Building a Game

Project Plan

Mohammad Ibrahim Salman

CS3821- full year project

Supervised by: Susnas Sourjah

Department of Computer Science

Royal Holloway, University of London

11/10/2024

Chapter 1: Abstract

If one were to define a game by textbook standards, it would be a structured activity with established rules, where the player(s) aims to achieve a specific objective, requiring skill, strategy, or luck. Games are gratifying competitive experiences and offer a rewarding challenge, testing players' ability to understand the rules and creatively approach tasks through decision-making. In today's world, games are predominantly associated with video games, which have become a powerhouse of entertainment, offering new modes of socialisation through online multiplayer interactions and immersive escapism. They are played on various devices, from touch-screen smartphones to dedicated gaming systems like the "big three": Nintendo, Sony's PlayStation, and Microsoft's Xbox, the latter of which has redefined gaming in once unimaginable ways. From intense arcade-style competitions to light-hearted adventures, intricate simulations, and deeply immersive storytelling experiences, modern games cover a broad spectrum of experiences.

The project I envision for a game aligns with Sid Meier's notion of "a series of interesting choices." My goal is to develop a game that emphasises decision-making, problem-solving, and interaction with design patterns, user interfaces, and graphics. This project will allow me to fully immerse myself in the game development ecosystem and research the technology behind past games to inform my work.

I will be using the open-source game engine Godot, which has proven to be an excellent starting point for new game developers due to its flexibility in supporting both 2D and 3D game development, and has its scripting language, GDScript, very similar to Python, making it easier to learn and work with for those familiar with Python. Additionally, Godot offers a wide range of tools and an intuitive interface, which may lead to developing a more refined game than Python-based frameworks like Pygame [1].

My project will address the problem of accidental complexity that arises when implementing game mechanics without guiding templates or patterns. As it is well known that game developers face deadlines, leading to compromises in code quality, which makes maintaining and extending the game difficult, the solution would be to introduce design patterns as templates for common mechanics, reducing complexity in their implementation. Introducing template instantiations for game mechanics,

mapping mechanics like agents, game worlds, or progress indicators to relevant design patterns that provide ready-to-use templates for developers. Additionally, it aims to evaluate the extendibility and reusability of these patterns through empirical testing, proving that their use makes future changes easier and less error-prone. The project also focuses on creating documentation through the GAME-DP repository to guide developers, enabling even novices to efficiently implement the patterns. Lastly, it provides real-world examples from games like Unity projects, offering concrete guidance for implementing similar mechanics [2].

The motivation behind the project also stems from the rapidly evolving nature of the video game industry, which has become a dominant force in global entertainment. With the shift towards digital distribution and the introduction of new technologies, the industry faces both significant opportunities and challenges. This project will focus on analysing the economic drivers fuelling this growth and exploring how innovative technologies, such as social media integration and in-game advertising, impact consumer engagement. Key goals include understanding the value creation process by examining the interrelationship between game content, platforms, and user interaction. Additionally, the project will investigate emerging business models like freemium pricing and digital distribution to assess their long-term viability. By studying technological advancements, such as cloud gaming, and social trends like gamification, the project will explore their broader implications for the industry. Moreover, the research will propose future opportunities in areas like multiplayer dynamics and platform differentiation, aiming to enhance our understanding of consumer behaviour and game development [3].

Chapter 2: Time Line

For a game development project that spans two terms, focusing on both the technical implementation and refinement, here is a structured timeline similar to your example:

Term 1: Implementation Phase

- **Week 1-2:** Preparing the project plan, reading on design patterns, APIs, and graphics.
- **Week 3:** Study game development frameworks (e.g., Godot) and familiarize with core concepts, including 2D/3D rendering, physics, and scripting in GDScript.
- **Week 4-5:** Define the game concept and mechanics. Create a game design document (GDD) that outlines core gameplay elements, rules, and objectives.
- **Week 5:** Implement the basic game engine architecture, focusing on setting up the scene, player movement, and core mechanics like physics, collision detection, and input handling.
- **Week 6:** Develop the initial game prototype, including key gameplay features (e.g., character control, environment interactions). Begin asset integration for basic visuals and sounds.
- **Week 7:** Research and integrate design patterns for game mechanics, such as agent behaviour and state machines for NPCs (non-playable characters) and enemies.
- **Week 8-9:** Fine-tune game mechanics and implement more advanced features, such as AI for enemies or puzzle logic. Begin testing for bugs and performance issues.
- **Week 10-11:** Prepare for the interim report and presentation, documenting the game's core features, progress, challenges, and next steps. Include the initial playable prototype for feedback.

Term 2: Refinement and Evaluation Phase

- **Week 1-2:** Refine and augment the game mechanics based on feedback. Re-evaluate and improve features like physics interactions, AI behaviour, or player progression systems.
- **Week 3:** Implement UI/UX elements, such as menus, scoreboards, and game controls. Create polished visual effects and animations to enhance user experience.
- **Week 4-5:** Encapsulate the game into a more modular structure using proper software engineering practices, ensuring the codebase is scalable and maintainable.
- **Week 6:** Integrate additional functionalities, such as multiplayer support, online leaderboards, or game saving/loading mechanics.
- **Week 7-9:** Conduct extensive playtesting and debugging. Collect feedback from peers and make necessary adjustments. Optimize performance, polish visuals, and ensure compatibility across platforms.
- **Week 10-11:** Prepare for the final report and viva, presenting the completed game. Include evaluations on performance, user feedback, challenges encountered, and future improvements.

Chapter 3: Risks and Mitigations

In any project, recognizing potential risks is crucial to ensuring smooth progress and minimizing setbacks. Below are five key risks identified for this game development project, along with their respective mitigations. Each risk is evaluated in terms of its likelihood and importance, with a structured approach to addressing and mitigating its impact.

1. Hardware Problems

One of the most basic but potentially damaging risks is hardware failure. Issues such as my computer not functioning correctly, storage becoming full, or data corruption could bring progress to a standstill. Without access to the development environment or game files, the project would struggle to proceed.

Mitigation: To address this risk, I will implement a **multi-tiered backup strategy**. By regularly committing my work to GitHub, I ensure that my code and assets are safely stored in the cloud and accessible from any machine. Additionally, I will maintain local backups on an external USB drive, which will be updated weekly. This ensures that, even in the case of hardware failure, work can continue with minimal disruption.

2. Poor Estimation of Tasks

One of the key risks in project management is the poor estimation of how long tasks will take to complete. Overlapping or overrunning tasks could significantly slow down progress, causing delays that might make it difficult to meet milestones.

Mitigation: To mitigate this, I will **break down tasks into smaller, manageable sub-tasks**. This approach allows me to allocate realistic timeframes to each sub-task, reducing the risk of misjudging the overall time required. Additionally, I will spend sufficient time **researching and planning** before beginning each sub-task, ensuring I have a solid understanding of the complexity involved. By tracking progress against each sub-task, I can identify any potential overruns early and adjust my schedule accordingly.

3. Bugs and Glitches

Bugs and glitches, whether in the game mechanics, controls, or graphical elements, are inevitable in any game development project. These issues can derail progress if not properly managed, potentially leading to larger technical debt if left unresolved.

Mitigation: My approach to mitigating bugs is to **document and fix them as they are encountered**, committing each fix to GitHub regularly. I will also adopt a **proactive debugging approach**, using tools to check for "code smells" and ensure that my code adheres to programming standards from the outset. Regular testing and debugging at every stage of development will ensure that issues are caught early before they compound into larger problems.

4. Overhead of Knowledge

At times, I may encounter knowledge gaps, whether in programming techniques, game development principles, or even the use of the game engine itself. Without a deep understanding of these aspects, it can be difficult to proceed with confidence and efficiency.

Mitigation: To address this, I will **allocate sufficient time for research** and continuous learning throughout the project. By researching ahead of implementation, I can better understand the technical and creative challenges, reducing the risk of delays due to unforeseen complexities. Additionally, consulting **documentation, tutorials, and forums** will be a regular part of my development process, ensuring I stay informed and can make well-founded decisions.

5. Conscious Experiments

One of the more abstract risks is the possibility of losing sight of the overall vision for the final game. Constantly shifting ideas, experimenting with new mechanics, and trying to implement features without a clear plan could lead to a confusing, half-baked product. Needless experimentation could consume valuable development time without contributing meaningfully to the final game.

Mitigation: To prevent this, I will follow **industry-standard software engineering practices**, ensuring that I adhere to a clear plan and avoid over-experimentation. By maintaining focus on the core mechanics and features defined in my initial game design document, I can prevent unnecessary scope creep. Additionally, I will **evaluate progress and the quality of the code at regular intervals** to ensure that experiments contribute positively to the project. This approach will allow for controlled, meaningful experimentation while keeping the end goal firmly in sight

Chapter 4: Bibliography

[1]: Cassone, F. (2020) Godot Engine Game Development in 24 Hours. Available at: https://books.google.co.uk/books?hl=en&lr=&id=3gBRDwAAQBAJ&oi=fnd&pg=PT16&dq=Cassone,+F.+(2020)+Godot+Engine+Game+Development+in+24+Hours.+&ots=YyodB2vaFY&sig=bLK01oiynexjBsLHNC2C-FwBRCI#v=onepage&q&f=false

[2]: Paschali, M.-E., Volioti, C., Ampatzoglou, A., Gkagkas, A., Stamelos, I., & Chatzigeorgiou, A., 2021. Implementing game requirements using design patterns. Journal of Software: Evolution and Process, 33(12), e2399.

[3]: Marchand, A., and Hennig-Thurau, T., 2013. Value Creation in the Video Game Industry: Industry Economics, Consumer Benefits, and Research Opportunities. Journal of Interactive Marketing, 27(3), pp.141-157.

[4]: Nystrom, Robert. (2014) Game Programming Patterns, Genever Benning; 1st edition. Available at: http://www.gameprogrammingpatterns.com/

[5]: Zechner, Mario. Green, Robert. (2012) Beginning Android Games; Apress Second Edition.

[6]: Griffith, Christopher. (2012) Real-World Flash Game Development; Routledge, 2nd Edition.