Final Year Project Plan

Full Unit - Project Plan

Building a Game

Submitted By Sulaimaan Bajwa

Created in order to plan out the final project critical to the completion of the 3rd year of

Msci (Hons) Computer Science with Software Engineering

Supervised by Hugh Shanahan
EPMS School, Department of Computer Sciences
Royal Holloway, University of London

Abstract **2**

1 Background **3**
1.1 Context 3
1.2 Goals 3

2 Created Materials **4**
2.1 Reports 4
2.2 Proofs of Concept 4
2.3 Testing Tools 4

3 Timeline **5**
2.1 Gantt Chart 7

3 Risk Assessment **9**
3.1 Data Loss[6] - High Impact 9
3.2 Report and Project Imbalance - High Impact 9
3.3 Lack of Generation Knowledge - Medium Impact 9
3.4 Unintended Hardware Creep - Low Impact 9
3.5 Scope Creep - Medium Impact 10

5 Bibliography **11**

References **12**

Abstract

This document serves as the project plan to my 3rd year project, essential to my qualification. During the course of this project I will be attempting to design, create and deploy a video game. I have chosen to create a Roguelike game due their current popularity and approachable but challenging nature. I hope to learn more about Unity as a game engine and how to develop and test with it, with the objective that it will be a strong foundation for my software engineering portfolio.

1 Background

1.1 Context

Video games are an almost unavoidable applied form of computer science. There exist educational tools[1] that also take the form of video games, created especially for their ability to captivate the mind of the young audience, and get them interested in the topics brought forward through creative fun and problem solving.

By definition, a video game is ‘an electronic game in which players control images on a video screen’[2]. True to this definition, games are often much deeper than just sprites, often containing problem solving, a progression system, collections and completion rewards, and specific challenges for exceptional players. To master these individual devices, with the use of exceptional programming can yield incredibly enthralling games, in their responsive, accessible and rewarding nature.

I am targeting the Roguelike genre in particular, as these games have recently become incredibly popular, with games like Hades[3] earning critical acclaim, and thoroughly building on the elements set out in the renowned classic The Binding of Isaac[4]. These such games share the common properties of exploring some generated or hand crafted space, collecting objects that alter the player - for better and/or for worse, and overcoming - or dying to - increasingly difficult foes/challenges to the player. Generally the game cycle is repetitive, and built on the motivation of a player wishing to get further than they managed in the previous attempt, or run[5]. Games like these are often fundamentally simple, with basic movement controls, collection of items, and some form of combat control. The challenge found within the game is entirely based on the combinations of items that the player chooses to collect, and the combat mechanics of enemies, both of which will largely need fine tuning during play testing.

1.2 Goals

The final goal of the project will be to have formulated, built and deployed a game, making sure to utilise design patterns and complex technologies. In order to achieve this, I need an understanding of which design patterns exist in the industry, and how they are generally included. A good example of such a design pattern may include the use of controllers for handling specific objects, and factories to make objects.

I’ll also need to pay special attention to technologies and standards used within the industry. An early assessment of technologies has thrown Unity as a good way to work practically with games, having both a very detailed, interactive approach to the front end of the game development cycle, and an even more granular C# module for integrating the back end into the game building environment.

My own goals for this project, in addition to what is expected of me, includes learning new programming skills as I pick up C# and start to understand Unity as an engine, as well as the software through which I interact with the engine. To create and publish a game will also help build my portfolio of presentable programming to potential employers, or future team members. These goals will end up forming the basis on which I am capable of defending my project decisions and software design choices in the interim evaluation.

2 Created Materials

2.1 Reports

My research will lead me to create the following reports:

- Design Document
    - Should have all the key elements of how my game will work, reinforced by later reports
- Major Elements of a Roguelike Game
    - Will form the basis of my target features for my game
- Current Major Titles
    - Will provide an insight into how to apply features from the Major Elements research
- List of Required Algorithms and Structures
    - The logical basis of the programming side of the game
    - A good way to initially think about design patterns
- Final Project Report
    - The final submitted report, a culmination- and evaluation of- decisions made during all prior research and development

2.2 Proofs of Concept

A list of likely proof of concepts required to complete the planned programming:

- Motion and collision
    - The basis of all character and enemy motion throughout
- Items and logical application of items
- Map generation
- Enemy logic and damage system
- Character visual logic

2.3 Testing Tools

Testing is quintessential to making a robust game, so these are the tools I will likely need to create:

- Item Spawner
- Map Roller/Descender/Ascender/Room Generator
- Enemy/Boss Spawner
- Stat Changer/Effect Applier/Damage Tester

3 Timeline

Academic Week	Milestones	Specific Notes
Term 1		
Week 1 19/09/2022	Write project plan	Important foundation knowledge for my whole project
	Research currently existing games	
	Gain familiarity with the targeted engine	
Week 2 26/09/2022	Conduct research on breakdown of important game mechanics, like visuals, motion, items and difficulty	Meeting Tuesday 27th 15:10
Week 3 03/10/2022	Finish plan, existing game research and engine familiarity	Meeting Monday 3rd 14:00 Friday 7th Project Plan Deadline
Week 4 10/10/2022	Start creating an early prototype for the game, featuring motion, damage, all major collision and simple items	Serves as the first major milestone in the programming section of the project
	Finish game mechanic research	
Week 5 17/10/2022	Research on how to devise an algorithm for keyed procedural generation	Primary complexity of the game is PCG
Week 6 24/10/2022	Identify missing research areas	Prototype will serve as the foundation for the final project files.
	Finish algorithm research	
	Finish early prototype	
Week 7 31/10/2022	Start creating a method for breaking down the composite pieces of complex items	Important design pattern section
Week 8 07/11/2022	Work on Interim Early Deliverable, modifying the prototype	Key part for reflection on decisions I have made.
Week 9 14/11/2022	Work on interim report	
Week 10 21/11/2022	Work on interim presentation	
Week 11 28/11/2022	Finish report and presentation, practice presentation	December 2nd submit all for interim
Week 12 05/12/2022	Working on procedural generation algorithm, to create rooms of a set size, and of a set number of rooms	December 5th presentation

Term 2		
Week 17 & 18 09/01/2023 & 16/01/2023	Create algorithm that randomly generates map	Major complexity of project, see report on PCG
Week 19 23/01/2023	Finish map generation algorithm	Bulk of final programming, allowing me to tunnel-vision and complete the code, mitigating a lack of report detail. Large majority of this section will fall under the crucial design patterns section of my final report, important to constantly reflect in my diary.
	Create template rooms to insert into randomised maps	
Week 20 30/01/2023	Finish templates, add them to generation algorithm	
Week 21 06/02/2023	Work on character visual modification algorithm	
Week 22 13/02/2023	Finish character visual	
	Work on enemy set templates	
Week 23 20/02/2023	Integrate the template enemy sets into the randomisation algorithm, using both a set of regular enemy sets, and boss enemies	
Week 24 27/02/2023	Draft up Final Project Report	Important for feedback on the level of detail and key organisation of the final report
Week 25 06/03/2023	Start finalising and testing programming side of project for submission	Ensuring major bugs are squashed and that the code runs as expected, on target systems.
Week 26 13/03/2023	Finish Final Report draft and start work on feedback	Crucial for an optimal Final Report
Week 27 20/03/2023	Finish report and project for submission	March 24th Final submission for all project documents

2.1 Gantt Chart

	Term 1											
Tasks x Weeks	19/9	26/9	3/10	10/10	17/10	24/10	31/10	7/11	14/11	21/11	28/11	5/12
Write project plan	🟩	🟩	🟩									
Research currently existing games	🟦	🟦	🟦									
Gain familiarity with the targeted engine	🟥	🟥	🟥									
Research important game mechanics		🟦	🟦	🟦								
Create an early prototype for the game				🟥	🟥	🟥						
Research algorithms for procedural generation					🟦	🟦						
Identify missing research areas						🟦						
Create a break down for effects of items							🟥					
Work on Interim Early Deliverable								🟥	🟥	🟥		
Work on interim report									🟩	🟩	🟩	
Work on interim presentation										🟩	🟩	
Practice presentation											🟩	
Create algorithm from generation research												🟥

	Term 2										
Tasks x Weeks	9/1	16/1	23/1	30/1	6/2	13/2	20/2	27/2	6/3	13/3	20/3
Create procedural generation algorithm to create rooms	🟥	🟥	🟥								
Create template rooms to insert into randomised maps			🟥	🟥							
Integrate the template rooms into the randomisation algorithm				🟥							
Work on character visual modification algorithm					🟥	🟥					
Work on enemy set templates						🟥					
Integrate template enemy sets into the randomisation algorithm							🟥				
Draft up Final Project Report								🟩	🟩	🟩	
Finalise and Test programming side of project for submission									🟥	🟥	🟥
Finalise Project Report and submit all project documents										🟩	🟩

3 Risk Assessment

3.1 Data Loss[6] - High Impact

Unity, especially with repeated use of the undo button can ironically crash quite frequently on my system. In order to mitigate loss while developing my unity project, all elements of my project are also backed up to a cloud-based extension of Git version control, hosted via the Royal Holloway CIM systems. In the case of a crash GitLab will have all of my previously synchronised data, avoiding significant loss of error as a result of a crash that could potentially irreversibly corrupt data.

3.2 Report and Project Imbalance - High Impact

I am relatively prone to tunnel-vision, where I often focus so heavily on either the report aspect, or the programming aspect of a project. This can lead to a very dramatic imbalance in the work I submit, as one of the two is always seemingly incomplete, or lacks the required attention and detailing that the other received. This particular work pattern is almost always due to a lack of appropriate time management, and as a result, the mitigation for this risk is this project plan itself. By creating the weekly plan, and the gantt chart derived from it, I prevent myself from neglecting one aspect of my project.

3.3 Lack of Generation Knowledge - Medium Impact

I have not worked with keyed generation before, and while I understand the premise, I have not approached coding a generation algorithm like this. If i am unable to produce an algorithm that can accept and output in a useable way from the procedural generation algorithm, i will not be able to complete this project in a way that allows for good user control, where a user may re-generate the same level from a previous run, either for practice, or competing with friends. In order to mitigate this, I have decided that I will fall back on a less strict algorithm, that does not accept a key, but instead fully randomises the layout of the level with the use of a key at all. This should be simpler to achieve in the scenario that I am unable to grant the user a key-based algorithm instead.

3.4 Unintended Hardware Creep - Low Impact

The system I use to develop this game will likely have better specifications than the system of an average player of this game. This means I must test the speed of my code on a machine more inline with restrictive specifications of a standard user. The mitigations for unintentionally creating a game that is not runnable unless the user has a very powerful system would have to be the running of the code on a much lighter system during testing. Running my system on its internal battery alone disables the much more powerful graphics card, and testing under that particular condition may be the best way forward, but I will also attempt to run the game on various systems, in order to make sure it can run on a number of environments.

3.5 Scope Creep - Medium Impact

In some cases, pushing a broad spectrum of mechanics and features into a game is a good thing, as it increases player agency and play time, however, in this simple, and time restricted project, it is unwise for me to attempt to put too much into the game at once. I have a currently defined scope to the project, which creates a game I am comfortable with. In the case that I do achieve my targets for functionality and robustness, I may eventually add more to the game, but I want to avoid doing so during times where I have planned, required elements. To prevent this, I will constantly refer to this project's diary.md which is present within my GitLab version control instance, as well as keeping as best as I can to the Gantt chart I have derived from my time plan. This will ensure I produce all the functionality I intend to produce, and will approximate how much time I have left for extensions to that functionality.

5 Bibliography

Ampatzoglou, A., & Chatzigeorgiou, A. (2007). Evaluation of object-oriented design patterns in game development. *Information and Software Technology*, *49*(5), 445-454.

- Key paper for ensuring my thoughts on design patterns can be backed by others, and helps me evaluate the use of design patterns within my project.

Khalifa, A., de Mesentier Silva, F., & Togelius, J. (2019, August). Level design patterns in 2D games. In *2019 IEEE Conference on Games (CoG)* (pp. 1-8). IEEE.

- The level design for my game seems rather simple by itself, so this paper may shed light on a more detailed approach to even an intentionally simplistic form of level design.

Sharif, M., Zafar, A., & Muhammad, U. (2017). Design patterns and general video game level generation. *International Journal of Advanced Computer Science and Applications*, *8*(9).

- This journal seems to encapsulate most of the complexity of my project, as a result, this is a vital source of information.

Harris, J. W. (2020). *Exploring Roguelike Games*. Taylor & Francis Group. https://doi.org/10.1201/9781003053576

- A book exploring the roguelike genre as a whole, likely vital foundation information for my design document.

Ahn, J. K. (2016). Mechanism of Permanent Death in Rogue-like Games. *Journal of Korea Game Society*, *16*(1), 33-42.

- Permadeath is an important element of roguelike games, and while I currently see it as a trivial element, this Journal should shed light on any important complexities, and the effect it has on the player.

Gellel, A., & Sweetser, P. (2020, September). A hybrid approach to procedural generation of roguelike video game levels. In *International Conference on the Foundations of Digital Games* (pp. 1-10).

- I have never approached procedural generation before, so tackling it in a manner that I can read about first will help me understand the fundamentals.

References

[1] Mojang, Microsoft. (2016, November 1). *Minecraft Education Edition*. Minecraft Education Edition. Retrieved September 26, 2022, from https://education.minecraft.net/en-us

[2] Merriam-Webster. (2022, September 16). *Video game Definition & Meaning*. Merriam-Webster. Retrieved September 26, 2022, from https://www.merriam-webster.com/dictionary/video%20game

[3] Supergiant Games. (2020, September 17). *Hades on Steam*. Steam. Retrieved September 26, 2022, from https://store.steampowered.com/app/1145360/Hades/

[4] McMillen, E. (2011, September 28). *The Binding of Isaac on Steam*. Steam. Retrieved September 26, 2022, from https://store.steampowered.com/app/113200/The_Binding_of_Isaac/

[5] Dotson, C. (2020, May 28). *What Is a Roguelike? The Beginner's Guide*. Lifewire. Retrieved September 26, 2022, from https://www.lifewire.com/what-are-roguelikes-4117411

[6] Team Asana. (2021, August 18). *7 Common Project Risks and How to Prevent Them • Asana*. Asana. Retrieved September 26, 2022, from https://asana.com/resources/project-risks