Logo

Description automatically generated

**22/FA-COP-2939-68831**

**Capstone Project Report**

**Game Development with GODOT: Abyssal**

Created By: Johnathan Webster

Student Number: 1040332

Instructor: Dr. Banisakher

Date: 11/10/2022

Table of Contents

[Proposal 5](#_Toc118976118)

[Program Writing and Executing 5](#_Toc118976119)

[Required Software and Hardware 5](#_Toc118976120)

[Project Schedule 5](#_Toc118976121)

[Risk Analysis 6](#_Toc118976122)

[Requirements 8](#_Toc118976123)

[1.1 Purpose 8](#_Toc118976124)

[1.2 Intended Audience 8](#_Toc118976125)

[2. Overall Description 8](#_Toc118976126)

[2.1 Product Overview 8](#_Toc118976127)

[2.2 User Classes and Actors 8](#_Toc118976128)

[2.3 Program Environment 9](#_Toc118976129)

[2.4 Use Cases 9](#_Toc118976130)

[User Case 1: Application launch 9](#_Toc118976131)

[User Case 2: Continue launch 9](#_Toc118976132)

[User Case 3: New game launch 10](#_Toc118976133)

[User Case 4: Options launch 10](#_Toc118976134)

[User Case 5: Exit launch 10](#_Toc118976135)

[3. Domain Requirements 10](#_Toc118976136)

[3.1 System and Platform Requirements 10](#_Toc118976137)

[3.2 Accessibility and Useability Requirements 11](#_Toc118976138)

[3.3 Safety and Security Requirements 11](#_Toc118976139)

[3.4 Maintenance and Reliability Requirements 11](#_Toc118976140)

[4. Future Requirements 11](#_Toc118976141)

[Appendix A: Potential Changes and Constraints 12](#_Toc118976142)

[Appendix B: Example User Interface 13](#_Toc118976143)

[Basic Structure 14](#_Toc118976144)

[Routine Breakdown 15](#_Toc118976145)

[Frameworks and Software 15](#_Toc118976146)

[Detailed Structure 16](#_Toc118976147)

[Main Menu 17](#_Toc118976148)

[Settings Menu 17](#_Toc118976149)

[Quit Menu 17](#_Toc118976150)

[Process: Initialize Scene Objects 17](#_Toc118976151)

[Objects and Classes 17](#_Toc118976152)

[1.1 Player Controls 18](#_Toc118976153)

[1.2 Dimitri 18](#_Toc118976154)

[1.3 Items 18](#_Toc118976155)

[1.4 Enemies 18](#_Toc118976156)

[Level 1 Scene 18](#_Toc118976157)

[Test Matrix 19](#_Toc118976158)

[Conclusion 20](#_Toc118976159)

[References 21](#_Toc118976160)

# Proposal

The application I intend to create will be a simple platformer game within the GODOT game engine. The game will include classic mechanics from platformer games including jumping, side-scrolling, health, coins, enemies, power-ups, main menu, and lives for the player. The game will be written in C# and game assets will be standard 32-64 bit.

# Program Writing and Executing

The program will be written both within the GODOT game engine and Visual Studio Code. GODOT will be responsible for compiling code and executing structured logic and sequencing of events. Visual Studio Code will be an external IDE (integrated development environment) I will use for all C# programming. GODOT comes with its own language, GDscript, but will not be utilized for any coding.

# Required Software and Hardware

The full software requirements will be:

* [GODOT 3.5](https://godotengine.org/download): Will need [Mono Version (C# support)](https://downloads.tuxfamily.org/godotengine/3.5/mono/Godot_v3.5-stable_mono_win64.zip)
* .[NET SDK](https://dotnet.microsoft.com/download): Framework for Mono Version
* 64-bit Windows operating systems
* [Visual Studio Code](https://code.visualstudio.com/): For external IDE development
  + GODOT extension in Visual Studio Code (C# tools for GODOT)
  + Mono Debug for code debugging
  + C# for Visual Studio Code
* [Kenney Game Assets All-in-1](https://kenney.itch.io/kenney-game-assets): Artwork for game
* [Trello](https://trello.com/?&aceid=&adposition=&adgroup=105703214328&campaign=9843285532&creative=430959026561&device=c&keyword=trello&matchtype=e&network=g&placement=&ds_kids=p53016490704&ds_e=GOOGLE&ds_eid=700000001557344&ds_e1=GOOGLE&gclid=CjwKCAjw3qGYBhBSEiwAcnTRLqFqu5s94mg_5UkCh4jl5AUThbPWSL-H66wyHoVY0-ZzozTLNSp73RoCujgQAvD_BwE&gclsrc=aw.ds): Used for Scrum/Agile development

# Project Schedule

|  |  |  |
| --- | --- | --- |
| Week | Description | Allotted time |
| 1 | Develop over-all outline of project, including software requirements and game assets | 10-hours |
| 2-3 | Create user and game requirements utilizing Scrum software | 20-hours |
| 4-6 | Have sample for gameplay and game logic finished and refine for quality-of-life improvements | 30-hours |
| 8-12 | Finish menu, save system, respawn, life counter, and any other systems that are standard for game design | 40-hours |
| 13-14 | Refine systems with assists and correct any lingering issues | 20-hours |
| 15-16 | Create presentation of application and documentation | 20-hours |

# Risk Analysis

Some potential issues that may arise include but are not limited to:

* Feature delay: Video game physics do not always align with the real world. There is a possibility that a certain physics mechanic will be delayed due to feature oversight or implications conflicting with other features.
* Allotted time: Although there is a specified amount of time for each iteration there is always a possibility the time is either not enough or is too much. The schedule will be adjusted accordingly.
* Feature redesign: The outline of the features is vague to incorporate multiple variations of the feature. This can cause a feature to be completely redesigned or scrapped entirely depending on complexity and time constraints.

These are just some of the main issues I expect the project to run into. There are many more (bugs, life, knowledge, reference material, etc..) but these will be addressed through systems unit tests, project analysis, and weekly evaluations. I will also utilize my professor for any feedback or variations needed to meet project requirements.

Delivery and Installation   
 The user will first need to download the zip file with all needed assets for the game and the .exe file that will launch the game. The application will be launched through a .exe file that can be opened from the file manager (or platform specific variations) When the application is launched the user will be presented with a main menu where they can select their save file and begin playing the game.

# Requirements

## 1.1 Purpose

The purpose of this document is to detail the software application Abyssal. The document will review the overall description of the application, functional requirements, domain requirements, nonfunctional requirements, and lastly any future requirements for the application.

## 1.2 Intended Audience

The intended audience for this document is for developers, testers, quality assessors, product manager(s), management involved in the application development, and any sponsors for the application.

# 2. Overall Description

## 2.1 Product Overview

This program will be developed for the Windows 10 operating system. The program will be developed in the Godot game engine and will utilize Visual Studio Code for external code development.

## 2.2 User Classes and Actors

The intended user classes will be individuals wanting to play a platformer game. Actors will be the developers, testers, quality assessors, product manager(s).

## 2.3 Program Environment

The game is designed to run on windows 10 desktop systems. The application will require a minimum of 2 gbs of RAM, and 1 gbs of available storage for the application to be installed. It is assumed that the intended customer’s computer will meet these requirements for successful download and installation.

## 2.4 Use Cases

Below is a set of normally use cases from the users’ perspective. Including screen descriptions, action inputs, and alternative flows.

## User Case 1: Application launch

User clicks and runs the .exe for the game application to launch. From there the user will be presented with the main menu where they can select generic options such as: Continue, New game, Options, Exit.

## User Case 2: Continue launch

User clicks the Continue option from the main menu which allows them to continue where they previously left off with the game application. (Not in final design)

## User Case 3: New game launch

User clicks the new game option from the main menu. This allows the user to start a fresh save for the game. Allowing them to start a new save if desired. (Not in final design)

## User Case 4: Options launch

User selects the options menu option to be directed to the game’s adjustable settings. Here the user can adjust game settings to better suit their personal preferences.

## User Case 5: Exit launch

User selects the exit option from the main menu prompt to exit the application.

# 3. Domain Requirements

Below are the requirements imposed by the domain of the program environment.

## 3.1 System and Platform Requirements

The program requires the Windows 10 operating system with a minimum of 2 gbs of RAM (random access memory), and 1 gbs of storage to successfully run the application.

## 3.2 Accessibility and Useability Requirements

The application will include a settings option where the user can adjust in game settings for accessibility and useability on their system. The program will be installed from a .zip file for easy extraction of the application.

## 3.3 Safety and Security Requirements

The program will not collect any personal data about the user. All data will be stored locally on the user’s system which they have complete access over. The application will be available only from our website from a secure connection to our server.

## 3.4 Maintenance and Reliability Requirements

The chosen frameworks and APIs will be with the current windows operating systems and Godot 3.5 version (the most stable version).

# 4. Future Requirements

There will be the implementation of features that will enable the user to access and run the application on other operating systems. Future implementations will have the ability to run on MacOS and Linux distributions.

# Appendix A: Potential Changes and Constraints

Potential limitations or constraints may affect future development. Limitations including discontinued support of required software and APIs the prevent compatibility for many users. Another being cost as currently these systems are free for most users where in the future there is a possibility these applications will charge for a subscription or service of some kind.

# Appendix B: Example User Interface

Below is the initial layout of the main menu screen. This will display when the user first opens the application.Graphical user interface, text, application

Description automatically generated

## Basic Structure

The overall structure of the project will rely on the GODOT game engine. The game engine uses a delta function that is called 60 times each second to keep system calls in a sync of 60 second intervals.

Diagram

Description automatically generated

## Routine Breakdown

Delta: method called very second 60 times to keep track of game state changes and interaction with objects.

Player\_Controls: Stores the players x, y, and z index to determine the position the play is facing or moving on the coordinate grid.

Dimitri : Player\_Controls: Using the position the player is on the coordinate grid we then pass delta to \_Process to call each method 60 times a second.

Player\_Direction: Determines which direction the player is facing and if they are currently pressing anything.

Player\_Jumping: Used to determine if the player is currently in the air and used to calculate gravity and friction of the player.

Play\_Physics: Used to determine gravity, speed, acceleration, deceleration, friction, and other physics-based attributes for the play. Also has a timer to sync with frame refreshes.

## Frameworks and Software

Software will be developed in GODOT while using visual studio code to write source code in C#. GODOT uses the .NET Core community framework.

The full software requirements will be:

* [GODOT 3.5](https://godotengine.org/download): Will need [Mono Version (C# support)](https://downloads.tuxfamily.org/godotengine/3.5/mono/Godot_v3.5-stable_mono_win64.zip)
* .[NET SDK](https://dotnet.microsoft.com/download): Framework for Mono Version
* 64-bit Windows operating systems
* [Visual Studio Code](https://code.visualstudio.com/): For external IDE development
  + GODOT extension in Visual Studio Code (C# tools for GODOT)
  + Mono Debug for code debugging
  + C# for Visual Studio Code
* [Kenney Game Assets All-in-1](https://kenney.itch.io/kenney-game-assets): Artwork for game
* [Trello](https://trello.com/?&aceid=&adposition=&adgroup=105703214328&campaign=9843285532&creative=430959026561&device=c&keyword=trello&matchtype=e&network=g&placement=&ds_kids=p53016490704&ds_e=GOOGLE&ds_eid=700000001557344&ds_e1=GOOGLE&gclid=CjwKCAjw3qGYBhBSEiwAcnTRLqFqu5s94mg_5UkCh4jl5AUThbPWSL-H66wyHoVY0-ZzozTLNSp73RoCujgQAvD_BwE&gclsrc=aw.ds): Used for Scrum/Agile development

## Detailed StructureDiagram Description automatically generated

## 

## Main Menu

The application begins with the opening main menu scene. The main menu scene acts as the first interaction the player has with the application. From there the player can branch into three directions. Play, settings, or quit scenes.

## Settings Menu

The settings menu serves as the scene where the player can customize in-game settings. The settings, in future installments, give player control over sounds, graphics, and screen space. Currently there is only one setting “Sound” where the player can toggle game audio on and off.

## Quit Menu

The quit scene isn’t a scene. It is a selection option that terminates game execution.

## Process: Initialize Scene Objects

Process is called when the player selects the “Play” option from the main menu. The process calls all objects that need to be instantiated in the scene.

## Objects and Classes

## Player Controls

Player controls inherits from the Kinematic Body 2D class. The Kinematic Body 2D class provides many behind-the-scenes functionality that allows the developer to assign global positioning, key input, and assign collision shapes that allow the player to interact with their environment.

## Dimitri

Dimitri directly inherits from the player controls class. Allowing the developer to better assign values for specific characters. In the diagram we override the \_Process() object and require it to include Player\_Direction, Player\_Jumping, and Player\_Physics. \_Process() is called every second, sixty times. The inclusion of the overridden functions will require them to be processed every time \_Process() is called creating a smoother experience for the player.

## Items

Items are objects the player can safely interact with. Currently there is only one, Coins, that when the player interacts with it increases the total count and is the condition for winning a level. Items are given a Collision Shape 2D to inform \_Process() when the player is currently touching the item. When the signal is called the game will respond appropriately.

## Enemies

Enemies are the opposite of items and are objects the player should avoid. Enemies are functionally the same as items.

## Level 1 Scene

Level 1 scene is the final step in the process. After all objects have been created, this scene is loaded and where the player can start interacting with all other objects.

## Test Matrix



## Conclusion

Abyssal was a fun and huge learning experience. This project taught me multiple things from time management, organization, how software programs interact with one another and most importantly to research. The amount of effort put into this simple program is one that I am proud of and truly admire all developers out there who provided all the content and resources I was able to utilize for free. I am proud I was able to deliver most of the original content that was described in the beginning, however, I quickly realized that I was being too ambitious. I was unable to include multiple features I originally planned and had to adjust my test matrix to meet requirements.

## References

Godot game engine documentation: <https://docs.godotengine.org/en/stable/>

Windows Operating System: <https://docs.microsoft.com/en-us/windows/>

Asset Artist: <https://www.kenney.nl/assets>

.[NET SDK](https://dotnet.microsoft.com/download): Framework for Mono Version

[Visual Studio Code](https://code.visualstudio.com/): For external IDE development

YouTube:

1. [GDQuest](https://www.youtube.com/c/Gdquest)
2. [BornCG](https://www.youtube.com/c/BornCG)
3. [FinePointCGI](https://www.youtube.com/c/FinePointCGI)