Progress Report

# Abstract

This project aims to find an efficient solution to cave mapping using multiple drones all initially unaware of their environment. The project will tackle various problems in the field of robotics such as localisation, frontier searching, teamwork and collision avoidance of both static and moving objects. The drones will gradually build an occupancy grid of its environment using built in sensors then proceed to map the entire cave by continually identifying and searching frontier cells which are boundaries between unoccupied areas and unknown areas.

Contents

[Abstract 1](#_Toc531032286)

[1. Introduction and Motivation 2](#_Toc531032287)

[2. Objectives 3](#_Toc531032288)

[2.1 Functional 3](#_Toc531032289)

[2.2 Non-Functional 4](#_Toc531032290)

[3. Project Overview 5](#_Toc531032291)

[4. Project State 6](#_Toc531032292)

[4.1 Completed/Ongoing objectives. 6](#_Toc531032293)

[4.2 Mapping 7](#_Toc531032294)

[5. Research 8](#_Toc531032295)

[6. Project Management 8](#_Toc531032296)

[6.1 Timetable 8](#_Toc531032297)

[6.2 Risks 9](#_Toc531032298)

[6.3 Methodology 9](#_Toc531032299)

[6.4 Testing 10](#_Toc531032300)

[Glossary 11](#_Toc531032301)

[References 11](#_Toc531032302)

[Appendices 12](#_Toc531032303)

# Introduction and Motivation

Cave exploration is high-effort, time-consuming and potentially dangerous to humans. This is due to factors such as flooding, hypothermia, tight spaces, falling rocks, harmful gases and particulates such as histoplasmosis from animal droppings [1]. Many caves are too extensive to fully map and are constantly being further explored. This is certainly the case for the Mammoth cave in Kentucky, USA the largest cave network on Earth which contains over 400 miles of cave [2].

This project aims to propose a safer and risk-free method of cave exploration and mapping by simulating a fleet of autonomous flying drones. Each drone will be equipped with capabilities to move in all directions, rotate around its y-axis, avoid collisions, sense its immediate environment, plan its search path and communicate information with other nearby drones. The project will initially be implemented for the 2D case where the environment is viewed from the top-down assuming altitude is constant. Expansion into the 3D case will be an extension to the project. The initial cave environment and each drone’s path will be trackable and viewed in real-time on the same GUI to allow for testing and review of the search algorithms employed.

The project will be split into three main stages during implementation:

1. **Generation and visualisation of the cave environment**

Cave simulations will be created by initially starting with random noise then passing a few iterations of a cellular automaton over it to create realistic cave looking environments. Certain parameters will be changed to allow customisability of the cave environment to ensure testing is more thorough. More complex and realistic caves will be able to be generated using Perlin noise.

1. **Exploration by a single drone**

At each time step the drone will search its local environment using the built-in sensor and add newly discovered information to its locally stored data structure mapping the cave. The drone will then have to calculate the configuration space from the work space by identifying areas near the cave wall which will cause a collision. Additionally, new frontier cells are calculated, and the best cell is chosen to be searched next.

1. **Exploration by multiple drones working together**

With multiple drones the collision avoidance algorithms will be updated to account for non-static entities such as other drones but could also be used for avoidance of cave fauna such as bats. The drones will possess the ability to communicate what they have learnt to other nearby drones, so they can decide on where to search to maximise exploration efficiency. Communication protocols will be employed to exchange exploration information to minimise mapping duplication and help decide where to explore.

Finally, this project is not solely limited to cave exploration, it may be modified to have applications with other real-world problems involving the use of multiple drones in an unknown environment such as: mass deliveries, rescue operations, cleaning and crop harvesting.

# Objectives

The overall objective of the project is to find a good model for cave exploration with multiple drones such that there is a significant increased efficiency with utilising more than one drone. Each objective has been given a label noting: i) if it is a core or stretch goal, ii) the priority and iii) the dependencies.

## 2.1 Functional

1. To be able to create realistic cave environments with a cell resolution of 10cm by 10cm and maximum size of at least 250m by 250m (i.e. max cave dimensions of 2500x2500). The representation of the cave should have different values to represent different types of cells (e.g. unoccupied and occupied) to be used by the visualiser and for statistical analysis.

(Core, High, Independent)

1. Allow cave creation customisability for the following parameters: Maximum environment size in all dimensions, amount of open space, jaggedness of cavern walls, size of tunnel areas, length of cavern *sections and variability/variety of cave.*

(Core, Medium, Dependant on 1)

1. Ability to view the generated cave environment on a GUI.

(Core, High, Dependant on 1)

1. Drones should be able to move in all directions when given a request and possess the ability to search its local environment up to a specified range.

(Core, High, Independent)

1. Drones should be able to detect and avoid collisions with the cavern walls.

(Core, High, Dependant on 4)

1. Drone should be able to locally store known areas of the cave environment. The drone object will use a quadtree data structure to store occupied cells.

(Core, High, Dependant on 4)

1. The drone will be able to calculate frontier cells and use an algorithm to decide the optimal frontier cell to explore next.

(Core, High, Dependant on 4,6)

1. The drone should be able to track its path from its starting location, so it can be used when exploration is complete and shown visually.

(Stretch, Low, Dependant on 4,6)

1. In instances with multiple drones, collision avoidance techniques should be used to avoid drone-to-drone collisions.

(Core, High, Dependant on 4)

1. When multiple drones are nearby they will communicate mapping information to avoid search repetition, increasing efficiency. Both drones will use the received communication to update their locally stored map.

(Core, High, Dependant on 4)

1. Display search statistics in real-time and at simulation completion such as: Total time taken to explore the cave, percentage/area explored by each individual drone and ratio of explored areas to unexplored areas mapped over time. (Stretch, Low, Dependant on 3, 4, 8)
2. Extend the project to be able to create environments and simulate drone behaviour in 3D.

(Stretch, Medium, Dependant on all previous objectives)

1. Assume the drones cannot localise themselves with 100% accuracy and so use localisation techniques with the sensed data to localise themselves in the environment accurately.

(Stretch, Low, Dependant on 4, 6, 8)

## 2.2 Non-Functional

1. To be able to create realistic cave environments in less than 10 seconds.
2. The searching simulation should run close to real-time speed.
3. To be able to view informative data on the environment and drones within the GUI.
4. GUI should be responsive, usable and aesthetically pleasing.

# Project Overview

The project will be written in the C++ language using Microsoft Visual Studio Community 2017 [3] on Windows machines and Atom [4] as the text editor for Linux. Visualisation will be achieved using GLUT [5]: an OpenGL utility toolkit which contains libraries written in C++ and will allow the creation of a GUI to showcase the cave and drones. Git [6] will be used for source-code management and version controlling. GitHub [7] will allow the Git repository to be stored remotely allowing work to be completed and transferred at home and at university, whilst also ensuring the project is backed up.

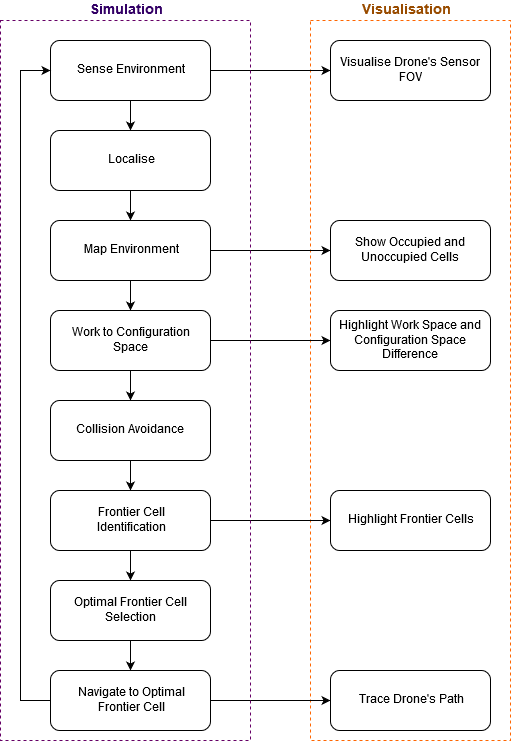


Figure 1 - Single Drone - Searching Process with visualisation processes.

# Project State

## 4.1 Completed/Ongoing objectives.

Partial progress has been made on the cave generation and visualisation which encompasses objectives 1 and 3. As you can see from the screenshot, the cave network is well connected, but does lack variety in its structure. All paths are of similar length, width and size. Some progress has been made for objective 2 to alter some cave parameters however additional research will be required for this as the complexity was initially not fully anticipated. Perlin/Simplex noise is a possibility for adding complexity to the cave structure and making it look more realistic [8].



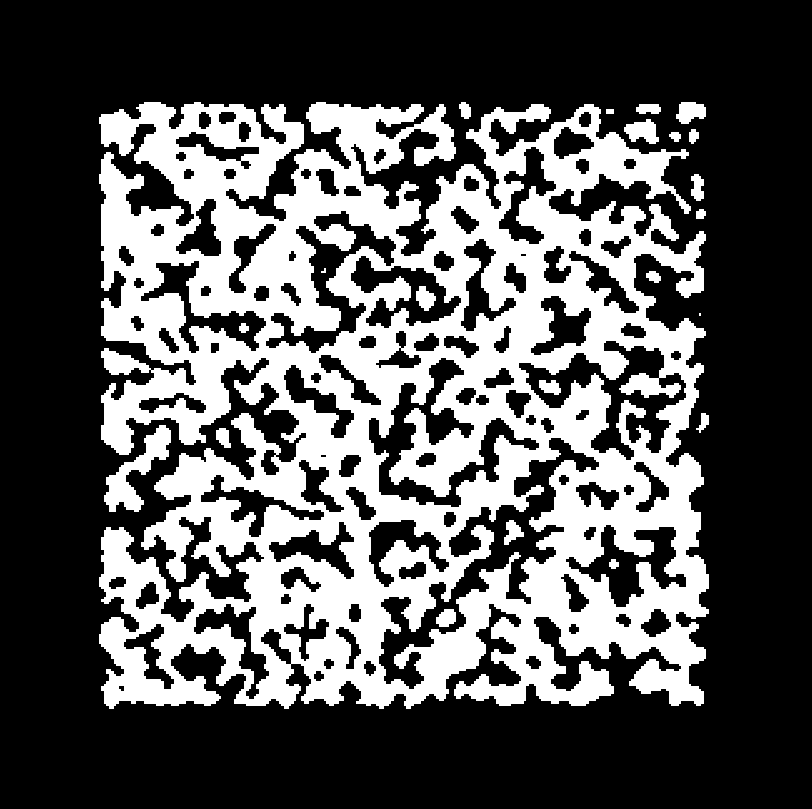


Figure 2 - Cave generation prototype zoomed in.

Figure 3 - Cave generation prototype, unoccupied cells are represented in white. 250x250 array.

Currently cave generation uses a naïve approach by generating a random array of values and thresholding each by a set amount known as the “fill percentage”. Then a cellular automata rule is applied across the entire cave a few times to smooth and generate the eventual “cave” texture.

The visualisation window has functionality for zooming in, zooming out and moving in both the positive and negative x and y directions achieved by binding keyboard key detections with GLUT.

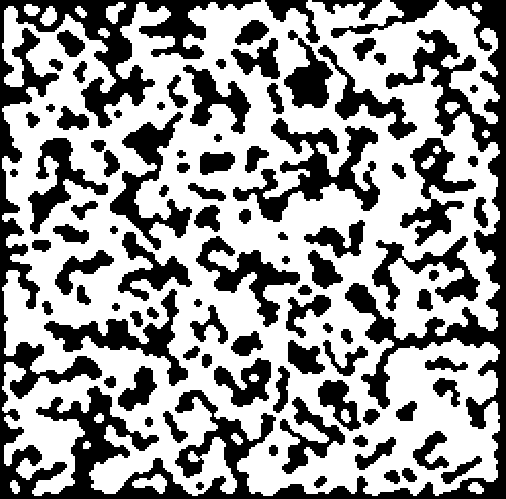
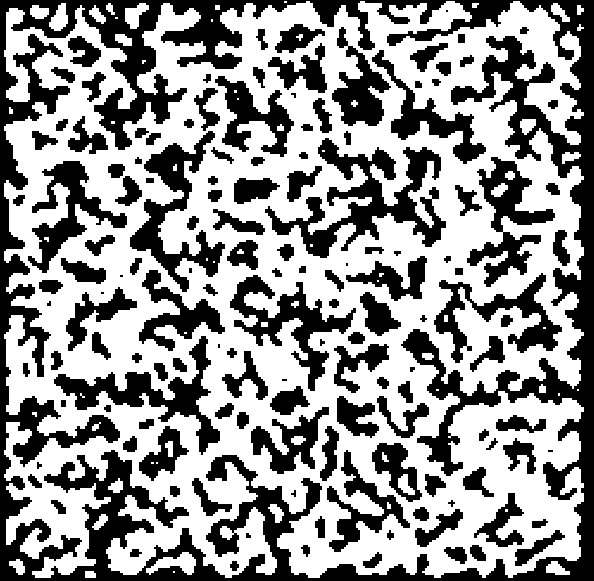
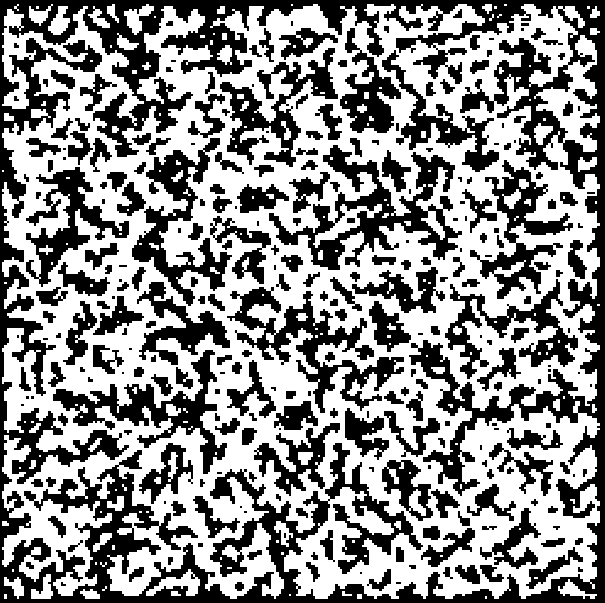
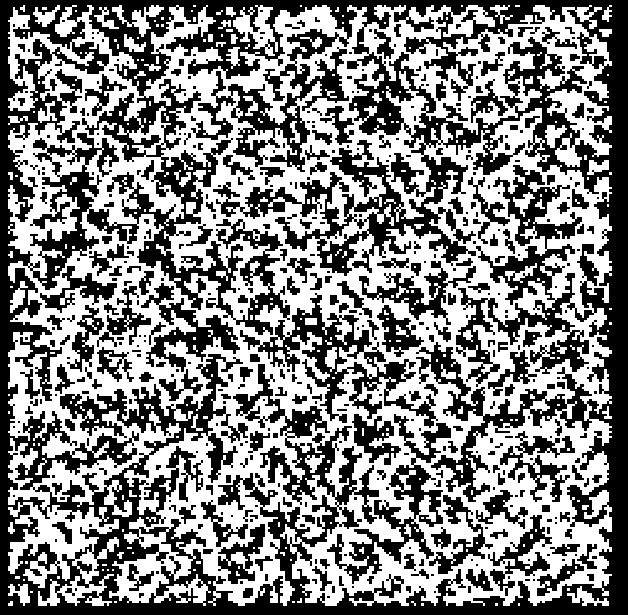
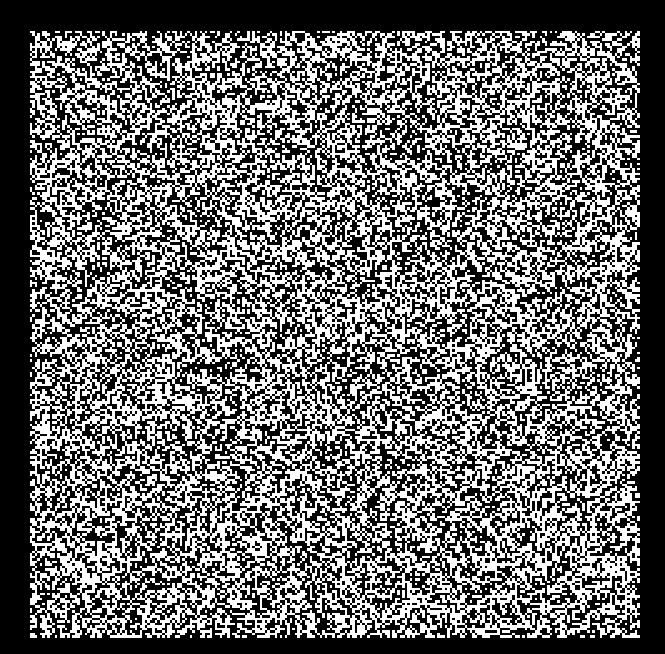


Figure 4 - From left to right: Initial Noise, 1 Smooth pass, 2 Smooth passes, 5 Smooth passes, 20 Smooth passes. All for a 250x250 cave.

For each smooth step: If a cell had more than 4 occupied neighbours there was a 100% chance to become occupied. On the other hand, if a cell had less than 4 occupied neighbours there was a 75% chance to become unoccupied.

## 4.2 Mapping

Various methods and data structures have been researched and compared to be used for each drone to map the environment. These include: 2D array, Quadtree, Spatial hashing [9] and Binary space partitioning [10]. Quadtrees have been selected to store occupied cells and frontier cells for each drone for the following reasons:

1. Reduces memory allocation at the start.
2. Expandable at runtime.
3. Combination of sparse and dense regions will not affect the data structure’s performance. Particularly good for cave environments where there will be large open areas (sparse) and cavern walls (dense).
4. Collision detection is more efficient as a small subset of nodes are checked.
5. Allows extensibility into 3D visualisation as mesh generation from quadtrees has been researched. [11]

# Research

There are various aspects of the project which I have identified that require further research including but not limited to:

* Cave generation using Perlin/Simplex noise:

Perlin noise is a well-researched and documented method for creating procedurally generated environments [8]. It can be used to build upon the naïve random noise method currently in use to add sparsity and realism.

* Collision avoidance:

Avoidance of cave walls can be calculated by defining a secondary border between the drone and the wall with a distance equal to the size of the drone. Starting with work space and assuming the drone is a point, configuration space can indicate where the drone can navigate to without collisions [12].

* Visualisation optimisations:

For example, drawing less polygons by combining groups of dense polygons to reduce the number of vertices.

* Tools to analyse efficiency and effectiveness of solution:

Using flood fill to calculate the maximum area a drone could explore.

* Localisation.
* Frontier searching.
* Optimal searching with multiple drones.
* Representations of the cave.

# Project Management

## 6.1 Timetable

The first version of the timetable can be found in the project specification. An improved timetable has been created, expanding stages and better accounting for free time during the Christmas break and term 2.

More time was dedicated for the cave generation stage (green), as well as for the single drone stage (blue) and teamwork stage (red). Each of these stages were also further split up into sub-stages to further improve time-management. The improved timetable can be found as Figure 5 in the appendices section.

## 6.2 Risks

In addition to the risks highlighted in the project specification the following additional risk has been identified. Cave generation has a possibility to slow down dramatically as the cave size increases and as increasingly complex features are integrated into the cave structure. For example, on a computer running an Intel i7-6700k CPU and NVIDIA GeForce GTX 970 GPU, cave generation runs with the following usages when repeatedly zooming in and out of the visualisation to force redraws.

|  |  |  |
| --- | --- | --- |
| **Cave Size** | **CPU Usage** (i7-6700k) | **GPU Usage** (GTX 970) |
| 50x50 | 17% | 0.8% |
| 250x250 | 18% | 9% |
| 1000x1000 | 30% | 35% |

Generation of caves sizes 50x50 and 250x250 ran smoothly on this particular setup, however a cave of size 1000x1000 ran noticeably slower with lag. This is not sufficient as for a 500mx500m map: 250,000 cells are generated and will need to run fast to satisfy objective 1. A cave of dimensions 1000x1000 will have a maximum of 4,000,000 vertices drawn to the screen, this number should be kept as low as possible as it could serve as a GPU bottleneck [13].

To compensate the risk, I will explore various methods and algorithms to optimise drawing cells onto the screen by grouping them together and store the vertices instead of recalculating on each draw command.

## 6.3 Methodology

An Agile approach will allow easier re-organising of certain sub-stages of the project. For instance, if cave generation takes too long, the individual drone searching stage which occurs next chronologically can still go ahead on time using a simpler than excepted generated cave. Then complex cave generation can be implemented later in development and this should not affect code which should have occurred chronologically afterwards.

At the end of each sub-stage in development which can be found on the improved timetable (Figure 5) a small amount of time will be dedicated to testing that the sub-stage works with the previous stages and as intending in the project objectives. This is to ensure problems do not cascadeunintentionally to the end of the project development and that there is a continual working product at the end of each stage.

## 6.4 Testing

Due to the graphical nature of the project and the visualisation, many aspects of the project will be easily tested by using the GUI. For example, errors in cave generation can easily be identified from viewing the displayed cave on the GUI. An example of this has already occurred when the orthographic view volume was set too small and no shapes were shown. Using the GUI, the view volume was set correctly so the cave could be viewed.

In the cave generation stage a set of unique cave environments will be used for testing purposes. The parameters of the generator and seed value will be stored to be able to reproduce them later.

# Glossary

|  |  |
| --- | --- |
| *Frontier cells* | Unoccupied cells that are adjacent to unknown cells. |
| *Localise* | Process of a robot identifying where it is in an environment. In the context of this project the robot will localise using multiple scans of the environment at different positions. |

# References

|  |  |
| --- | --- |
| [1] | “Histoplasmosis,” [Online]. Available: https://www.healthline.com/health/histoplasmosis. |
| [2] | “Mammoth Cave National Park,” [Online]. Available: https://www.nationalgeographic.com/travel/national-parks/mammoth-cave-national-park/. |
| [3] | Microsoft, [Online]. Available: https://visualstudio.microsoft.com/. [Accessed 20 November 2018]. |
| [4] | “Atom,” [Online]. Available: https://atom.io/. [Accessed 10 November 2018]. |
| [5] | “GLUT - The OpenGL Utility Toolkit,” [Online]. Available: https://www.opengl.org/resources/libraries/glut/. [Accessed 20 November 2018]. |
| [6] | [Online]. Available: https://git-scm.com/. [Accessed 10 November 2018]. |
| [7] | [Online]. Available: https://github.com/. [Accessed 10 November 2018]. |
| [8] | Wikipedia, “Perlin Noise,” [Online]. Available: https://en.wikipedia.org/wiki/Perlin\_noise. [Accessed 20 November 2018]. |
| [9] | E. J. Hastings, J. Mesit and R. K. Guha, “Optimization of Large-Scale, Real-Time Simulations by Spatial Hashing,” [Online]. Available: http://www.cs.ucf.edu/~jmesit/publications/scsc%202005.pdf. [Accessed 22 November 2018]. |
| [10] | Wikipedia, “Binary space partitioning,” [Online]. Available: https://en.wikipedia.org/wiki/Binary\_space\_partitioning. [Accessed 23 November 2018]. |
| [11] | M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, “Quadtrees - Non-Uniform Mesh Generation,” in *Computational Geometry Algorithms and Applications (3rd Edition)*, 2008, pp. 307-315. |
| [12] | M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, “Robot Motion Planning - Getting Where You Want to Be,” in *Computational Geometry Algorithms and Applications (3rd Edition)*, 2008, pp. 284-290. |
| [13] | “Unity - Manual: Optimizing graphics performance,” Unity Technologies, [Online]. Available: https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html. [Accessed 21 November 2018]. |

# Appendices

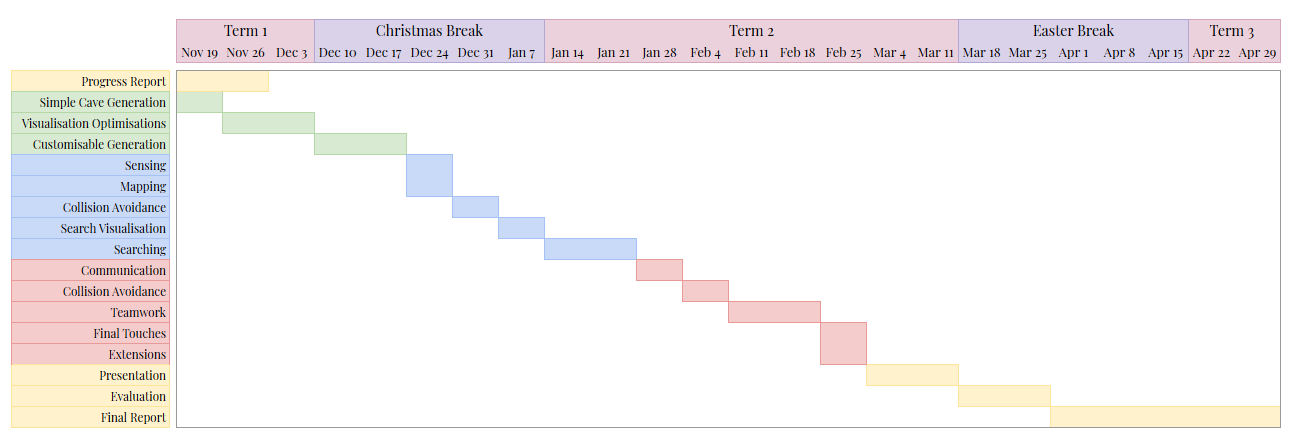


Figure 5 - Improved Timetable