Mixed Reality Tabletop War Game Assistant

Submitted April 2024, in partial fulfilment of

the conditions for the award of the degree:
BSc Hons Computer Science

20363169
School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature: NB
Date: 13/04/24

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.*

Table of Contents

In this document, there are 6385 words all up.

1. Abstract

2. Introduction and Motivation

Tabletop war-gaming is a popular hobby that is quickly gaining popularity, however, with a high barrier to entry, it remains niche and inaccessible to many. The rules to tabletop war-games can be complex and difficult to learn. This can be daunting for new players, putting them off the hobby as well as causing disagreement between seasoned players over rules interpretations.

Some of the most popular war-gaming systems are produced by *Games Workshop* [1]. One of their more popular systems, *Warhammer 40k*, has a core rule-book of 60 pages [2] and the simplified version of another game system, *Kill Team*, is a rather dense three page spread [3]. This complexity can be off putting to new players. Many tabletop / boardgames suffer from a players first few games taking significantly longer than is advertised due to needing to constantly check the rules.

As well as this, new players are likely to take longer to make decisions as they are not familiar with the game's mechanics of what constitutes a "good" move (sometimes referred to as "analysis paralysis"). This can be exacerbated by the game's reliance on dice rolls to determine the outcome of actions. Meaning that seemingly "optimal" moves do not always result in favourable outcomes, causing an extended learning period for an already complex game.

Kill Team is a miniature war-game known as a "skirmish" game. This means the game is played on a smaller scale with only ~20 miniatures on the table at one time. The aim of the game is to compete over objectives on the board for points. Each player takes turns activating a miniature and performing actions with it. These a selection of actions are: moving to another location, shooting at an enemy, melee combat and capturing an objective. The game uses dice to determine the results of your models engaging in combat with each other with the required rolls being determined by the statistics of each "operative" (a miniature) involved and the terrain.

Figure 1: An example of the *Kill Team* tabletop game using the *Gallowdark* terrain. The terrain we will focus on here are the flat walls and pillars. [4]

Video games help on-board new players by having the rules enforced by the game itself. This project aims to bring a similar methodology to tabletop war-gaming, specifically the *Kill Team Lite* [3] system

using the *Gallowdark* setting. The *Kill Team Lite* rules are publicly available from *Games Workshop's* website and is designed to be played on a smaller scale to other war games, making it a good candidate for a proof of concept. As well as this, the *Gallowdark* [5] setting streamlines the terrain used and removes verticality from the game, making implementation much simpler.

Developing a system that can digitally represent a physical *Kill Team* board would allow for the creation of tools to assist players. For example, a digital game helper would remove the burden of rules enforcement from the players and onto the system, allowing players to focus on the game itself or allow players to make more informed game decisions by being able to preview the options available to them. This could also be utilised to record a game and view a replay or be used for content creation to make accurate board representations to viewers with digital effects.

2.1. Project Intention and Background

This project will focus on the development of a system that can track the position of miniature models and terrain pieces on a *Kill Team* board which can subsequently be displayed digitally. From here, we aim to implement proof of concept visualisation of a few select game rules on the digital board to demonstrate that the tracking system provides the necessary information to process said rules.

To determine the specific goals for this project, it is important to provide some more context on the gameplay and community surrounding *Kill Team*.

2.1.1. Gameboard Components

Before looking at the game rules, we will first break down the physical components of the gameboard to determine what needs to be tracked and problems that may arrise from this.

A game of *Kill Team* is comprised of two players, each with a group of "operatives" (miniatures) referred to as a "Kill Team". Each Kill Team has it's own unique rules, with each operative having it's own set of unique statistics and special abilities. A miniature is comprised of a circular base with a model placed on top. The diameter of the base is either 25mm, 28.5mm, 32mm, 40mm or occasionally 50mm.

Figure 2: An example of a *Kasrkin Kill Team* operative on a 28.5mm base [6]. The height of the operative is ~35mm exclusing the base. This project will focus on getting the system functional with the *Kasrkin* models on 28.5mm bases.

It is worth noting that is is common for models to extend past the edge of their base as seen in Figure 2. Whilst the example above is somewhat minor in it's overlap, different models can be more extreme in their extension. As a result of this the system will need to be able to locate and identify an operative from only a partial view of the base, or the surroundings of the base, from a bird's eye view. The detection system will either need to be above the board if using visual detection methods or below the board if using, for example, an RFID method. This is due to the terrain potentialy surrounding an operative, so having a camera on each side of the board will not guarantee it is visible.

The game is played on a 30″ by 22″ board, referred to as a "Killzone". The *Gallowdark* ruleset instead uses a 27″ by 24″ board with the specialised terrain shown in Figure 1. As stated previously, the terrain we are focusing on here are the thin walls with pillars connecting them. The terrain is all at the same height, but the operatives are shorter than the terrain. As a result, the system will need to find a solution to detect operatives behind terrain. From a birds eye view, the terrain will block part of the operative due to parallax. For this project, we will focus on using a half sized board to simplify this problem.

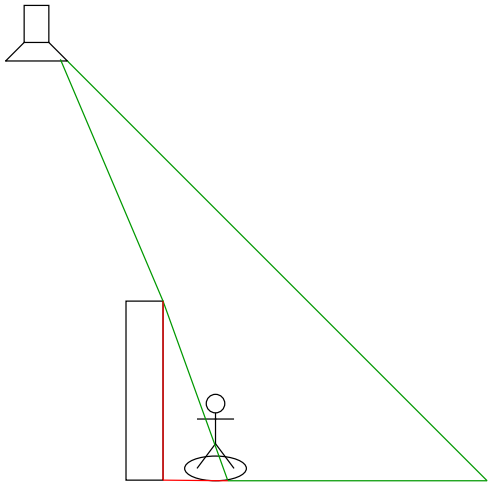

Figure 3: An example of the problem introduced by parallax. The camera is placed above the board offset from the operative. The operative is partially obstructed by terrain. The green triangle represents the parts visible to the camera. The red lines demonstrate the areas blocked by the terrain.

The further away the camera is from the operative on the x axis, the more the terrain will block the operative. However, as the camera travels away on the y axis, the terrain will block less of the operative. For a camera implementation this would also mean we lose quality in the image. As a result, a camera based system would need to find an ideal distance from the board which provides enough quality in the image, whilst also being able to see enough of an operative if it is obscured by terrain.

2.1.2. Gameplay

The gameplay of Kill Team focuses around a turn based system with a full game consisting of 4 rounds - called "turning points". Each player takes turns "activating" a single operative and performing actions with it. The number of actions available to an operative is defined by it's statistics[1]. Once an operative has performed it's maximum number of actions play is handed to the other player. That same operative may not be "activated" until every other operative, still in play, on their team has been activated. Once every operative on both teams has been activated the next turning point begins resetting the activation of each operative still in play. Once all 4 turning points have been completed the game ends and a winner is decided.

This project is focussed on creating a digital representation of the board, and then implementing a few select rules from the game to demonstrate the system's capabilities. The rules for "movement" and "shooting" present themselves as good candidates for this due to both their complexity and frequency within the game.

2.1.2.1. Movement

There are two types of movement available: "normal move" and "dash". A normal move allows an operative to move a set distance as defined in it's statistics (typically this is 6″). A dash allows an operative to only move 3″. Movement has several restrictions attatched.

1. Movement must not exceed the operative's movement characteristic.
2. Movement must be made in straight line increments. This means no curves but diagonals are fine.
    1. Increments can be less than 1″ but are still treated as 1″ for the purpose of total movement.
3. An operative cannot move through terrain unless it is a "small obstacle" (such as a barricade)[2]
4. An operative may not move over the edge of the board or through any part of another operative's base.
5. An operative CAN perform a normal move and dash in the same turn.
6. An operative CAN perform a normal move OR dash and then perform a "shoot" action.

This is quite a complex set of rules to enforce as a result this would be a good candidate for the system to assist with.

2.1.2.2. Shooting

2.1.3. Community

[1] There are a lot of exceptions to this with abilities being able to modify the stats of themselves or other operatives, but for the sake of simplicity we are going to ignore this and assume stats are set.

[2] We will not be encountering small obstacles in this implementation.

3. Related Work

3.1. Previous Mixed Reality Tabletop Systems

Some companies, such as *The Last Game Board* [7] and *Teburu* [8], sell specialist game boards to provide a mixed reality experience.

The Last Game Board achieves this through utilizing the touch screen to recognise specific shapes on the bottom of miniatures to determine location and identity. *The Last Gameboard* is 17″ x 17″, as a result the number of game systems which are compatible is limited. However, you can connect multiple systems together. The drawback of this is the price point for the system is rather high, with boards starting at ~$350. It is also worth noting that this system has not recieved great reviews, with alpha users reporting: long load times, low FPS, graphical and sound glitches, lack of updates and even screens being 50% black.

Figure 4: The Last Game Board touchscreen tabletop system [7]

Teburu [8] instead takes an RFID based approach, providing a base mat that allows you to connect squares containing RFID receivers and game pieces containing an RFID chip. *Teburu* connects to a tablet device to provide the digital experience as well as to multiple devices for individual player information. *Teburu* games allow for game pieces to either be in predetermined positions or within a vague area i.e. within a room.

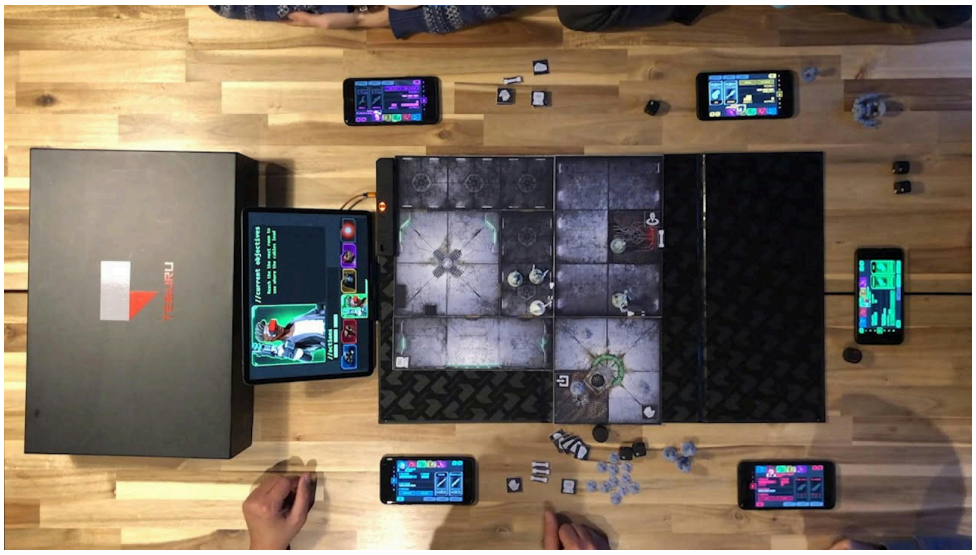

Figure 5: The Teburu Game System [9] showcasing *The Bad Karmas* board game. The black board is the main game board. The squares above connect to the board below to transmit the RFID reader information back to the system for display.

An RFID based approach is also used by Steve Hinske and Marc Langheinrich in their paper [10] which places an antenna grid below the game board to detect RFID chips within models. This allowed them to find what chip is in range of what antenna, allowing them to find the general location of a game piece. This worked particularly well for larger models where you could put RFID chips far away from eachother on the model. Using the known positions of the chips and dimensions of the model combined with which antenna said chips are in range of allows you to determine an accurate position of each model. They also go into alternate RFID approaches which will be discussed later when outlining the chosen methodology.

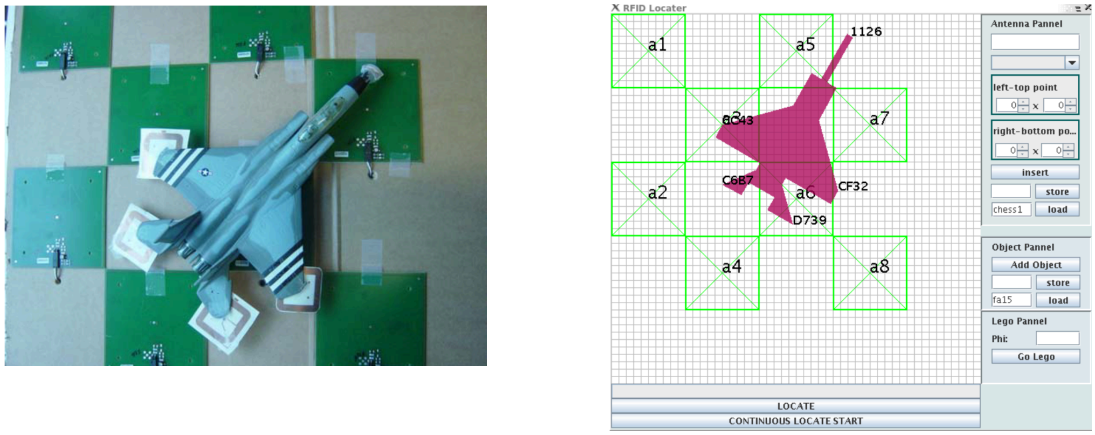

Figure 6: An example of Steve Hinske and Marc Langheinrich's approach depicting the antenna grid, RFID tags and physical model alongside the computer's prediction of the model's position

Surfacescapes [11] is a system developed in 2009 by a group of masters students at Carnegie Mellon as a university project. *Surfacescapes* uses a *Microsoft Surface Tabletop* (the product line was rebranded to *PixelSense* in 2011). This uses a rear projection display, and 5 near IR cameras behind the screen [12]. This allows the *PixelSense* to identify fingers, tags, and blobs touching the screen using the near IR image. *Surfacescapes* utilises this tag sensing technology to track game pieces using identifiable tags on the bases of miniatures.

Figure 7: An example of *surfacescapes*' in use on the *Microsoft Surface Tabletop*[13]. The position of the models have been tracked by the system and outlined with a green cricle.

Foundry Virtual Tabletop [14] is an application used to create fully digital *Dungeons and Dragons* tabletops. These can either be used for remote play or in person play using a horizontal TV as a game board. *Foundry VTT* allows for the creation of modules to add new functionality to your virtual tabletop. One such module is the *Material Plane* [15] module which allows the tracking of physical miniatures on a TV game board. This functions by placing each miniature on a base containing an IR LED with an IR sensor then placed above the board. This can be configured to either move the closest "virtual" model to where the IR LED is or (with some internal electronics in the bases) can be set up to flash the IR LED in a pattern to attach different bases to specific models. An indicator LED is present to show when the IR LED is active.

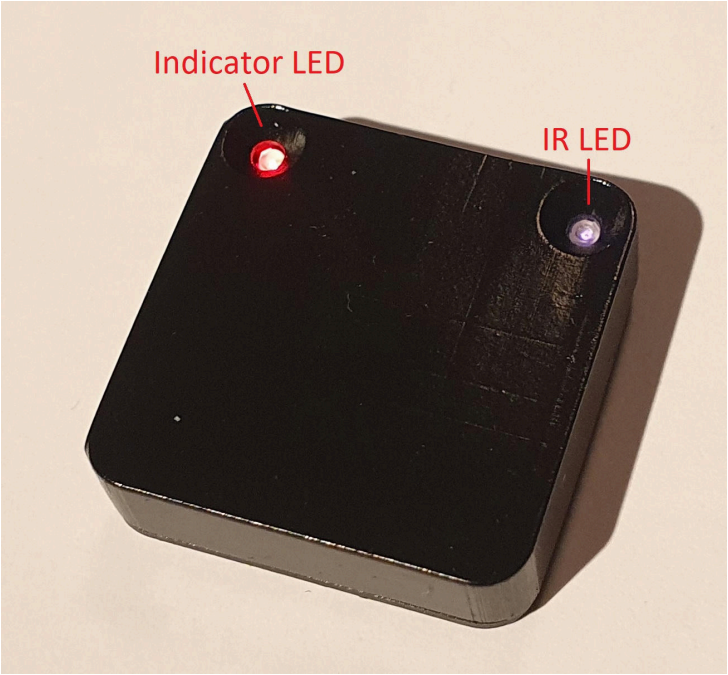

Figure 8: An example of one of the *Material Plane* bases. A miniature would be attached to the top. [16].

3.2. Object Detection and Tracking Technologies

Finding a method to detect and find the positions of miniatures on a game board, whilst not obstructing the game, is the main focus of this project. This section will outline the different technological approaches that could be taken to achieve this.

3.2.1. RFID

An RFID approach is the somewhat obvious solution. This would involve embedding RFID chips underneath the bases of the miniatures. Then some method of reading these chips would then need to be embedded either within or underneath the game board. There are a number of different approaches that could be taken to locating RFID chips which have been outlined by Steve Hinske and Marc Langheinrich [10] in their work on a similar project.

RFID solutions would require either an antenna grid underneath the game board or multiple individual RFID readers. This would be a viable option as hiding an antenna grid below a board is a relatively simple and unobstructive task. The same goes for hiding RFID readers beneath a table.

The main drawback of RFID is that a reader (at its core) can only detect if a chip is in range or not (referred to as "absence and presence" results). Due to this, some extra methodology would need to be implemented to determine the position of a chip.

One approach utilises increasing the range of an RFID reader to its maximum and then subsequently reducing the range repeatedly. Using the known ranges of the readers it would be possible to deduce which tags are within range of which reader - and subsequently deduce from which ranges it is detectable in the position of the RFID chip. This approach could in theory provide a reasonably accurate position of the RFID chip. However, this approach would take a longtime to update as each RFID reader would need to perform several read cycles. Combined with problems caused from interference between the chips / readers, the fact that being able to vary the signal strength of an RFID reader is not a common feature and the need for multiple RFID readers, this approach does not meet the requirements for this project.

Another approach utilises measuring the signal strength received from an RFID chip and estimating the distance from the reader to the chip, known as received signal strength indication (RSSI) . This approach is much quicker than the previous method needing only a single read cycle to determine distance. Most modern day RFID readers can report RF phase upon tag reading. However, current RSSI methods have an error range of ~60cm caused by noise data, false positives / negatives and NLOS (non-line of sight) [17]. This won't work for this project given than the board size is 70 x 60 cm.

Triliteration is a process in which multiple receivers use the time of arrival signal from a tag to determine it's position. This suffers from similar problems to other RFID methods in that it produces an area in which the tag could exist within - as opposed to it's exact position. Combined with the need for 3 RFID readers, this approach fails to be accessible to the target audience.

The approach previously mentioned in Steve Hinske and Marc Langheinrich's paper, which utilised an antenna grid below the game board, seemed promising.

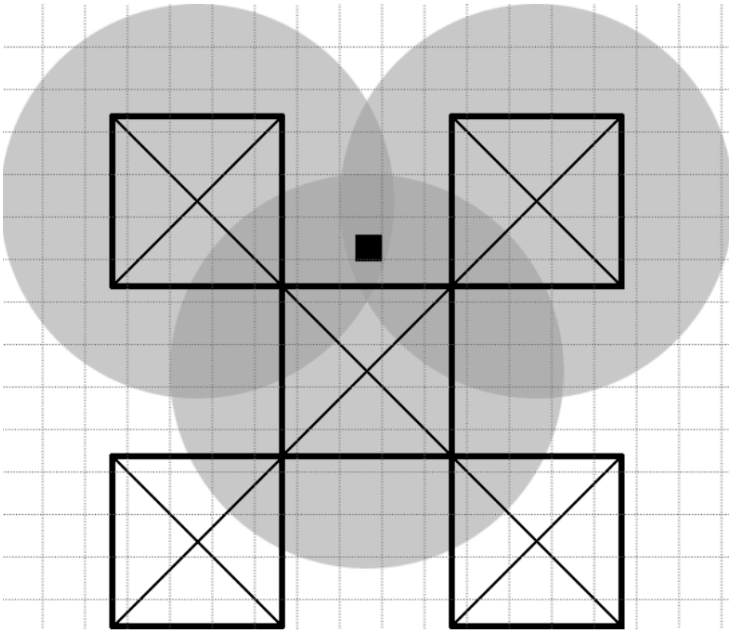

Figure 9: An example from Steve Hinske and Marc Langheinrich's paper [10] describing their overlapping approach. The black square represents the RFID tag whilst the grey circles show the read area of each RFID reader. The intent is to use which readers are in range of the tag to determine an "area of uncertainty" of the tag's location.

They were attempting to do this for *Warhammer 40k*, a game played on a much bigger board typically with larger models. As a result they were able to put RFID tags across larger models, such as vehicles, and utilise the known position of each tag relative to the model and the estimated position from the RFID readers to determine not only an accurate position but also orientation. Unfortunately for this project the size of the miniatures in *Kill Team* would require the use of one tag or two in very close proximity.

A potentially valid method utilizing RFID could be to use the approach outlined by Zhongqin Wang, Ning Ye, Reza Malekian, Fu Xiao, and Ruchuan Wan in their paper [18] using first-order taylor series approximation to achieve mm level accuracy called *TrackT*. However, this approach is highly complex and can only track a small amount of tags at a time.

3.2.2. Machine / Deep Learning

Modern object tracking systems are often based on a machine learning or deep learning approach. In this case a classifier model would be used to identify each unique miniature on a team, and then subsequently locate it within the game board. The biggest drawback to this approach is the amount of training data needed for each class in a classifier. According to *Darknet's* documentation (a framework for neural-networks such as YOLO) [19] each class should have 2000 training images.

Since each user will use their own miniatures, posed and painted in their own way, they would have to create their own dataset for their miniatures and train the model on them each time. As a user's miniatures are likely to follow a similar paint scheme, ML classification could potentially struggle to identify between miniatures from top down and far away if not enough training data is supplied.

3.2.3. ARKit

Apple's ARKit supports the scanning and detection of 3D objects [20] by finding 3D spatial features on a target. Currently any phone running IOS 12 or above is capable of utilising the ARKit. In this system,

you could scan in your models, then use the ARKit to detect them from above. This information could then be conveyed to the main system. This could also allow for the system to be expanded in the future to use a side on camera as opposed to just top down detection, allowing for verticality within other *Kill Team* game systems. Combined with certain Apple products having an inbuilt LIDAR sensor (such as the *iPhone 12+ Pro* and *iPad Pro* 2020 - 2022), which further enhances the ARKit's detection, this could be a viable approach.

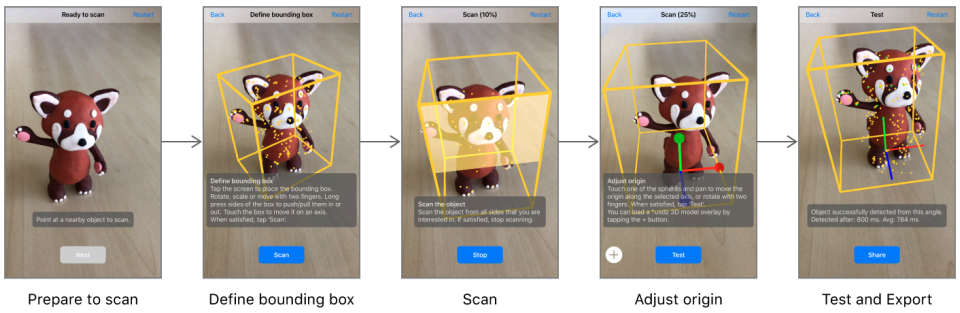

Figure 10: Registering an an object in ARKit [20]

After some testing utilizing an iPad Pro with a LIDAR scanner, some drawbacks were found with this approach. The object detection required you to be too close to the models detect them, and when being removed and re-added ARKit either took a long time to re-locate a model or failed to do so at all.

As a result I won't be using this approach for the project. Although, for future work, this could be an interesting option to explore allowing you to implement AR features such as highlighting models on your phone or tablet or displaying information above the model.

3.2.4. Computer Vision without neural networks

After considering all the options above, the best method found was using computer vision techniques.

One technique is blob detection. A blob is defined as a group of bright or dark pixels contrasting against a background. In this case the miniatures would be the blobs.

Utilizing blob analysis would allow locating general position of miniatures (as they would very clearly stick out from the background) but getting their exact center may then prove difficult (which would be needed for calculating movement and line of sight). Combined with the addition of terrain adding clutter to the image and the miniatures potentially being any colour, this approach could prove difficult to implement in the given time frame.

The computer vision method explored the most was utilizing coloured tags to identify miniatures.

The use of coloured tags would also mean there would be access to specific measurements allowing for an accurate location to be discerned. However, some external modification to the the game board or miniatures would need to be made. The challenge here is finding a way to do this without obstructing the flow of the game or the models themselves.

3.3. Computer Vision

3.3.1. Fiducial Markers and Tag Detection

3.3.2. Occlusion solutions

3.3.3. Object Detection

3.3.4. Parallax Solutions

3.3.5. Calibration Solutions

4. Project Description

This project aims to create a system that can track tabletop miniatures, in a game of *Kill Team Gallowdark*, using only materials easily accessible to the average miniature war-gamer. Then, utilise this system to implement a "helper" program for the game, providing a digital representation of the physical game state as well as additional information to the players to assist in their decision making process and alleviate the burden of rules enforcement.

As a result, the project can be broken down into two main goals.

1. Detection of the models and terrain to create a virtual representation of the game board.
   a. The position of the miniature must be tracked accurately.
   b. The system must be able to identify between different miniatures.
   c. The system must aim to be noninvasive to the miniatures.
   d. The system must be ambivalent to a model's shape and colour.
   e. The system must be able to complete this task whilst being accessible to the average miniature war-gamer.
2. Implementing the game logic in the virtual board to guide players through the game.
   a. Allow users to select a model and a subsequent action (normal move, shoot, dash, capture objective).
   b. Calculate the distance a model can move and display this on the virtual board.
   c. Account for terrain that blocks movement.
   d. Calculate the line of sight between the selected model and opposing models then display this on the virtual board.
   e. Account for terrain that blocks line of sight.
   f. Display information about the selected model's odds to hit a target.

4.1. Model and Terrain Detection

4.2. Game Board Representation

5. Methodology and Design

5.1. Design

The general design of the system would place a webcam or camera above the game board. Miniatures would have easily recognisable tags attached to allow for easy identification via computer vision.

The tag would need to be unobstructive to the game (big enough to be seen by the camera but small enough as to not get in the way of the game whilst also not looking too "out of place"). This ruled out sticking QR codes on top of miniatures, or attaching them to wires a pointed upwards from a miniatures base. This meant a unique design was needed.

Before attempting find tags some viewpoint / perspective correction is needed. The end goal is to get a bird's-eye view of the game board, however a camera placed above the board will produce a slightly distorted view of the game board. This process would involve finding some set reference points on the board, in this case the corners, and using these to perform a perspective transform creating a top down view of the image. This means that when the tags are located their location can be provided directly to the game board display.

The detection problem can be split into two components: locating and identifying the miniature. Tackling the location problem first seemed like the most sensible approach to begin with before moving onto identification.

In the end it was chosen to 3D print a small rim that could fit around the base of a miniature. This rim would be a high contrasting colour to the background and miniature. The benefits of 3D printing the rim is that the colours and dimensions are customizable to the user. Miniature war-gamers paint their models in a variety of different ways, so having a rim they can decide the colour of themselves keeps the system accessible to the target audience and open to many different miniature sizes and colours.

Figure 11: An example of the basic contrasting rim design.

Some important observations here are that models may poke over the edge of the rim and models will block the rim due to parallax, so a rim detection system will need to find a full circle from only a partial amount of visible rim. When terrain is introduced this will cause further issues. The proposed fixes for this will be discussed later.

The next step was to find a way to detect the rim. The best solution found for this was *openCV* [21]. An alternative image processing suit would have been to utilise *MATLAB*, but *openCV* can be used directly within *Python*. As a result, the final system could be written entirely within *Python* signficantly streamlining the development process without needing to connect *MATLAB* with an external program for the game board logic / display.

This led to the development of a processing method which will locate the center point of circles of a specific colour in a provided image.

The image processing steps taken:

1. Sort the image to only contain the colour of the rim (in this case yellow).
2. Grayscale the image.
3. Apply a blur to the image (this helps to remove noise from the image).

4. Apply hough circle detection [22] on the processed image.
5. Draw the detected circles onto the original image.

It is important to note that hough circle detection can easily mistake two close but separate circles as one circle. As a result of this, the hough circle detection function in *openCV* has several arguments that will need to be tweaked as the system is developed.

Through some testing we were able to locate singular circles even with the presence of terrain pieces blocking parts of the rim.

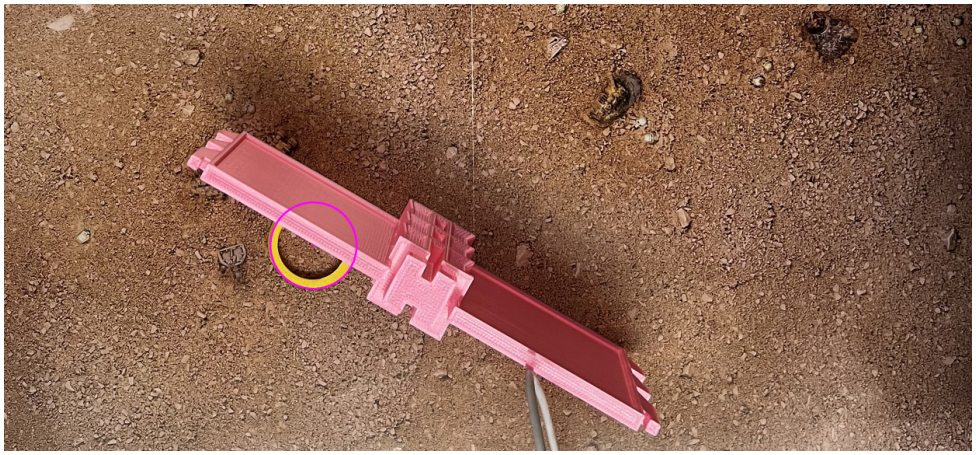

Figure 12: Circle detection on a gameboard with example terrain.

Parallax is a problem that still needs to be addressed. Through testing it was found that using a standard game board size, the parallax was too great to expect to locate a rim. As a result, we have settled on using a two camera solution. This solves the parallax problem by having each camera watch a different half of the game board.

However, one problem with this approach is that calibration will be needed to ensure that the overlap between the cameras does not cause issues. This could be achieved by either calibrating the cameras to the game board, as it will be of a known size, by placing tags on each corner of the board. Alternatively the middle of the board could be marked with a piece of tape, or tags on the edges. This should then allow for the two images to be stitched together accurately.

Due to the nature of this project being aimed at following the *Kill Team Gallowdark* rule set, the terrain pieces used are placed along a grid. This prevents a situation where a terrain piece is very close to the edge of a board with a miniature placed behind resulting in the miniature being blocked from view.

Once we have located a model we need to identify it. In our implementation of *Kill Team* we will assume that each player will have up to 14 models on their team (known as “operatives”). As a result we will need to create an encoding capable of representing at least 28 different options.

Another rim will be placed around the bright contrast rim from before. This rim will be split into 6 sections. The first section will indicate the starting of the encoding and will use a unique colour. The other 5 sections will use 2 different colours to represent 1 or 0. Once a rim has been located, the system will then determine the orientation of the encoding circle. As the size is known, we can then rotate around the image of the encoding circle reading each section to eventually get a binary number. Each binary number will then represent a unique model on the board.

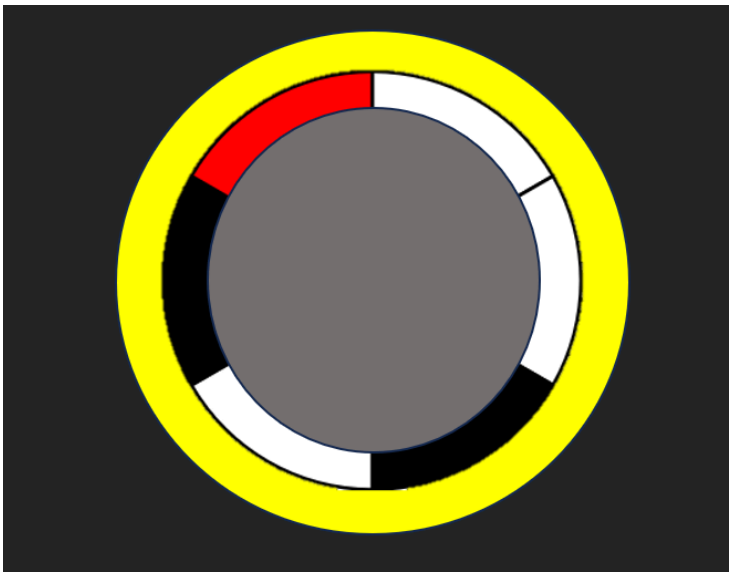

Figure 13: An example of what the encoding circle could look like. The red section indicates the start of the encoding. The system would be able to determine which direction to read in based on the start encoding section being in the bottom or top half of the image.

Another encoding method that was considered was to use one rim for both detection and identification. By using one rim of known colours, the image could be split into 3 images each containing only the colour of each segment. Each image would then be converted to black and white and recombined to create a singular image with the segments connected to create a full circle. After circle detection has been performed the system could then do the normal encoding reading. This approach would allow the rims of the miniatures to be significantly smaller and only require three contrasting colours.

One problem faced by this approach is that vision of the entire circle is needed to read the encoding, as a result if it is partially obscured then the system will be unable to identify the model.

This could be counteracted by splitting the encoding into quarters around the rim, so that as long as 1/4 of the encoding is visible, the system can correctly identify it. We can also exploit *Kill Team* being a turn based game and simply compare the current game state with the previous game state to identify which pieces have been moved. A combination of active tracking and being able to determine which piece has moved, should allow for a robust system.

Terrain detection will be performed using two possible solutions, the user can either select predetermined terrain set ups or the system can detect the terrain pieces.

The current approach for terrain detection is to use QR codes on the top of each terrain piece. In *Gallowdark* the size and shape of each terrain "module" is already known. As a result, the QR code can be used to determine the type of terrain and orientation. Then, since the terrain type has been established we already know the size and shape. So we can then fill it in on the virtual board.

Once the system has correctly identified and located each model and the terrain, a digital representation of the game board can be created.

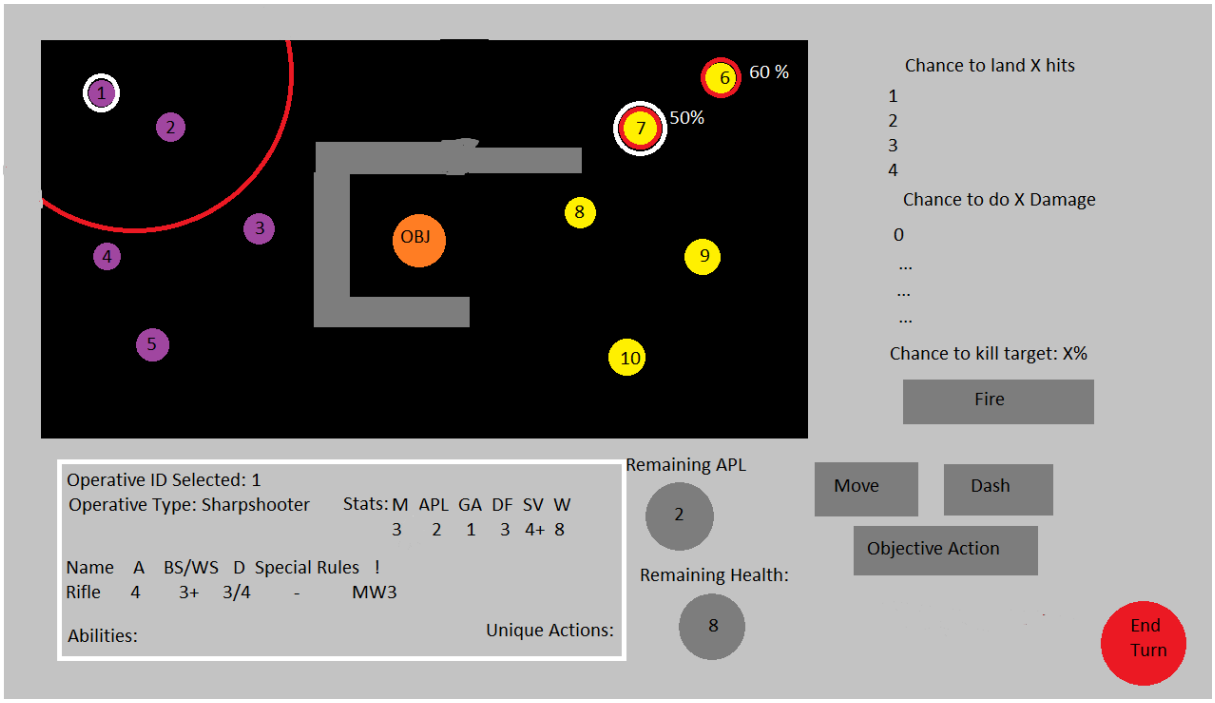

Figure 14: A proposed GUI for the game helper.

In Figure 14 the red semi-circle represents the valid area of movement for a selected operative. This can be calculated by splitting the game board into a very small grid, we can then apply a path finding algorithm (Dijkstra's, A* etc) to the grid from the operative's starting point. We can then calculate all grid positions reachable by using a movement value less than or equal to the current operatives available movement characteristic.

Red circles around opposing operatives show they are in line of sight. This would be calculated using simple ray casting. A white and red circle indicates a selected opposing operative. This is to calculate the statistics on the right hand side of the screen to be specific to the opposing operative.

The information for the selected operative is displayed below the game board. A user can select the action they have performed / want to perform on the right hand side.

The GUI will be built in *PyQt* [23] which is a python wrapper for the *Qt* framework. One potential stretch goal is for the GUI and the detection system to be separate entities. For example, the detection system should be able to detect and track pieces without input needed from the virtual game board i.e. which model is selected. This would allow for other detection systems to be swapped out in the future or for other applications to be developed using the detection system.

6. Implementation

6.1. Camera Setup

6.1.1. Design and Calibration

6.1.2. Homography

6.2. Model Detection

6.2.1.1. Colour Thresholding

6.2.2. Rim Detection

6.2.2.1. Hough Circles

6.2.3. Model Identification

6.2.4. Optimisations

6.3. Terrain Detection

6.4. Game Board Representation

6.4.1. Linking of Detected models and terrain to the virtual board

6.4.1.1. Operatives

6.4.2. Line of Sight

6.4.2.1. Obsucring and visible

6.4.2.2. Cover

7. Evaluation

8. Summary and Reflections

8.1. Project Management

As per the initial Project Proposal the original goal for the first semester was to have ring detection and terrain detection completed. Currently, simple ring detection has been completed. This still needs to be expanded to include identification, though work has begun on this. Terrain detection has been moved backwards to allow for more time to make a good detection system for the miniatures.

Choosing to tackle the ring detection first was a good choice. This allowed for me to encounter larger issues and provide solutions to them whilst still in the research stages on the project. This has allowed for me to develop my computer vision skills and take approaches I would not have otherwise considered. For example, using a singular rim of multiple colours and combining the images to create a full circle for identification.

Supervisor meetings were held every week to check for blockers or advice on reports. Uniquely, our supervisor meetings were conducted as a group with the other two dissertation students my supervisor had. As all three of us were working on tabletop game based projects we were all knowledgeable in the area. This meant we could provide feedback or ideas for each others projects from a student perspective.

I found the Gantt chart to be unhelpful in managing the project. Gantt charts work well in well structured work environments. However, due to the nature of this semester having a large amount of other courseworks and commitments that needed to be balanced. An agile approach was much more effective, doing work when time was available. The nature of development work being much more effective when done in long, uninterrupted sessions meant that, with this semester's courseworks and lectures being very demanding at 70 credits, being able to find long swathes of uninterrupted time was very difficult between lectures, courseworks, tutorials etc. As a result, development was done in smaller, more frequent chunks which were not as effective as longer, less frequent chunks. I expect this to change next semester when the workload decreases down to 50 credits.

A much preferred method, that I will likely go ahead with, is Kanban. I personally found most use from the Gantt chart in that the project is naturally broken up into sub tasks. This approach of sub-tasking when, combined with a less structured agile approach, is something that Kanban works really well at and have found effective in the past in GRP (COMP2002).

In the interests of being able to show current progress in comparison to previous I have included an updated Gantt chart (Figure 15) along with the previous one (Figure 16).

Looking at the time remaining I will focus on getting the main parts of the system functional. These are: model detection and tracking, terrain detection, virtual game board representation, movement preview and line of sight preview. If there is time available then I will aim to also implement the flow of the game (breaking it down into each phase, providing guidance of what to do in each phase, statistics etc). The most technically interesting part of this is the virtual game board and tracking technology. So placing a focus on having them work fluidly is more important to the dissertation.

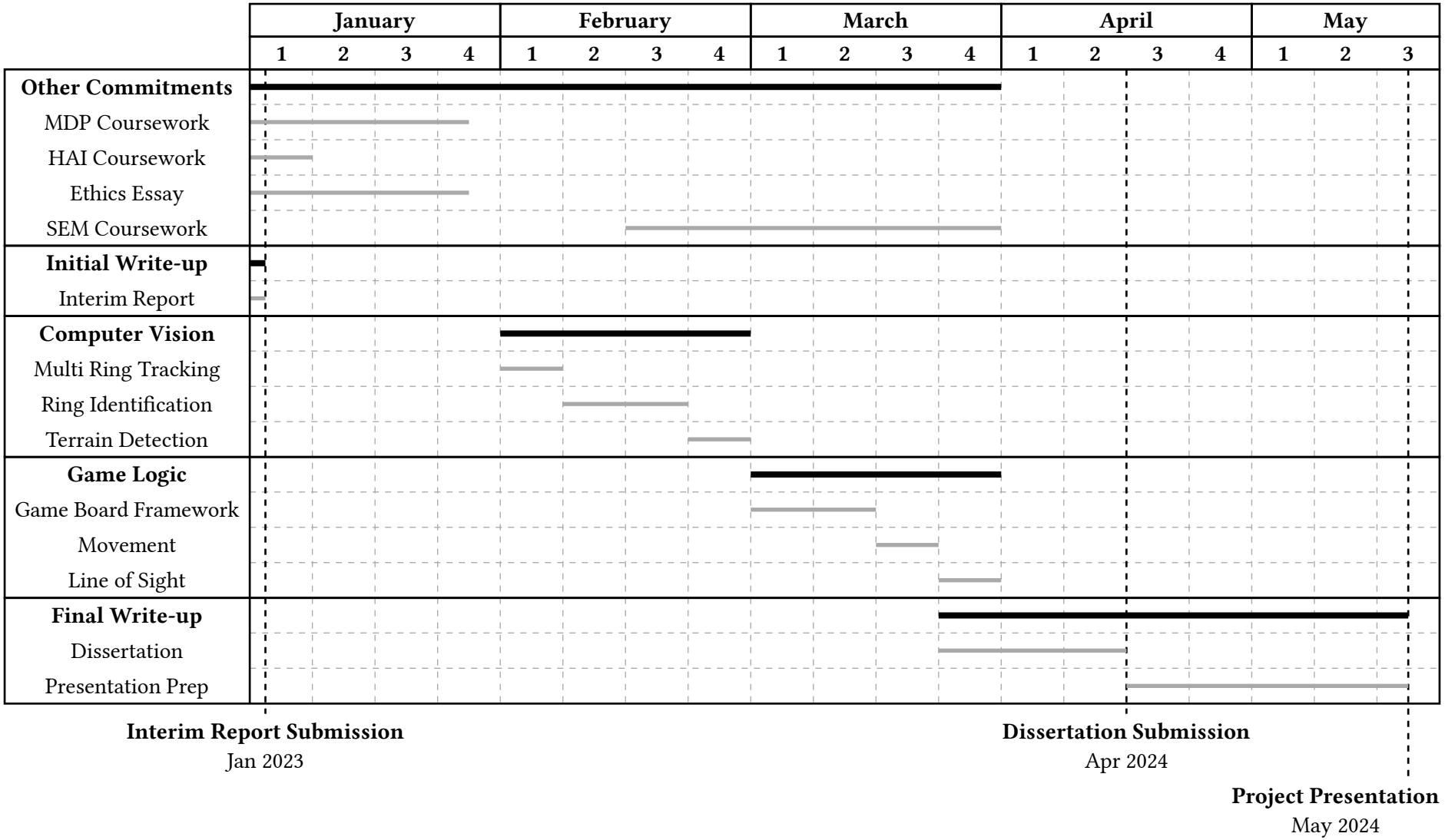

Figure 15: The Updated Gantt chart

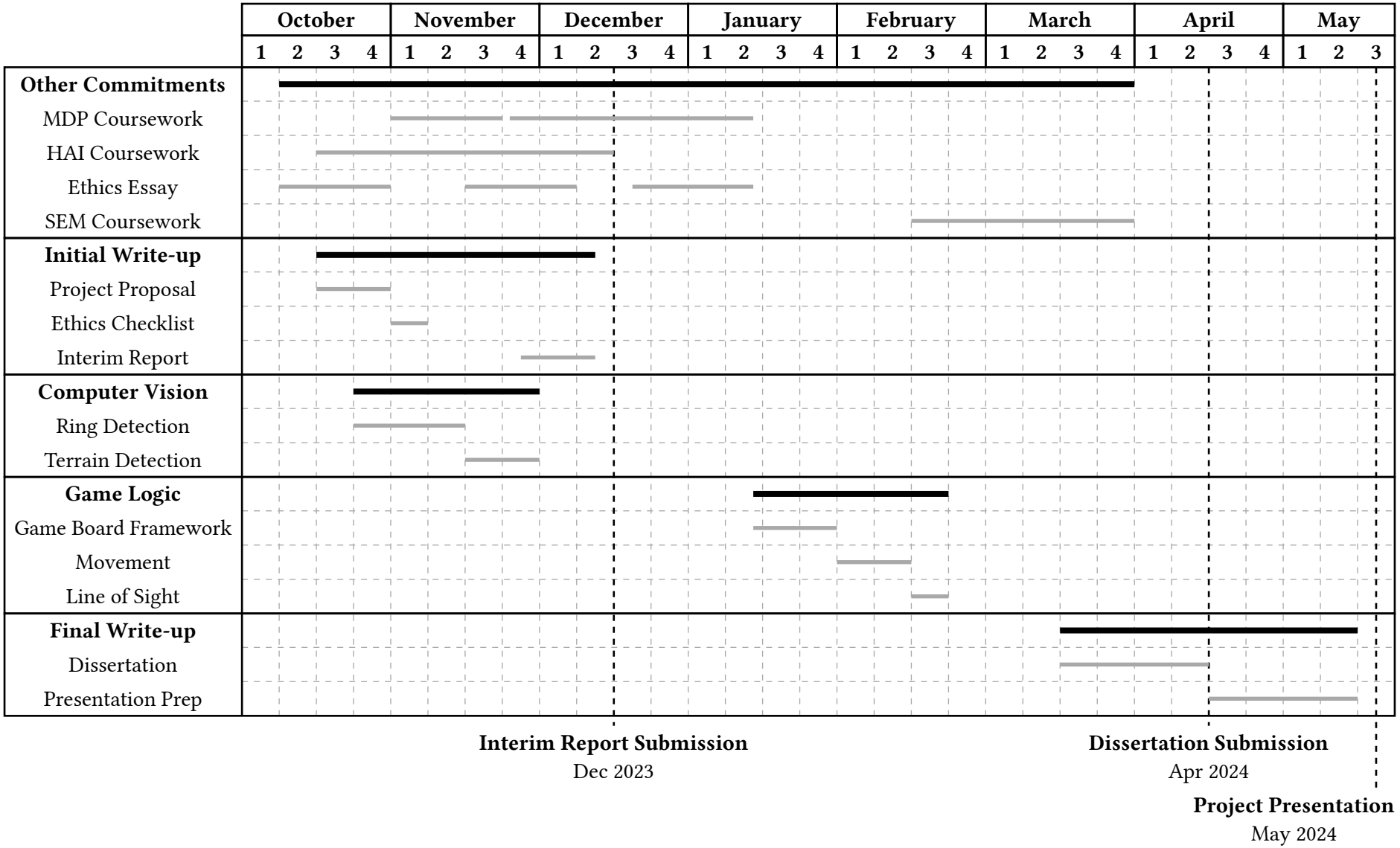

Figure 16: The original Gantt chart

8.2. Future Work and Reflections

Something I underestimated was the amount of time needed to research and determine which method should be used for model detection. A large amount of time was spent determining the viability of different approaches that may make the system more robust and easier to produce. As a result of this, development started later and took longer than was expected.

The biggest impact on this project was needing to get an extension on other coursework deadlines. This extension had a knock on effect of other courseworks which took priority over this project. The timings allotted in the Gantt chart did not take into account extensions being needed.

However, I am happy with the progress made so far. Having completed the methodology to find the circular tags even when partially obstructed is very promising to the viability of the chosen approach. As a result, I believe that a solid strategy has been chosen for model detection and identification and the work done so far has successfully laid a good foundation for the project.

8.2.1. Order Tokens and Objective Markers

8.2.2. Odds to Hit

8.2.3. Advanced Terrain

8.2.4. Height

8.2.5. Larger Game Board

8.2.6. Increasing the total number of operatives

8.2.7. Server Client Architecture For Game Board and Detection

8.2.8. Detection Upgrades

Hemming Distance

8.3. Development Reflections

Official openCV python documentation is somewhat lacking.

8.3.1. Design Approach

8.4. LSEPI

The main LSEPI issue you could see here is copyright issues as *Kill Team* is a copyrighted entity owned by *Games Workshop*. The way this project will handle this is by having the system implement the *Kill Team Lite* rule set previously mentioned. This is publicly available published by *Games Workshop*. One issue is that different "*Kill Teams*" (groups of operatives) will have different and unique rules as well as their own statistics pages. These are copyrighted and won't be able to be directly implemented. As a result of this, this project will implement basic operatives with fake statistic pages based off of the publicly published *Kill Team* information. If this proves to be problematic then the game rules will be

based off a very similar system *Firefight* [24]. This is published by *One Page Rules*, who produce free to use, public miniature war-game systems.

8.4.1. Accessability

9. Bibliography

[1] Games Workshop, "Games Workshop Investor Relations Statement." Oct. 17, 2023. [Online]. Available: https://investor.games-workshop.com/our-history

[2] Games Workshop, "Warhammer 40k Core Rules." Jun. 02, 2022. [Online]. Available: https://www.warhammer-community.com/wp-content/uploads/2023/06/dLZIlatQJ3qOkGP7.pdf

[3] Games Workshop, "Kill Team Lite Rules." Aug. 16, 2022. [Online]. Available: https://www.warhammer-community.com/wp-content/uploads/2022/08/ekD0GG2pTHlYba0G.pdf

[4] Warhammer Community, "Kill Team: Into the Dark – What's in the Box?." Accessed: Dec. 31, 2023. [Online]. Available: https://www.warhammer-community.com/2022/08/04/kill-team-into-the-dark-whats-in-the-box/

[5] Games Workshop, "Killzone Gallowdark." Accessed: Dec. 12, 2023. [Online]. Available: https://www.warhammer.com/en-GB/shop/killzone-gallowdark-2023

[6] Games Worksshop, "Kill Team: Kasrkin." [Online]. Available: https://www.warhammer.com/en-GB/shop/kill-team-kasrkin-2023?queryID=4bbc70cecccea9006fc261421f8bc787

[7] Last Gameboard, "The Last Gameboard Kickstarter." Accessed: Dec. 12, 2023. [Online]. Available: https://www.kickstarter.com/projects/gameboard1/gameboard-1

[8] Teburu, "Teburu." Accessed: Dec. 12, 2023. [Online]. Available: https://teburu.net/#home

[9] "Teburu: a sneak peek." Accessed: Dec. 13, 2023. [Online]. Available: https://www.youtube.com/watch?v=5paNoKA4b5A

[10] Steve Hinske and Marc Langheinrich, "An RFID-based Infrastructure for Automatically Determining the Position and Orientation of Game Objects in Tabletop Games." Jun. 05, 2007. [Online]. Available: https://vs.inf.ethz.ch/publ/papers/hinske-pg07-rfidtabletop.pdf

[11] Whitney Babcock-McConnell, Michael Cole, Stephen Dewhurst, Bulut Karakaya, Dyala Kattan-Wright, and Maokai Xiao, "Surfacescapes." Accessed: Dec. 13, 2023. [Online]. Available: https://www.etc.cmu.edu/projects/surfacescapes/index.html

[12] Microsoft, "Microsoft Surface 1.0 documentation." Accessed: Dec. 13, 2023. [Online]. Available: https://social.technet.microsoft.com/wiki/contents/articles/8017.microsoft-surface-1-0-sp1-getting-started-guide-hardware-overview.aspx#Tabletop

[13] Abhay Buch, "With Surfacescapes, Dungeons & Dragons becomes high-tech." Accessed: Dec. 13, 2023. [Online]. Available: http://thetartan.org/2010/4/5/scitech/surfacescapes

[14] Foundry Gaming LLC, "Foundry VTT." Accessed: Dec. 23, 2023. [Online]. Available: https://foundryvtt.com/

[15] Material Foundry, "Material Plane." Accessed: Dec. 23, 2023. [Online]. Available: https://www.materialfoundry.nl/

[16] Material Foundry, "Material Plane Github." Accessed: Dec. 23, 2023. [Online]. Available: https://github.com/MaterialFoundry/MaterialPlane/wiki/Beta-Hardware-Guide

[17] "Robust and Accurate Indoor Localization Using Learning-Based Fusion of Wi-Fi RTT and RSSI." 2022. Accessed: Dec. 27, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9002808/

[18] Zhongqin Wang, Ning Ye, Reza Malekian, Fu Xiao, and Ruchuan Wang, "TrackT Accurate tracking of RFID tags with mm-level accuracy using first-order taylor series approximation." Dec. 15, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870516302736

[19] "YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)." Accessed: Dec. 28, 2023. [Online]. Available: https://github.com/AlexeyAB/darknet

[20] Apple, "ARKit." Accessed: Dec. 27, 2023. [Online]. Available: https://developer.apple.com/documentation/arkit/arkit_in_ios/content_anchors/scanning_and_detecting_3d_objects

[21] G. Bradski, "The OpenCV Library." 2000. Accessed: Dec. 12, 2023. [Online]. Available: https://github.com/opencv/opencv

[22] G. Bradski, "Hough Circle Transform." Accessed: Dec. 27, 2023. [Online]. Available: https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html

[23] Riverbank Computing Limited, "PyQt." Accessed: Dec. 31, 2023. [Online]. Available: https://riverbankcomputing.com/software/pyqt/intro

[24] One Page Rules, "Firefight." Accessed: Dec. 31, 2023. [Online]. Available: https://www.onepagerules.com/games/grimdark-future-firefight