University of Tripoli

Faculty of Information Technology

Development of a 3D Puzzle Game for Desktop Computers

In partial fulfillment of the requirements for the degree Bachelor of Science in Information Technology

By

Islam Ghretlli 215185139

Supervised by:

Dr. Abd Annasur Addiaf

Abstract

Game development

In a game project, the product is a game. But, and here comes the point: A game is much more than just its software. It has to provide content to become enjoyable. This has an important effect on the game project as a whole. The software part of the project is not the only one, and it must be considered in connection to all other parts: The environment of the game, the story, characters, gameplay, the artwork, and so on.

All of these parts individually, are passions and skills loved and tended to by the author.

There is one major difference. In my opinion, it's the only difference that really matters.

We can go over the technical details of why it's different, sure. 3D engines, particle physics, lots of different things come into play.

But lots of different forms of software have strings attached. Modeling software has to do a lot of the same things. Every piece of significant software has some specialized library it has to use.

So what makes GAMES different?

Here it is: Software is designed to fill a business need. You want an inventory system? You can define what types of items you ahve to handle. You can define what you want for your production scheduling. You can do all that. Or if you want banking software, you can define what you want to do with it.

With games, your business need is "fun". Try writing a technical specification for "fun".

That, in my humble opinion as a developer, is what makes games different than regular software. You simply can't say "Great! This software is now feature complete as per the client's requests!" because all they want to do is have fun.

That being said, you don't need 3d graphics and extravagant physics for something to be fun. Why do people still play Tetris? Its physics consist of "move block down" "don't let block go out of bounds" and "stop block when it hits something", and while over the years there have been numerous versions, some with fancier graphics than others, but the bottom line is -- it's fun!!

So if you want to be a great game developer, do not throw out what you've learned as a regular software developer. It's still very useful stuff. And @Sion is right about separating your components, just like you would in a regular piece of software. But the single most important feature you can add to your game is fun. Fun fun fun fun fun. That's why game development exists, that's what you need to make your game successful. And trust me on this however fun it is to play, it's at least 10x as fun to make!!

Dedication

Table:

Glossary

Chapter 1

## Introduction

The gaming world while vastly explored in other parts of the world, is grossly overlooked and neglected in Libya. An aspiring or budding game developer is not provided with the means to flourish and further this interest, as a career or hobby. This, besides being a waste of real potential and the end of young dreams, is an overlooked opportunity distinguish and enhance our country to the rest of the world, by means of widely renowned and profitable product.

PC gaming, while traditional, is an undeniably prominent faction of the gaming industry; and held as a personal interest and love; the author have chosen this platform for my intended project. Undoubtedly, the world of gaming is branching anew, and strongly, towards both online and in ways of smartphones. While not denying the potential, nor importance of this, it is equally important to not forget that PC gaming is neither deprecated nor obsolete in this industry, and pursuing distinction in this platform is not adding another drop of water to an already vastly explored sea; but rather should be respected and valued in its own right; acknowledging its more than impressively large fan-base and, shamelessly, financial opportunity.

The genre is an experimental blend between Point-and-Click, Multiple-Ending, and First Person Puzzle Games. Taking key aspects of each and setting them in a First-Person, 3D, Free Moving game-world setting.

The detailing of the different genres of games, as well as technology options and choices made for this project will be done in later chapters

# 1.2. Game Development

Game development is the procedure of creating video games which involves working on some platform/software to develop the game. The development is undertaken by a game developer which can be one person or a team of **Game Engineers** (for constructing the overall structure and game play), **Designers** (for designing the objects and animations), **Coders** (for coding and giving life to the game including logics, score and other calculations), **Project Manager** (for managing and promoting the game), **Game Tester** (who will test the game after it has been developed to find bugs in the game, if any). All of these human resources come under Game developers and the process of developing the complete game is termed as **Game Engineering**.

# 1.2.1Game Development Concepts

Game development is one of the most exciting fields of computer science and a major part of the software development industry. Computer games comprise of a large and ever expanding market world-wide.

Interactive digital media and games along with entertainment applications have an enormous craze amongst the daily users of PCs and play an important role in providing economical strength(as games are not free), a factor which cannot be easily neglected.

So game development will also continue giving birth to new ideas and interactive devices for making the game play more attractive and exciting. For this game developers need to be well trained and equipped with their development skills.

# 1.2.2 Game Development is Software Development

The steps and stages required for a game's development follow similar stages like that of software development, along with a few extra stages for successful completion of the game. Since game development is a major part of the software development industry, therefore game development also goes through the various phases of SDLC (Software Development Life Cycle) along with some extra phases. Let's explain these phases in details.

Software Development Life Cycle is a well-structured and arranged sequence of phases in software engineering for developing the intended software product. Same structure needs to be followed by a game developer also. These stages are:

1. Communication:

Here, the user initiates the request to develop a desired game. Then he/she contacts the developer or project managing firm and tries to discuss the terms. Then after a successful agreement, stage 2 will begin. If the user himself is a game developer, then it starts from stage 2 directly.

2. Requirement Gathering:

In this stage, the game development team discusses the requirements to carry on the project and the project manager will decide the number of human resources required for the project. The team will carry out the discussion with varied stakeholders to discuss problems with various domains with a motive to bring out as much information as possible based on their requirements.

3. System Analysis:

In this stage, the developers will decide the roadmap of the plan for a successful game development till release and try to come up with the best software model (we will discuss about software engineering models in next chapter, in details) appropriate for the project. This phase also includes a proper understanding of the product's limitations or changes required in existing systems beforehand.

4. Systems Design:

In this phase of development, desired features and detailed work, which includes game-play, setting up of objectives and levels, screen layouts, player and game object model; creating animations in game, business rules, process diagrams (UML, DFD), pseudo-code and other documentations (GDDs) are done.

5. Development Phase:

The real code for your game will be written in this phase. This can be from a pseudo code or a set of algorithms written in the GDD (Game Design Document).

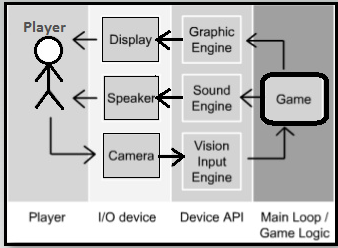
6. Integration and Testing Phase:

In this phase, a demo version of the game is released, with a trial period of 15 days or 1 month. This is done officially by the team itself to check for errors and popularity. With this partial release of game, the promotion is also started on websites like YouTube and other gaming sites. At the same time tester(s) are hired to check for bugs in the game.

So all these stages must be followed to make a game and to make it successful.

# 1.2.3 Architecture of a Game

The architecture and structure of a game is similar to that of a software. But it does have some additional components which makes it different from a software. Every game has the following components:



* Graphics Engine
* Sound/Audio Engine
* Rendering & Vision-Input Engine
* I/O Devices (like, Mouse, keyboard, speaker, monitor etc)
* DLL files and Drivers/Device APIs

# 1.3.1Defining a game

Well, you all have heard about the term **Game**. A Game can be defined as an activity enrolled for amusement as a prepared form of play that basically undertakes for enjoyment and/or as an educational tool.

Alternatively;

# Defining a video game

A video game can be defined as a game that runs on electronic devices which involves humans interacting with a user interface(UI) for generating visual responses on any video device like TV screen or computer monitor. This electronic game when played by a user, deals with a set of graphics, images or even audio to make the game more interactive.

In simple words, video games are the digital entertainment platform (which is a form of software) that humans "play" via a computer, TV, smart phones, tablets or other gaming consoles.

# 1.3.3 Different Genres of Games

Genre of a game defines the exact category of the game and can be relayed through the similar gameplay characteristics, like - the type of objectives and storyline, the levels and camera point (i.e. FPS, TPS), the features and the storyline that the game is showcasing. Genre of a game is not defined by the content or the playing mode, but by the common challenges and characteristics, that the game is having. For example, **FIFA** and **PES - Pro Evolution Soccer** have similar properties and objectives as both of them are soccer games and the player's get points or the levels of difficulty raises as the team wins the matches by scoring goals, like a real life football match. Genre of a game is decided by the resemblance found and listed under a common heading, here, FIFA and PES will come under *Sports-Genre Game*.

The Genre of interest here is Puzzle games;

Puzzle games focus on logical and conceptual challenges. While many action games and [adventure games](https://en.wikipedia.org/wiki/Adventure_game) include puzzle elements in level design, a true puzzle game focuses on puzzle solving as its primary gameplay activity.

Rather than presenting a random collection of puzzles to solve, puzzle games typically offer a series of related puzzles that are a variation on a single theme.

### Trial-and-Error puzzles

This sub-genre includes point-n-click games that often exhibit similarities with adventure games and walking simulators. Unlike logical puzzle games, these games generally require [inductive reasoning](https://en.wikipedia.org/wiki/Inductive_reasoning) to solve. The defining trait is that you must experiment with mechanisms in each level before you can solve them. Puzzle elements often do not have consistency throughout the game, and thus require guessing and checking.

These include [Myst](https://en.wikipedia.org/wiki/Myst" \o "Myst), [Limbo](https://en.wikipedia.org/wiki/Limbo_(video_game)), [The Dig](https://en.wikipedia.org/wiki/The_Dig_(video_game)), [Monument Valley](https://en.wikipedia.org/wiki/Monument_Valley_(video_game)), and [escape room](https://en.wikipedia.org/wiki/Escape_the_room) games such as [The Room](https://en.wikipedia.org/wiki/The_Room_(video_game)).

# Video Games Platforms

Video games have evolved a lot since their inception—in both how they look and how they’re played. With every passing year, the variety of video game platforms expands. These platforms give gamers different experiences: some virtually catapult you into a fictional world, while others are simpler and better for gaming on the go. To gain a better understanding of the many gaming experiences now available, here’s a guide to some of the different types of video game platforms.

## PC

PC gaming is one of the best and oldest ways to experience video games. Playing on a PC has many advantages, including higher-quality visuals and more versatility. PC gaming also gives you a wider variety of games to play, like the ones available through programs such as Steam. Certain games, such as World of Warcraft and League of Legends, are also only available on PCs.

## CONSOLES

Video game consoles from Microsoft, Sony, and Nintendo have been leading the way in gaming platforms for years now. In fact, a [2018 study](https://www.pewinternet.org/2018/05/31/teens-social-media-technology-2018/) from the Pew Research Center states that “84% of teens say they have… access to a game console at home.

## MOBILE

A recent trend in gaming is the increased development of games for mobile phones and tablets. This is an area on which the gaming industry has recently capitalized with popular titles such as Clash of Clans, Fortnite, and Angry Birds.

## VIRTUAL REALITY

VR puts you in the middle of a virtual world, and it’s one of the most immersive gaming experiences available at the moment. VR requires you to put on a headset, which allows you to see the virtual world; Oculus, Sony, HTC, Samsung, and more brands offer many different types of VR headsets.

# PC Gaming in depth

A **PC game**, also known as a **computer game** or **personal computer game**, is a type of [video game](https://en.wikipedia.org/wiki/Video_game) played on a [personal computer](https://en.wikipedia.org/wiki/Personal_computer) rather than a [video game console](https://en.wikipedia.org/wiki/Video_game_console) or [arcade machine](https://en.wikipedia.org/wiki/Arcade_cabinet). Its defining characteristics include: more diverse and user-determined gaming hardware and software; and generally greater capacity in input, processing, video and audio output. The uncoordinated nature of the PC game market, and now its lack of physical media, make precisely assessing its size difficult.[[1]](https://en.wikipedia.org/wiki/PC_game#cite_note-stuart10-1) In 2018, the global PC games market was valued at about $27.7 billion.[[2]](https://en.wikipedia.org/wiki/PC_game#cite_note-2)

[Home computer](https://en.wikipedia.org/wiki/Home_computer) games became popular following the [video game crash of 1983](https://en.wikipedia.org/wiki/Video_game_crash_of_1983), leading to the era of the "bedroom coder". In the 1990s, PC games lost [mass-market](https://en.wikipedia.org/wiki/Mass_market) traction to [console games](https://en.wikipedia.org/wiki/Console_game), before enjoying a resurgence in the mid-2000s through [digital distribution](https://en.wikipedia.org/wiki/Digital_distribution).[[1]](https://en.wikipedia.org/wiki/PC_game#cite_note-stuart10-1)[[3]](https://en.wikipedia.org/wiki/PC_game#cite_note-economist12-3)

[Newzoo](https://en.wikipedia.org/w/index.php?title=Newzoo&action=edit&redlink=1) reports that the PC gaming sector is the third-largest category (and estimated in decline) across all platforms as of 2016, with the console sector second-largest, and mobile / [smartphone](https://en.wikipedia.org/wiki/Smartphone) gaming sector biggest. 2.2 billion video gamers generate US$101.1 billion in revenue, excluding hardware costs. "Digital game revenues will account for $94.4 billion or 87% of the global market. [Mobile](https://en.wikipedia.org/wiki/Mobile_operating_system) is the most lucrative segment, with [smartphone](https://en.wikipedia.org/wiki/Smartphone) and [tablet](https://en.wikipedia.org/wiki/Tablet_computer) gaming growing 19% year on year to $46.1 billion, claiming 42% of the market. In 2020, mobile gaming will represent just more than half of the total games market. [...] China expected to generate $27.5 billion, or one-quarter of all revenues in 2017

# PC vs Other Platforms

Recent study by DFC Intelligence reveals that over 3 billion people on the planet play video games, which is about 40% of the world's population. Of those people, a whopping 1.5 billion are PC gamers, proving the enduring popularity of non-console and non-mobile play.

With almost [half the world playing video games](https://gamerant.com/3-billion-gamers-report/), it makes sense that there would be a variety of platforms that different people enjoy, given the variety in tastes and experiences in the global population. However, a significant portion of gamers, 48%, show their preference for PCs as their primary video gaming platform. That doesn't mean almost half of all gamers only play on PC, as this number includes people who also play on console or mobile.

According to the data, console-exclusive players make up a small minority of the worldwide gaming population, with only 8% of gamers primarily playing on console. However, PC gamers are more likely to be "casual" gamers, since [video game console sales makes more money](https://gamerant.com/most-popular-game-console-ps2-playstation-wii-nintendo/) as console gamers spend more on average. That explains why PC games, while popular, don't always get as much hype as new consoles or console games as the big money just isn't in PC gaming.

1.42 billion of the world's gamers are in Asia, and the [video games industry in Japan is growing](https://www.thegamer.com/the-japanese-game-industry-grows-for-third-straight-year-thanks-nintendo/) especially fast. However, a large portion of these gamers play primarily on mobile, taking up most of the world's population. Still, PC gaming has a significant holding in Asia. The next largest portions of gamers are in Europe, Latin America, MENA, and North America. Mobile gaming is the fastest-growing gaming demographic, and may even eventually overtake PC as the popular platform of choice.

In a very reassuring finding for our line of work, a study by DFC Intelligence has found that almost half of all players around the world are on PC. According to the report, over three billion people are gaming consumers, in one form or another.

The research was published this August, and although it lives behind a paywall, stats gleamed by IGN give a breakdown on its findings. It’s believed that 3.1 billion, some 40% of the population of the planet, consume video games, and of those nearly half are into mobile gaming, which is the fastest growing audience.

Onto the good stuff, though: 48% of the people of that 3.1 billion are on PC. That’s 1.5 billion if you don’t have a calculator near. Comparatively, only 8% are considered to be primarily console users, but they have the highest spending per player. Asia leads the regional breakdowns of the available data, contributing 1.42 billion gaming customers, and 53% of the mobile marketplace. Europe is next, with 668 million, and 17%, respectively. Latin America has 383 million and 11%, while North America has 261 million and 4%, and the last percentage is held by MENA, at 7% of the worldwide mobile players.

Mobile gaming continues to be one of the most important sectors in the industry. So much so, Fortnite developer Epic and Apple are in the middle of a large-scale legal tussle over how Apple runs the iOS marketplace.

So far this year, Call of Duty: Modern Warfare is leading in sales. Companies that make an effort with PC releases tend to see tangible benefits, like SEGA with Persona 4: Golden this summer. Many of Steam’s top-sellers keep selling for years, and some, like Counter-Strike: Global Offensive, continue to outdo their own records.

# Proposed Game Overview

Description of the main points of what the game is.

* + - 1. **Game Title:**

Autopilot

* + - 1. **Game Genre:**

Puzzle, point and click, multiple ending

* + - 1. **Game Perspective:**

First Person POV

* + - 1. **Game Mode(s):**
* Single player only
* Offline
  + - 1. **Target Audience:**

Everybody

**1.4.1.6 Core Idea:**

The main character, Grey, becomes trapped in a virtual world of his own creation, and he needs to use elements in his environment and manipulate his coworkers into triggering his waking.

* + - 1. **Goal:**

Escape the virtual world by waking up from daydream.

* + - 1. **Topic of game:**

Escape room puzzles in a semi-open world environment.

# Game Background and Game flow

Information that doesn’t directly change the game-play. However, it does weave an over-arching narrative the game follows, and several smaller side stories.

# Background story:

Grey is your run of the mill guy with a run of the mill job, personality, and life. He works in a boring office doing boring work. So with the never changing routine he started to retreat into his mind and daydream while his body works on autopilot. So it's just another day, another dollar and he logs in and gets to working and goes into autopilot and goes into that place in his mind where ever memorized detail of his office floor exists. But he's all alone; just him to do whatever he wants; until he realizes that he snap out of it. And has to figure out how to wake himself up from his daydream

# Other characters:

The other characters who take part of the story are the ones who may hold the key to winning the game.

* His coworker Jeremy, who has a daughter with cancer.
* His coworker Maisy, who is a secrete drug dealer.
* His coworkers Paul and Jane, who both have feelings for each other.
* The janitor, Sally, who no one interacts with.
* His boss, Richard Tator, who is difficult to get along with.
  + 1. **Game Play**

The aspects of which outline what the player’s experience will consist of, as well as what they can expect from the in-game behavior (Story, world, and characters).

* + - 1. **Objective(s):**

The motivations provided to the player to trigger progress in the game. As well as the end results and goals the player is ultimately working towards.

* + - 1. **Standard objectives:**
  + When fulfilled the game can be considered “played” and an end reached for the game.
  + Solve puzzles marked in game.
  + Complete a sequence that leads to one of the game’s endings.
    - 1. **Completionist objective:**

Additional goals and tasks that can be completed but are optional, and not necessary to finish the game.

* Solve all the puzzles.
* Get all the endings.
* Find secrete ending(s).
  + - 1. **Game Logic:**

The rules and laws inside the game-universe that dictates a structure the game must follow. I.e. if magic does not exist within the universe, it cannot be used as part of the game story telling tools, nor as a game-play mechanic.

* Fourth wall breaks are viable.
* Actions in the virtual world affect the in-game “real” world.
* If all the puzzles in a specific route are solved, an ending is triggered.
* Player can initiation and solve one or more puzzles from multiple routes at once.
* Characters abide by world physics.
* Only objects that relevant to puzzles can be used.
* Jumping out of window or throwing objects are consequence free, character or object will respawn back inside the building.
* Dialogue will reflect tips and character thoughts.
  1. **Project Aim and Goals**

## Objective

The aim is to create a functioning game for PC, playable on most processors (Core-i3 and above), that provides an engaging, enjoyable, and challenging experience in equal parts.

It is important to successfully and seamlessly combine the genres intended, namely: Point-and-Click, Multiple-Ending, and First-Person Puzzle game. Which is a rarely seen meeting of genres.

Such an effort provides a risk; each genre has its conventions and fan-base which live according to them, a combination of such risks alienating or creating a distasteful new feel to long-term, hard core gamers of each respective genre; it is the author’s attempt to create such a game that rather than failing to meet their expectations, the game should enhance their experience with a new flavor from the best of the other genres fore-mentioned.

In respect to the point-and-click aspect of the game, the player should find the world, while limited in size, easy to navigate and interaction with the objects within easy and natural feeling, in a manner similar to its original form.

## Goals

* Develop a playable game.
* Create a sequence of solvable puzzles
* Produce multiple endings.
* Launch game to be play-tested by normal people.
* Fulfill requirements of a software engineering graduation project.

### Agile Model

In this model, the product is broken down into a set of features, and hence it is used for quickly delivering a working product and so considered as a very rational development method. This model generates ongoing releases of your project, each having small and incremental changes updated from the previous released version. At each cycle, the project is tested and then released.

* 1. **Summary**

Game development, while it is largely still software development varies slightly in its development cycle, and measure of success. Video games, and PC games specifically, make up a large sum of the world’s profit, and is the predominant console for consuming electronic entertainment games. Therefore, this project aims to be a part of that market, and push it forward to continue competing with rising gaming industry in mobile and console gaming; as well as serving as the author’s graduation project as a software engineer. Using the agile model to develop as it is most fitting for game development.

Chapter 2

# Introduction

While analyzing the viable options for the development of this game, the time constraint, limited experience, lack of budget and the development team consisting of one person were the drive behind keeping the ideas and scope of the game as limited as possible. Relying on the internet, books, similar previous projects, as well as the experience of the developer, simplicity and cost efficiency (monetary and time wise; learning curve considerate) was the main strategy.

# Game Engines

For every game, game engine plays a major role since the game engine helps the game designers to bring characters of the game to life, by helping in scenes, characters and graphic generation, sound, artificial intelligence, scripting animation, networking etc. Game Engine is like an integrated development environment, with a readymade suite of visual development tools and reusable software components. It turns the complex task of game development simple, by providing an abstraction layer, which makes a lot of big tasks look very easy, while the game engine does all the hard work in the background. In other words, it is a framework that is designed specifically for the construction and development of video games. Developers use these game engines for creating games for consoles, mobile devices and personal computers.

A Game Engine is created to develop games, just like any other IDE for any particular language programming. All the components in the game engine are built and integrated to support the motive of game development.

# Choosing a Game Engine

Choosing the right game engine comes down to personal preference. It depends on the answer of the question, what type of games do I want to make? For example, do I want to make a game on Roblox? Maybe, an RPG? If so, you can just use the OGRE engine, or RPG Maker engine. That’s not to say that it has to be those engines in order to make those games.

## [Godot](https://godotengine.org/)

Godot, is another great choice to choose from, it’s very flexible meaning that you could make 2d/3d games. It’s open source, and all the work you develop, is 100% yours. You own all the rights to it, and you won’t need to pay any fees/royalties in the future. This is a great choice for an indie game developer, but you need to take into account that there isn’t as much documentation, guides/videos as Unity or Unreal as the community is much smaller for this Engine.

|  |  |
| --- | --- |
| Pros | Cons |
| * Very flexible | * Not as beginner friendly |
| * Organized, great way to understand and see your work | * Small community |
| * Open source | * Documentation needs a lot of work |
| * MIT license, meaning that the games you develop are 100% yours, and you do not have to pay any sort of fees later on. | * Not as beginner friendly |
| * High Level Debugger | * Small community |
| * Engine is very actively developed | * Documentation needs a lot of work |

## [Unreal](https://www.unrealengine.com/)Engine

It’s true that you could get better graphics using the Unreal engine, but at the end of the day, it all depends on the assets, and visual artists. If you have great visual artists, you can make games on Unity look better than games made with Unreal. From personal experience, and others feedback, Unreal is not as beginner friendly. I’m not saying Unreal is bad, it is one of the best engines out there, and also, it’s backed with that Fortnite money.

One thing that Unreal does that not many other game engines provide, is grants. Unreal, is willing to support indie game developers with money to help fund their games. Which is really neat. Also, like Unity, Unreal has plenty of documentation, guides, and videos all over the internet.

|  |  |
| --- | --- |
| Pros | Cons |
| * Known for AAA game development | * Need a licensed copy for working |
| * Backed by Fortnite | * Not beginner friendly |
| * More tools | * More suitable for a team of developers |
| * Asset store | * Need a licensed copy for working |
| * More efficient rendering technology | * Not beginner friendly |
| * Offers grants | * More suitable for a team of developers |

## [Unity](http://unity3d.com/)Game Engine

It does get some hate, for having “bad graphics” compared to Unreal, but I believe you could make the same exact game on both engines, and have very similar graphics. In my opinion the Unity Game Engine, is very beginner friendly, and I highly recommend it as a first choice getting into indie game development, as there are plenty of documentation, guides, and videos all over the internet.

|  |  |
| --- | --- |
| Pros | Cons |
| * Beginner Friendly | * Unity is heavy, takes a lot of space on your hard drive |
| * Cross-platform friendly | * Even small games are built with a large .exe size |
| * Excellent asset store | * Does not offer grants |
| * Cheap cost / free to use | * If your games profit 100k/annually, you have to use Unity Pro (which is fair, but could be a con if you don’t want to pay a monthly fee) |
| * Endless possibilities | * Unity is heavy, takes a lot of space on your hard drive |

## Game Engine Findings

If you’re looking to get into indie game development, I recommend either Unreal, or Unity. For the main reasons that there are plenty of documentation, guides, videos, all over the internet, and it’s backed by great communities, and money. I personally like Unity more, but, at the end of the day, it all comes down to what type of game you want to make, and what your personal preferences are. Don’t be afraid to try out multiple engines at first. Find what works best for you, and try your best to stay motivated throughout the game development process. Good luck, with the indie game development journey.

It boiled down to either Unreal Engine or Unity Engine. And Unity was ultimately selected for the following project specific reasons:

* + Graphics are meant to be simple and cartoon-like for the purpose of the stylized game, therefore Unreal’s edge in ways of superior graphic quality was rendered irrelevant.
  + Beginner friendly: As the author both had some previous experience with Unity, and it is already more beginner friendly for learning purposes. The delay having to familiarize with UE’s environment was another setback against it.
  + Free to smalltime monetization:
  + Access to free assets and tutorials. As Unity is geared more towards a vaster audience it would be more comfortable for a quick development process.
  1. **Unity Game Engine in Depth**

Unity is a 2D/3D engine and framework that gives you a system for designing game or app scenes for 2D, 2.5D and 3D. I say games and apps because I’ve seen not just games, but training simulators, first-responder applications, and other business-focused applications developed with Unity that need to interact with 2D/3D space. Unity allows you to interact with them via not only code, but also visual components, and export them to every major mobile platform and a whole lot more—for free. (There’s also a pro version that’s very nice, but it isn’t free. You can do an impressive amount with the free version.) Unity supports all major 3D applications and many audio formats, and even understands the Photoshop .psd format so you can just drop a .psd file into a Unity project. Unity allows you to import and assemble assets, write code to interact with your objects, create or import animations for use with an advanced animation system, and much more.

Unity has done work to ensure cross-platform support, and you can change platforms literally with one click, although to be fair, there’s typically some minimal effort required, such as integrating with each store for in-app purchases.

**2.5.1 Unity Architecture and Compilation**

Unity is a native C++-based game engine. You write code in C#, JavaScript (UnityScript) or, less frequently, Boo. Your code, not the Unity engine code, runs on Mono or the Microsoft .NET Framework, which is Just-in-Time (JIT) compiled (except for iOS, which doesn’t allow JIT code and is compiled by Mono to native code using Ahead-of-Time [AOT] compilation).

Unity lets you test your game in the IDE without having to perform any kind of export or build. When you run code in Unity, you’re using Mono version 3.5, which has API compatibility roughly on par with that of the .NET Framework 3.5/CLR 2.0.

**2.5.2 Important Classes**

This section provides an overview of some of the most commonly used and important built-in classes in Unity that you may want to use when scripting.

These serve as a starting point for the discovery of scripting basics in Unity, and do not cover all classes in Unity, or even every member of the classes which are covered.

• GameObject Represents the type of objects which can exist in a Scene

• MonoBehaviour: The base class from which every Unity script derives, by default.

• Object: The base class for all objects that Unity can reference in the editor.

• Transform: Provides you with a variety of ways to work with a GameObject’s position, rotation and scale via script, as well as its hierarchical relationship to parent and child GameObjects.

• Vectors: Classes for expressing and manipulating 2D, 3D, and 4D points, lines and directions.

• Quaternion: A class which represents an absolute or relative rotation, and provides methods for creating and manipulating them.

• ScriptableObject: A data container that you can use to save large amounts of data.

• Time (and framerate management): The Time class allows you to measure and control time, and manage the framerate of your project.

• Mathf: A collection of common math functions, including trigonometric, logarithmic, and other functions commonly required in games and app development.

• Random: Provides you with easy ways of generating various commonly required types of random values.

• Debug: Allows you to visualise information in the Editor that may help you understand or investigate what is going on in your project while it is running.

• Gizmos and Handles: allows you to draw lines and shapes in the Scene view and Game view, as well as interactive handles and controls.

* 1. **Looking at Similar Games**

These games directly inspired and influenced the style and flow of the idea for the intended game; taking elements of what made them a unique, enjoyable, and memorable experience. Then emulating and incorporating the same or similar elements into the project.

**2.6.1 Portal (PC, 2007)**

Portal is a 2007 puzzle-platform game developed and published by Valve. It was released in a bundle, The Orange Box, for Windows, Xbox 360 and PlayStation 3, and has been since ported to other systems, including Mac OS X, Linux, and Android (via Nvidia Shield).

Portal consists primarily of a series of puzzles that must be solved by teleporting the player's character and simple objects using "the Aperture Science Handheld Portal Device", a device that can create inter-spatial portals between two flat planes. The player-character, Chell, is challenged and taunted by an artificial intelligence named GLaDOS (Genetic Lifeform and Disk Operating System) to complete each puzzle in the Aperture Science Enrichment Center using the portal gun with the promise of receiving cake when all the puzzles are completed. The game's unique physics allows kinetic energy to be retained through portals, requiring creative use of portals to maneuver through the test chambers. This gameplay element is based on a similar concept from the game Narbacular Drop; many of the team members from the DigiPen Institute of Technology who worked on Narbacular Drop were hired by Valve for the creation of Portal, making it a spiritual successor to the game.

Aggregate reviews for the standalone PC version of Portal gave the game a 90/100 through 28 reviews on Metacritic.[72] In 2011, Valve stated that Portal had sold more than four million copies through the retail versions, including the standalone game and The Orange Box, and from the Xbox Live Arcade version.[77]

**2.6.2 Stanley Parable (PC, 2011)**

The Stanley Parable is an interactive drama and walking simulator designed by American developer Davey Wreden. The game was originally released on July 31, 2011 as a free mod for Half-Life 2.

The game has no combat or other action-based sequences. Instead, the gameplay involves guiding a silent protagonist named Stanley alongside narration by British actor Kevan Brighting. As the story progresses, the player is confronted with diverging pathways. The player may disobey the narrator's suggestions, which will then be incorporated into the story. Depending on the choices made, the player will encounter different endings before the game restarts.

Within two weeks of its release, the mod was downloaded more than 90,000 times.[3] Responses of most players were positive, and Wreden became "an overnight internet sensation among hardcore gamers."[32]

**2.6.3 Normality (PC, 1996)**

Normality is a 3D graphical adventure game, released in June 1996 by Gremlin Interactive. All cut-scenes in the game are pre-rendered. Normality has humor and is well paced, the puzzles can be hard for this post-syberia times, but it is just a matter of using lateral thinking and the 1984-style story is adequate enough to make you push the game forward.

"Normality" was a little sensation back then, being the first adventure to use a 3D engine to navigate your character through the game. And there was much to navigate. Many locations to visit, many objects to discover, many obstacles to overcome. The story and setting is still quite fun to follow and explore. Maybe the graphics are not that, what we are used to today, but from a gameplay standpoint "Normality" has aged very well.

* + 1. **Game Specifics**

The desired features that make up the intended game:

* First Person
* Game world, includes:
  + 3D Objects
  + Characters
  + Rooms with walls, doors, and windows
* Windows platform deployed
* Opening scene
* Single player
* Multiple endings
* Intractable puzzles
* Sound effects
* ~~NPCs in the background~~
* In-Level HUD
* Menus
* Lighting effects
* Reflection effects
* Interactive objects
  + Puzzle components
  + Easter eggs
  + Extra objects for realism
    1. **Choice of Game Engine**

The game engine selected is Unity Real-Time Development Platform. Unity is a cross-platform engine the engine itself currently supports building games for more than 25 different platforms, including mobile, desktop, consoles, and virtual reality. Some of the reasons the engine was selected are:

* Unity has a broad reach that makes recruitment easier
* Unity is a good engine for beginners
* Unity is fast and agile
* Unity makes portability easier
* Unity has a large and varied asset store

As well as the author’s prior experience with Unity, its good compatibility with the 3D modeling software of choice, and programming language of choice.

* + 1. **Game Layout**

The layout diagram illustrates the layouts of the In-Game world areas

The first area is the main space the game will take place in, solving the different puzzles inside to ultimately achieve the end task of escaping that virtual world.

Other sections can be reached through puzzle solving and will lead to one of the endings.

The Main area is shown in the figure below:

//TBD

The Elevator Ending Area shown in the figure below:

//TBD

The Vent Maze Area shown in the figure below:

//TBD

* 1. **Tools**
     1. **Jasc PaintShop Pro X7:**

2.8.1.1 Version:

* + - 7.0

2.8.1.2 About PaintShop Pro X7:

* + - It includes content-aware features such as "Magic Fill"
    - "Smart Edge"
    - as well as support for [XMP](https://en.wikipedia.org/wiki/Extensible_Metadata_Platform) sidecar files that preserve edit settings for [raw formats](https://en.wikipedia.org/wiki/Raw_image_format).

2.8.1.3 What it will be used for:

* + - Textures e.g. notes
    - General appearance of building and objects
    1. **Blender:**

2.8.2.1 Version:

* + - 2.8x

2.8.2.2 about Blender:

Modeling 3D Objects with lighting and cameras

Animation and simulations for 2D and 3D

Texture and shading objects including scripted and complex costume shaders

2.8.2.3 What it will be used for:

Modeling the assets and characters used in the game

Most of the animations for characters, objects, and UIs

Texturing and shading the environment

* + 1. **Audacity:**

2.8.3.1 Version

* + - 2.0.x

2.8.3.2 Features:

Recording audio from multiple sources

Post-processing of all types of audio

Adding effects such as normalization, trimming, and fading in and out

2.8.3.3 What it will be used for:

Editing, mixing, and recording sound effects.

* 1. **3D Objects, Terrain, and Scene Management**
     1. **Objects:**

The objects will be modeled and animated in Blender and exported in the FBX format. The format was established to provided interoperability between digital content creation applications. There are many tools available for conversion from/to FBX file format.

FBX is standard for data exchange between DCC apps and game engines like Unity and UE4. FBX was originally a native format for Motion Builder scenes, so in theory it can hold whole scene data in a node-based structure.

The main materials and textures are to be applied in Unity’s editor, with the exception of textures that need special mapping will be done in Blender and exported in the FBX file.

The scaling, positioning, and rendering of objects and their animations will be done in Unity’s native editor.

* + 1. **Environment:**

The environment will be created and controlled during the development phase in Unity’s editor. It will primarily utilize the models exported from the modeling software. In addition, the lighting, reflections, and manipulation will utilize the tools provided by unity’s development environment.

* + 1. **Scene Management:**

Loading, rendering, and control of the scene and its object will be handled by the Unity’s editor.

* + 1. **Coding**

The main coding of the game will be carried out in the Visual Studio Community IDE, using the latest available version at any given time. Later to be added into the game assets in the Unity development environment and attached to GameObjects.

* 1. **Game-flow Layout**

//TBD Chart of all possible routes

* 1. **Summary**

Chapter 3

3.1 Introduction

3.2 Project Requirements

3.2.1 Functional Requirements

General Game Requirements:

* 1. Main Menu Launch
  2. Start New Game
  3. Open Options
  4. Open Controls
  5. Open Settings
  6. Set Full Screen
  7. Change Volume
  8. Change Resolution
  9. Change Quality
  10. Return to Main Menu
  11. Load Level
  12. Pause Game
  13. Resume Game
  14. Open Options
  15. Quit to Main Menu
  16. Display Quit Confirmation
  17. Cancel Quit Confirmation
  18. Quit to Desktop
  19. Show New Task
  20. Open Task Menu
  21. Scroll through Tasks
  22. Remove Finished Task
  23. Display Tip
  24. Display Dialogue
  25. Display Pop Ups
  26. Move Camera
  27. Click on Object
  28. Activate Puzzle
  29. Play Music
  30. Play Sound Effects
  31. Respawn Dropped Item
  32. Save Progress
  33. Load Progress
  34. Win Game
  35. Play Cutscene

Puzzle and Interaction Specific Requirements:

* 1. Pickup Object
  2. Drop Object
  3. Reposition Object
  4. Activate Item
  5. Hide Item
  6. Turn On Screen
  7. Play Video
  8. Show Puzzle Screen
  9. Hide Puzzle Screen
  10. Type Password
  11. Use Log-in Button
  12. Use Mail App
  13. Open Mail Content
  14. Minimize Mail
  15. Close Mail
  16. Switch to Clock Camera
  17. Exit Clock Camera
  18. Change Time
  19. Display Phone Camera
  20. Save Screenshot
  21. Change Photo Material
  22. Interact through proxy camera
  23. Move Security Camera
  24. Detect Security Breach
  25. Switch Camera Views
  26. Open Elevator Doors
  27. Close Elevator Doors
  28. Open Elevator Number Pad
  29. Close Elevator Number Pad
  30. Record Number Clicks
  31. Record Collected Items
  32. Unlock Doors

Character, Animation, and Extra Cosmetics Requirements:

* 1. Change Character State
  2. Play Character State Animation
  3. Play Idle Animation
  4. Stop Character Idle Animation
  5. Move Character
  6. Lock View
  7. Release View
  8. Zoom In
  9. Show Reflections
  10. Turn On Light
  11. Allow Door Interaction
  12. Allow Window Interaction
  13. Allow Drawer Interaction
  14. Lock Items in Place
  15. Pour Liquids
  16. Turn On Fans
  17. Display Random Tips
  18. Display Random Texture
  19. Display Random Animations
  20. Change Menu Lighting
  21. Keep Track of Playtime
  22. Display Autosave Icon
  23. Change Courser Icon
  24. Play Credits
  25. Skip Credits
  26. Skip Cutscene

3.2.2 Non Functional Requirements

Performance Requirements

Upon sensory overload or crash, the game must prompt with a force-close window detailing the error that has occurred.

Players should be able to access all in-game function of the game.

Safety Requirements

Game should not overheat or otherwise harm the user’s device. It will not contain malware of any sort.

Security Requirements

The game does not require internet access, nor any of player’s personal information and may not access it thus no special security measures need to be taken.

However, the in-game profiles require no authentication so it is possible for unauthorized users to access them.

Software Quality Attributes

Upon starting the game it should not take more than 2 minutes for the GUI to load.

Players should be able to click multiple buttons sequentially without causing the game to crash.

Response time for the buttons should not exceed 3 seconds.

Loading time for the puzzle should not exceed 6 seconds.

Game should not crash spontaneously.

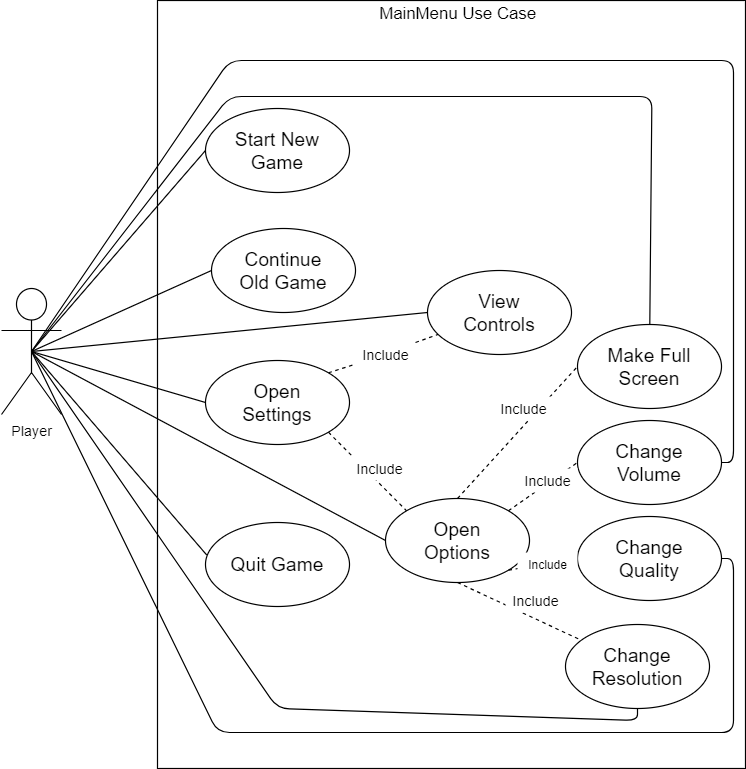
Business Rules

The game should not break any ethical, legislative, or copyright laws. Its content may not break with the conventions for the age-appropriate rating set.

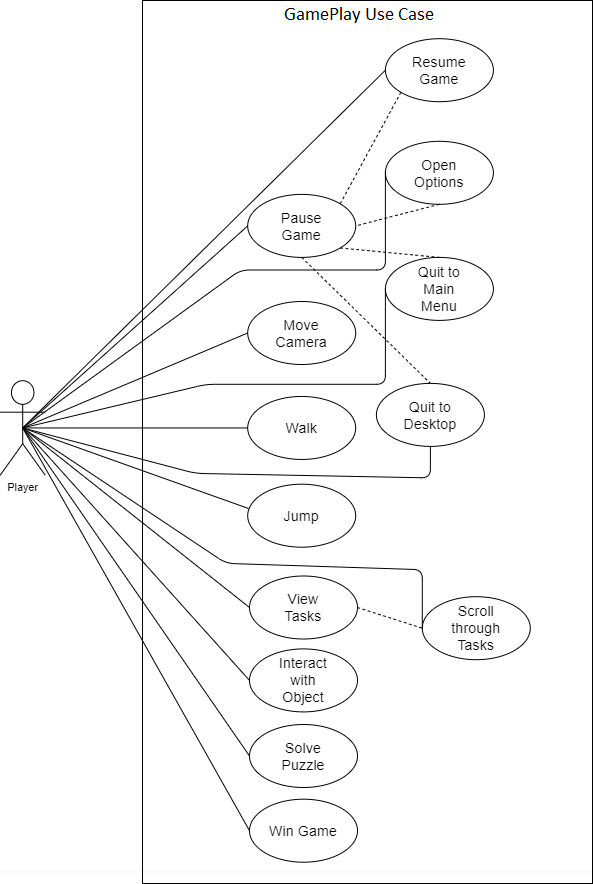
3.3 UML Diagrams

3.3.1 Use Case Diagrams

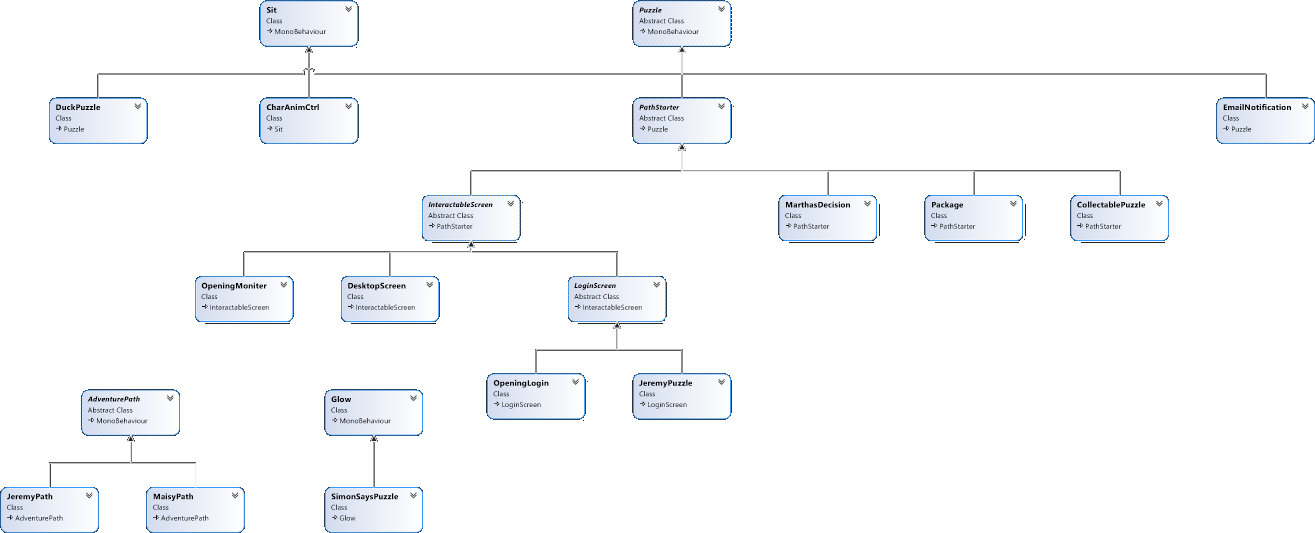
Main Menu Use Case



GamePlay Use Case

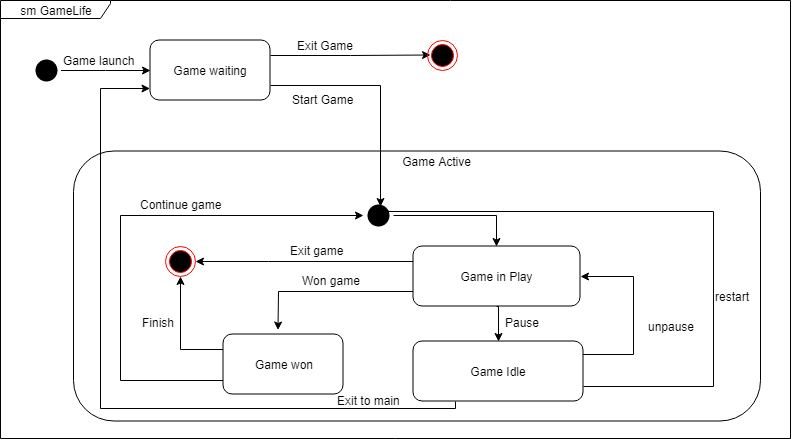


3.3.2 Class Diagram



3.3.3 Activity Diagram

GameLife



3.3.4 Sequence Diagram

Display Tip SQ