ASHESI

ASHESI UNIVERSITY

DYNAMIC STORY WRITER FOR COMPUTER ROLE-PLAYING GAMES

APPLIED PROJECT

B.Sc. Computer Science

Oluwatobi Oluwajimi Adelaja

2019

ASHESI UNIVERSITY

DYNAMIC STORY WRITER FOR COMPUTER ROLE-PLAYING GAMES

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

Oluwatobi Oluwajimi Adelaja

2019

DECLARATION

I hereby declare that this Applied Project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

…………………………………………………………………………………………

Candidate's Name:

…………………………………………………………………………………………

Date:

…………………………………………………………………………………………

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

…………………………………………………………………………………………

Supervisor's Name:

…………………………………………………………………………………………

Date:

…………………………………………………………………………………………

Acknowledgements

My appreciation to God for keeping me through the course of this project

To my supervisor who guided me to the completion of this project.

To my Mother, Seyram Kartey and Ayomide Adelaja for supporting me during this project.

ABSTRACT

Computer Role-Playing Games (CRPG) are a genre of video games were the player controls at least one character, the character has attributes and relationships with other game objects and is important to the story. They are one of many entertainment media that have storytelling as one of its major components. The better the story component of a game the better the gameplay experience is. This project explores an approach to improving gameplay of RPG games by personalization of game stories. This is achieved by having an AI system generate the game story at run time by considering the decisions and choices the player makes. A benefit of this will be that all players will have unique stories and whenever a player restarts the game they will get a new story. This creates an amount of gameplay experience with each story and players get to be actively involved in the narrative direction of the game.

Key Terms

Role Playing Games (RPG), Computer Role Playing Games (CRPG), Non-Playable Characters (NPC), Player Personalization, Dynamic Story

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Background and Problem

Grand Theft Auto V [24], the third best-selling video game in the world, took three years to develop [1] but took some gamers just days to finish. Marvel's Spiderman, the seventh-best selling PlayStation 4 game, took at least two years to develop [2], but three days after its official release there were full gameplay walkthroughs of the game on YouTube. Most Action-Adventure and Computer role-playing games (CRPGs) use static story structures [3]. When gamers finish the main storyline, they complete the objective of the game and the experience ends. Most games attempt to extend the gaming experience by providing Downloadable content – additional content provided for a released video game. Although, game studios take years to develop a game, the length of a game's story limits how much gaming experience a gamer could get. A solution to this problem would be to have multiple stories that a gamer will follow that would depend on the choices they make while playing the game [3]. The challenge this poses is, how do you extend the gaming experience in Computer Role-Playing Games by having multiple storylines that are affected by the gamer's decisions without significantly increasing development cost and time of a video game?

1.2 Proposed Solution

My proposed solution to this problem is having an Artificial Intelligence (AI) that generates game stories, missions, quests and non-playable characters throughout the game and alters the game story depending on the choices the player makes. The Game story will be generated at runtime giving a wide range of gameplay possibilities that are dependent on the choices the player will make. This will be possible by having a set of possibility variables for every component of a story – characters, actions, Locations, Questions, cutscenes, Timelines, Decisions, etc. that AI will use to craft the story. It will do this at the same time considering variables that cannot be together because the story may not make logical sense. The algorithm will use story writing techniques and styles that are relevant to an action-adventure RPG game. Some of these techniques adopted were inspired from John Truby's The Anatomy of a Story: 22 Steps to Becoming a Master Storyteller [4].

1.3 Objective

The objective of this project is to build a system that independently creates a story for an Action-Adventure CRPG game, directs the future of the story depending on the actions of the Non-Playable Characters (NPCs) and the player, and uses some reinforcement learning to affect how the NPC's act.

1.4 Related Work

This section explores the various games and research work done in the field of player personalization, adaptive stories and dynamic games.

Personalization by Player Decisions

Detroit: Become Human is an adventure game developed by Quantic Dream and published by Sony Interactive Entertainment for PlayStation 4, released worldwide on 25 May 2018 [5]. It is story driven interactive game that explores the lives of three androids with their decisions being controlled by the player. Every decision and the order of decisions have varying impacts on the future of the story. This game improves gaming experience by involving the players in the direction of the story. Personalizing player experience within the game by making their choices have consequences is essential in capturing and maintaining the interest of the player [3].

Personalization by Adaptive Learning of NPCs

Another game that involves player personalization is *Middle-earth: Shadow of War*, an action role-playing video game developed by Monolith Productions and published by Warner Bros. Interactive Entertainment [6]. This game uses the 'Nemesis System' that creates unique personal stories with every enemy and follower. Unlike other RPG games were the player dies and re-spawns like the previous actions never happens, the enemies in Shadow of War remember this experience, learn from it and improve because of it. The dynamic NPCs make the game more realistic and personalize player gameplay.

AI Generated Game Environment

No Man's Sky is an action-adventure survival game developed by Hello Games [7]. It achieves player personalization by having an infinite amount terrains and locations to explore. This was made possible by using AI to build the worlds.

Related Research

In the growing field of research on improving player personalization, researchers provide various approaches to achieve a degree of player personalization. Lopez and Bidarra [8] propose a model where the game records player performance and creates a model of players' actions, preferences, or personality. Given any state that the game is in, the model predicts the desired experience for the next game state. These models are then used to control the adaptation and generation engine that adjusts the appropriate game components. Fig 1.0 shows the overview of the architecture of the principle. Ramirez and Bulitko [9], in their paper on *Telling Interactive Player-Specific Stories*, propose "a system that uses a player model both at the design time and at the play time to generate and select stories fitting a specific player”. They achieve this by combining two systems, Automated Story Director (ASD) proposed by Riedl et al. [10] and Player-Specific Stories via Automatically Generated Events (PaSSAGE) proposed by Thue et al. [11]. They use PaSSAGE to tell player-specific stories based on different player models by foreseeing the possible ways the story can go at design time and use ASD as a planner to foresee other ways the story could go and create contingency narratives for it.

The system that is proposed in this paper to achieve player personalization focuses on giving each player a different game story within the same world. All the related work presented above use a singular story with the same set of characters and achieve player personalization by having alternative endings and various paths to those endings. The general story idea and character set remains the same. Characters will behave the same way in every play of the game and the beginning chapter/missions will be the same for every player. The system proposed attempts to give every player a completely different story, with different characters, that behave differently and will lead the player to different ends. This approach ensures that full story personalization can be fulfilled, where every player has his own story and a player can always replay the game and get a completely different story. *Figure 1.2* and *1.3* show diagrammatically how current player personalized game systems work and the proposed story writer system.

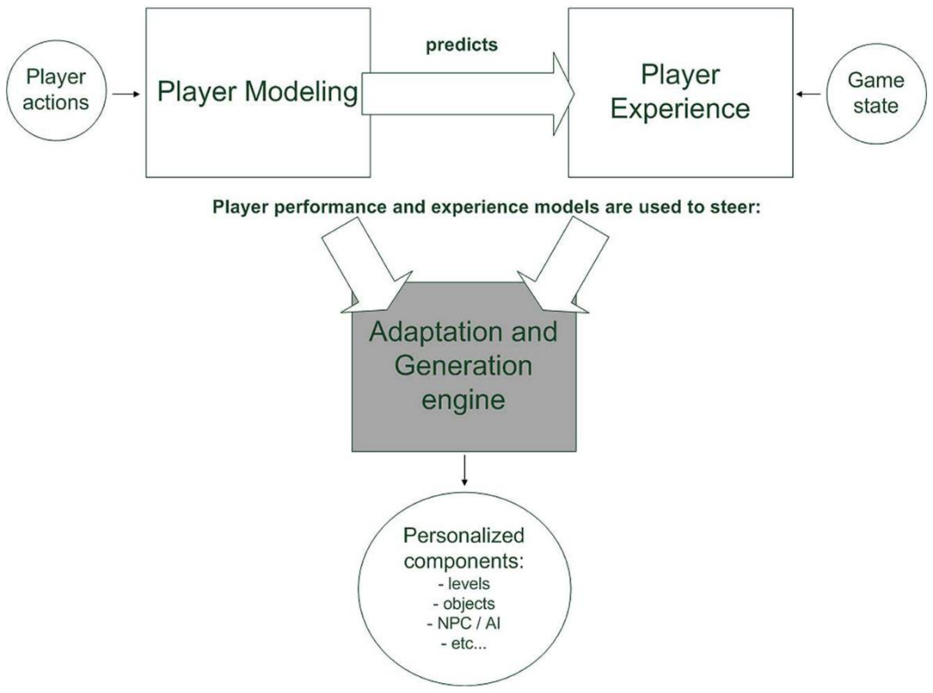


Figure 1.1: Overview of game adaptivity architectural principles [8]

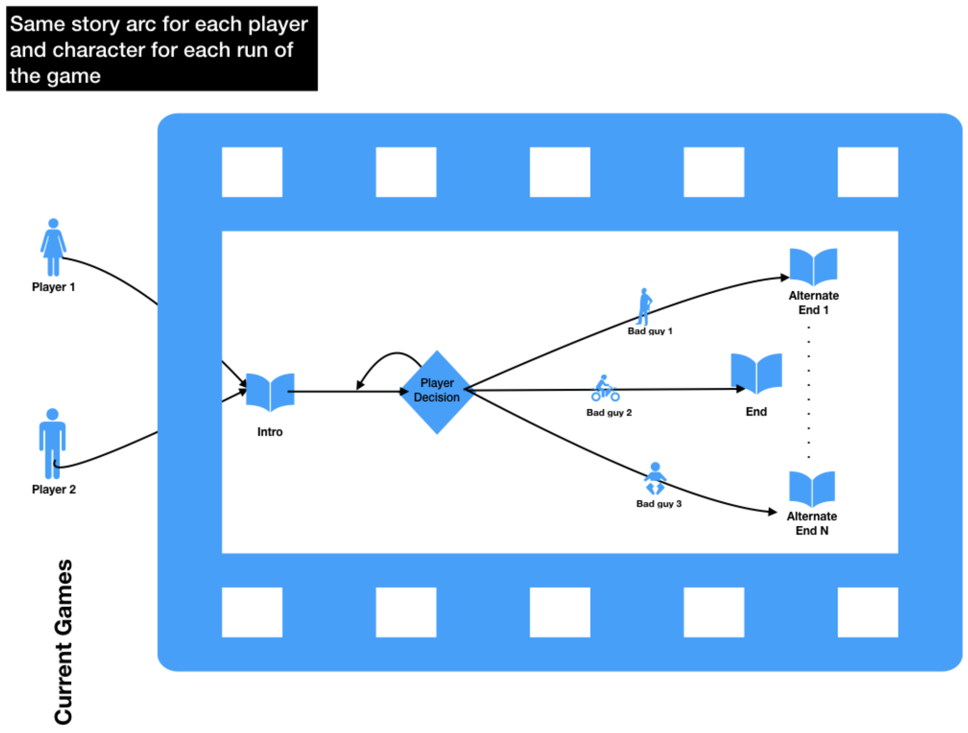


Figure 1.2: Overview of current player personalization systems

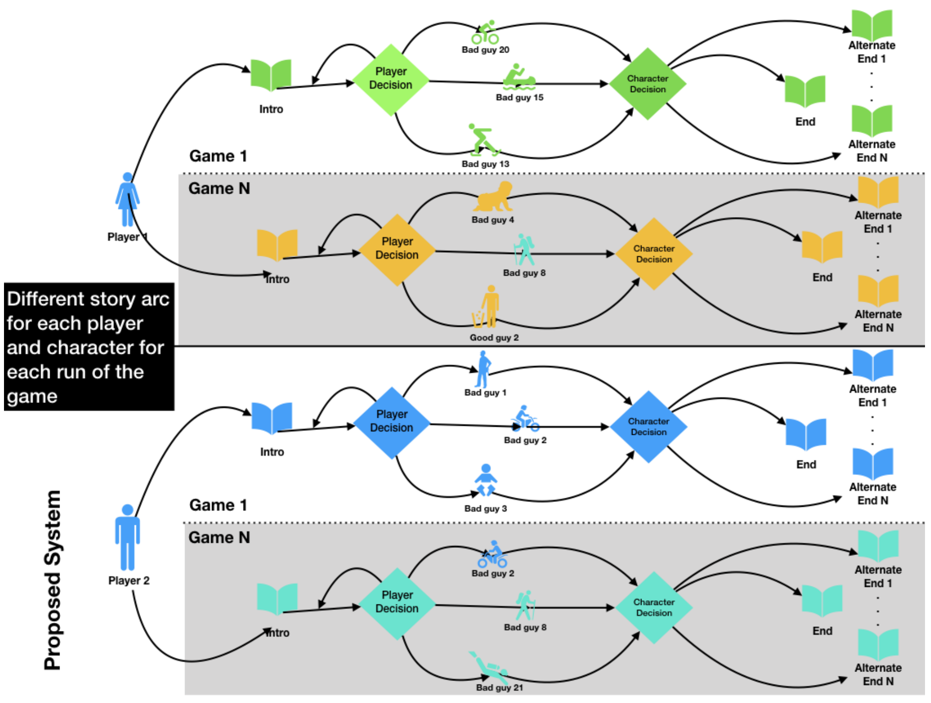


Figure 1.3: Overview of proposed dynamic story writer

Chapter 2: System Requirements

2.1 Overview

This chapter covers the use of the system, the users of the system and the different functionalities the system is meant to perform for the users. Section 2.2 describes the users of the system, 2.3 gives the scenarios of when and how the users will use the system, 2.4 explains the process of requirements gathering and analysis. The functional and non0functional requirements of the system will be explained in sections 2.5 and 2.6 respectively.

2.2 Description of Users and Scope

The users of this game system are video gamers, specifically expert gamers. All gamers should have an opportunity for a unique personalized story experience that make their choices have effects. Having a personalized character is one of the main reasons people play RPGs and a "personalized player experience is a key factor in making players feel involved in a virtual world" [3]. Expert gamers have a unique advantage with this kind of system, if they end up completing the game they can always replay and get a new story with a new gameplay experience. When games are replayed, the gaming experience is depleted because the player still plays in same story and may get bored, but a change in story will always give a refreshing set of experiences. If by chance they get a very similar story with one they have previously played, a different set of decisions that they may take will cause the game to adapt differently. In the perspective of the problem, this kind of system can keep gamers enjoying one game before a new game comes out. Due to regenerative gameplay experience a gamer can get out of such a system, it will be much better at keeping gamers entertained for longer periods than static storyline games.

2.4 Requirements Gathering and Analysis

The process of gathering my requirements required me to use previous academic work as a secondary source of requirements for the system. With the knowledge of what problem the system should solve and how it should solve it, the secondary data defined the functionalities that a system within this problem space should have.

2.5 Functional Requirements

Multiple Story Possibilities: The core of the system is to have it create the story for a CRPG. Some approaches have been proposed by researchers like dynamically controlling the quests [3], dynamically adjusting game elements like the Non-Player Characters [8], modelling the narrative based on predetermined player styles [11] or using a case-based approach to adapt NPC's from a knowledge base [12]. This functionality is needed to have the system always have a way of creating a story for the game. A distinct difference that this functionality will have from other related work is that, the function of the Game AI will be extended to be the primary Plot Writer and Game Designer. Unlike some of the other work done where the story has certain perimeters or fixed set of characters, the system should know how to write the story from scratch with new set of characters with new set of behaviors.

Realism and Storytelling: Every story is about characters who interact with each other having wants, needs and flaws [4]. Storytelling is an important factor that makes a game compelling and interesting to complete and for good storytelling to work well in games characters must have substance and relatable motivations for their actions [13]. Even after players begin to lose interests in the game mechanics, a good story will pull them forward. A good story is one where the main character has a desire/want, a need and a flaw that the story addresses [4, 13]. What makes characters relatable to the audience are these three components and this keeps them interested in the game. The system should be able to generate new stories every time, but the stories must be realistic and relatable to be a good story. Although there are no set rules that stories must follow in order to be good, it is necessary for game characters to be believable. NPC's need to have some self-awareness, intentional states and self-impelled actions [14] and although it may be impossible to create a fully believable character, getting these qualities embedded in them can help incorporate richer character design that will affect gameplay and narration.

Adaptation (Reinforcement Learning): This involves making the game components like NPCs and the environment adaptive to the actions that they perceive. NPCs should be designed to treat the player character and other NPCs based on their reputation and their knowledge of the character [3]. This can be done through a basic reinforcement learning approach were the NPC assigns values to each character for each category that determines how they will act towards the character. Every time that player characters act each of their values will be updated affecting how they perceive each other and therefore directing their judgment of the character for each category. This kind of dynamic modelling of NPC behavior towards each character makes them self-aware and intelligent, thereby contributing to the personalization gameplay and the realism of storytelling [8]. Like human beings, Characters learn from their past actions and encounters with other people and this shapes how they will behave in the future. Zhao's work on *Behavior Patterns of Characters in Story-Based Games*, even goes further by changing the learning rate of a character over time based on changes in the outcome of action trends [15]. It is crucial for NPCs and other game components to have behavioral intelligence because it is important in making the narrative realistic and compelling to gamers.

Decision System: To get players involved in the gameplay, the player should make decisions that have effects on the game storyline [3]. This is achieved by having a decision system within the game were the player can make choices. “The idea is to allow the player to develop his own character through his actions and choices within the interactive narrative” [16] which makes the player feel involved within the growth of the narrative.

2.6 Non-Functional Requirements

Scalability: The system should be able to handle and increasing number of characters within the game story without crashing or having a significant effect on performance

Playability: The player should be able to find the game easy to play and controls unambiguous.

2.3 Scenarios and Use Case Diagrams

The section gives a description of the system model and how the users interact with the system.

2.3.1 Class Diagram

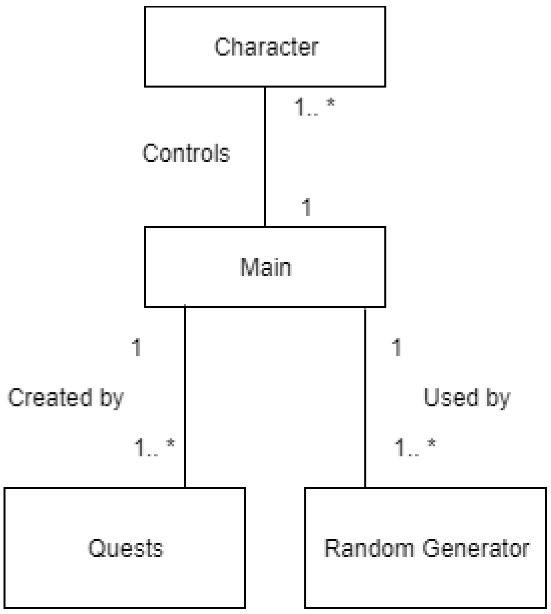


Fig 2.1: UML Diagram of Classes and Associations

Character Class: This class defines the attributes of a character within the story and their perceptions of other characters. It holds values that determine their behavioral disposition with other characters and when learning takes place within the characters these values are updated based on the rewards given for the action.

Table 2.1 Character Class

Character
Name Sex Nationality Known Characters Trusted Characters Desire Acquired Objects Action History Acquired Information Rivalry Trust Ideology

setCharacter( )
printCharacterDetails( )

Main Class: This class creates characters and quests for the player. It determines the actions characters can take, how they will act, how they will learn and how they will optimally behave in the future based on what they have learnt. It holds global variables of various data sets that characters and game objects can be defined by like possible name or locations.

Table 1.2 Main Class

Main
possibleMaleCharacterNames possibleFemaleCharacterNames possibleNouns questTypes setDesires possibleLocationsNames possibleTerritories possibleMoneyTreasures desireProbAction allCharacters deadCharacters noOfCharactersProb count

generateRandomCharacter()
charProbAction()
optimalRival()
optimalTrust()
learn()
desireActions()
act()
start()
Main()

Quests Class: This class holds he details and structure of quests that the player can perform

Table 2.3 Quest Class

Quest
questName questType questLocation

Random Generator: This is the class that defines a secure random generator using the RNGCryptoServiceProvider. It uses clock time along with other factors like system environment variables, to create less predictable random characters which will be used for picking random data variables for the characters in the Main class.

Table 2.2 Random Generator Class

Random Generator
RNGCryptoServiceProvider csp

RandomGenerator()
Next()

Chapter 3: Architecture and Design

3.1 Overview

This chapter covers the architecture and design of the system. Section 3.2 will give the architectural details of the Story Generator and Section 3.3 for the Game as a whole.

3.2 Architecture of Story Writer

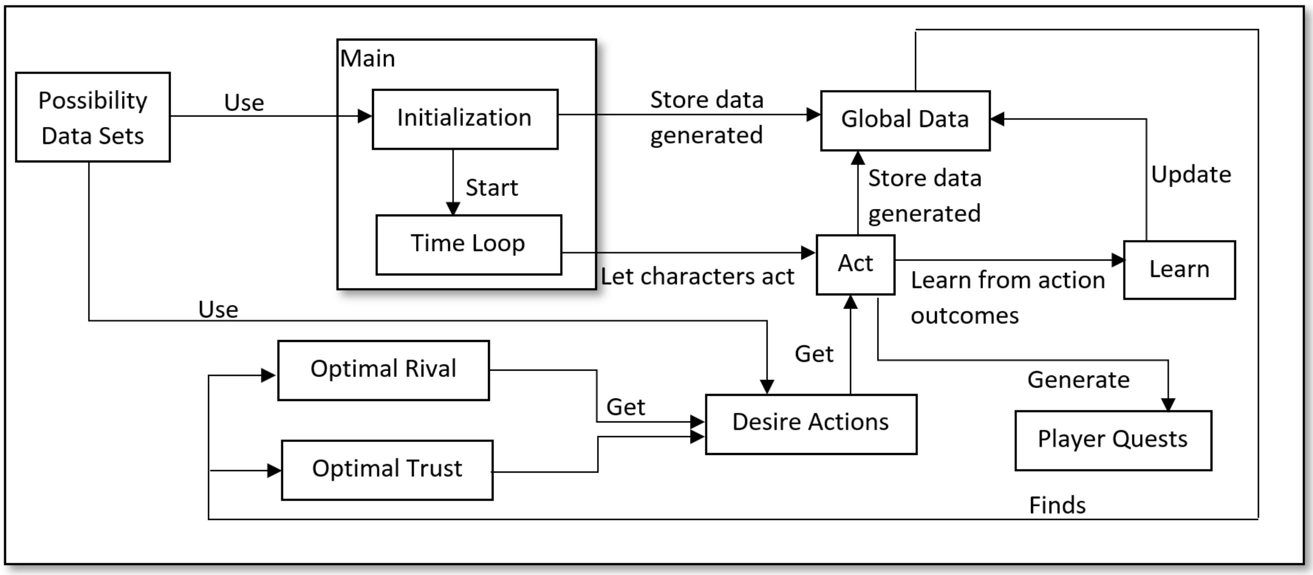


Figure 3.1: Architecture of Story Writer

From the diagram above, the story writer system begins from initializing all characters and setting their properties using data from the possibility Data sets. After initialization, the system enters a time loop that calls the *act()* functions on all the characters to make the characters act. The act function gets the action that the character should take from the *desireAction()* function which decides what action the character should take based on his desire. It considers the optimal rival (the character with the most rivalry) and the optimal trust (the character with the least distrust) in determining who a character should act on based on learning from past events. When a character acts, he and other characters learn from the outcome of that action that will alter how they will act in the future. All the data from the processing is continuously saved in Global data sets which will be sent to the Role-playing game occasionally.

3.3 Architecture of Role-Playing Game

MVC (Model-View-Controller Architectural View)

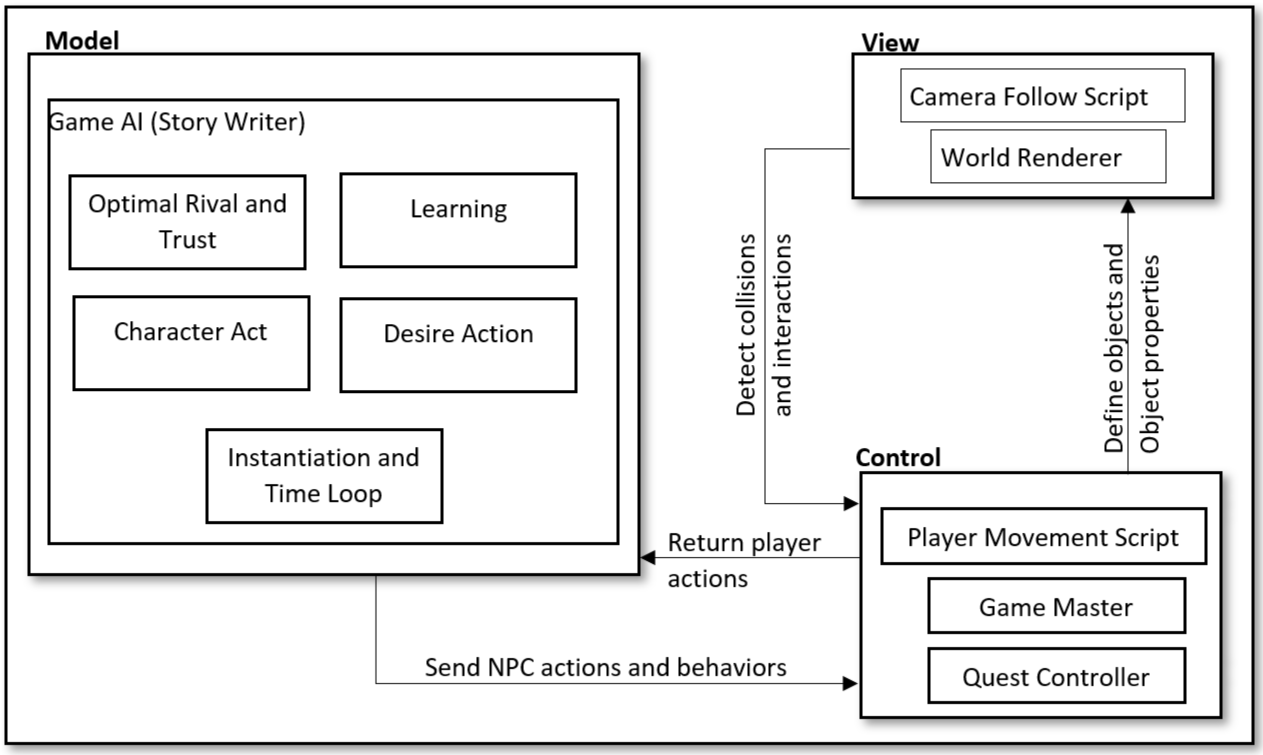


Figure 3.2: Model-View-Controller Architecture of Role-playing Game

The model component of the MVC architectural view manages data and does the data processing. The Model component of the RPG consists of the Story Writer which tells what the characters in the story are doing and their change in properties. It also sends the available quests that the player can take to the game master which is a control component. The Story Writer performs all data processing of the story and gives results to the Game Master.

The Control component manages user input and interactions with the system and within the RPG three components perform this, the player movement script, the Game Master script and the Quest Controller Script. The Game Master is the main controller that collects character data and actions from the Story Writer and represents this in the game by controlling game objects within the 3D world. When a character wants to perform a quest, it switches to quest mode from roam mode and give control over to the Quest controller that manages quest objects and interactions. When the player is done performing a quest, the Quest controller calls the game managers to switch modes and update the players actions in the Story Writer with whatever quest the player has completed and display all the actions that have taken place to the player as text in the 3D world. At the start of the roam mode, all the actions of the NPC's are also showed to the player as text.

The View component of a system controls what and how the user sees the data. In the RPG games, Unity handles the rendering of the 3D world and this serves as one of the components that controls the view of the game to the player. Another component that controls the view is the camera follow script. This is a script that controls the movement of the camera to follow the player based on the movement inputs that the player gives. This camera follow script sets the camera to view the player model in third person view, were the player character is visible on-screen during gaming. The third person view was used so that the player can be more aware of his surroundings.

Pipe and Filter Architectural View

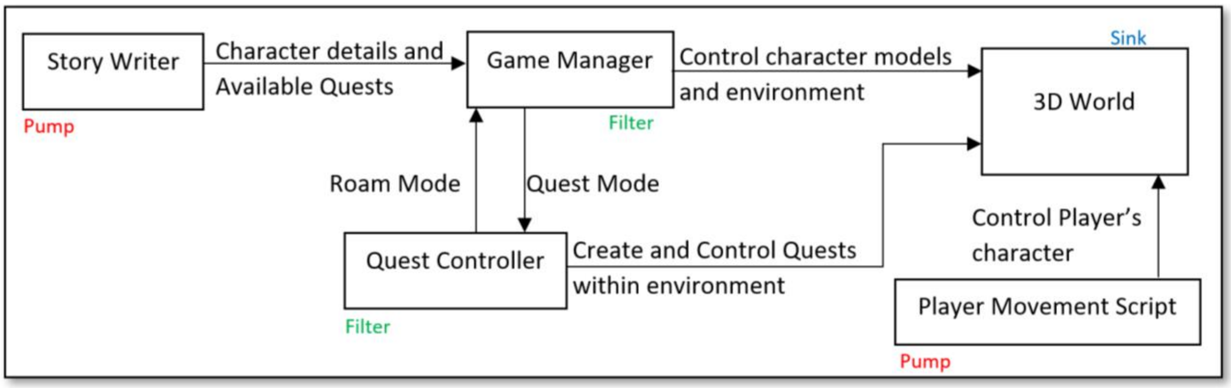


Figure 3.3: Pipe and Filter Architectural View of the Role-playing Game

The system architecture can also be viewed as a Pipe and Filter Architectural View. Sinks are the final destination for the data, which in this system is the story represented as a game. This is what the player will see when playing the game. Pumps are components that generate data or receive input from the user. The Player Movement Script receives input from the user as keys and is used to move the player's character within the game (the sink). The Story Writer rather generates data as characters, character actions and quests for the player to take. This data is sent to the Game Manager. The Game Manager and the Quest Controller are filters, which perform some processing on the data. The game manager converts the characters data and Quests into Unity Game objects that will exist within the game. When it is time to perform the quest, control is switched to the Quest Controller which does data processing on quests and turns them into activities that the player does in the game. The game objects and activities are created, controlled and displayed within the 3D world of the game.

Chapter 4: Implementation

3.1 Technologies Used

This section covers the various technologies used in the making of this system.

C#: a general object-oriented programming (OOP) language developed by Microsoft within its .NET framework. C# was used because it is the programming language used for the Unity game engine, so connecting the story writer system to a role-playing game will require it be coded in C# for ease of encapsulation.

Unity and Unity Engine Library: Unity is a cross-platform real-time engine developed by Unity Technologies with the ability to create games and interactive experiences in both 2D and 3D. It offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality. The UnityEngine Library contains components and classes that enable developers to control game object in Unity using script. Unity was chosen because it is free, was easier in access and learn. It also enabled easy importing of 3D models from Blender, a 3D application on the same platform.

Blender: is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender was used because it was lightweight, and the models are imported into Unity through the file without any additional plugins or functionalities. This made creating and updating models from blender into Unity seamless.

3.2 Story Writer Approach and Description of Components

The approach used in designing the story generator/writer adopted principles from Behavioral Based Control in the field of Robotics. “Behavior-based control (BBC) involves the use of ‘behaviors’ as modules for control. Thus, BBC controllers are implemented as collections of behaviors.” [18]. When various autonomous agents using the same behavioral control interact within the same environment emergent behaviors may occur. For example, a swarm of robots may have the behavior to be close to any nearby object that they detect and stay within a range of 10cm. This could cause an emergent behavior of clustering, were each robot will detect other robots in the swarm as an object and move close till 10cm. Some of the robots will tend to gather together since they are all trying to keep themselves 10cm close to each other. In applying this concept to designing the story writer, each character will be an autonomous agent like the swarm robots, and they will have a set of behaviors or actions they can perform based on some set of conditions. When interactions begin to take place some of the characters will begin to cause an emergent behavior were they mostly interact within themselves based in the conditions of their actions. This emergent behavior that will be exhibited by the characters will be the game story.

Stories that can be written for Role-playing games are a bit restricted because the story must be genre specific. For example, RPG stories are hardly have comedy as a central element because there is not much inciting action that could take place. Due to some creative restrictions like this, actions were limited to killing, allying with a character, acquiring information or items, destroying locations and a variable action for some specific task that doesn’t fall into any of the other categories.

When it comes to creating character journey’s or arcs, John Truby mentions that every character in great stories that have been written have a desire/want, a flaw and a need [4]. The desire “is what your hero wants in the story, his particular goal” and the “story doesn't become interesting to the audience until the desire comes into play”. The flaw or weakness is something that is holding the character back or is missing within him that is ruining his life and the need is “what the hero must fulfill within himself in order to have a better life”. In addition to what makes characters unique are their “evil tendencies” that I simplified to “ideologies”. This is a value that represents the probability of doing an action that is ethically evil. In a Blog Post by Massimo Pigliucci, a Professor of Philosophy at CUNY-City College, he outlines five types of moral concerns from researchers that determine how people exercise moral judgment [19].

1. Harm (to others)
2. Fairness
3. In-group Loyalty
4. Respect for Authority
5. “Purity,” meaning respect for the sacred or the proper.

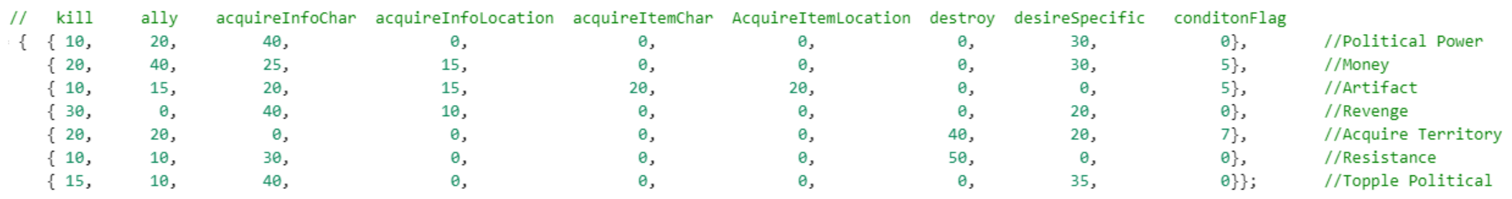
Only some of the criteria could be explored and implemented into the design of the system, Harm and In-group Loyalty. Ideology values range from 1 – 99 and the higher the ideology value the more “evil” the character will be and the lower the ideology value the more “good” character will be. The more evil the character, the higher the probability for the character to either kill another character (harm) or be more disloyal to their alliance.

Each character has demographic characteristics like Name, Sex and Nationality, in addition to personality characteristics like ideology and desire/want. With these properties, the following systems were implemented to achieve the requirements stated in chapter 2:

1. Desire and Desire Journey: Desire Journeys are the actions that characters take in order to get to their desire. Characters will not know when they achieve their desires, but desire journey keeps characters acting in a way that specific to their desire. The actions and probability of occurrence for each action are determined by a probability matrix *(Figure 4.1)*, with each desire having their own set of probabilistic values for each action.

2. Conditioned Randomization: At initialization of the story, character's properties are randomly selected from the possible data sets like possible character names. Randomizations were used to make choices when there is no data available to make that decision or no rational basis for the decision like at initializations. In situations where characters have no basis for choosing who they should kill, randomizations are used. When they are more than one optimal character for selecting who to ally or kill, randomizations are used to select any of the optimal characters. Although this will cause some actions within the story to not make sense, randomization is the only way to make decisions when there is no basis for making that decision (except from having default actions). For the RPG game that was being designed, having the characters perform default actions instead of randomizing was not desirable because this will make the story more predictable. For each desire, randomizations are used in conjunction with the probability matrix (Figure 4.1) to select which action is to be taken. When the randomized number value falls within a range of numbers, which their difference to the ratio of the possible numbers will form the probability for that action, that action will be taken. For example, if a character is taking an action based on randomization and the randomized number produced has the value of 32 (random between 0 to 100)

and the character has the “money” desire, the value will be cross referenced with the probability matric in Figure 4.1 to return an action of allying with another character. This way the kind of actions that characters with a certain desire are going to take will be fairly controlled in a way that makes sense to the desire.



```
//   kill     ally    acquireInfoChar  acquireInfoLocation  acquireItemChar  AcquireItemLocation  destroy  desireSpecific   conditonFlag
{  { 10,      20,           40,                  0,                0,                  0,             0,          30,             0},        //Political Power
   { 20,      40,           25,                 15,                0,                  0,             0,          30,             5},        //Money
   { 10,      15,           20,                 15,               20,                 20,             0,           0,             5},        //Artifact
   { 30,       0,           40,                 10,                0,                  0,             0,          20,             0},        //Revenge
   { 20,      20,            0,                  0,                0,                  0,            40,          20,             7},        //Acquire Territory
   { 10,      10,           30,                  0,                0,                  0,            50,           0,             0},        //Resistance
   { 15,      10,           40,                  0,                0,                  0,             0,          35,             0}};       //Topple Political
```

Figure 4.1: Probability Matrix of Action to perform based on character desire

3. Ideology, Rivalry and Trust Learning: Characters have a choice to perform or not perform certain actions like killing or destroying locations but not performing an action prevents a character from achieving his desire (theoretically). When characters do not perform actions, they learn that they have to and so their ideology updates to so that they have to take those actions even though they have to kill more. When a character has an ally killed by another character, their rivalry towards that character and his allies increase. The rivalry towards the killer’s ally increases based on the trust the killer has to each ally. When the character wants to perform a “kill” action, they select they kill the character with the highest rivalry value, the character they have the most opposition with. This way, there is tension and opposition between characters, a key component to good story telling [4]. When the rivalry between characters increase, so will how much their distrust the character and if the character wants to perform an ally action they select the character with

the least distrust i.e. the character with the most trust. This way they trust characters that they have the least issues with or conflicts of interest. At initialization, all rivalry values between characters are initialized to 0 but trust values are initialized based on the desires between characters. Desires that have a conflict of interest have higher distrust values than desires of similar interests and this makes characters ally with people they have similar desires with

4. Questing System: The system was built by treating the player character as another NPC within the story. Quests will be a means by which the player can perform actions and affect the game story. When it is the player’s turn to act like other characters within the game, a quest is created and added to the list of available quests that they can perform. Then the player is given an option to perform it. To have more than one quest options that they can pick from, allies of the character also create quests for the player to perform. Within the RPG, players will walk to the locations where the quest will begin in order to perform a quest. This will trigger a detector that will tell the Game that the player wants to perform a certain quest at a location, and this will cause the quest to initiate.

3.3 Implementation of Role-Playing Game

Using the Unity Game engine, a default scene called “World” was created and this serves as the 3D environment where the player will interact with the Non-player characters and the environment. The Role-playing game serves as an implementation of the story writer within a real game. The RPG contained the basic elements of an RPG – Ability for player to control the main character, choice of quest to perform at any given time and completing quests. The various components are described below:

1. Game Story Writer: This is the dynamic story writer described in the previous section that produces each Non-playable character's behavior and generates quests for the player to perform. The actions that NPCs take, and the quests generates are sent to the Game Master which will control the 3D environment to represent them to the player.
2. Game Master: Imports and creates an object of the Game Story Writer to generate characters and character actions. At game start, Unity runs the code within the *start()* function. Within the function, Game Master collects the player's Avatar Name and Gender and sets them as attributes within the Game AI object so that it creates a character object within it for the player. The start method within the Game AI object is called and the Game Master pulls the NPCs generated within the object to instantiate NPC models within the 3D world and displays NPC actions (*Figure 4.2 and Figure 4.3).*
3. Quest Controller: This component of the game environment when a player is performing a quest in a way that is specific to the quest type of the quest being done. For example, if the player must defend a location or a person, the player has to prevent the enemies from taking over the location or killing the person to be protected. This can involve the quest controller to instantiate enemies to target the location or person for a given amount of time or number of enemy swarms. If the player can defend the location/person, then the quest will be achieved. This will be different from when the player is performing an assassination type quest. The quest controller will create a target enemy that the

player must destroy and when the player achieves that the quest will be completed. For different quest types, the quest controller should set the game environment differently. The quest details are then displayed unto the screen and the quest goal marked by a blue pin *Figure 4.4*.

4. 3D World and Models: The 3D models used in the RPG were basic cubes as characters and objects as markers for quests and quest goals. Since the purpose of the game is solely for implementing the story writer in a game, the game models were kept basic and not detailed. A sword (figure 4.2) was used to mark the position of available quests that the player can perform. When the player goes to that location, the quest will begin and game objects and markers that have nothing to do with the quest are removed. The Quest controller then sets the quest goal that is marked by a location pin as seen in figure 4.4.

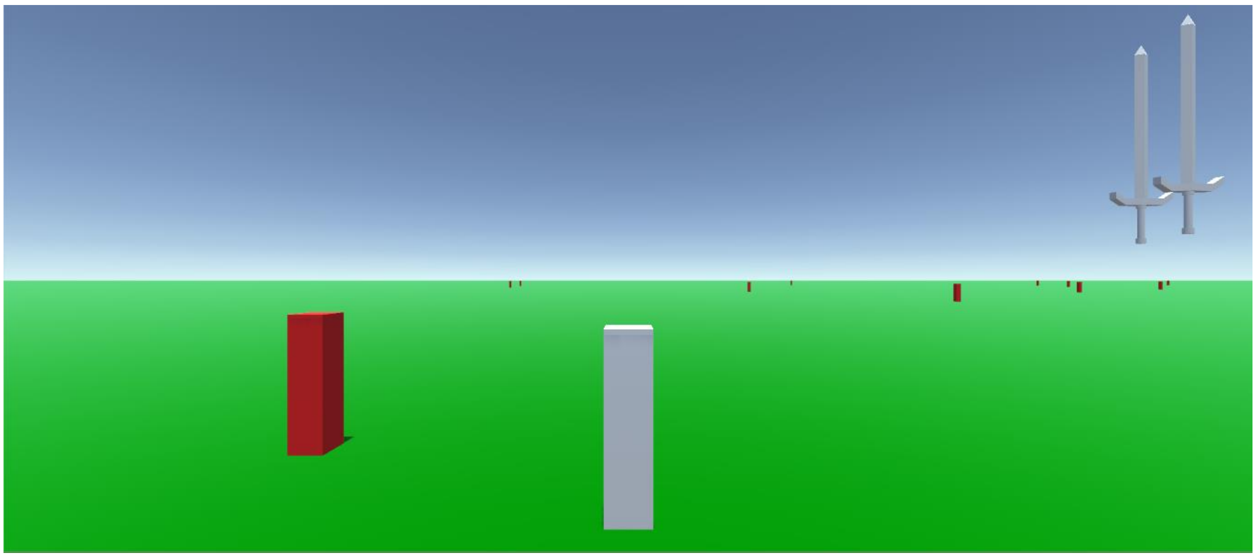
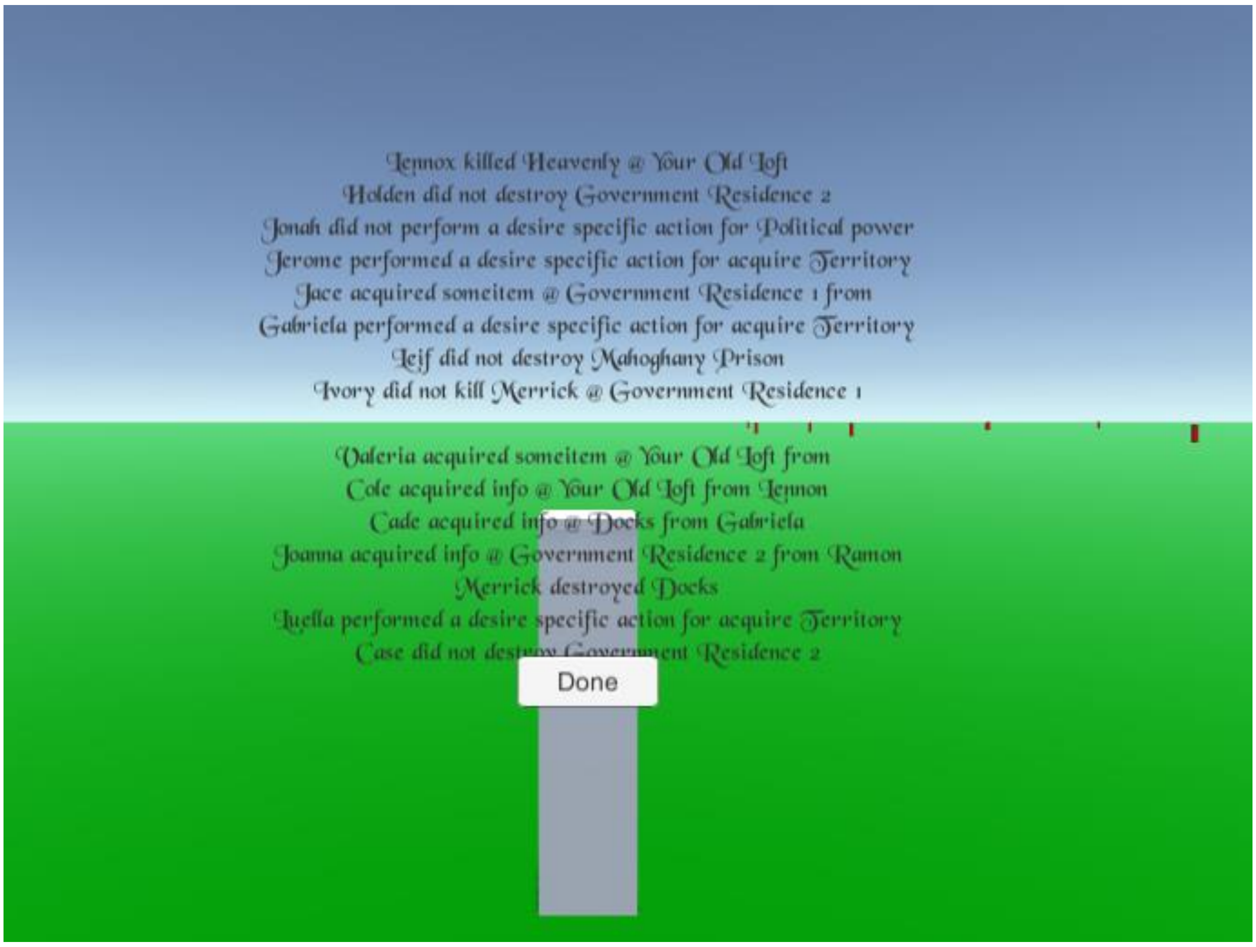
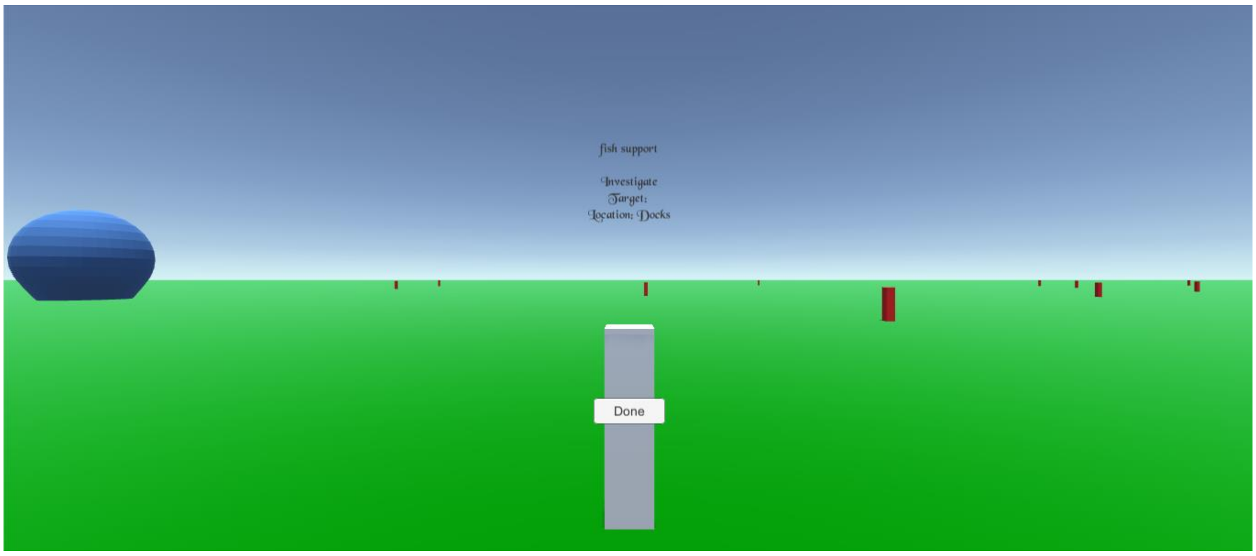


Figure 4.2: Game Environment showing NPCs (red cuboids), the player character (grey cuboid) and available quests locations


Lennox killed Heavenly @ Your Old Loft
Holden did not destroy Government Residence 2
Jonah did not perform a desire specific action for Political power
Jerome performed a desire specific action for acquire Territory
Jace acquired someitem @ Government Residence 1 from
Gabriela performed a desire specific action for acquire Territory
Leif did not destroy Mahoghany Prison
Ivory did not kill Merrick @ Government Residence 1

Valeria acquired someitem @ Your Old Loft from
Cole acquired info @ Your Old Loft from Lennon
Cade acquired info @ Docks from Gabriela
Joanna acquired info @ Government Residence 2 from Ramon
Merrick destroyed Docks
Luella performed a desire specific action for acquire Territory
Case did not destroy Government Residence 2

Figure 4.3: Start of Roam Mode. Game Master displays NPC actions generated by the story writer


fish support

Investigate
Target:
Location: Docks

Figure 4.4: Initialization of Quest mode. The blue ellipse is the location of the quest goal.

3.4 Pseudo Code of Key Algorithms and Functions

3.4.3 Learning Function

```
//Function updates the learning variables after an action outcome
//Input: The effector character, the affected character, degree of change of learning variable and the category of learning

function learn(effector, affected, change, category):
  if category == "Kill":
    for i : all affected's trusted characters:
      if i is not null: //i.e. dead
        i rivalry to effector += change
        i trust to effector = 30% *i trust to effector + 70% *i rivalry to effector

        //alter the degree of rivalry to the effectors ally's based on how much the effector trusts the ally
        for j : all effector's trusted characters:
          if j is not null: //i.e. dead
            i rivalry to j += change*(effectors trust to j)
            i trust to j = 30% *i trust to j + 70% *i rivalry to j
```

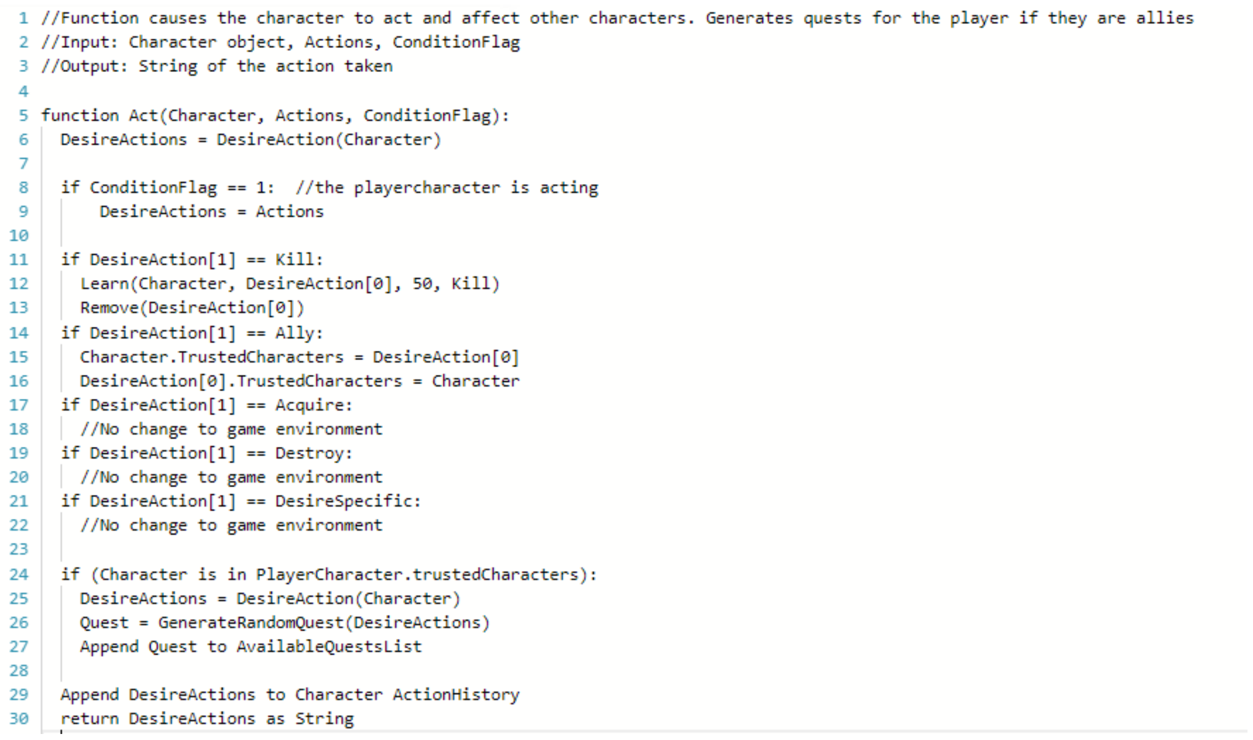
Listing 3. 2: Learning Function pseudo code

3.4.4 Desire Actions Function

```
//Function determines the action a character should take
//Input: Character object
//Output: Array of Action description [TargetCharacter, Action type, Location, Acquirable]

function DesireAction(Character):
  randomNumber =  RandomInterger(0:100)
  Cross reference randomNumber to the probability matrix to determine the action
  if action == kill:
    TargetCharacter = OptimalRival(Character)   //get the character that has the maximum rivalry value in character's rivalry array
  if action == Ally:
    TargetCharacter = OptimalTrust(Character)   //get the character with the minimum distrust value in character's trust array
  if action == Acquire Information from Character:
    TargetCharacter = RandomCharacter()
    Acquirable = Info
  if action == Acquire Information from Location:
    Acquirable = Info
  if action == Acquire Item from Character:
    TargetCharacter = RandomCharacter()
    Acquirable = Item
  if action == Acquire Item from Location:
    Acquirable = Item
  if action == destroy:
    //nothing specific
  if action == desire specific:
    //nothing specific. Specific actions will be hangled by game master.

  Location = RandomLocation()
  return [TargetCharacter, action, Location, Acquirable]
```

Listing 3. 3: DesireActions Function pseudo code

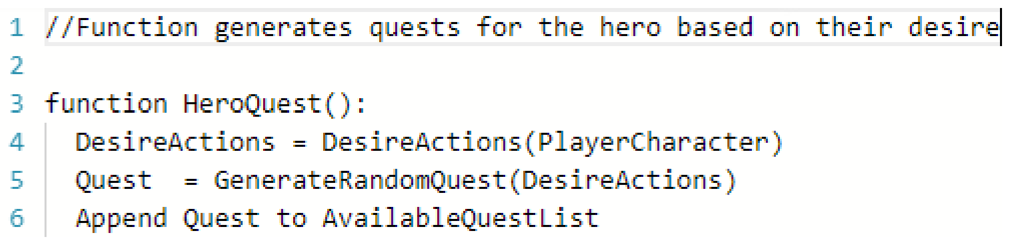
3.4.5 Act Function (NPCs)



```
//Function causes the character to act and affect other characters. Generates quests for the player if they are allies
//Input: Character object, Actions, ConditionFlag
//Output: String of the action taken

function Act(Character, Actions, ConditionFlag):
  DesireActions = DesireAction(Character)

  if ConditionFlag == 1:  //the playercharacter is acting
      DesireActions = Actions

  if DesireAction[1] == Kill:
    Learn(Character, DesireAction[0], 50, Kill)
    Remove(DesireAction[0])
  if DesireAction[1] == Ally:
    Character.TrustedCharacters = DesireAction[0]
    DesireAction[0].TrustedCharacters = Character
  if DesireAction[1] == Acquire:
    //No change to game environment
  if DesireAction[1] == Destroy:
    //No change to game environment
  if DesireAction[1] == DesireSpecific:
    //No change to game environment

  if (Character is in PlayerCharacter.trustedCharacters):
    DesireActions = DesireAction(Character)
    Quest = GenerateRandomQuest(DesireActions)
    Append Quest to AvailableQuestsList

  Append DesireActions to Character ActionHistory
  return DesireActions as String
```

Listing 4: Act Function pseudo code

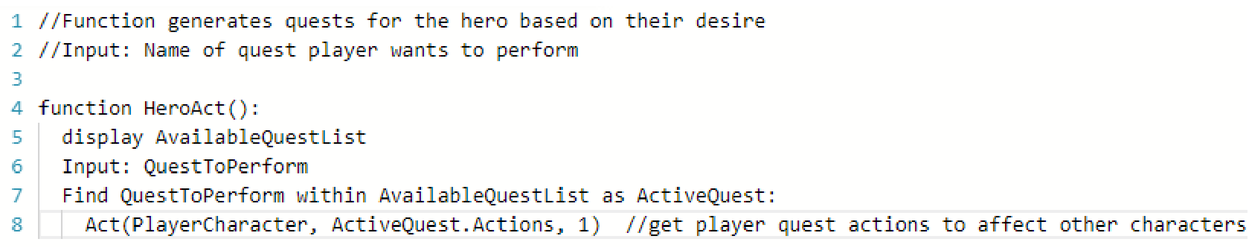
3.4.6 Hero Quest Function



```
//Function generates quests for the hero based on their desire

function HeroQuest():
  DesireActions = DesireActions(PlayerCharacter)
  Quest  = GenerateRandomQuest(DesireActions)
  Append Quest to AvailableQuestList
```

Listing 5.5: HeroQuest Function pseudo code

3.4.7 Hero Act Function



```
//Function generates quests for the hero based on their desire
//Input: Name of quest player wants to perform

function HeroAct():
  display AvailableQuestList
  Input: QuestToPerform
  Find QuestToPerform within AvailableQuestList as ActiveQuest:
    Act(PlayerCharacter, ActiveQuest.Actions, 1)  //get player quest actions to affect other characters
```

Listing 5. 6: HeroAct Function pseudo code

In addition to these functions, other helper functions were written to perform basic functionalities like finding the Optimal Rival (maximum rival value), Optimal Trust (minimum trust value), appending character objects to lists and finding characters within a list given their name.

Chapter 5: TESTING

3.1 Overview

This chapter covers the testing performed on the story writer and the RPG game created and the results from the performed tests

3.2 Unit Testing

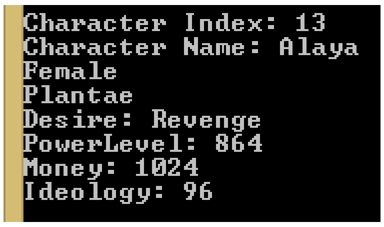
3.2.1 Test Case: A character object is created and assigned a random desire. Testing the Character class

Expected Results: The desire actions/behavior should reflect the desire that has been assigned to the character based on the probability data of each desire in the probability matrix.

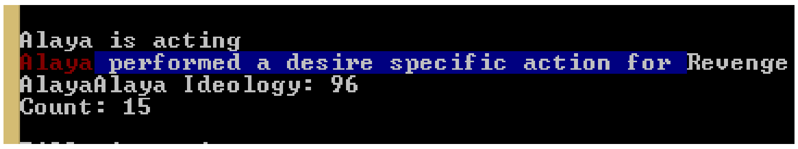
Test Results:

Table 5.1: Test Results for Creating the Character Object

Valid Input	**Result**
When a random character is generated, they are made to perform actions	As seen in *Figure 5.1*, characters behave based on the desire that is assigned to them
Invalid Input	**Result**
The character is assigned a desire that isn't represented in the probability matrix	The character doesn't perform an action



```
Character Index: 13
Character Name: Alaya
Female
Plantae
Desire: Revenge
PowerLevel: 864
Money: 1024
Ideology: 96
```



```
Alaya is acting
Alaya performed a desire specific action for Revenge
AlayaAlaya Ideology: 96
Count: 15
```

Figure 5.1: Results from Testing the Character Class

3.2.2 Test Case: Characters act and learn from the outcomes of the actions. This should feed into the Rivalry system

Precondition: Character should not be dead

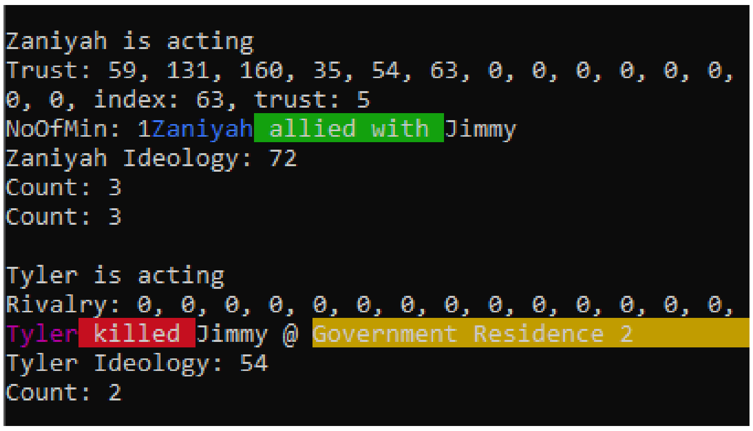
Expected Results: After the outcome of an action is known by the character, the character should adjust its learning variables, and base its future actions on these values.

Test Results:

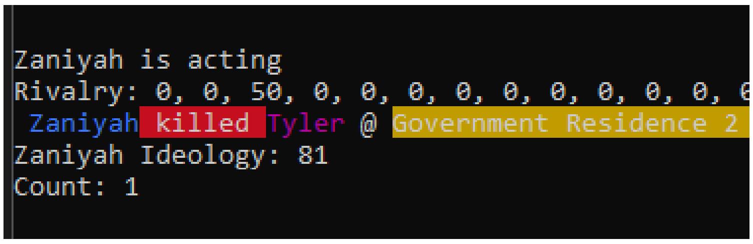
Table 5.2: Test Results of Character learning

Valid Input	**Result**
When characters interact with themselves, and two characters A and B ally, if a character C kills character B	The rivalry value between of character C to character A should increases and this prompts A to kill C. This scenario is seen in *Figure 5.2*.
If a character has more than one max rivalry value with other characters	One of the characters with the max rivalry value is chosen randomly and killed
Invalid Input	**Result**

When characters interact with themselves, and two characters A and B ally, if a character C kills character B but character C dies later on because of another character	Character A doesn’t attempt killing character C again because C is dead



```
Zaniyah is acting
Trust: 59, 131, 160, 35, 54, 63, 0, 0, 0, 0, 0, 0,
0, 0, index: 63, trust: 5
NoOfMin: 1Zaniyah allied with Jimmy
Zaniyah Ideology: 72
Count: 3
Count: 3

Tyler is acting
Rivalry: 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
Tyler killed Jimmy @ Government Residence 2
Tyler Ideology: 54
Count: 2
```



```
Zaniyah is acting
Rivalry: 0, 0, 50, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
 Zaniyah killed Tyler @ Government Residence 2
Zaniyah Ideology: 81
Count: 1
```

Figure 5.2: Scenario were the rivalry between characters change because of the outcome of an action

3.2.3 Test Case: Camera Follows Player Characters in third person view when the player moves

Precondition: The game must have started

Expected Results: The camera follows the player from behind as the player moves

Test Results:

Table 5.3: Test results for Camera Following player

Valid Input	**Result**

The player moves left, right, forward or backward	The camera moves left, right, forward or backward respectively following the player

3.2.4 Test Case: Receiving Player Input and performing quests by detecting the player in a certain location

Precondition: The game must have started

Expected Results: The player's character moves in the direction corresponding to the key movement mapping

Test Results:

Table 5.4: Test results for receiving player input

Valid Input	**Result**
When the player presses right, left, up or down arrow key	The player's character moves right, left, forward and backward respectively
The player presses the space bar	The player's character jumps
Invalid Input	**Result**
If the player presses any other key	The game doesn't respond

3.3 Component Testing

This section shows the testing of various components of the Story Writer and the Role-playing Game and the results of the testing

3.3.1 Test Case: Multiple Story Possibilities

Precondition: None

Expected Results: For every new game started the story should be different from the previous game played

Test Results: Some actions of NPC's in some games were recorded and the storyline of actions were represented as a chart of actions that the characters took

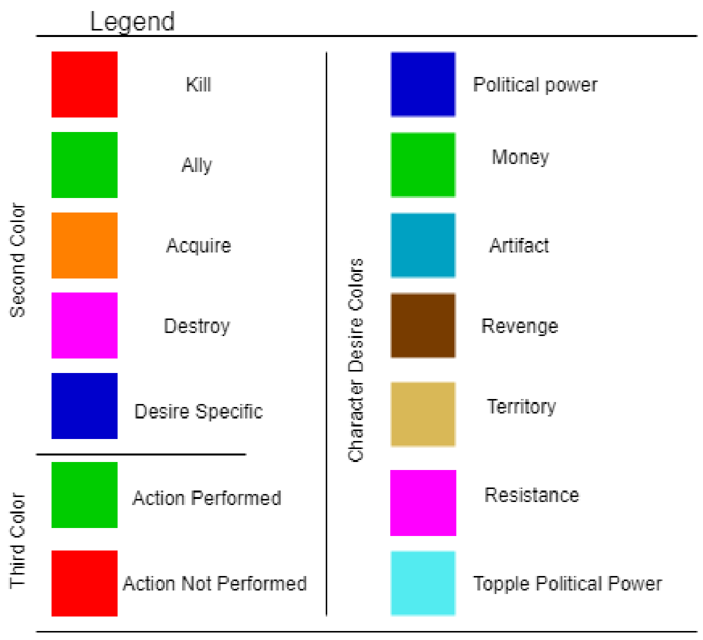


Figure 5.3: Color Legend for Game story charts

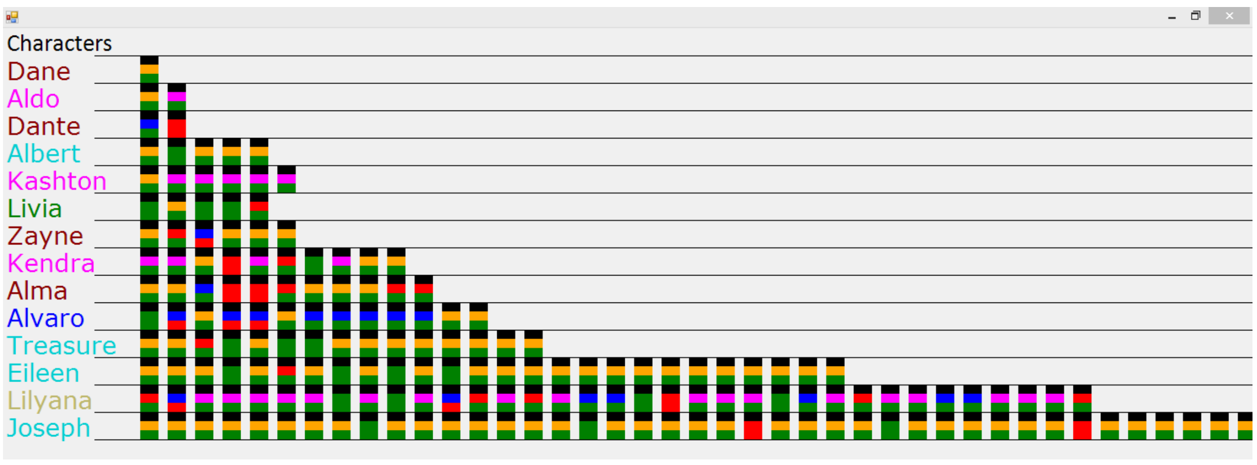


Figure 5.4: Game Story Outcome 1

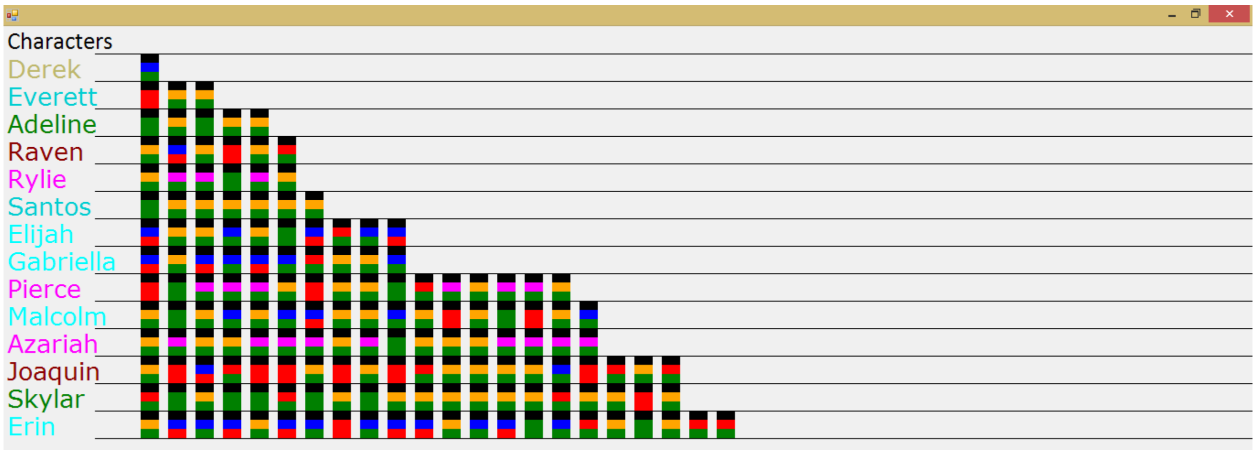


Figure 5.5: Game Story Outcome 2

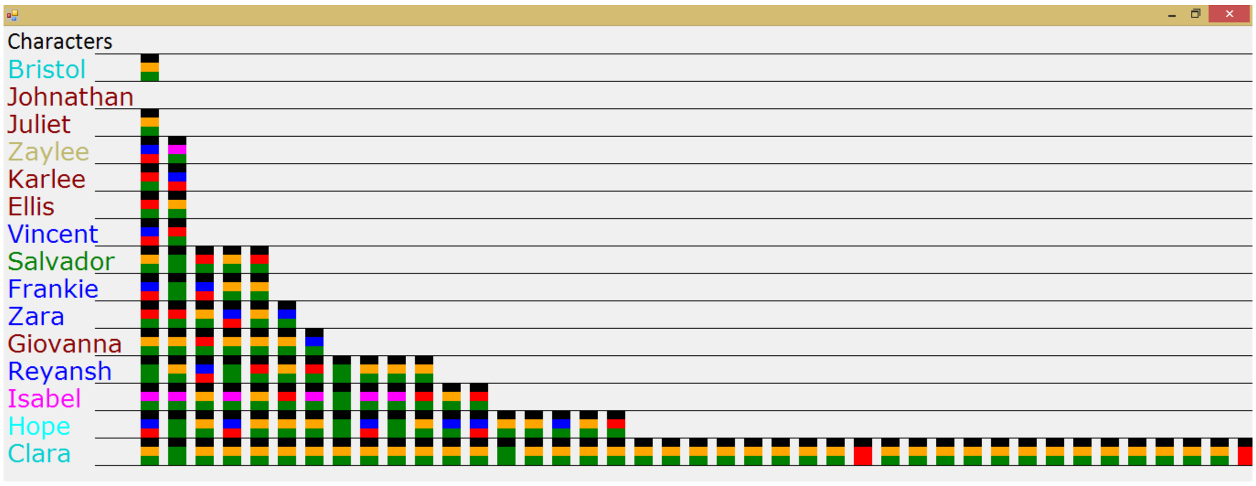


Figure 5.6: Game Story Outcome 3

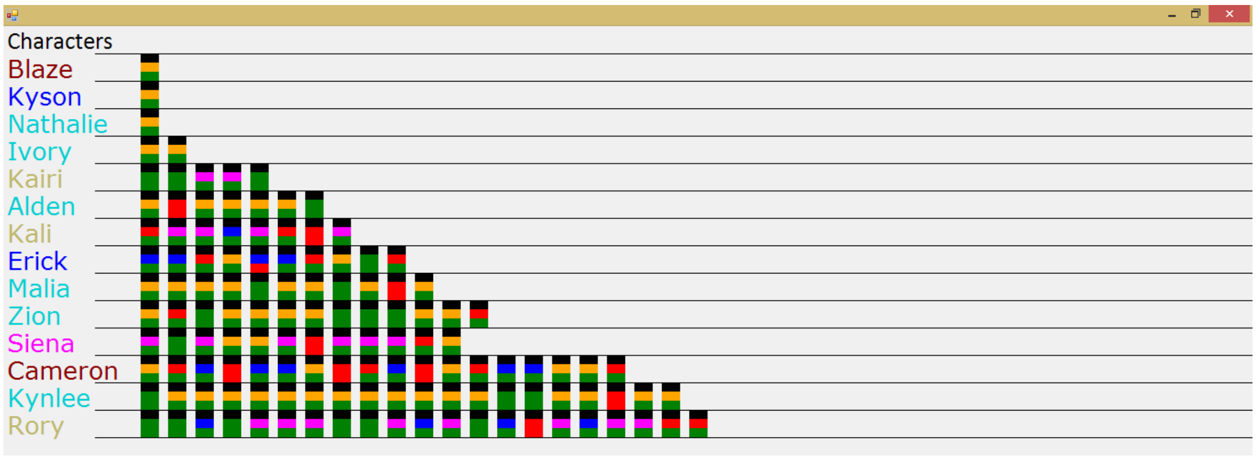


Figure 5.7: Game Story Outcome 4

3.3.2 Test Case: Capturing Avatar Details within the Role-playing Game

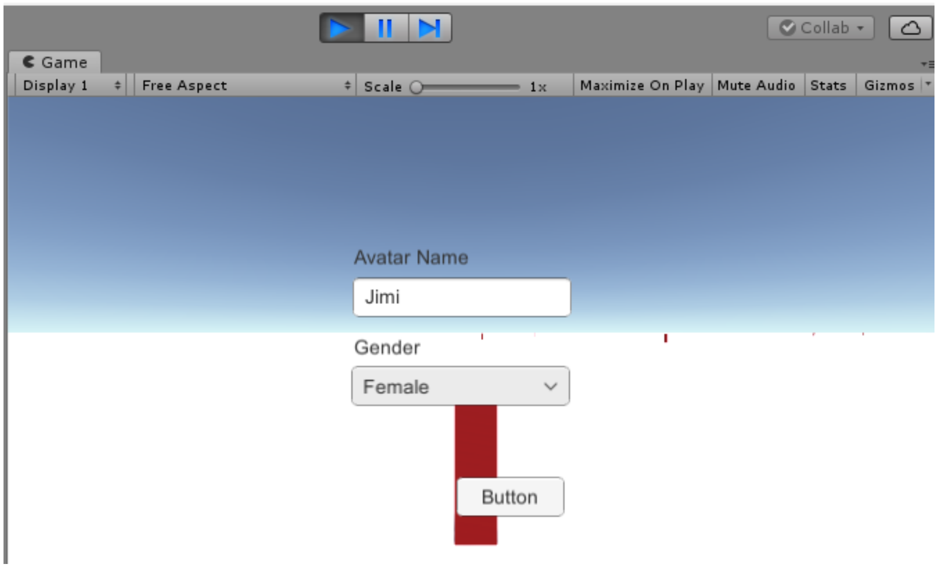
Precondition: At the start of the game

Expected Results: The game takes in player data and sends the data to the story writer

Test Results:

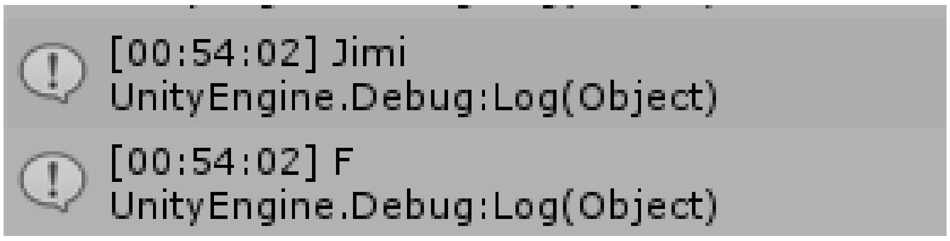
Table 5.5: Test results for Capturing Player Avatar Details

Valid Input	**Result**
The player types in anything as their avatar name	These values are saved into the game story writer (*Figure 5.8)*
Invalid Input	**Result**
The player doesn’t type anything as their avatar name	The avatar name is sent to the story writer as an empty string



Avatar Name: Jimi
Gender: Female

Figure 5.8: Inputting Avatar Names into the Game



```
[00:54:02] Jimi
UnityEngine.Debug:Log(Object)
[00:54:02] F
UnityEngine.Debug:Log(Object)
```

Figure 5.9: The Avatar Details that the Story Writer Receives

3.3.3 Test Case: Instantiation of NPCs within the RPG Game

Precondition: The game starts, the player has inputted their avatar details and the Game master receives the NPCs from the Story Writer

Expected Results: The Game Master instantiates/spawns NPCs around the game environment based on the locations that the story writer has designated to them

Test Results:

Table 5.6: Test results for the instantiation of NPC objects

Valid Input	**Result**
The player inputs their avatar details and presses the create button to start the game	The Game Master Instantiates NPCs (represented as red cuboid) as seen in *Figure 4.2*

3.3.4 Test Case: Testing Game Master's Free Roam Mode and State Switching

Precondition: The game starts, the player has inputted their avatar details and the Game master receives the actions of NPCs from the Story Writer

Expected Results: At game start, the game is at roam state i.e. the player is free to roam around the game world and perform any quest of their choosing. When the player goes to the location of an available quest the game master will switch to quest mode where the quest goals are set up and the player will perform the quest. Once a quest goal is reached the game switches back to roam state

Test Results:

Table 5.7: Test Results for Game Master's Free Roam Mode and State Switching

Valid Input	**Result**
At game starts after the player has inputted their avatar's details	The game is in roam mode were the player can start a quest of their choosing (*Figure 5.10).* At the start of this state, NPC actions are displayed *(Figure 5.2)*

The player goes to the location of an available quest	The game master switches to quest mode and initializes the quest. Available quests are hidden, and locations made dormant till quest is completed. (*Figure 4.4*)
The player reaches the active quests goal	The game master switches to roam state were the player can choose to start any of the available quests. The game master sends data to the story writer of which quest that the player has completed so as to update the players actions into the game story

3.3.5 Test Case: Quest Controller instantiating and controlling a quest

Precondition: Quest state is switch to by game master

Expected Results: The quest controller within the game master displays the quest details and aim, and sets an object at the quest goal location

Test Results:

Table 5.8: Test results for Quest Controller instantiating and controlling a quest

Valid Input	**Result**
Quest state is switch to by the Game master	The quest controller displays quest details on the screen and sets the quest goal object at the quest goal location. Quest goal object is a blue

	location pin that just indicates where the quest goal is

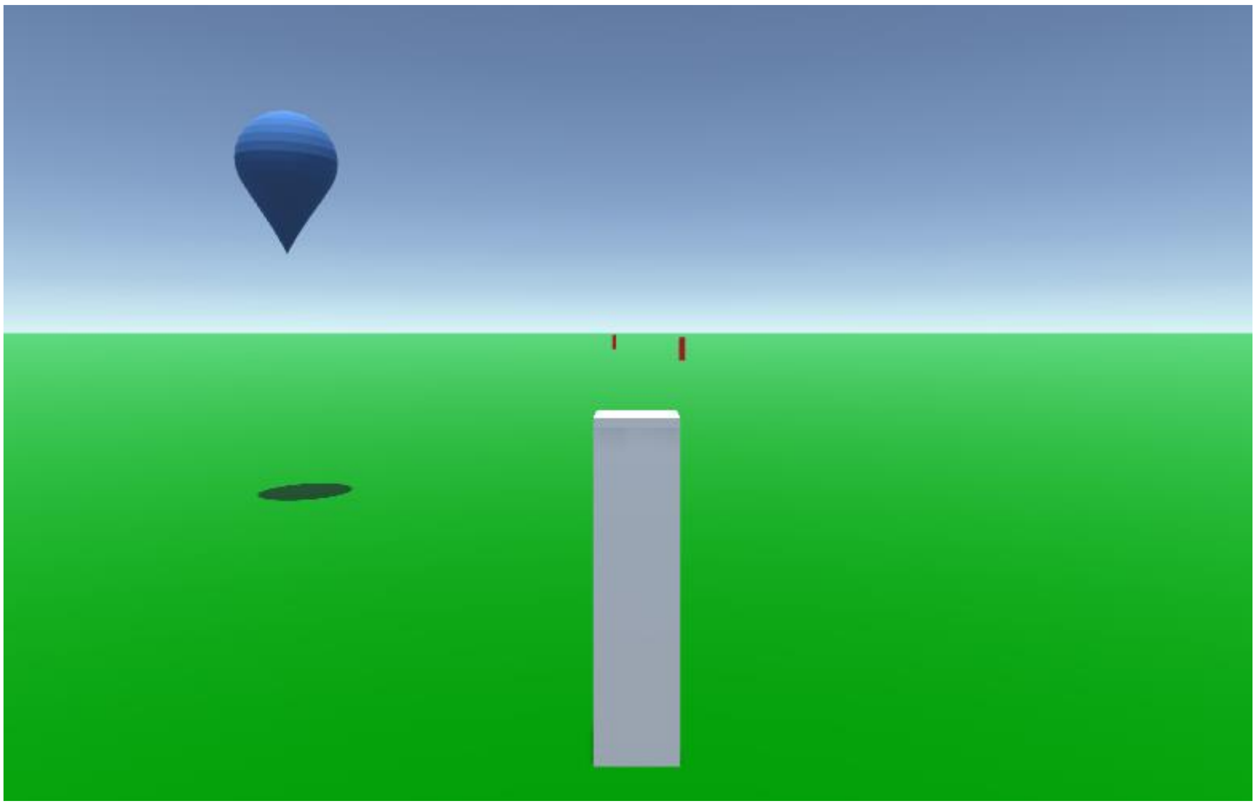


Figure 5.7: Player close to quest goal

3.4 System Testing

When all the components of Story Writer and the Role-playing game were put together to create a full dynamic story telling role-playing game prototype, all components performed and interacted as they were designed to. Situations did occur were when the player is killed early the game master just brushes through the rest of the NPC actions without player intervention. Not all expected outcomes were seen since the story writer behaves independent of external influence and so the certainty of those functionalities actually working were dependent on analyzing the algorithms syntax.

Chapter 6: CONCLUSION

3.1 Impact to the Game Industry

When the iPhone was released in 2007, it changed the telephone industry with the invention of touchscreen smartphones. This technology changes the way we used our phones and has caused changes in various industries [21]. Companies began producing touchscreen phones with bigger screen sizes, although some companies said that this kind of phones wouldn't sit well with the public, it was still able to flood the market. Other forms of technology like Machine learning and Big Data Tools have also changed how we live and “over the last two years alone 90 percent of the data in the world was generated” [20]. Companies are using various methods to gather data from their customers which they will feed into their machine learning models. This will help predict customer behavior and make their marketing and services more efficient. With every technological breakthrough, the industry associated with that technology begins to adapt, change and adopt this technology. This same behavior is most likely to happen to the game industry with the invention of AI that dynamically writes game stories. If customers adopt and are willing to buy this technology, it will cause other game studios to release games with the same technology. No one uses phones with keypad buttons anymore because companies were forced to change. In a future where this is the case, the game industry will shift to having more Role-playing games that have dynamic narratives. The only set of Role-playing games that will not be affected are those that have to follow some story accuracy of a historical figure in their time.

With the possible adoption of dynamic storytelling technology into games in the near future, games are more likely to be played for long because of the near limitless experience that gamers want to enjoy. Game studios will be less reluctant to release another generation of the game soon, because customers haven't reached the maximum satisfaction level of the game. Gamers have to be willing for the next game in order to buy it, but they won’t be willing until they are tired of the previous [22]. This will increase the average rate at which one studio will release games but in turn will increase game experience.

3.2 Future Work

The work done in this project serves as just the backbone for storytelling in games. In Role-playing games, dialogues and cutscenes are also components of story narrative. Modern CRPGs are visually interactive and should not be presented in text form. Relationships and actions between characters are presented in both dialogues and acting, but these components are not covered within the scope of the work.

For characters to have dialogues between themselves, simple actions in the story have to be creatively expanded into dialogue that showcase the use of past information they have about each character, the environment and the action that is about to take place. This process will have to involve another AI component that will be dedicated to just handling dialogues. Performing this could be another project of its own.

Dialogues will take place within cutscenes, were characters will act out the scenes. Without the presence of a pre-made animated scene, another AI will be needed to direct the actions that each character will perform within a cutscene. This would involve the AI knowing what is in the objects in the environment, the relationship between characters and the purpose and outcome of the scene. With this information, the AI will have to animate each character’s model, and to do this it will have to know how to animate each rig realistically. It could render the animations before the cutscene starts or at runtime.

Rendering at runtime would have a huge strain on processing power of the game console but rendering before the cutscene will increase how long loading the cutscene will take. Some other issues like controlling the camera position and angle will have solved for an AI to direct cutscenes.

With both dialogues and cutscenes, a game with a fully implemented dynamic story will still require speech. The dialogues will have to be converted to speech that the characters will during a cutscene produce. This could be done by using third party application or plugins that convert text to speech. That way the system will just have to feed in text to return the speech equivalent. In order to properly do this some issues will need to be resolved. In order to make the speech realistic, there must be natural fillers and pauses that makes it seem human. Google has been able to achieve this with Google Assistant's Call Screen [17]. The second problem comes from the fact that all characters have different voices. Whatever does the text to speech conversion should also be able to create unique voices for each of the characters.

By putting all these systems together, a fully automated story writer for Role-playing games will be achievable and meet up to the quality of games written, directed and designed by humans.

3.3 Limitations

The following functionalities in the system were not implemented as expected of the requirements.

Decision System: This system was expected to give the player the ability to make choices that have consequences within the game story. The system developed only gave players the choice to select which quest that they want to perform. This limits the amount of choices the player can make. An improvement upon this is having choice options within quests that determine the outcome of the quest that they perform. The quest outcome will then have consequences on the game story.

Passing of Available Quest data to the RPG: The story writer was tested on quest generation and succeeded. When the writer was incorporated into the RPG, the generated quests are expected to be pulled by the Game master and display them in the game environment during roam mode. This didn’t happen for unknown reasons even after multiple tests to determine where the faults were. The Game rather kept on generating just one quest, even when the number of NPCs in the story increased. Some advanced testing may be required to fix this and in a worse case, the whole game master will have to be redone.

3.4 Recommendations

This section discusses some suggested areas to improve the AI that generates the game story.

- Implementing a Character Localization and Timing system: In a 3D environment like games, objects have 3D locations and so will characters. The locations of characters should affect their ability of taking some actions because they will have to travel a distance which can take a long time. Implementing the writer in such a way that factors location will also involve implementing time. How long characters will take doing an action and how much time it will take for a character to perform an action.

- Permanent Environmental Change: When characters perform the “destroy” action on locations, there should be a permanent environmental change where other characters can’t perform actions in those locations because the locations have been destroyed. In addition, when characters conquer territories, there should be repercussions for characters they have rivalry with
- Incorporating Flaws and Needs: Character flaws and needs are essential to making a story have more relatable characters. Having to implement this and causing it to play a role in the actions they take place will improve the system.
- Include Personality Models/Types to define how a character will act: The Enneagram institute provide the different categories of personalities and the specific behaviors of these personalities can exhibit. Incorporating personality behaviors in the dynamics of the characters will improve the storytelling of the game writer.
- Information Localization: Realistically humans aren’t omniscient, they learn about new information by acquiring it from people who were directly involved with how the information was formed. This should be taken into consideration, characters should learn about the actions of other characters by acquiring the information from characters that have it. A system should be implemented such that, information should be replicated and passed to characters like an object. Characters should only be able to make informed decisions based on the information they have acquired even though some information about a decisive action hasn’t gotten to them.

3.4 Conclusion

This project sought to develop a solution to increase the gameplay experience in Role-playing games by having the game story personalized to each player. The proposed solution to this problem was by using AI to write the game story so that way each player gets their own written story. The approach to how the AI will work was adopted from Behavioral Robotics, where each character will serve as an AI agent with their own set of unique behaviors that are realistic or like the way humans will behave in the same situations. When these characters interact in a series of actions, the story is created. The by which characters behave were gotten from John Truby’s outline for how to make good stories [4]. This approach was adapted into a Role-playing game and was tested resulting in desirable outcomes. This kind of static-less game story technology has the potential to give games the ability for story flexibility. Although the basic underlining features of this approach were only implemented within this project, further improvements and story realistic approaches have the potential to improve the capabilities of dynamic story writers to be as good as human writers.

References

[1] Michael French. 2013. Inside Rockstar North - Part 1: The Vision. *MCV*. Retrieved October 10, 2018 from https://www.mcvuk.com/development/inside-rockstar-north-part-1-the-vision

[2] Samantha Callender. 2018. Spider-Man PS4 Game Development Is Complete. *ScreenRant*. Retrieved October 10, 2018 from https://screenrant.com/spider-man-ps4-game-development-gone-gold/

[3] Juha-Matti Vanhatupa. 2011. Guidelines for Personalizing the Player Experience in Computer Role-playing Games. In *Proceedings of the 6th International Conference on Foundations of Digital Games* (FDG ’11), 46–52. DOI:https://doi.org/10.1145/2159365.2159372

[4] John Truby. 2008. *The anatomy of story: 22 steps to becoming a master storyteller*. Faber and Faber, New-York (N.Y.)

[5] PlayStation. *Detroit: Become Human*. Retrieved March 18, 2019 from https://www.playstation.com/en-us/games/detroit-become-human-ps4/

[6] Shadow of War. *Middle-Earth: Shadow of War*. Retrieved March 18, 2019 from https://www.shadowofwar.com/

[7] No Man’s Sky. *No Man’s Sky*. Retrieved March 19, 2019 from https://www.nomanssky.com/

[8] Lopes Ricardo and Bidarra Rafael. 2011. Adaptivity Challenges in Games and Simulations: A Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3, 2 (2011), 85–99.

[9] Alejandro Ramirez and Vadim Bulitko. 2012. Telling Interactive Player-specific Stories and Planning for It: ASD + PaSSAGE = PAST. In *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (AIIDE'12), 173–178. Retrieved December 14, 2018 from http://dl.acm.org/citation.cfm?id=3014629.3014660

[10] Mark O. Riedl, Andrew Stern, Don M Dini, and Jason M Alderman. 2008. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications* 4, 2 (2008), 23–42.

[11] David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. 2007. Interactive storytelling: a player modelling approach. In *Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (AIIDE'07), Jonathan Schaeffer and Michael Mateas (Eds.). AAAI Press 43-48.

[12] C. R. Fairclough. 2004. *Story games and the OPIATE system*. Ph.D. Dissertation. Trinity College, Dublin, Ireland.

[13] Colin Campbell. 2019. How to write a video game story. *Polygon*. Retrieved March 19, 2019 from https://www.polygon.com/features/2019/1/10/18165611/how-to-write-a-video-game-story-narrative-building-tips

[14] Lankoski Petri and Björk Staffan. 2007. Gameplay Design Patterns for Believable Non-Player Characters. *DiGRA '07 - Proceedings of the 2007 DiGRA International Conference: Situated Play* 4, (2007). Retrieved from http://www.digra.org/wp-content/uploads/digital-library/07315.46085.pdf

[15] Richard Zhao. 2009. *Applying Agent Modeling to Behaviour Patterns of Characters in Story-Based Games*. Master's Thesis. University of Alberta, Edmonton, Alberta. Retrieved March 22, 2019 from https://www.researchgate.net/publication/255993939_Applying_Agent_Modeling_to_Behaviour_Patterns_of_Characters_in_Story-Based_Games

[16] Magy Seif El-Nasr. 2007. Interaction, narrative, and drama: Creating an adaptive interactive narrative using performance arts theories. *Interaction Studies* 8, 2 (2007), 209–240. DOI:https://doi.org/10.1075/is.8.2.03eln

[17] Screen your calls before answering them - Phone app Help. Retrieved March 25, 2019 from https://support.google.com/phoneapp/answer/9118387?hl=en

[18] Maja J Matarić. 2008. *The robotics primer*. The MIT Press, Cambridge, Mass.

[19] Massimo Pigliucci. 2010. Rationally Speaking: What set of moral criteria? *Rationally Speaking*. Retrieved November 17, 2018 from http://rationallyspeaking.blogspot.com/2010/10/what-set-of-moral-criteria.html

[20] Bernard Marr. 2018. How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. *Forbes*. Retrieved March 30, 2019 from https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/

[21] Tim Bajarin. 2017. How Apple's iPhone Changed These 5 Major Industries. *Time*. Retrieved March 30, 2019 from http://time.com/4832599/iphone-anniversary-industry-change/

[22] Knowledge@Wharton. 2008. How To Market Tired Products. *Forbes*. Retrieved March 30, 2019 from https://www.forbes.com/2008/05/02/hummer-miami-marketing-ent-sales-cx_kw_0501whartonmarketing.html#7a56a05232ed

[23] Sam Liberty. 2017. What Are Role-Playing Games Even? How are they that? *Medium*. Retrieved April 18, 2019 from https://medium.com/@SA_Liberty/what-are-role-playing-games-even-how-are-they-that-50071c5552e2

[24] Grand Theft Auto V. *Rockstar Games*. Retrieved April 18, 2019 from https://www.rockstargames.com/games/info/V