**MINOR PROJECT REPORT**

**ON**

**Role Playing Game Model**



**BACHELOR OF ENGINEERING**

**In**

**Computer Science and Engineering**

**SUBMITTED BY**

Aryan Sharma(2020a1r194)

Aditya Partap(2020a1r113)

**Model Institute of Engineering and Technology**

**Kot Bhalwal, Jammu – 181122**

**www.mietjammu.in**

**DECLARATION**

I, **Aryan Sharma**, hereby declare that this project is the record of authentic work carried out by me during the final term of my B.Sc. and has not been submitted to any other University or Institute towards the award of any degree.

**Signature of the Student**

**Abstract**

This report focuses on the development of 2D Top-Down RPG Desktop game, **Riko: The Dragon Hunter**. The project is based on Retro Pixel games containing 3 different levels for the player to play. It has multiple achievable and separately upgradeable weapons, player’s experience gaining system with 3 different player skins, computer controlled intelligent enemies

- both Melee and Ranged to challenge the player along with simple NPCs and other interesting objects like chests, crates. Unlike the traditional Top-Down RPG games where player attack with sword, this game rather has guns to attack just like Survival Shooting games.

This project explores a new dimension to the traditional Retro Pixel RPG games by mixing the features of Survival Shooting games where player shoots enemies to survive. With the simplicity of this game it simply aims to bring fun and make you look back to your childhood’s Roguelike, Pokémon, Mario games with some new modern features.

**Table of Contents**

[Chapter 1 – Introduction](#_bookmark0)

* 1. [Game Overview](#_bookmark1) 
     1. [Target Audience](#_bookmark3)
     2. [Game Description](#_bookmark4)
  2. [Game Objectives](#_bookmark5)
  3. [Why game as a project?](#_bookmark6)
  4. [Methodology](#_bookmark7)

[Chapter 2 – Project Scope](#_bookmark8)

* 1. [Features and Gameplay Elements](#_bookmark9)
  2. [Limitations and Constraints](#_bookmark10)

[Chapter 3 – Game Design](#_bookmark13)

* 1. [Game Rules:](#_bookmark14)
  2. [Game Structure](#_bookmark15)
  3. Game play
  4. Camera
  5. HUD
  6. Animation States
  7. Audio
  8. Attempt to make game more fun

[Chapter 4 – Technology and Tools](#_bookmark21)

* 1. [Python Programming Language](#_bookmark22)
  2. [Pygame](#_bookmark23)
  3. [Asset pack](#_bookmark24)

[Chapter 5 – Code](#_bookmark29)

* 1. [Code view](#_bookmark30)
  2. Screen shorts

[Chapter 6 – Conclusion](#_bookmark57)

[6.1 Future Plans](#_bookmark58)

[References](#_bookmark57)

# Chapter 1 – Introduction

This report describes the process involved in making a 2D Top-Down RPG game, Riko: The Adventurer with Unity3D. This game is for both PC Standalone and Android devices. This chapter discusses the game overview including the synopsis, target audience, description, then focuses on the game design, describing how the game is implemented.

## Game Overview

The primary objective of the player is to gain experience by killing the enemies, he can only achieve that if the clears the entire dungeon that he has been assigned to. Throughout the dungeon, there are various enemies, chests, obstacles. Each area of the dungeon has different enemies with different health, and damage, he can only go further if he clears the current area that he can enter and exit. This section elaborates the synopsis, target audience, software used and finally game description.

### Target Audience

2D RPG games are extremely popular especially in mobile devices. The RPG game called Soul Knight has over 10,000,000+ installs on google play. This shows that a large base of players are actively interested in this genre and gives me a good opportunity to tap into that market by following the traditions that have already set for this genre. But I’m mainly targeting to casual gamers aged from 13-30.

### Game Description

This game involves a single player entering a random dungeon with a gun as the weapon after selecting a character from menu scene. The dungeon is inhabited by various creatures that will attack the player if he goes into the area that the creatures inhabiting. If the player successfully defeats the creatures the player gains experience points and if the experience points are max as the current grade adventurer, the player gets promoted and the process goes on until he is the max grade adventurer. But he does not entirely become the max grade adventurer within a single

dungeon; he has to defeat creatures by teleporting to different dungeons. The player can also change the gun if he acquired more than one and upgrade separately. In this game, I have designed 4 dungeons for the player to play and besides the creatures the dungeons have NPCs, obstacles like crates, health restorations. Although there are 4 dungeons, the dungeons are not considerably big for the player to take a lot time to clear the entire dungeon thus it will only take 5-10 minutes on average without pausing the game.

## Game Objectives

The major objectives of this game project are:

* To create a RPG game that will have all the functionality of traditional RPG games.
* To implement gun attack as the ability instead of usual sword attack.
* To implement multiple weapons with different bullets and power.
* To implement individual weapon upgrading system
* To implement various kinds of enemies that have the purpose to simply attack the player until the death of the player.
* To add sound effects to player, enemies, weapons and other objects like doors.
* To design the game levels that will be simple yet beautiful.
* To make a User Friendly Interface that will be pleasant to look at.
* To add animations to all sorts of game objects like player, enemies, doors, water including UI transitions.

## Why game as a project?

Video games are not just any computer software which are made to benefit user’s daily life, games are rather made for user’s entertainment purpose, so more than anything we need to pay attention to what the user wants from the game, how to make it more entertaining, just making any game will not do, that is why it’s more challenging because I always have to carefully consider if I’m making developing it correctly to entertain users. I also have to invest a lot of time on the proper game designing to make it visually accepted. And to add that game requires a lot of scripts. The scripts are like pieces of a puzzle which you need to put all of them together to make it work. Thus I think game is a perfect project to prove myself as a CSE student.

## Methodology

The software methodology I have used to develop this game is Agile methodology which is an approach for developing products (in this case: **games**) using short iterations. The main idea is not making the whole project from start to finish, but making small features for the current project in small periods of time. In this way the results of each iteration are used to adjust the project plan.

* Each iteration is like a short project in itself
* Uses “inspect and adapt” practices to adjust the goals and measure progress

# Chapter 2 – Project scope

This chapter describes the Scope of the project that come as questions to both the developers and other users during the development of software.

## Features and Gameplay Elements

Riko: The Adventurer is a single player offline game so there is no hosting cost, if it developed to be a web game using WebGL, it will only consume internet data to load the ads there.

1. **Character Creation**: Allow players to create and customize their own characters, including choosing their appearance, abilities, skills, and attributes.
2. **Quests and Objectives**: Provide a variety of quests and objectives for players to complete, which can range from main story quests to side quests and optional tasks. These quests often involve exploration, combat, puzzle-solving, and interaction with non-player characters (NPCs).
3. **Leveling and Progression**: Implement a leveling system where characters gain experience points (XP) by completing quests and defeating enemies, allowing them to level up and improve their abilities, skills, and attributes.
4. **Skills and Abilities**: Offer a wide range of skills, abilities, and spells that characters can learn and use during gameplay. These skills can be categorized into combat skills, magic spells, crafting abilities, and more.
5. **Inventory and Equipment**: Include an inventory system where players can manage and equip various items, such as weapons, armor, consumables, and other equipment. This allows for character customization and strategic decision-making.
6. **Combat Mechanics**: Incorporate combat mechanics, which can vary depending on the RPG subgenre. This may include turn-based combat, real-time combat, or a combination of both. Provide a variety of enemies with different strengths, weaknesses, and attack patterns.
7. **Non-Player Characters (NPCs)**: Populate the game world with NPCs that interact with the player, offering quests, providing information, or engaging in dialogue. NPCs can have their own stories, personalities, and relationships with the player and other characters.
8. **Exploration and Open World**: Create a vast and immersive game world for players to explore freely, discover hidden areas, interact with environmental objects, and encounter various creatures and challenges. This can include dungeons, towns, cities, forests, mountains, and other diverse locations.
9. **Choice and Consequences**: Introduce choices and consequences that impact the storyline and the development of the player's character. These choices may influence relationships with NPCs, alter the outcome of quests, or determine the character's moral alignment.

## Limitations and Constraints

1. **Technical Limitations**: RPG development may face technical constraints, such as limited processing power, memory, or storage capacity. These limitations can affect the game's graphics quality, the number of characters or objects on screen, or the overall performance of the game.
2. **Budget Constraints**: The available budget for the RPG project may impose limitations on the scope of development. Financial constraints can impact aspects such as the size of the development team, outsourcing options for artwork or sound design, marketing efforts, and overall production value.
3. **Time Constraints**: RPG development requires a significant amount of time to design, develop, test, and refine. Time constraints can limit the depth and complexity of the game world, the number of quests or storylines, and the overall polish and quality of the final product.
4. **Scope Management**: It is essential to manage the scope of the RPG project effectively. Without proper scope management, the project may become overly ambitious, leading to delays, budget overruns, or a compromised final product. Setting realistic goals and prioritizing features and content is crucial.
5. **Resource Limitations**: The availability of skilled developers, artists, writers, and other resources can impact the development of an RPG. Limited access to specialized talent can affect the quality and diversity of game assets, the depth of storytelling, or the implementation of complex gameplay mechanics.
6. **Platform Limitations**: If the RPG is developed for specific platforms, such as consoles or mobile devices, there may be limitations imposed by the hardware or software capabilities of those platforms. This can affect graphical fidelity, performance, or compatibility with certain features or technologies.
7. **Design Constraints**: Certain design decisions may be constrained by the limitations of the chosen game engine, development framework, or programming language. These constraints can impact the flexibility and creativity in implementing certain gameplay mechanics, visual effects, or AI behaviors.
8. **Player Expectations**: RPGs often have dedicated and passionate fan bases with high expectations for the genre. Meeting player expectations while staying within project constraints can be challenging. Balancing player feedback and desires with practical development considerations is important.
9. **Legal and Licensing Constraints**: When using pre-existing intellectual properties or licensed content in an RPG, there may be legal and licensing constraints to consider. These constraints can affect the use of specific characters, settings, or themes and may require negotiation or modification of the game's content.

It is crucial to identify and address these limitations and constraints early in the development process to ensure realistic project planning and successful execution. Properly managing these constraints can help deliver a high-quality RPG within the project's limitations.

# 

# Chapter 3 – Game Design

Game design is the study of how to make the game functional by setting game rules, game mechanics, gameplay which I already briefly discussed in Chapter 1 but now I will go through them thoroughly. I will also discuss more about what I did to make the game more fun rather than visually accepted with the existing resources I have and that is part of game design because even if we make a fancy game with good graphical work if the game design is not well enough it will not attract the user. So I invested a lot of time on doing this section, since without it the game is incomplete.

## Game Rules

The game starts with player entering the menu scene where player can select a character, he can also unlock skin during selection if wish, in menu scene the player can move around freely until he wishes to enter the dungeon. In the dungeons the player can also move freely around, shoot bullets and switch weapons if he has more than one. There are multiple enemies inhabiting with different damage points and health that will attack the player if it enters in their inhabitant area. Initially player has no target so he can freely shoot but when player goes near to any kind of enemies the weapon locks the target so player can only shoot it when triggering the shooting event. If player successfully kills the enemy it gains experience points, in the process if experience is max to current grade of the player, the player gets promoted with gaining back the full health and the level only completes if he killed all the enemies. The levels are also filled with chests in random places which can grant the player gold or new weapons. In games scenes there is also NPCS that player can communicate with.

## Game Structure

ShopScreen

PauseScren

LosingSreen(if Died)

DoorOpened to teleport(if all

enemies died)

ChooseCharacter

Screen

Gameplay(in Random

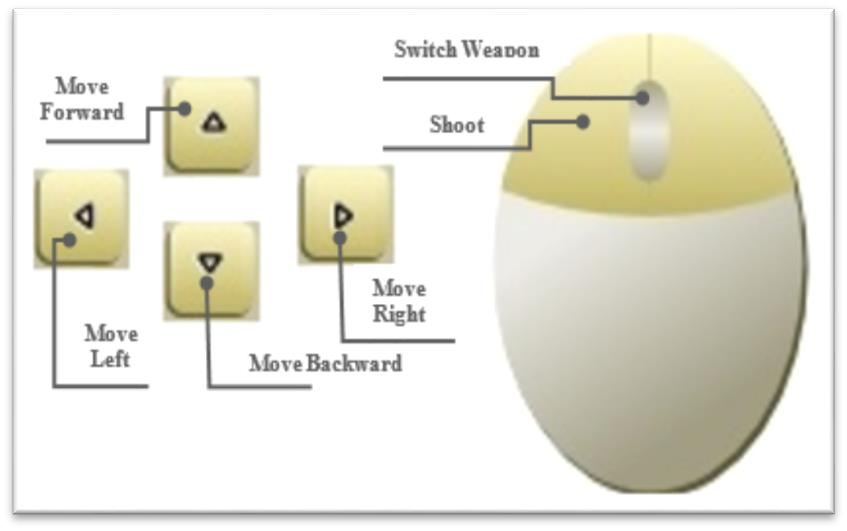
Dungeon)

Character Selection

QuitScreen

## Game Play

The following figures illustrates the inputs to control the player. For PC we are using keyboard keys to move the player and mouse to shoot and switch weapons. For mobile we are using mobile joystick touch inputs and 2 separate buttons for shooting and switching weapon.

* + 1. **Game Controls (PC)**

**Move Backward**

Figure 2: PC Keyboard Inputs

## 3.4 Camera:

### Menu Camera

At the start of the game on menu scene if the player touches on the screen the camera zooms in to the available character skins. If any of the player skin is touched it simply zooms in again and ChooseCharacterScreen is shown, after selection and going back from the ChooseCharacterScreen the camera zooms out and locked its screen to selected skin for player to player and follows the player.

### Game Camera

Game camera does not have the fancy zoom in, zoom out effect like menu camera but it follows the player more smoothly and additionally has another script attached to itself to shake camera when player is shooting- this feature is also part of Attempts to Make the Game More Fun which is the last (6.8) section of this chapter.

## HUD

* + 1. **HUD (PC)**

Figure 3: HUD (PC)

The HUD for both platforms are pretty much same the only thing that differs is on android devices we need virtual joystick to move, and buttons to shoot and switch weapon, thus we have them and in PC it is not needed.

## Animation States

### Player

The player has a total of 5 running animation states: player\_side, player\_up, player\_down, player\_sideUp, player\_sideDown, this is 8 directional movement. According to the direction of the player looking at the state changes. If the player has weapon, its position also changes according to the player’s states. The following image illustrates the idea of 8 directional movement:

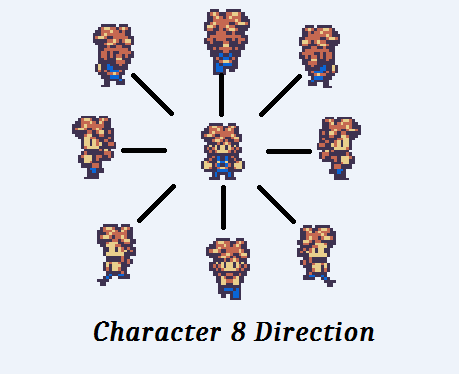


Figure 4: Character in 8 directions

### Enemies

They have two animation states: idle state, attack state. Initially enemies’ animation state is idle, if player enters in their inhabitant area the enemies’ state transition to attack state. This mechanism applies to all of the enemies.

### NPCs

The NPCs only have one animation state, which is idle.

### Environment

Few of the game objects that were used to design environment have animations which only have a single state. The following game objects have animations in them

* + - * River: It has the river wave animation which can be seen in Dungeon\_1 scene.
      * Water Fountain: It has water flowing animation which can be seen in Dungeon\_1 scene.
      * Lava: It has lava wave animation which can be seen in Dungeon\_2, Dungeon\_3 scene.
      * Water: It has water wave animation which can be seen in Dungeon\_2 scene.
      * Fire Torch: It has fire movement animation which can be seen in all of the dungeons.
      * Healing Fountain: It has water flowing animation which can be seen in Dungeon\_1 scene.

### UI

There are animation transitions to show the different UI panels, and all of were animated in Unity and have two animations states: “Hide” and “Show”. Hide simply hides the panel by either fade in transition or moving the panel to out of the screen space and show animation does the opposite of hide animation.

## Audio

The following table shows the list of audio clips with description I have used to develop this game, each of the audio clips may play in several objects like same audio clip is being played in several enemies, but I have changed the pitch of the audio clip to distinguish sound effect for enemies.

|  |  |  |
| --- | --- | --- |
| Name | Category | Description |
| Auvic - Heal Me | Background Music | Plays during the game |
| Brainwave - Contingency Of  Pink | Background Music | Plays during the game |
| Glitch HopAuvic ft. Pipo  Fernandez - Indignation | Background Music  Background Music | Plays during the game |
| Orange Wolke - Stephen's Tiny  Cafe | Background Music | Plays during the game |
| Thaehan - Pixelated | Background Music | Plays during the game |
| Wontolla - Up To No Good | Background Music | Plays during the game |
| Shurk - Haunted (JNATHYN  Remix) | Background Music | Plays during the game |
| SGV - Lighthearted | Background Music | Plays during the game |
| Odd Chap - Midnight Sway | Background Music | Plays during the game |
| game-over-arcade | FX | Plays when the player gets defeated |
| lemonjolly\_hurt4-unh | FX | Plays when player gets hurt |
| littlerobotsoundfactory\_gate- open | FX | Plays when gates or barrier opens |
| musiclegends laser-shoot | FX | Plays when player shoots with the gun, Flame Thrower. |
| jeckkech projectile | FX | Plays when player shoots with the gun,  Canon. |
| cabled-mess laser-shots | FX | Plays when player shoots with the gun, MG or Spazer. |
| shoot1 | FX | Plays when player shoots with the gun,  Matter or Rocket. |

|  |  |  |
| --- | --- | --- |
| shoot\_laser | FX | Plays when player shoots with the gun or  laser. |
| shoot3 | FX | Plays when player shoots with the gun,  Pistol or shotgun. |
| littlerobotsoundfactory jingle-  achievement-00-victory | FX | Plays when player completes the level. |
| littlerobotsoundfactory pickup-  gold-02 | FX | Plays when player receives gold from  chest. |
| syseq good |  | Players when player receives weapon from  chest. |
| newagesoup fun-explosion | FX | Plays when enemy dies. |
| BittenVonKrumpen\_Hurt | FX | Plays when the enemy Devil/ Bitten Von  Krumpen gets hurt. |
| sharesynth hurt02 | FX | Plays when the enemy imp / Swampy/ Masked Orc/ Wogol/ Muddy/ Zombie/ Orc Shaman/ Skelet/ Slime/ IceZombie gets  hurt. |
| Boss\_impact4 | FX | Plays when the enemy BigZombie/ Ogre/  Minotaur/ OneEye/ BigDemon gets hurt. |
| robotlike\_monster\_impact3 | FX | Plays when the enemy Chort/ gets hurt. |
| small\_monster\_impact2 | FX | Plays when the enemy TinyZombie/  OrcWarrior/ Muddy gets hurt. |

Table 4: List of Audio Clips with descriptions

## Attempts to make game more fun

Although common mechanics can make the game functional but that’s not enough for entertainment purpose, to entertain the users we need to pay attention to other parts of the game. With keeping that mind set I have included the following features in the game to make the more fun.

**Screen Shake:** When player shoots the camera is slightly shaking that gives an impression that an event is occurring.

**Environment Animations:** I have added river wave, lava, water wave, background animations to make the game visually more beautiful.

**Background Music:** Instead of playing a single BG music I have added multiple BG music that we usually get to hear in car racing with the order being totally randomized.

**Sound Effects:** Instead of playing same sound clip when player is shooting, I have added different sound clips for each of them, which helps us to distinguish the weapons, and added other sounds effects that have been playing as player hurt, enemy hurt, victory, game over clip.

**Character Color Change:** Just hurt sound clip being played does not entirely indicate the character got hurt because you cannot tell who got hurt that is why

changing the color comes in handy in minimal games like this and I implemented that as well.

**Explosion effect:** Just destroying the bullets when come in contact seems static and looks bad that is why adding a simple explosion effect makes a lot difference.

**Shooting effect:** Adding a weapon trigger effect also makes a lot difference.

**Enemy death effect:** A simple death effect instead of enemy vanishing suddenly outspokenly better.

**Enemy health bar:** Previously my demo version had no health bar and when my supervisor was trying the demo version, he could not see the progress of hitting enemies which is universal for all the gamers, if there are no health bar for the enemies we cannot tell if we are really damaging the enemy, thus added enemy health bar in my current version.

**Popup Text:** This is a must have feature for all the 2D RPG games, and thus added in my game.

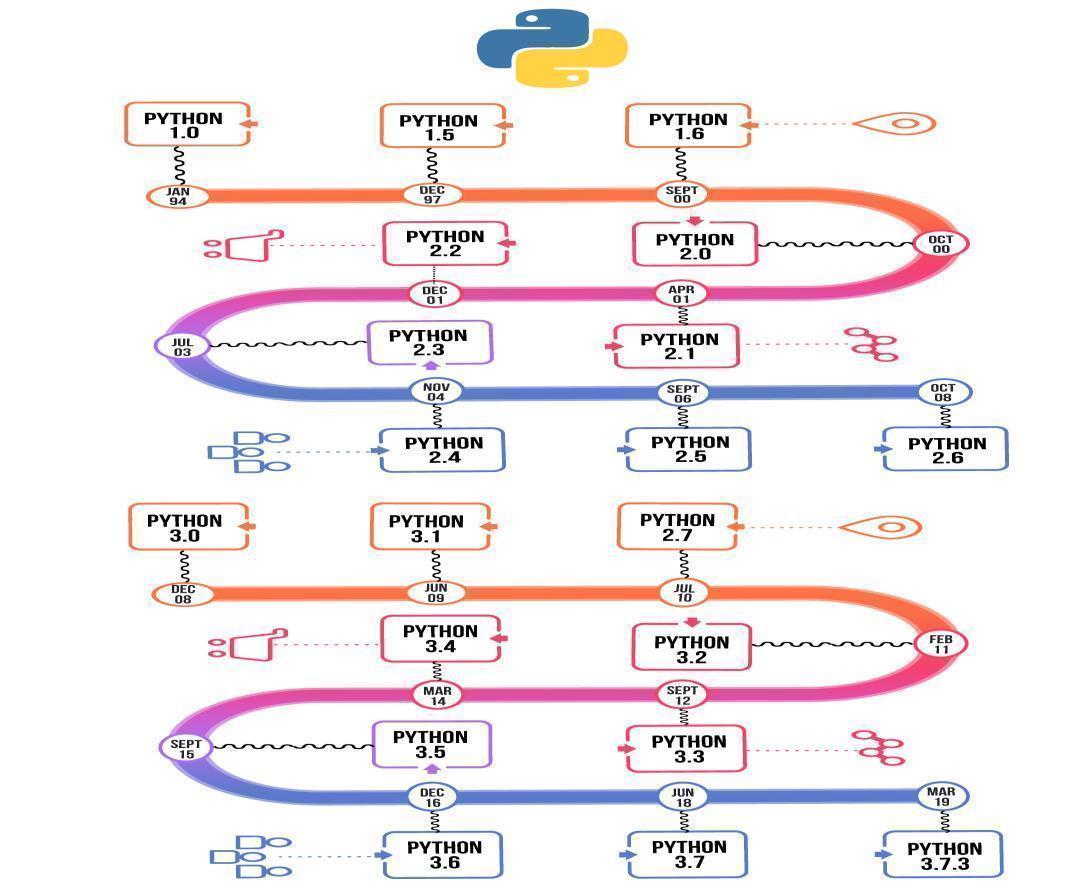
**Toast Message:** Popup text can be used to in various places to inform the player but it is not a good approach to use for the important things like when game ends a text pop up that level is completed but that does not sound rewarded enough so I have added toast message for the special events like level completion, character selection, new weapon reception.

# Chapter 4 – Technology and Tools

## 4.1 Python Programming Language

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is a dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). It was created by Guido van Rossum and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions: Python 2 and Python 3. Both are quite different.

## HISTORY OF PYTHON

In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application-based work in December of 1989 at Centrum Wiskunde & Informatica (CWI) which is situated in the Netherlands. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language in which Python is said to have succeeded is ABC Programming Language, which had interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did was really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language that had removed all the flaws. The inspiration for the name came from BBC’s TV Show – ‘Monty Python’s Flying Circus’, as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the “Benevolent dictator for life” (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox. The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data types exception handling and functions. 16 Following are the illustrations of different versions of Python along with the timeline.

**WHY PYTHON**

Python is consistently rated as one of the world's most popular programming languages. Python is fairly easy to learn, so if you are starting to learn any programming language then Python could be your great choice. Today various Schools, Colleges and Universities are teaching Python as their primary programming language. There are many other good reasons which makes Python as the top choice of any programmer:

* Python is Open Source which means it's available free of cost.
* Python is simple and so easy to learn.
* Python is versatile and can be used to create many different things.
* Python has powerful development libraries including AI, ML etc.
* Python is much in demand and ensures high salary

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

**Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented** − Python supports the ObjectOriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

### 4.2 Pygame

Pygame is a popular open-source library for game development in Python. It provides a set of modules and functions that simplify the creation of 2D games and multimedia applications. Pygame is built on top of the Simple DirectMedia Layer (SDL), a cross-platform development library that offers low-level access to audio, keyboard, mouse, and graphics hardware.

Key features of Pygame include:

1. **Graphics Rendering**: Pygame allows developers to create graphics, draw shapes, and display images on the screen. It provides functions for handling sprites, managing animations, and applying transformations to game objects.
2. **Input Handling**: Pygame enables the detection and handling of user input, including keyboard presses, mouse movements, and mouse clicks. Developers can easily respond to user actions and implement interactive gameplay mechanics.
3. **Audio Support**: Pygame provides features for playing and manipulating sound effects and music tracks. It supports various audio formats and allows developers to control volume, playback speed, and other audio properties.
4. **Event Handling**: Pygame offers an event-based system for handling user events, such as keyboard and mouse inputs, window resizing, and system events. Developers can define event handlers to respond to specific events and update the game state accordingly.
5. **Collision Detection**: Pygame includes functions for detecting collisions between game objects. Developers can check for collisions between sprites, rectangles, or other shapes, allowing for accurate collision detection and response in the game.
6. **Font and Text Rendering**: Pygame provides functionality for rendering text on the screen. Developers can specify font styles, sizes, and colors, and display dynamic or static text elements within their games.
7. **Game Loop Management**: Pygame simplifies the creation of the main game loop, which controls the flow of the game. Developers can define game logic, update the game state, and render graphics within the loop, ensuring smooth and responsive gameplay.
8. **Cross-Platform Compatibility**: Pygame is compatible with various platforms, including Windows, macOS, and Linux, making it easy to develop games that can run on different operating systems without major modifications.
9. **Extensibility**: Pygame is an open-source library, allowing developers to contribute to its development and create custom extensions or modules to enhance its functionality.

Pygame is a popular choice for beginners and experienced developers alike, as it provides a straightforward and accessible way to create 2D games using the Python programming language. Its simplicity, extensive documentation, and active community make it a valuable tool for prototyping, learning game development concepts, and building smaller-scale games

### Asset packs

Asset packs for games are pre-made collections of digital assets that game developers can acquire and use in their games. These assets include graphical elements, audio files, 3D models, animations, textures, scripts, and other resources that can be incorporated into a game project. Asset packs are typically created by artists, designers, or specialized asset providers and are available for purchase or free download from various online marketplaces or platforms.

The purpose of asset packs is to save development time and resources by providing ready-made content that can be used to enhance the visual and auditory aspects of a game. They offer a wide range of creative assets, allowing developers to quickly populate their game worlds, create appealing visual effects, and add immersive audio elements without having to create everything from scratch.

Asset packs can be tailored to specific genres or themes, such as fantasy, sci-fi, horror, or platformers. They often include assets in various file formats compatible with popular game engines or development frameworks, such as Unity, Unreal Engine, or Godot. Some asset packs also come with documentation, sample scenes, or example scripts to help developers integrate the assets seamlessly into their projects.

By utilizing asset packs, game developers can:

Save Time: Asset packs provide ready-to-use content, eliminating the need to create assets from scratch. This saves development time and allows developers to focus on other aspects of their games.

Improve Visuals: High-quality art assets in asset packs enhance the visual appeal of games, whether through character designs, background scenery, special effects, or user interface elements.

Enhance Audio: Asset packs often include sound effects, music tracks, and ambient audio that can significantly enhance the immersive experience of a game.

Maintain Consistency: Asset packs often have cohesive styles and themes, ensuring visual and audio consistency throughout the game.

Learn from Examples: Some asset packs provide sample scenes or scripts that developers can study and learn from, helping them understand best practices and techniques for integrating assets effectively.

However, it's important to note that while asset packs offer convenience and ready-made content, they may not always perfectly align with a game's unique vision. Developers should carefully consider the licensing terms and restrictions associated with each asset pack and ensure that the chosen assets fit well within the overall design of their game.

Overall, asset packs are valuable resources for game developers, providing a range of pre-made assets that can accelerate development, enhance the game's visuals and audio, and contribute to the overall quality and polish of the final product.

# Chapter 5 – Code

## 5.1 Code view

# Declaring variables to be used through the program

vec = pygame.math.Vector2

HEIGHT = 350

WIDTH = 700

ACC = 0.3

FRIC = -0.10

FPS = 60

FPS\_CLOCK = pygame.time.Clock()

COUNT = 0

# Create the display

#displaysurface = pygame.display.set\_mode((WIDTH, HEIGHT), flags, 32)

displaysurface = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption("Game")

# Run animation for the RIGHT

run\_ani\_R = [pygame.image.load("Player\_Sprite\_R.png").convert\_alpha(), pygame.image.load("Player\_Sprite2\_R.png").convert\_alpha(),

             pygame.image.load("Player\_Sprite3\_R.png").convert\_alpha(), pygame.image.load("Player\_Sprite4\_R.png").convert\_alpha(),

             pygame.image.load("Player\_Sprite5\_R.png").convert\_alpha(), pygame.image.load("Player\_Sprite6\_R.png").convert\_alpha(),

             pygame.image.load("Player\_Sprite\_R.png").convert\_alpha()]

# Run animation for the LEFT

run\_ani\_L = [pygame.image.load("Player\_Sprite\_L.png").convert\_alpha(), pygame.image.load("Player\_Sprite2\_L.png").convert\_alpha(),

             pygame.image.load("Player\_Sprite3\_L.png").convert\_alpha(), pygame.image.load("Player\_Sprite4\_L.png").convert\_alpha(),

             pygame.image.load("Player\_Sprite5\_L.png").convert\_alpha(), pygame.image.load("Player\_Sprite6\_L.png").convert\_alpha(),

             pygame.image.load("Player\_Sprite\_L.png").convert\_alpha()]

# Attack animation for the RIGHT

attack\_ani\_R = [pygame.image.load("Player\_Sprite\_R.png").convert\_alpha(), pygame.image.load("Player\_Attack\_R.png").convert\_alpha(),

                pygame.image.load("Player\_Attack2\_R.png").convert\_alpha(), pygame.image.load("Player\_Attack2\_R.png").convert\_alpha(),

                pygame.image.load("Player\_Attack3\_R.png").convert\_alpha(), pygame.image.load("Player\_Attack3\_R.png").convert\_alpha(),

                pygame.image.load("Player\_Attack4\_R.png").convert\_alpha(), pygame.image.load("Player\_Attack4\_R.png").convert\_alpha(),

                pygame.image.load("Player\_Attack5\_R.png").convert\_alpha(), pygame.image.load("Player\_Attack5\_R.png").convert\_alpha(),

                pygame.image.load("Player\_Sprite\_R.png").convert\_alpha()]

# Attack animation for the LEFT

attack\_ani\_L = [pygame.image.load("Player\_Sprite\_L.png"), pygame.image.load("Player\_Attack\_L.png"),

                pygame.image.load("Player\_Attack2\_L.png"),pygame.image.load("Player\_Attack2\_L.png"),

                pygame.image.load("Player\_Attack3\_L.png"),pygame.image.load("Player\_Attack3\_L.png"),

                pygame.image.load("Player\_Attack4\_L.png"),pygame.image.load("Player\_Attack4\_L.png"),

                pygame.image.load("Player\_Attack5\_L.png"),pygame.image.load("Player\_Attack5\_L.png"),

                pygame.image.load("Player\_Sprite\_L.png")]

# Animations for the Health Bar

health\_ani = [pygame.image.load("heart0.png").convert\_alpha(), pygame.image.load("heart.png").convert\_alpha(),

              pygame.image.load("heart2.png").convert\_alpha(), pygame.image.load("heart3.png").convert\_alpha(),

              pygame.image.load("heart4.png").convert\_alpha(), pygame.image.load("heart5.png").convert\_alpha()]

class Item(pygame.sprite.Sprite):

      def \_\_init\_\_(self, itemtype):

            super().\_\_init\_\_()

            if itemtype == 1: self.image = pygame.image.load("heart.png").convert\_alpha()

            elif itemtype == 2: self.image = pygame.image.load("coin.png").convert\_alpha()

            self.rect = self.image.get\_rect()

            self.type = itemtype

            self.posx = 0

            self.posy = 0

      def render(self):

            self.rect.x = self.posx

            self.rect.y = self.posy

            displaysurface.blit(self.image, self.rect)

      def update(self):

            hits = pygame.sprite.spritecollide(self, Playergroup, False)

            # Code to be activated if item comes in contact with player

            if hits:

                  if player.health < 5 and self.type == 1:

                        player.health += 1

                        health.image = health\_ani[player.health]

                        self.kill()

                  if self.type == 2:

                        handler.money += 1

                        self.kill()

class Player(pygame.sprite.Sprite):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.image = pygame.image.load("Player\_Sprite\_R.png").convert\_alpha()

        self.rect = self.image.get\_rect()

        # Position and direction

        self.vx = 0

        self.pos = vec((340, 180))

        self.vel = vec(0,0)

        self.acc = vec(0,0)

        self.direction = "RIGHT"

        # Movement

        self.jumping = False

        self.running = False

        self.move\_frame = 0

        #Combat

        self.attacking = False

        self.cooldown = False

        self.immune = False

        self.special = False

        self.experiance = 0

        self.attack\_frame = 0

        self.health = 5

        self.magic\_cooldown = 1

        self.mana = 0

        # Sound

        self.slash = 0

  def next\_stage(self):  # Code for when the next stage is clicked

            self.stage += 1

            #print("Stage: "  + str(self.stage))

            self.enemy\_count = 0

            self.dead\_enemy\_count = 0

            if self.world == 1:

                  pygame.time.set\_timer(self.enemy\_generation, 1500 - (50 \* self.stage))

            elif self.world == 2:

                  pygame.time.set\_timer(self.enemy\_generation2, 2200 - (30 \* self.stage))

class Castle(pygame.sprite.Sprite):

      def \_\_init\_\_(self):

            super().\_\_init\_\_()

            self.hide = False

            self.image = pygame.image.load("castle.png").convert\_alpha()

      def update(self):

            if self.hide == False:

                  displaysurface.blit(self.image, (400, 80))

## 5.2 Screen shorts





# Chapter 6 – Conclusion

In this game, I have implemented character unlock and selection system, achievable weapons and its individual weapon upgrade system, player experience gaining system, various types of AI enemies with different behavior and activities with following the idea of OOP, player and enemy receiving damage system, player reward system, added different effects to most of the player’s intractable objects and also added animations to all of the possible game objects including the user interface.

This game is solely made for the user’s entertainment purpose and its principle objective was to make a game that is similar to old Pixel Art RPG games but yet to have modern vibe.

## Future Plans

This game is made for PC and Android at the present. There are many scopes available for the improvement of this game including the additions of other functionalities.

Some of the plans for the future of this application could possibly be:

* + - To make an IOS version.
    - To change the game art entirely since for now I’m using free assets to design the levels.
    - To add more dungeons and increase player grade’s system.
    - Add more different enemies in the game
    - Adding mission giver NPCS.
    - Adding pet that follows the player and help him/her in the combat.

# References

Youtube.com

Javapoint.com

Pygame online forms

Open-source asset pack websites

Geekforgeeks.com

Github.com

https://www.google.com/aclk?sa=l&ai=DChcSEwiz15bY6oL8AhWtgksFHSVMD88YABAAGgJzZg&sig=AOD64\_3GNc9wz8NJXQ3sV0Um2In-5q08VQ&q&adurl&ved=2ahUKEwjB\_I\_Y6oL8AhX61HMBHTYbCRwQ0Qx6BAgHEAE

https://github.com/

https://www.geeksforgeeks.org/python-programming-language/