Final Report

Title page

Name: Thomas Wenham

Registration Number: 1704969

Supervisor: Joseph Walton-Rivers

Second Assessor: Liu, Zilong

Degree Course: Computer Science (Computer Games)

Acknowledgements

Joseph Walton-Rivers (Supervisor) – Advice, ideas and misc. programming support.

Abstract / Summary

This document details the Tile-based RPG that I have created. It is written from scratch – a custom game engine – meaning that all the code is either detailed in the technical documentation or is part of the Java language. The game is played on a freely explorable grid-based world viewed from a top-down perspective. The player has access to a range of abilities with which to defeat enemies – using different equipment, they can use short and long range attacks with different effects. Enemies are controlled by a simple artificial intelligence that decides how they should move and attack the player. There is a system for loading levels; text files of a special format can be read to populate the game's grid system. All graphics are drawn by me; static objects were made to be repeating when put together. There is a menu system that allows the creation of hierarchical menus and a user to navigate the menus. In the game, this is used for an inventory that the player uses to view their items. The final outcome is a game that incorporates all of these elements.

Table of contents

1. List of symbols

The following is a list of abbreviations and words with non-standard meanings; that are used in this document.

- RPG
  Role-playing game. In this document, RPG will refer to the genre of video games, typified by its use of quests from which characters gain experience points to improve various attributes [1] – such systems coming from role playing board games, specifically *Dungeons & Dragons* (1974).

- Game, the Game
  Depending on the context it is used in, may refer to either the whole developed product, the software that makes it up, the specific Game class in code, or as an instance of the program running such that a person could play it.

- GO
  For brevity, GameObject is sometimes abbreviated to GO. GameObject itself is a Class of Java object specific to the game. Full definition of GameObject is proved in the Specification.

- NPC
  Non-player character. In-game characters that are not controlled directly by the player.

- Lag

  In this document, lag refers to time spent waiting after performing an action (in-game). Related to but distinct from standard definitions, since in terms of games it usually refers to a delay of an action occurring after it should, for example of players in an online game. Whereas this definition originates from the Super Smash Bros. community [2]. This may be referred to as cooldown or recovery in other games.

2. Main Text

2.1. Context

The problem to be solved is ‘How do you create a Tile-Based RPG’. In order to solve this, I will look at the context of how game design concepts AI are used in recent games, show its importance by looking at similar games and consider important social factors.

2.1.1. General Game Design

The game has real time combat, so I had to look how the problems when designing combat were solved in other games. While developing *God of War* (2018), Mihir Sheth (Sony Santa Monica Studios) [2] identified 4 key features of combat. These were:

1. “Responsive and combo-oriented gameplay”
    - The player should receive immediate feedback and rewarded for playing well e.g. with combos
2. Fighting groups of enemies
    - The player should be able to split attention to fighting many enemies and not follow certain games with slow and defensive 1 on 1 combat.
3. Having attacks that “move you forward physically through the space”
    - When fighting a group, being able to push the enemies back and build momentum
4. “Gameplay and controls are accessible, hassle free and are on the player’s side”
    - Of course, the controls should be good

While *God of War* is a 3D over-the-shoulder action game, these concepts can be applied to the combat in a 2D top-down view game.

While *Horizon Zero Dawn* (2017) is another modern 3D action game, the level design is another transferrable element. As Blake Rebouche (Guerrilla Games) discusses [3], the purpose of level design in an open world game is to provide a wide area where the player has to figure out how to traverse it, using level geometry to “funnel” the player towards important areas. He also talks of “macro level design” -- connecting open spaces with corridor to be able to control the player’s experience and “micro level design” -- to create the open spaces with multiple paths through to give players freedom.

2.1.2. Artificial Intelligence and Games

Although AI is not a focus of the developed product, there is a wealth of AI research. Some of it by academics researching its use in games and also independently by game developers.

For example, the behaviour of enemies in *God of War* (2018) [2] has enemies avoid clustering together so that they are always visible to the player. It also has a token system for enemy aggression so that combat is kept fair for the player. While in *Uncharted 4* (2016) combined scripted behaviours and automated decision making [4]. Naughty Dog used many different techniques such as a ‘Heat’ system -- places player might be are ‘hot’, in the NPCs vision is ‘cool’, defining limits on the number of enemies at certain points to prevent clustering and giving enemies different roles so that decision making can be varied.

As described by Yannakakis et al [5] , the use of planning algorithms and procedural content generation can produce more interesting gameplay such as in *F.E.A.R. (2005)* and *Fallout 3 (2008)* and that procedural content generation is found in many critically and commercially successful games such as *Spelunky (2008)*, *Minecraft (2011)*, *Diablo 3 (2012)* and *Civilization V (2010)*.

Van der Linden et al [6] explain some approaches to the procedural generation of a 'dungeon' level such as Cellular Automata, Generative Grammars and Genetic Algorithms.

2.1.3. Similar Games

The Legend of Zelda (1986) [7] is a major inspiration for this game. Despite lacking traditional RPG mechanics such as experience, it is considered as a spiritual predecessor of modern action RPGs and open world games. Common elements include the 2D top-down view, tile-based map (although Link is not restricted to the grid) attacks such as Link's sword, arrows, bombs, key-door puzzles as well as the general openness and aesthetic.

Legend of Grimrock 2 (2014) [8] is another action RPG although it is rendered in 3D with a first-person perspective. The similarities are its non-linear gameplay, the strict grid-based movement but also real-time combat.

Crypt of the NecroDancer (2015) [9] is also a 2D top-down game with grid-based movement, with combat and enemies fitting similarly to the grid. The difference is that it is also rhythm, which the developed game is definitely not. Furthermore, *Crypt of the NecroDancer* is a roguelike [10] game rather than being an open adventure.

2.1.4. In Society

Accessibility: The developed product has some issues regarding accessibility. Only keyboard input is natively supported, meaning that it would not be possible (or requires external software) to play if one cannot use a keyboard. Furthermore, two hands are intended to be used to player, so if one can only use one hand, it makes the game more difficult to play. Another issue is that text is not scaled well, meaning that it may be too small for some people to read.

Legal: Some assets that have been created take heavy inspiration from outside sources and may be infringing on a 3rd party's copyright. There could be other copyright issues if graphics or music were added from other artists – does the designs and assets become mine as the 'owner' of the game or do the original artists maintain their Intellectual Property? Also, there could be a legal issue if I were to distribute the game to minors since it has not age-rated by an agency such as the VSC (Video Standards Council) Rating Board.

2.2. Objectives

The following are the objectives outlined before development started:

- A fun and interesting gameplay loop
- A large open-world map with things to discover, in terms of the world and the gameplay
- Either procedural generation of levels or a system for level creation
- Satisfying controls
- A variety of attacking options for the player

- A variety of enemy types, and multiple different AI behaviours.
- Modularity so further additions do not impact (in terms of the code) on existing parts
- Theme must not be Fantasy or Sci-fi.

2.3. Specification

The specification is split into broad sections based on their function. Each section is broken down to better represent how the section's title is actually implemented.

2.3.1. Game Logic

Summaries

The main part of the game that handles the running of the game, the main physics and control by the player and for NPCs.

2.3.1.1. Game

The entry point to the program. Contains the main game loop that handles the running of the game – with some parts processed within it and the rest deferred to other classes – in this case acting as an interface between them. Mainly concerned with the directing program flow and managing the game database.

Full Documentation

2.3.1.2. Keys

Processes the user's keyboard input. Defines what certain to do when keys are pressed by setting predefined flags that can be read elsewhere in the program.

Full Documentation

2.3.1.3. Controller

Control system for NPCs. Similar to Keys but does not have external input. Extensions can use basic AI to decide. Implemented AI that chases the player if they get close, and attacks when possible.

Full Documentation

Technical Achievement

A custom game engine created, comprised of expected components; game physics system, user input management/player control, an AI system with a basic AI implemented.

2.3.2. Game World Representation

Summaries

Programmatic representation of the world that the game is played in.

2.3.2.1. GameGrid

The 'grid' central to this "Grid-based RPG". A 2D array of GameObjects with an operation to move them around on it.

Full Documentation

2.3.2.2. HitboxGrid

Data structure that allows hitboxes to interact with each other and objects on the GameGrid. Actually a list and not a grid.

Full Documentation

2.3.2.3. Hitbox

Models the game concept of a hitbox in relation to this game. E.g. they have a grid position and stats relevant to the game.

Full Documentation

Technical Achievement

Created GameGrid which is the ‘physics` of the game that handles collisions between the objects within it. HitboxGrid system created as a data structure for storing hitboxes and handling their lifetimes. Hitbox created as a system for modelling damaging effects. It allows flexibility, such as being able to move by itself, have its own sprite -- or not have one – have different lifetime, be able to not affect certain game characters and be able to increase its area of effect.

2.3.3. Game Objects

Summaries

All objects that exist as entities in the game world.

2.3.3.1. GameObject

An object that sits in the game’s grid system. Basic GameObject only has a position, but extensions add many more properties. Examples include characters (player and NPC), doors, signs.

Full Documentation

2.3.3.2. OffGridGO

An object that is in the game world but does not take up a grid space. This is so that it may overlap with other objects and/or move smoothly between grid spaces.

Full Documentation

Technical Achievement

Created a logical hierarchy for in-game objects. It starts with basic objects that only have a position and branches out to varied specific implementations. For example, interactable objects, pick-up-able items and Projectiles.

2.3.4. Menu System

Summaries

A system for creating menus. Defines structure for building a hierarchy, traversing the menu and how to display them.

2.3.4.1. MenuContainer

Container that contains every other element in a menu – always the top of the hierarchy. Also handles how to navigate the menu from user input.

Full Documentation

2.3.4.2. MenuBox

The basic level of sub-menu. Defines operations for adding new sub-menus and for maintaining the hierarchy.

Full Documentation

2.3.4.3. MenuControls

Gets keyboard input in order to control menu navigation

Full Documentation

2.3.4.4. MenuAction

Used by programmer implementation to define what to do when a sub-menu is selected.

Full Documentation

2.3.4.5. MenuOption

Used by programmer implementation as a sub-menu that executes a task when selected.

Full Documentation

2.3.4.6. MenuGrid

A specialised type of MenuBox that displays as a grid (different to default)

Full Documentation

Technical Achievement

Created something like a library for Menu implementation, defining an original implementation of a Tree that also handles traversing the tree, and is able to display it graphically.

2.3.5. In-Game Menus

Summaries

Implementation of the MenuSystem specific to the game's menus.

2.3.5.1. PauseMenu

Specific implementation of MenuContainer to represent the menu when paused.

Full Documentation

2.3.5.2. InvMenuGrid

Specific implementation of MenuGrid that represents the inventory screen – displays inventory in a grid.

Full Documentation

2.3.5.3. InvSlot

Specific implementation of MenuBox and MenuAction specific to holding Items.

Full Documentation

Technical Achievement

Demonstrates how the MenuSystem can be used to create actual menus and the flexibility of it.

2.3.6. Rendering

Summaries

The system that allows the game to be rendered on-screen

2.3.6.1. MainFrame

The frame, i.e. window in which the game is displayed. Can be full screen.

Full Documentation

2.3.6.2. DrawingPanel

The panel (Swing) the all parts of the game are drawn onto. This handles positioning while it delegates to objects to drawn themselves.

Full Documentation

Technical Achievement

Created a game rendering system using only basic operations built-in to the language. It manages drawing the game world, in-game objects, HUD, menus and the textbox in the right place, while each of these define how to look themselves. Mathematics is used to calculate how to render the game to fill and be centred to the window and similarly for the menu inside it.

2.3.7. Utilities and Other

Summaries

2.3.7.1. LevelLoader

System to load (and save) levels from a file into the game.

Full Documentation

2.3.7.2. ImageManager

System for loading images (for the game – game assets) from files and making them available to rendering system.

Full Documentation

2.3.7.3. TextboxWriter

System for writing text to the screen so that it appears letter by letter instead of all at once.

Full Documentation

Technical Achievement

Created an original (text) file format for storing level data and a kind of parser to read it create a level from it. Also for the reverse; get level data and write a file from it.

Used Java tools to create a system that can read all files in the graphics directory, convert them to an Image object and save them to a Map with the filename (no extension) as a key.

Created simple system to gradually write text. From Game and Keys: Made it so that the user can skip the text, and close the textbox when they want to, while only using a single button.

2.4. Testing

2.4.1. Development Testing

This section will be a selection of the most significant problems and bugs encountered during development. Testing was continuously held during development to reduce the number of bugs before a new feature was implemented.

- When implementing screen scrolling, walls appear to move around the map and the physics do not match the graphics.
- Frame rate display always shows 62
- Hitboxes can pass through everything
- ‘Dead’ objects stay in the game
- When shooting, the shot goes the opposite way to the way the player is facing
- Enemies always try and stay 1 tile away from the player
- Enemies can never hit the player
- Enemies can always hit the player, instantly killing them
- Attacking graphic is always rotated 90° to the correct orientation
- Sign dialogue instantly closes when trying to skip to end.
- Cannot navigate menu unless running with IDE’s debugger.
- Items cannot be picked up
- Items are rendered on screen even if they should be out of view
- ‘Grenade’ object’s ‘explosions’ spawn only on one side.

2.4.2. Play Testing

The following is a list of comments on the gameplay that were shown through playtesting.

- The controls are responsive. Starting and stopping moving, and changing direction occur without a lag.

- However, the lag that stops movement while attacking feels poor.
- The control layout is not very intuitive or comfortable
  - With movement on arrow keys
  - Attacks of x to v
  - Pause on p, nowhere near other keys
  - Also randomly having t and g bound
- The shooting effect is hard to track on screen
- Combat is somewhat confusing, as there is a lack of feedback
  - When player is hit, their health bar decreases but nothing else
  - For enemies, only feedback is if they die (or in debug mode and can see their health)
- The menus are intuitive to use
  - Albeit do not have much functionality
  - Inventory is neat, but has no real use
- There is no way of knowing what door a key would open
  - Also, door sprite could be more representative of a door
- Attacks either do not do enough damage, or the basic enemies have too much health
  - They take several hits to take down
  - Especially true to the bomb attack, which looks like a big explosion but does the same amount of damage as regular attack but takes much longer to do.
- Can throw 'bombs' in very quick succession, creating overlapping explosions, seeming unfair
  - Somewhat contradictory to the previous point, but actually valid

2.4.3. Designed Tests

2.4.3.1. MenuTest

The MenuTest class was designed to be able to test all aspects of Menus that had been developed.

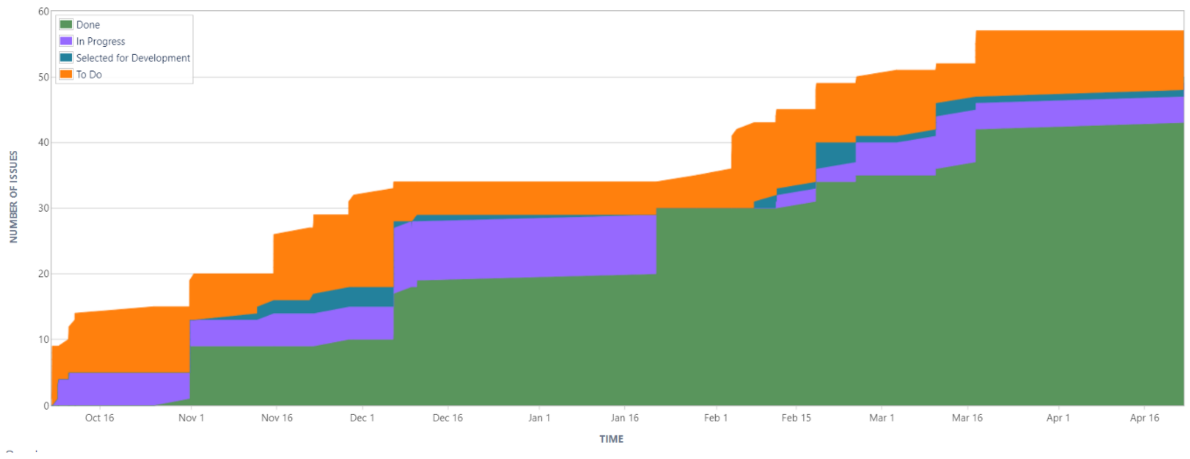
Result: Graphical menu displayed accurately. Is able to reorient to different sizes but has a problem with flickering. Able to navigate up and down the levels of menus. When selecting and option, shows that re-direction or executing another action is possible.

3. Project Planning

3.1. Maintaining Momentum and Adapting to Change

I think I was quite successful at maintaining momentum, making good progress every week during term time. I did however find it difficult to maintain momentum during vacations. The cumulative flow diagram (below) shows this disparity. This diagram is not a perfect indicator as it relies on the number of issues, when some issues may involve more work to complete than others. It would also require issues to be updated frequently, whereas I would update many issues all at once.

Figure 1 - Cumulative Flow Diagram of Issues (from Jira)



3.2. Identifying and Dealing with Risks.

I cannot say that I was successful at identifying risks as only 1 risk has been logged on Jira. As such, risks were dealt with when they became an issue. For example, a risk I did not identify was that other modules would impact the time I had to focus on this project. I dealt with this by reducing output when another module became higher priority (i.e. before a deadline) and then do more than usual afterwards. The one risk that was logged was that adding new classes (specifically extensions of GameObject) may cause previously developed systems. This had to be dealt with as it came, with a fix applied when it needed to be. This however may have led to lower quality code which could have impacted the pace of further development. I did not identify a pandemic as being a potential risk, though I could deal with it as I had already been able to work fully from home.

3.3. What I Have Learnt

I have learnt that:

- Over-scoping is practically unavoidable
  If you are starting from scratch, then it is impossible to know what can be done in the given time. It is easier to think in terms of what is wanted to be present in the final product. It is easier to start aiming for a bigger and favourable goal and then cut back than it is to reach a smaller goal and then try to make it more of what was wanted in the first place – having to redefine scope mid project.

- Being your own project manager is confusing

When developing something alone it is more natural to think: “What am I going to do now?” and then do it, rather than “What am I going to do after I tell myself what I just did and what I am about to do”. And maybe what you want to do next is not what you think you *should* do next.

- Estimating how long an unknown process will take is difficult and finding how long it actually took is difficult too
  Without using specific application because it is very unlikely that you will work on something for start to finish in a single sitting without stopping or being distracted. Trying to compare guessed time estimates with guessed time-taken seems to not be a very useful statistic.

3.4. Methodology

My methodology did prove suitable, as I was able to maintain a steady pace of development. Rapid cycles of development and testing allowed features to be built up on top of each other – making sure that the previous feature is working as intended before moving on. This is suitable as this is a individual project, meaning that it is okay to leave it in a broken state because the one person working on it will know what the problem is to be able to fix it – do not have to communicate this to someone else.

Areas which my methodology was not so successful was in the use of Jira’s management tools. On many occasions an Issue would be created for a task has already been completed. This is better than having not done the related work but worse than having foreseen that work and creating the Issue beforehand. Another problem is that some issues were made and then left unresolved. This was either because it was an abstract or vague ‘Story’ and not a descriptive task or because it was something that was thought should be done but did not become a priority.

4. Conclusions

4.1. Comparison to Objectives

4.1.1.

Tile-based RPG:

The game is based on tiles, which I have referred to as a grid throughout. Some parts of the game have moved away from the original grid, such as the descriptively named OffGridGO.

The game has few RPG mechanics. It has stats: HP, ATK, DEF that are a staple in RPGs but are found in non-RPGs as well.

4.1.2.

A fun and interesting gameplay loop:

'Fun and interesting' is completely subjective, so it is difficult to say if this objective has been met quantitively. Furthermore, no further details of gameplay were decided upon at the time of writing the first objectives, so one cannot compare the gameplay produced to any requirements as there were not any.

However, this is not to say that we cannot judge whether what was produced is 'fun and interesting' or not. Though it was not formally stated, it was always my intention to base the gameplay on *The Legend of Zelda* [7]. I would say that the game is not particularly 'fun and interesting' as is even though the gameplay matches *Zelda* quite closely. This would be mainly due to the lack of content in the game.

4.1.3.

A large open-world map with things to discover, in terms of the world and the gameplay:

The map is somewhat large, but this is distorted by the size of the view. So, while the map is many screens worth of width, the actual size of the world is not very big. Making the world larger would not fix this problem as there is not much to do anyway. There are no objectives or secrets with meaningful rewards. The gameplay stays the same throughout with no progression, so there is nothing to discover in that regard either.

4.1.4.

Either procedural generation of levels or a system for level creation:

Procedural Content Generation (PCG) turned out to be far beyond the scope of this project. Along with the obvious problem of actually coming up with a procedural generator, the game would need to be adapted to make procedurally generating content have a purpose such as having more variety of objects and graphics for existing objects as well as having better defined gameplay.

A dedicated tool for level creation was also far out of scope. However, the system that was created for storing levels does allow for easier level creation. The format is a text file that is human readable

– provided that they understand the representation. The file shows a representation of the grid with spaces being occupied by certain characters for different object. Also, the format also allows it to be opened into a spreadsheet application to be able to visualise the grid – where each cell is one object – instead of having to assume the alignment in the text file is correct.

4.1.5.

Satisfying controls:

Significant time was spent making the movement system feel good. The end result is a system with a buffer that allows the player to naturally switch between directions by pressing their next direction before lifting off the previous direction's key. The controls are satisfying in this regard; however, this is not the case for attacks. The player cannot move at the same time as attacking, which is rather unsatisfying. This partly a game design choice and partly due to a limitation of the game's systems.

4.1.6.

A variety of attacking options for the player:

I was quite successful in this area. While 3 of the 5 suggested types were fully developed, others would be less difficult to add, as there are now systems in place to handle it.

I was able to implement: Short range attack – spawns a hitbox directly in front of the player. Shooting attack – spawns a hitbox directly in front of the player that then moves quickly in the direction the player was facing. Distance attack – spawns an area of hitboxes far from the player.

The other two were: Area of effect – this would be almost trivial to add, as it is almost already in the game. Trap – would require changing OffGridGO system to allow more interactions with enemies.

4.1.7.

A variety of enemy types, and multiple different AI behaviours:

Development of multiple AIs was also a stretch of what was possible to have in this project. What was actually developed: Random – pick random moves, attack if the player gets close, Chase – move towards the player and attack when in range.

These would probably not be considered 'complex' behaviours as for example, the chase AI cannot navigate around walls if the player is on the other side. Having complex AI was would be because it is better for gameplay to have enemies with multiple behaviours, rather than it being an important part of the project.

4.1.8.

Modularity so further additions do not impact (in terms of the code) on existing parts:

This is another area that is hard to measure because it is not a yes/no question. A fair attempt was made at keeping the code modular but as time went on it likely became less so, as new things were added that did not fit with what was already there and so needed a special case. An example of success in modularity is the MenuSystem. It was made as a structure for building menus off of and I was able to do that with the pause/inventory menus. On the other hand, the level files system did not expand well and was somewhat left behind as more things were added and were hard coded to be loaded into the level.

4.1.9.

Theme must not be Fantasy or Sci-fi.
I came up with an idea for the theming to be music based in order to fulfil this objective. However, the lack of art and assets for the game means that the theming is weak. Also, the placeholder sprite for basic enemies, which is clearly fantastical, did not get replaced.

4.2. Overall Results

Overall, I am somewhat pleased with the results that I have produced. I was able produce a game engine from scratch with many useful features, and an original system for producing menus as well. I would have liked to have developed the gameplay further and had a more complete package. I certainly thought that I could do more in the beginning, but I now know that I had underestimated the time needed and overestimated the time available.

4.3. Future Development

4.3.1. Short Term

- Add new enemy types
  Use what has been made for the player to make new types of enemies that use different attacks.
- Improve the current AI
  Add pathfinding to fix the current AI's main problem
- Add experience and drops
  Give the player experience for beating enemies, that increases their stats. Also allow enemies to hold items that they drop when defeated.
- Peaceful NPCs
  Essentially walking signs
- Better level saving
  It needs to save all types of objects. A better system for saving dynamic and OffGrid objects.

4.3.2. Long Term

- Usable items
  For example, make 'grenades' an item so you can't spam them. Gives items a use.
- Shops
  NPCs that you can trade with, giving money a use.
- Quests
  Objectives for the player to give them purpose
- Save files
  Save the current state of the game so data can survive after closing the program

4.3.3. Far Future

Significant development would be needed.

- Tiling system
  So that the tile displayed can be different for the same object depending on other factors
- Good AI
  AI that displays some actual intelligence
- Good Artwork
  Make everything in the game have a unique and good sprite
- Good Gameplay
  More RPG mechanics such as skills, better combat mechanics
- Well-designed world
  Design a world that has places to explore and looks visually interesting.

5. References

[1] W. L. Hosch, “Role-playing video game,” Encyclopædia Britannica, inc., 30 July 2019. [Online]. Available: https://www.britannica.com/topic/role-playing-video-game. [Accessed April 2020].

[2] M. Sheth, “Evolving Combat in God of War,” in *Game Developers Conference*, San Francisco, 2019.

[3] B. Rebouche, “Balancing Action and RPG in 'Horizon Zero Dawn' Quests,” in *Game Developers Conference*, 2018.

[4] M. Gallant, “Authored vs. Systemic: Finding a Balance for Combat AI in 'Uncharted 4',” in *Game Developers Conference*.

[5] G. N. Yannakakis and J. Togelius, “A Panorama of Artificial and Computational Intelligence in Games,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 7, no. 4, pp. 317 - 335, Dec 2015.

[6] R. van der Linden, R. Lopes and R. Bidarra, “Procedural Generation of Dungeons,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 6, no. 1, pp. 78 - 89, March 2014.

[7] Nintendo EAD, *The Legend of Zelda,* Nintendo Co., Ltd., 1986.

[8] Almost Human Games, *Legend of Grimrock 2,* Almost Human Games, 2014.

[9] Brace Yourself Games, *Crypt of the NecroDancer,* Brace Yourself Games; Klei Entertainment, 2015.

[10] Wikipedia, “Roguelike - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Roguelike. [Accessed April 2020].

6. Tables, graphs, figures and equations