Honours in Game Design & Development

Team Project & Group Dynamics Module

Proposal, Technical Project

to be presented on 21/10/2019

Jordan O'Loughlin K00225361

Thomas Jones K00232078

Elijah Omotosho K00227425

Project Supervisor(s): Jacqueline Humphries

Contents

1. INTRODUCTION

1.1 PURPOSE

The goal of the project is to provide a short adventure styled game that utilizes an android phone in order to activate certain attacks along with puzzles to engage the player at the end of every level. The expected benefits of the project are that this project aids our group in gaining experience on game development by going through each stage of the game development process and giving each of us in our group a taste of what it's like when creating smaller games.

1.2 BACKGROUND

In this background the team will be looking at Maze puzzle games in general, then the group will be looking at how to make a circular design of a maze, explaining what pixel perfect is and how it will work inside of the group project and finally the group will state the inspirations of the project and what elements of our inspirations are going to be going into the project.

The maze game genre is a video game genre that is described as the game uses quick player actions that required interactions with an enemy by outmaneuvering them or attacking them while navigating their ways out within a time limit or completing a specific objective.

The most famous maze game would be Pac Man which was a game with the titular character Pacman trying to escape from the four ghosts in-game which are Pinky, Blinky, Inky and Clyde. The goal was to collect all the pelts inside a maze before the ghost got to you and took out your lives. It gives the user multiple opportunities to kill the ghost for a limited time.

Branching off to puzzle video games, they focus on more the logical and conceptual challenges, it mainly focuses on solving puzzles as the main and primary activity as gameplay. The gameplay consists of shapes and colors or symbols the players must manipulate in order to directly or indirectly create a specified sequence.

The genre of puzzle games is very vast and requires some level of thinking which may uses numbers, colors, shapes, or complex rules. A common trope in puzzle games is giving the player lives to give them a sense of urgency to complete the number of tries.

The goal in this section for the group is to combine both genre into one and make a game surrounding that incorporates all the tropes and traits of each genre while adhering to the brief given to us for the project.

Moving unto the circular maze design, the group have decided to create a circular maze. To accomplish this assignment, the group are going to acquire the services of the website ‘http://www.mazegenerator.net/’ to create a random pregenerated mazes for us.

After creating the mazes, the team will then transfer the mazes into visual studios and modify them to covert the white pixels inside of the generated mazes into transparent pixels so that we can place

object and they will stay on the same plane as the pregenerated maze. After all that the team will then apply the pixel perfect collision detection method to make the maze detect collisions to it.

Pixel perfect collision is an accurate kind of 2D collision detection that uses bitmasks to determine if two objects collide. Bitmasks are used to create every object in our game because it will give the team something which will allow us to accomplish collision detections.

The team was primarily inspired by *Zelda: A link to the past* and *Stardew valley*. These games are both top-down adventure games. The difference between the two games is that one game focuses on the action-adventure aspect of the game while the other focuses on the one more role-playing aspect of the game through my reach into both games. But both games have maze and puzzles aspect to them.

Zelda: A link to the past will be the game the group will be mainly taking from since it will have similar but inspired gameplay and it has the more maze section to the game than *Stardew valley*. The place in *Zelda: A link to the past* that the group will mainly be taking from is the dungeon section of the game since it will be mainly focusing on maze and in the dungeons inside of the games it also incorporates puzzles which meet our puzzles subgenre of the game. (Anon., 2019)

While in the game *Stardew valley*, the group decided on taking the general movement of objects and GameCharacter in the project, so this will include the games mechanic like jumping, dodging, moving, attacking, defending and interacting with object like for example picking up items. (Anon., 2019)

1.3 NEEDS STATEMENT

The game is going to be set apart from the others because it will be utilizing circular maze types. Most maze type games that are being produced will be geometrically angular. The team will be using an android controller, which will also set the game apart from its competitors. There is a gap in the market for circular based maze games that use separated controllers and their sensors.

1.4 SCOPE

The team must produce a game as a component of our third-year project. This project must be submitted as an entry to the Games Fleadh competition in March. The theme for Games Fleadh 2020 is "Maze". The team can use an OO language such as C++, C#, or Lua. It is expected of us to use Simple & Fast Multimedia Library to build a desktop game. The team can use any assets the team creates, and/or any assets that are free to use and to whom you give copyright.

The game needs to have a maze element, as this is the genre of the game. The team would be aiming to use circular mazes rather than a traditional square.

The game needs to include an enemy of sorts, that challenges the player throughout the maze.

Another challenge in the game would be a mini-game system. At certain points in the game, the player is required to complete a minigame to progress in the game.

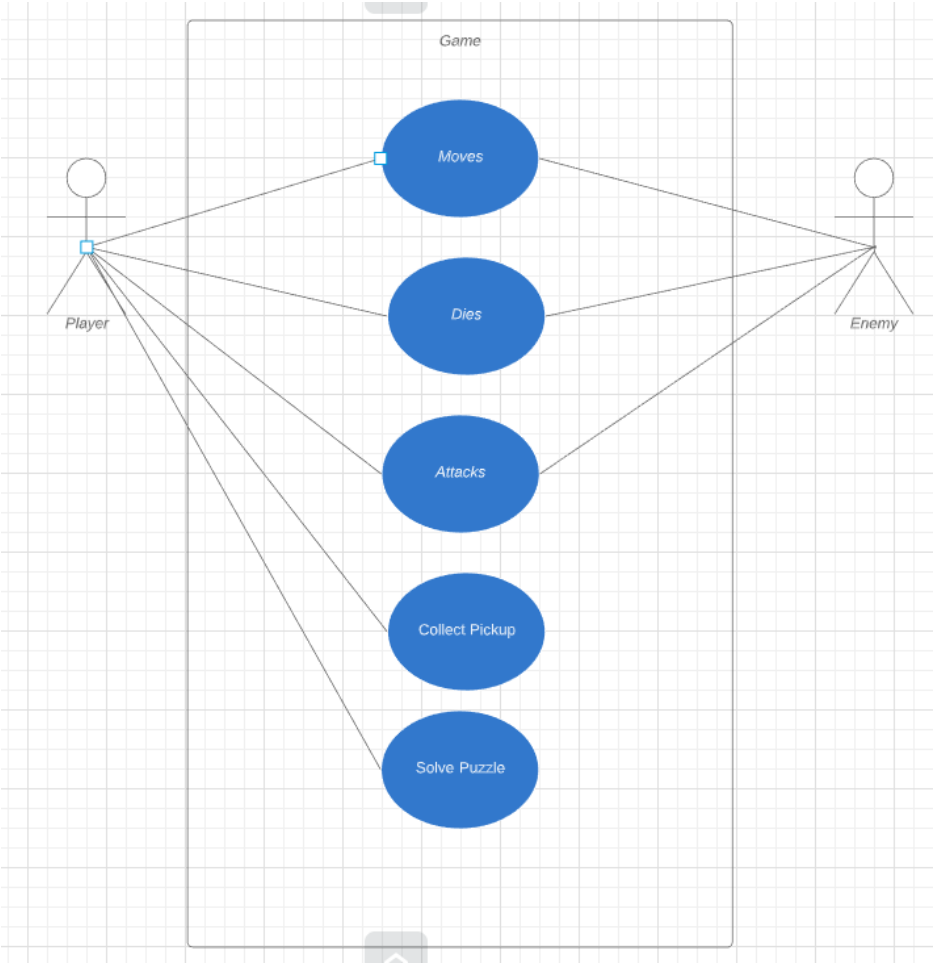
1.5 PROJECT MEMBERS

Team Member	Role	Contact Information	Responsibilities
Thomas Jones	Champion	K00232078@student.lit.ie	Makes sure that the project is going into the correct direction and doesn't get derailed.
Elijah Omotosho, Jordan O'Loughlin	Stakeholder	K00227425@student.lit.ie, K00225361@student.lit.ie	
Jordan O'Loughlin	Project Manager	K00225361@student.lit.ie	
Thomas Jones, Elijah Omotosho, Jordan O'Loughlin	Architect	K00232078@student.lit.ie, K00227425@student.lit.ie, K00225361@student.lit.ie	
Thomas Jones, Elijah Omotosho, Jordan O'Loughlin	Analyst	K00232078@student.lit.ie K00227425@student.lit.ie K00225361@student.lit.ie	
Thomas Jones, Elijah Omotosho, Jordan O'Loughlin	Developer	K00232078@student.lit.ie K00227425@student.lit.ie K00225361@student.lit.ie	Coding the Games and writing up documents that are involved with the game.

2. PROPOSED TECHNICAL APPROACH

2.1 REQUIREMENTS

Figure 1: Use Case Diagram



ISO-9126

The international organization of standardization developed the ISO-9126 standards for software engineering product quality, and this is what the project will be using for its goals and objectives. Keeping to the six main key pillars of the standard; maintainability, efficiency, usability, functionality, reliability, and portability. (Anon., n.d.) The project being in its infancy and being design for a simple college project won't need much development under portability. In fact, the game isn't going to be ported.

With the use of Object Orientated Programming, it is much easier to maintain, and test the project to make sure it is fully maintainable. With different methods and classes, Maven can be deployed to aid testing. Using a variety of white-box tests to examine the coverage of the code, making sure there aren't any unnecessary pieces of code or in executable methods.

The project's efficiency is mainly going to be measured by its resource utilization. The use of pointers and references in C++ to prevent the unnecessary duplication of objects in the code is going to be a key point for efficiency of the project.

The game will be aimed at those who don't necessarily play games, as such a lot of effort will be put in to assure that the game can be played by most, with plenty of instructions. The controls will be simple movement and it is not going to be a complex concept. This assures that the project's understandability, learnability, and operability will be of a high standard.

With regards interoperability, the project is not going to be exchanging much data between a multitude of different systems. The main problem that will occur is the interoperability between the SFML library, C++ and the Android system.

The game won't be subjected to stress conditions; we won't be employing a database, and as such, it should always be reliable. No unexpected crashes, no runtime errors, and no compile errors. Each instance of the game is going to be isolated from the others, as such there is a high fault tolerance for the project by design.

The project's response time with regards to the controller and input devices should be aiming at an input lag between 15-40ms. This is an ideal range for gaming in a single player fashion, an online multiplayer would have to be 15ms or lower, as it is generally more noticeable.

The screen should be updated at least 30 times in a second at a minimum, and 60 at maximum. As the project is still in its infancy it's difficult to propose a solid goal, however, between 30 and 60 is a solid foundation to work from.

Cohesion: For most of the project high cohesion will be used. Each of the game classes will be focused on what it should be doing and not have a large variety of things that it could be doing. This means each class will have a specific reason for existing and its functionality will have an exact reason for existing. High cohesion also allows for more robustness, responsibility, high reusability and easier readability when developing object-oriented programming. The project classes will aim to have a high cohesion within classes.

Coupling: Low coupling was also chosen due to the use of OO. Also, a lot of classes will be derived from a base class these derived classes will have the freedom to stretch and expand on the

base class it is derived from. Decoupling will play a major part in the project, allowing classes to have major changes and not have any effects on any other classes e.g. Player and Enemy will be derived from GameCharacter but will have varying behaviors such as the user controlling the player movement and attacks whereas the enemy's movement will be based on the player's position and range to its current location. If we later change the player, this should not affect the enemy's movement. This allows for high flexibility if a class needs to be changed or modified in the future.

2.2 GAME MECHANICS

Players will progress through a cave. Enemies will patrol the cave and attempt to kill the player. The player will have a sword and a bow to defend themselves. Combat will be like that of our researched games Stardew valley & link to the past. Players can approach enemies and hit them from a certain range or, take it easy and shoot from a distance.

All player input will be controlled from an android controller. The project will use a gyroscope to determine the position of the controller. This will allow us to create systems that allow the player to rotate the controller in a way to perform an attack. E.g. shake and flick for the charged attack.

Puzzles at the end of each level will challenge a player. This will make the player stop and think. Puzzles can become easier if the player pays attention as they traverse each level as there will be hints on where or how to solve each puzzle. This will give the player a sense of wonder and give them the motivation to explore each level to its full extent.

Powers ups can also be found within the level to give the player an advantage. Power-ups will include but not limited to increased visibility, increased attack power and defense, invincibility.

The player's eyesight will be limited. This will a challenge to the game. It also gives the player incentive to search the cave more for power-ups to increase their eyesight.

The player moves through a maze. Player attacks any enemies along the way. The player can collect power-ups that will help them. At certain points, the Player must solve a puzzle in a mini-game (TBD). Once the player has completed the mini-game, they progress to the next level.

2.3 PROTOTYPE/STORYBOARD

Figure 1-Prototype Design Unity

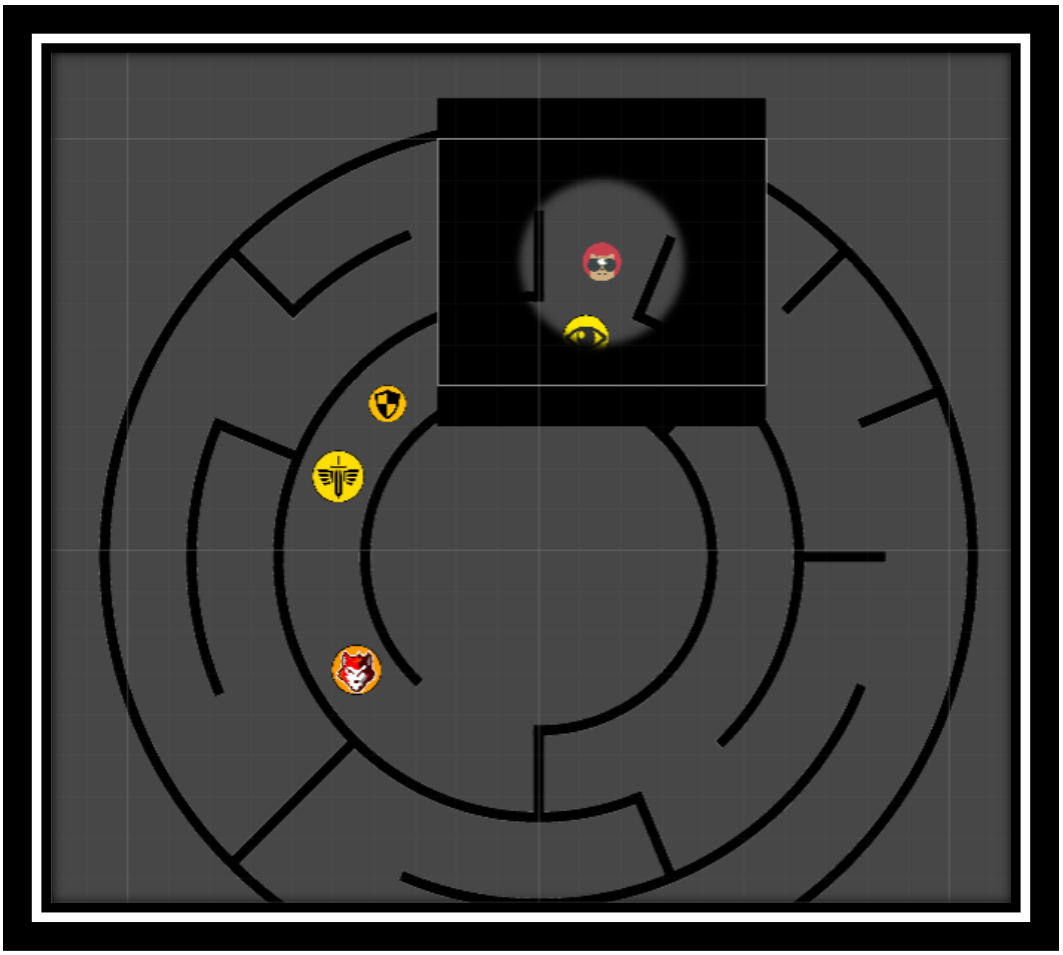


Figure 3 - In-Game View Unity

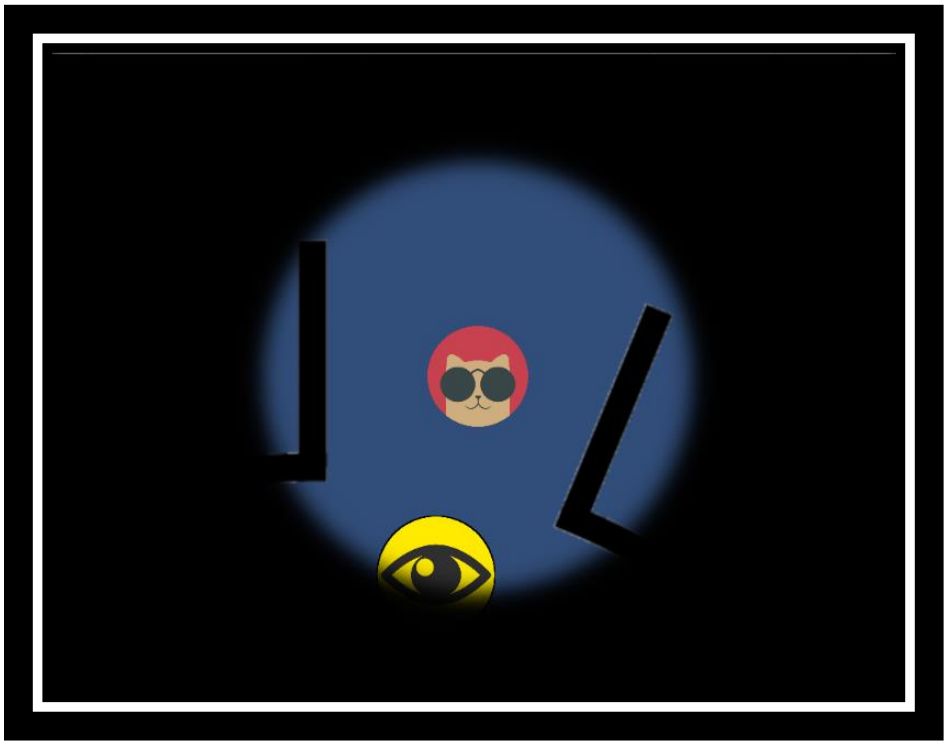
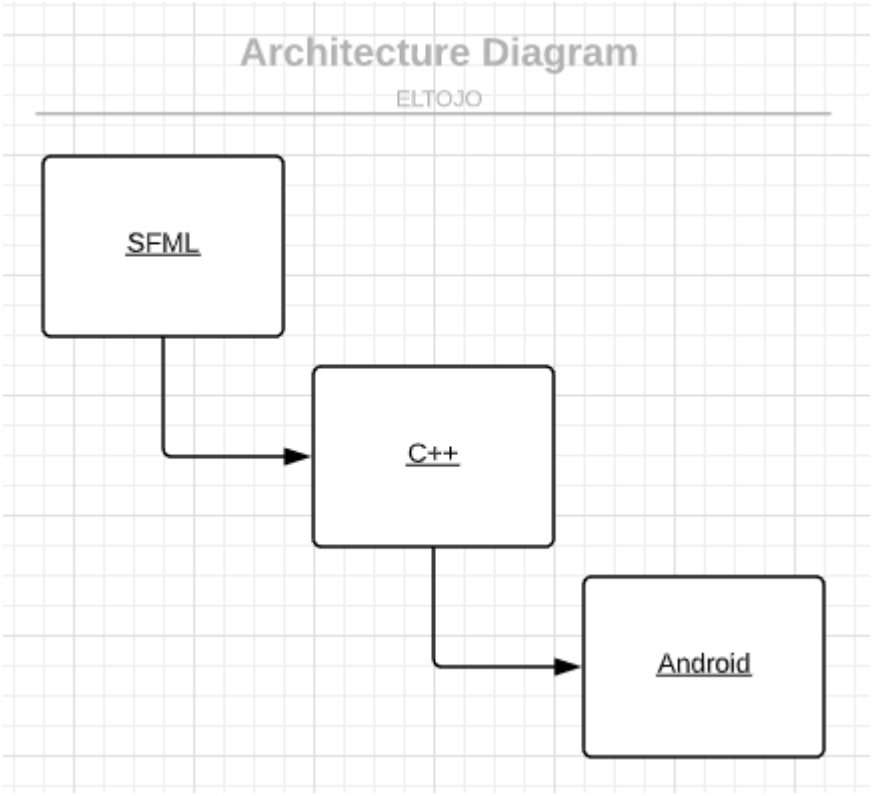


Figure 4 - Enemy and Player Prototype design

2.4 ARCHITECTURE DESIGN

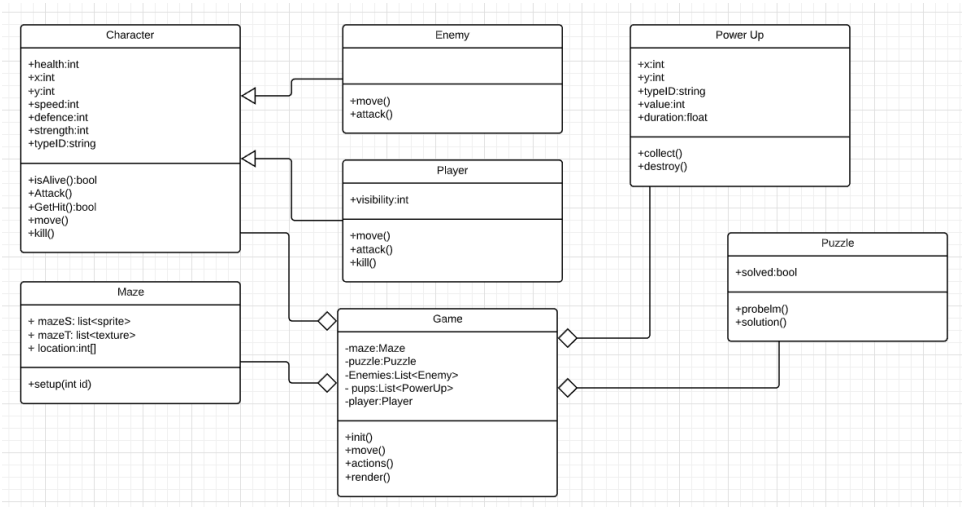
The main architecture of the system is based solely around C++ and the Simple Fast Media Library. The most difficult part of the system is that it needs to communicate with the Android Operating System. This isn't too bad for a language like C++; therefore, the system will use the C++ code libraries to communicate with the Android OS. The team isn't sure how to accomplish the communication; whether it will be Bluetooth, a network connection, or another unresearched method. It seems that the network connection might be the most reliable.

Figure 5: Architecture Diagram



The class and object structure as shown in Figure 6 is a basic overview of what the team will begin with. By the third sprint, this is expected to change. The system will use a generic Character object and base Enemy and Player from its behaviour. The power-up object will hold all stats about a power-up that can be created. The puzzle object will hold the data involved in the puzzles. The Maze object will hold all information regarding the maze such as sprites and textures. The Game class will be the controller for how each of these objects interacts with the system.

Figure 6: Class Diagram



2.5 IMPLEMENTATION

The project will be using the methodology of Object Orientated Programming over a Structured methodology. The reasons for this are that with OO, we have a lot more flexibility and readability with the code. We can create a basic class for game characters and use inheritance to create different types of GameCharacter. This will allow us to create objects with similar behaviour without having to duplicate code. OO will also allow the creation of generic objects to help with game mechanics such as power-ups and pick-ups. This is very helpful as the game might incur a lot of enemies or pickups and OO allows us to create these objects without having to duplicate unnecessary code. OO will also allow for some code hiding to make the project easier to look at and read; having complicated methods encapsulated in base classes and simply calling the functions needed in another. The project will be using a camel-case type naming structure. Variables will be written lowercase first and uppercase second. Methods will be a capital case.

2.6 QUALITY ASSURANCE PLAN

Deadlines: One risk of software development is missing deadlines. To ensure we do not miss any deadlines we will meet weekly to discuss upcoming deadlines for certain module uploads or features to be implemented.

The complexity of implementing an android controller: The main drawing appeal of our game is the android controller. However, we know this to be the hardest part of the project. If the android feature turns out to be too much of a challenge, then a gamepad controller will be used instead.

SFML: We are building the game using the SFML library however we have little experience with using this library. To overcome this and instead of waiting for SFML to be covered in class we have taken the initiative and researched done completed an SFML course on LinkedIn to help start our learning of SFML

Version Control: Through each iteration, we may encounter a bug that has been caused by a previous feature that was added. To overcome this, we will be using GitHub to track each iteration of the build and through each commit. This ensures that if any issues occur that we can rollback

Clear memory: To ensure high optimization the game will utilize deconstructions for objects and pointers in order to allocate dynamic memory and free up unnecessary memory. This will be done due to C++ not having a built-in garbage collector. Without this, there would be a lot of dangling pointers and wasted memory.

3. EXPECTED PROJECT RESULTS

By the end of the project we should have, a fully working deliverable maze type game. Circular maze with enemies to fight, power-ups, and a puzzle or two to solve. The projected amount of levels is 3, and we aim to have a working leaderboard. We aim to have a communication link between the game and an android device to make use of its internal sensors.

3.1 MEASURES OF SUCCESS

The game needs to be playable; which means a player must be able to complete each level. The game needs to connect to an android device and make use of the sensors to enhance the gameplay. Our circular type maze needs to be implemented to make it stand out from others. For aesthetic purposes we have chosen to use 2.5D dimension, the art style should cope with this. Our player movement, with a circular design, we've adopted to go with either 8-way direction or a fluid movement style.

During the development of the project, we will be using a success rubric as shown below in Figure 7. The project will focus on these, and upon achieving the acceptable outcome, this will be the measure that the project is successful.

Figure 7 – Success Rubric

Aspect	Acceptable Value / Range	Current
Input Lag	15-40ms	
FPS	30-60	
Complexity	< 10 (Cyclomatic)	
Runtime Errors	0	
Compile Errors	0	
Android Connection	Yes	
Controller	Android / Xbox	
Sensor usage	Yes	
Mini-Puzzles	3	
Leaderboard	Yes	

4. PROJECT MANAGEMENT

4.1 DEVELOPMENT METHODOLOGY

This methodology allows us to have weekly meetings which will maintain our focus along with maintaining a clear vision on what we wish to work on for any given week. This will allow us to implement features on a regular base as well as ensuring we can complete regular testing to ensure that each game feature is working as attended.

This will be done through unit testing along with white & black box testing. Whitebox testing will allow us to commence testing at an early stage of production as this type of testing does not require a GUI to be available. Black box testing will be used to test components of the game where the functionality of the software is not known. An example of the testing the team will be using is as shown in Figure 8 below.

Figure 8 - Testing example

Testing Example				
Method	Condition	Expected	Actual	Result
Boundary	Enemy Removed from character array	No Error		
Boundary	Player Removed from character array	No Error		
Boundary	Enemy added to character array	No Error		
Boundary	Player added from character array	Error		
Boundary	Power up added to list	No Error		
Boundary	Power up removed from list	No Error		

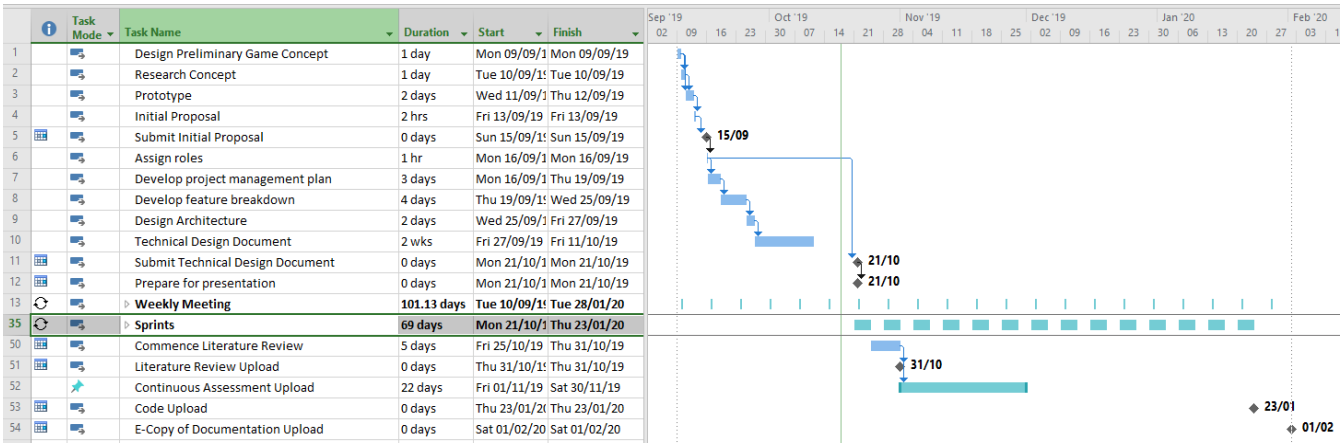
It will also allow us to take advantage of scrums. These scrums will allow us to complete sections of the game during sprints. This will allow us to implement features at a regular pace.

By breaking down the project into manageable units we can focus on high-quality development testing and collaboration. By conducting frequent builds, testing and reviews during each iteration quality will be maintained while finding and fixing any bugs or problems we might run into.

We have chosen to use the agile methodology approach to software development. We have chosen this as from what we learned last year of the different approaches to software development we found this to be the most effective when developing projects. This is due to the constant pace at which features are implemented along with a maintained pace of testing and reviewing.

4.2 SCHEDULE

Figure 9 - Gantt Chart



	Task Mode	Task Name	Duration	Start	Finish
1		Design Preliminary Game Concept	1 day	Mon 09/09/1	Mon 09/09/19
2		Research Concept	1 day	Tue 10/09/1	Tue 10/09/19
3		Prototype	2 days	Wed 11/09/1	Thu 12/09/19
4		Initial Proposal	2 hrs	Fri 13/09/19	Fri 13/09/19
5		Submit Initial Proposal	0 days	Sun 15/09/1	Sun 15/09/19
6		Assign roles	1 hr	Mon 16/09/1	Mon 16/09/19
7		Develop project management plan	3 days	Mon 16/09/1	Thu 19/09/19
8		Develop feature breakdown	4 days	Thu 19/09/1	Wed 25/09/19
9		Design Architecture	2 days	Wed 25/09/1	Fri 27/09/19
10		Technical Design Document	2 wks	Fri 27/09/19	Fri 11/10/19
11		Submit Technical Design Document	0 days	Mon 21/10/1	Mon 21/10/19
12		Prepare for presentation	0 days	Mon 21/10/1	Mon 21/10/19
13		**Weekly Meeting**	**101.13 days**	**Tue 10/09/1**	**Tue 28/01/20**
35		**Sprints**	**69 days**	**Mon 21/10/1**	**Thu 23/01/20**
50		Commence Literature Review	5 days	Fri 25/10/19	Thu 31/10/19
51		Literature Review Upload	0 days	Thu 31/10/1	Thu 31/10/19
52		Continuous Assessment Upload	22 days	Fri 01/11/19	Sat 30/11/19
53		Code Upload	0 days	Thu 23/01/2	Thu 23/01/20
54		E-Copy of Documentation Upload	0 days	Sat 01/02/20	Sat 01/02/20

Figure 10 - Game feature

Feature	Android Controller	Combat	Enemies	Puzzles	Power ups	Maze	Movement
What is it?	Allow user to use an android controller to control player. Attacks such as shaking the controller.	Melee combat. Ranged combat also availble. Attack strength will be determined by the	Varies of enemies that patrol the player and attack the enemy.	End level puzzles to progress. - Fix the pattern - movement sliding	Power ups the player can pick up. Such as improved visibility,invincibility , increased health.	Players to traverse the maze to complete the level	How each GameCharacter will move.
Sprint 1							Player/enemy movement(2hrs)
Sprint 2		Enemy attacks player. (1hr) Player attacks enemy (1hr) Enemy/player loses health(1hr) detect enemy in attack area(2hrs)			health increase(1hr) invincibility(1hr) attack increase(2hrs) vision increase(2hrs)		
Sprint 3			Create enemy movement behavior(5hrs)				
Sprint 4			Create second enemy type(3hrs)				
Sprint 5				Puzzle type 1- Match the colours(5hrs)			
Sprint 6				Puzzle type 2 - sliding ice (6hrs)			
Sprint 7						Maze collsion(6hr)	
Sprint 8						Creating level deisgn(5hrs)	
Sprint 9						placing enemy,player,puzzles into level	
Sprint 10	connecting to android (8hrs)						
Sprint 11	Mapping android controls to player movement and actions						
Sprint 12	Additional features/quality of life improcements						
Sprint 13	Additional features/quality of life improcements						
Sprint 14	Additional						

4.3 BUDGET

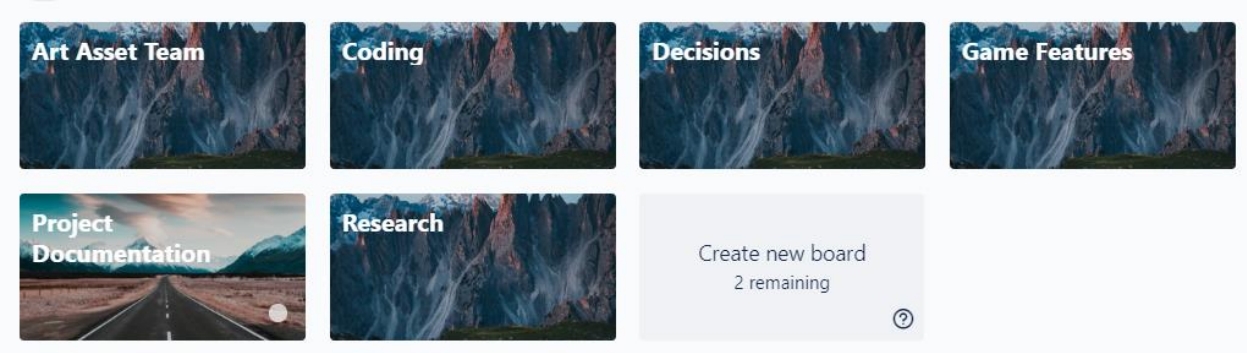
No budget required

4.4 COMMUNICATION & COLLABORATION PLAN

A swathe of collaboration tools will be used to communicate and effectively collaborate. The project began with Favro as it was the most advanced and effective tool for task management and had a built-in Gantt chart system however later moved to Trello shortly after due to pricing issues. Trello has been set up into several boards with each board tracking different aspects of the project. This was done to keep completely different sections of the project in separate areas for efficiency e.g. project documentation and research have separate boards.

Each member of the team is expected to attend a weekly meeting. These meetings will be the main source of weekly tasks. These meetings are broken up in each Trello board. Each board is discussed for a period and if there any new tasks that need to be added to the board, we take notes of said task then later add it to the board. See figure 11 to see Trello boards.

Figure 11 - Trello boards



Each board is split into *to do, doing & done*. After each weekly meeting, one of the meetings a project member writes up that weekly tasks or tasks that will need to be completed in the future. Each task is given due date in order to keep up a steady pace of progression. During each meeting, we delegate who would be best suited for each task and if said member agrees to take on that task. Once a task has begun being worked on said member moves that card into doing. Once they have this done, they then move it into doing.

So far Trello has been an efficient and simple service that has immensely helped with project productive and tracking. Being able to see what any given member is working on or has still to complete is effective for the project in order to track the pace at which each feature is being implemented.

Along with Trello for task management the team also uses GitHub for code management and version iteration. GitHub is being used to keep the game project code in one easily accessible place. This allows each member of the team to easily upload each change they make to the project.

With each code commit, all members can see what files have been changed and what exact lines in each file have been changed. This allows everyone to keep track of each change to the game. GitHub also allows version control. If in the case of a new commit that inadvertently breaks something in the game project you can simply roll back to a previous stable build. Also due to the fact we are using the SFML library which requires some configuration inside of visual studio means we can easily set up the game project to work on each member pc. All they simply must do is have the same path for SFML as the one in the config.

For communication, we are primarily using slack. Slack was chosen due to its wide range of apps that integrate seamlessly with GitHub and Trello. It allows us to track trello tasks inside of slack. This increases efficiency in communication as you can move tasks through slack instead of having to open the main app. This also allows allocating tasks through slack. This comes in handy when we must create a task out of college hours. All that has to be done is simply contact the person that would be best suited for that given task and you can create a task, assign them to it all within the app

Discord is then used primarily for our weekly video calls. These calls are focused on asking each member what they have done so far this week, how long they expect it will take them to finish their tasks & if they require help. This allows the secondary meeting to act like a “checkup”.

Gantt charts are being used to manage deadlines along with sprints. Having a Gantt chart allows for project management to ensure that project documentation deadlines are known and keep on top of. This along with trello allows for the efficient allocation of tasks.

Alongside the Gantt chart, a game feature chart was made. The chart explains each key feature of the game project and breaks it down into parts. Each part of the feature will then be completed incrementally completed in weekly sprints that the team will complete. Theses sprints are aimed at implementing key gameplay features weekly to progress through the months. Each part of a feature is also given an estimate of time to complete.

5. REFERENCES

Anon., 2013. *Intelligent 2D Collision and Pixel Perfect Precision.* [Online] Available at: https://www.gamedev.net/articles/programming/general-and-gameplay-programming/intelligent-2d-collision-and-pixel-perfect-precision-r3311/ [Accessed 20 October 2019].

Anon., 2019. *List of maze video games.* [Online] Available at: https://en.wikipedia.org/wiki/List_of_maze_video_games

Anon., 2019. *Maze Generator.* [Online] Available at: http://www.mazegenerator.net/ [Accessed 15 August 2019].

Anon., 2019. *Pac-Man.* [Online] Available at: https://en.wikipedia.org/wiki/Pac-Man

Anon., 2019. *Puzzle Video Game.* [Online] Available at: https://en.wikipedia.org/wiki/Puzzle_video_game [Accessed 20 October 2019].

Anon., 2019. *Stardew Valley.* [Online] Available at: https://en.wikipedia.org/wiki/Stardew_Valley

Anon., 2019. *The Legend of Zelda: A Link to the Past.* [Online] Available at: https://en.wikipedia.org/wiki/The_Legend_of_Zelda:_A_Link_to_the_Past

Anon., n.d. *ISO 9126 Software Quality Characteristics.* [Online] Available at: http://www.sqa.net/iso9126.html

Josikakar, n.d. *Software Engineering | Coupling and Cohesion.* [Online] Available at: https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/

Systems, S., 17 auguest 2017. *SFML 2.4.x Tutorial 013 - Pixel Perfect Collision Detection,* s.l.: s.n.

https://wiki.allegro.cc/index.php?title=Pixel_Perfect_Collision