking, but I feel like spriting that would be a lot of work and I’ll be pretty busy with school for the following weeks due to exams. Either way, at this point, I consider the game finished, as it’s now perfectly playable. One last point I have to make; I don’t know how difficult any of the levels are, or if they’re even all possible. I have about 24 levels in the game, but I haven’t played all of them. I just made them up as I went along. I know, I could probably do better, but for the purposes of this game, designing intricate and complicated levels is really unnecessary in my opinion. |
| Sources: | -No sources were used. |

## Finished game- review

I ended up really liking the little game I came up with. Admittedly, it’s got a lot of flaws. The gameplay isn’t all that solid from a design perspective, it doesn’t look great visually, it’s got no real pause menu, no system to pause the game, etc. All things I wanted to implement at some point, but I got really busy with work and school near the end, which resulted in me calling the game finished and leaving it at what it is. That being said though; I’m really quite proud of the pathfinding algorithm and the level loading.

In the very beginning I made some hypotheses about how the project would go, establishing some in scope and out of scope goals, alongside laying out difficulties I thought to find along the way. I will go through every one of these, and comment on it now the game is finished.

### In-scope goals:

**-Isometric turn-based strategy game, where you compete against an AI controlled enemy.**   
This one I essentially had to finish, because this is the basis of the game itself.

**-You control various units, different fighters, scouts, and a queen which can produce more units (up to a cap)**  
This one is pretty interesting; instead of a queen, I implemented the objective tiles, and instead of using a system like scouts and fighters I implemented different movement patterns and attack patterns. That being said though, the idea stayed the same.

**-Different mission types: destroy cities, capture objectives, destroy all enemy units, etc**  
This actually surprised me when I read this back; I had totally forgotten that this was one of the in scope goals I wrote for this project. I didn’t get to implement this, which is a shame, but not the biggest problem in the world.

### Out-of-scope goals:

**-Different types of enemy factions, which all have different powers and pose different challenges to the player**  
This one I did actually implement; although not as in detail as I wanted it to be. I wanted to create a whole fantasy universe with characters, but ended up scrapping most of that.

**-Upgrades to the player’s units, or different units the player can select. (different types of ants, long ranged ants vs melee ants, etc)**  
This one I didn’t implement at all, which is pretty sad, but then again; this might’ve brought me too close to the border between inspired by and ripping off Into the Breach.

**-Randomly generated challenge, where you face off against a randomly generated map of enemies for rewards.**  
This one, again, I didn’t implement at all. Looking back, I know how I’d implement this, probably using noise on an image and downscaling it, but that would’ve been very complicated and this was definitely a reasonable out of scope goal.

### Expected Difficulties

**-Usage of C++. I’m personally more used to higher-level programming languages, such as javascript or java, so adopting C++ is going to be a bit of a struggle.**  
While I’d used C++ before, it definitely wasn’t my strong suit when starting this project. Now, about half a year later, and I love the language. It was a bit of a learning curve, but I think I managed to get there in the end.

**-Implementing the enemy AI. I have never really tackled AI before (or at least not written one myself), so this will be quite difficult. I’m expecting this to be my biggest hurdle, but we’ll see.**  
This one was pretty difficult, but also really fun and exciting to work on. While calling it an AI might be a bit presumptuous, I’m really happy with how it turned out.

All in all; this was a really fun project, and even if I don’t get into BUas, I’m glad I had the experience and will definitely continue to make games. Even if I don’t get in, I’ll keep working on various hobby games by myself, and will probably end up working in the industry anyways.