FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Prototype for RPG-based Game
Student:	Sergei Abmanzin
Supervisor:	Ing. David Bernhauer
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2022/2023

Instructions

The aim of the work is to create the basis for various RPG-based games.
1. Do research on at least 3 games with RPG elements.
2. Do research on at least 3 game engines. Focus on concepts:
 - 2.5D graphics,
 - game control (mouse and keyboard),
 - creating or generating map(s),
 - dialogs,
 - scripting,
 - availability and quality of documentation and tutorials.
3. Analyze and design a game that will allow the player to navigate the map and interact with NPC characters through dialogues.
4. Implement and test a prototype of this game and describe how the programmer can define the map, element placement, and dialogs.

Electronically approved by Ing. Michal Valenta, Ph.D. on 9 February 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Prototype for RPG-based Game

Sergei Abmanzin

Department of Software Engineering
Supervisor: Ing. David Bernhauer, Ph.D.

May 11, 2023

Acknowledgements

First of all, I want to express my gratitude to my supervisor Ing. David Bernhauer, who made a significant contribution to this work. I would also like to thank the university for teaching me the most important lesson ”Rely only on yourself”. As a result, this led to the creation of this thesis, and also to the development of my faith in myself, therefore I also express gratitude to the most important and main person in my life - Myself.
Additionally, I would like to express my gratitude to all the people who contributed to the creation of the work by selecting the right words in English.

Declaration

In Prague on May 11, 2023 . . . .

Citation of this thesis

Abmanzin, Sergei. *Prototype for RPG-based Game.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Abstrakt

Cílem bakalářské práce je hlavně implementace prototypu počítačové hry na hrdiny. Základní principy hry jsou popsány v dokumentu. Sekce obsahují také srovnání existujících řešení, jako je Zaklínač 3: Divoký hon, Divinity: Original Sin II a Fallout 4. Důležitým prvkem práce je porovnání softwaru pro vytváření her. Sekce obsahují srovnání herních enginů jako Unity, Unreal Engine a CryEngine. Práce se týká vytvoření systému dialogů. Výsledkem práce je funkční prototyp hry na hrdiny vytvořené pomocí Unreal Engine a pomocí programovacího jazyka C++. Výsledná hra je k dispozici jako spustitelný balíček pro platformu Windows.

Klíčová slova Unreal Engine, Unity, sections, Zaklínač 3: Divoký hon, Divinity: Original Sin II, Hra, rolová hra, Herní engine, C++

Abstract

The purpose of the bachelor's thesis is mainly to implement a prototype of a computer role-playing game. The basic principles of the game are described in the document. The sections also contain a comparison of existing solutions such as The Witcher 3: Wild Hunt, Divinity: Original Sin II and Fallout 4. An important element of the work is a comparison of softwares for creating games. The sections contain a comparison of such game engines as Unity, Unreal Engine and CryEngine. The work concerns the creation of a system of dialogues. The result of the work is a functional prototype of a role-playing game created on the Unreal Engine using the C++ programming language. The resulting game is available as an executable package for the Windows platform.

Keywords Unreal Engine, Unity, sections, The Witcher 3: Wild Hunt, Divinity: Original Sin II, Game, role-playing game, Game Engine, C++

Contents

List of Figures

Chapter 1

Introduction

Video games are becoming increasingly popular, they offer certain advantages and benefits, such as social interaction, mental stimulation, and even help with mental health restoration. In recent years, various social problems have occurred that have affected modern generation. First motivation in creating this thesis is to create a game that will help an ordinary person to relax and distract from external problems. Second motivation is to learn more about gaming technologies and their latest progress, thereby starting one's own journey in game development.

1.1 Goals

In the theoretical part of the thesis, a definition of the game will be given, role play games concepts will be defined and a review and analysis of existing games of this genre will be carried out. Also a definition of a game engine will be provided, along with the identification of important criteria for its selection. Based on these criteria, an analysis of three game engines will be conducted. As result of theoretical part a game engine will be chosen as the primary for development, and the fundamental role play game concepts will be described.

In the practical part, the game design document, as well as the design of dialogue and quest systems, will be created based on the described concepts and chosen game engine. Furthermore, a prototype will be developed using the selected game engine, building upon the created designs. Moreover, user testing will be conducted to gather feedback and result of development, which will help guide the direction of the prototype.

1.2 Structure of work

The second chapter, Analysis, focuses on existing solutions, defines the basic concepts and specifies the requirements for the game. Moreover, **the second chapter** analyzes the existing game engines and compares them. One will also be selected to use. **The third chapter, Design**, focuses on design all described concepts. **The fourth chapter, Implementation**, contains the implementation and suggestions for the expansion of the game. Furthermore, provides valuable insights for utilizing the designed systems effectively. Moreover, **the fourth chapter** contains user testing and its result analyze.

Chapter 2

Analysis

In this chapter, we will define the basic concepts and define the key concepts of computer role play games. Also in this chapter, research for existing solutions will be conducted. As a result of the research, the best features of the games will be determined. Also this chapter provide research of existing game engines. As a result of the research, a game engine will be chosen for implementation of the prototype.

2.1 Definition of basic concepts and problems.

According to site statistics the number of gamers in 2020 has grown by 39% worldwide [1]. Games are becoming chart-topping. They provide an opportunity to escape from routine, meet new acquaintances and relax. Corresponding to Bernard Suits, the game is an activity that has its own rules and restrictions, and only by accepting them, you can start this activity [2].

Games provide us a choice commitment. For instance, what kind of world or situation to be involved in or what rules to accept. One of the genres of games is role playing games (RPGs). Henriksen defines RPG as "a media where, through immersion in the role and the world of this role, the player (character) is given the opportunity to interact with this world."[3] However for Simkins "An RPG is a game, not a game system or product, but a game experience that a player plays, in which the player portrays a character in a setting. Each player's portrayal of their character must include three components: immersion, experiencing the character; acting, performing in character; and gaming, obeying and manipulating rules and goals in character."[4] Both definitions of RPG have the same parts that speak about the role (immersion of the player in the setting) and interaction through this role with the world. For the continuous discussion, Sebastian Deterding and Jose Zagal simplified the definition of RPG to the combination of four phenomena: "roles, play, games and media culture."[5]

Different criteria are emphasized in different definitions, some focuses on the ”game” action that is being performed, some orients on the character we are immersed in, whose ”role” we assume, and some concentrates on the surrounding world and interaction (”play”) with him. In addition, there are different subtypes of RPGs emphasizing different concepts, such as tabletop role play games (TRPGs), live action role play games (LARPs), computer role play games (CRPGs) and multiplayer online role play games (MORPGs). In this paper the author will focus on CRPGs and the following concepts:

- Storytelling

  As the National Storytelling Network notes, ”Storytelling involves a two-way interaction between a storyteller and one or more listeners. The responses of the listeners influence the telling of the story. In fact, storytelling emerges from the interaction and cooperative, coordinated efforts of teller and audience.”[6] Undoubtedly, a player immersed in a character and define himself in an unfamiliar world, at this moment the player needs to simultaneously understand all these aspects: who is he, where is he and most importantly why is he here. During this interaction, the player gets answers to his questions and moves into the depths of events. Different cultures and situations are important, ”Different cultures and situations create different expectations for the exact roles of storyteller and listener - who speaks how often and when, for example - and therefore create different forms of interaction.”[6]

  The crucial part is the player’s emotions throughout the storytelling. One of the emotions is dread. This is the exact idea conveyed by the author of the article ”Storytelling That Moves People” Bronwyn Fryer, ”One of the principles of good storytelling is the understanding that we all live in dread.Fear is when you don’t know what’s going to happen. Dread is when you know what’s going to happen and there’s nothing you can do to stop it.”[7] Storytelling has a humongous power over the player and immerses him more and more into a new world, forcing him to attach to the environment and stay as long as possible in-game.

- Non-player character (NPC)

  Immersed in his story, the player often gets acquainted with other characters who later accompany him. The player is immersed in their story too. He begins to understand them, and sometimes even empathize with them. The game provides such an opportunity with the help of cutscenes, notes inside the game or dialogues with these characters.

- Exploration

  Immersed in a new world, having received his role, the player is given the opportunity to explore. Sometimes the player is guided by the game itself, moving him strictly according to the plot. Sometimes the player has the opportunity to move independently, stopping in different parts of the map, watching NPCs and their situations. Thus, the player studies his environment, learning more about his essence.

- Character progression

  Every person requires development, the character is no exception. With every hour spent in an RPG, the game character gets a gaming experience, this is also called ”levelling”. Thus, the player develops himself, improves his characteristics, sometimes gets extra points, for which he can unlock special skills. Thereby increasing the excitement of the player, and luring him more and more.

2.2 Research of existing solutions

Each game focuses on different concepts, giving them more or less time. Someone creates a strong line, others unforgettable characters, and someone gives full control to the player. In addition, different games may have different graphics or controls. In this work, we will take the following games as examples: The Witcher 3: Wild Hunt, Divinity: Original Sin II and Fallout 4.

2.2.1 The Witcher 3: Wild Hunt

The Witcher is an CRPG game developed and released by CD Projects in 2015. In the same year, it received a huge number of awards, including ”Best RPG” and ”Game of the Year” according to The Game Awards 2015. In a huge list of awards, there is one specific award that this game received called ”Excellence in Storytelling” in 2016 [8]. And it is the narration, the filling of the environment and diversity that are one of the obvious advantages. In addition, Jakub Szamalek said ”There will be some very touching moments in The Witcher 3, and you will have the time, since it’s a very long game, to establish relationships with people that are close to you from the very beginning, and see the relationship grow or falter, and this might be a pretty intense experience as well.” And also the developers have done tremendous work on a new mimic system, increased the number of animations, made more complex dialogues, which later helped to convey the emotions of the characters stronger than in previous parts, as well as immerse the player deeper into the plot [9]. Moreover, there are 36 endings in the game, as noted by IGN [10]. Of course, not all of them are related to the main line of the story, variations of the destinies of non-key characters are added to this figure, but no less time is given to them. Having seen all the features that the Witcher possesses, you can say that he coped very well with storytelling. A long history of 200 hours, decisions on which the future depends, an open world with an area of about 142 square kilometers of the world, demonstrates how The Witcher qualitatively approached such concepts as Storytelling and Exploration.

As mentioned earlier, the Witcher has a considerable number of storylines, naturally this means a larger number of characters in which the player can further observe twists in fate. The Witcher has more than 50 NPCs with whom the player can interact, start a dialogue and get a quest, as indicated in the pipeline from the developers [11]. Sometimes the player meets already known characters once again, thereby plunging back into the story of the character and empathizing with him even more. As the developers themselves point out, their priority is non-linear storytelling, and dialogues are one of the ways to achieve this goal, because they paid due attention to their creation, and obviously the Witcher gave great priority to such RPG concept as filling the world with non-player characters.

With the time of completing quests, the character receives not only the continuation of the story and immersion in the world, but also experience, which later gives new levels to the character. In the Witcher, character development is almost entirely based on the unlock-by-level system. The level depends on what you can equip your character with, how many ability points you will have, how many abilities and mutagens will be active.

As the level increases and the character develops, the opponents also become stronger, which does not allow the character to relax and forces the player to develop further. For each level, the player receives an ability point, which he can spend on discovering or developing an ability (a set of passive boosts that can affect every single aspect of your gameplay). The player can get such points with the help of special places of power scattered on the map, which makes the game not completely unlockable. But it doesn't end there. A player can have no more than twelve active abilities, and only one cell is available to him at the start. With a certain level , new cells are unlocked to the player (at levels 2, 4, 6, 8, 10, 12, 15, 18, 22, 26 and 30). Mutagens are also present in the game (rare alchemical ingredients that can be worn as equipment for permanent passive reinforcement). Initially, one cell is open to the player for such an ingredient, but with the development of the character, three more are opened (at levels 3, 9, 16). Also, the items received by the player are related to the level. For example, if your character is of insufficient level, you will not be able to use such an item until you reach the desired development. The Witcher provides a huge opportunity for character development. Due to the limited number of active abilities and a large selection of skills, players create their own unique builds for their character.

The development of the game began in 2011. During all this time, a lot of work was invested on the part of the studio, which later gave them a high rating on metacritic from critics and users [12]. A good story, a strong immersion, a not-hard character development and many other things brought success to the studio, because already in 2015 their annual sales revenue increased 8 times [13].

2.2.2 Divinity: Original Sin II

Divinity: Original Sin II was developed and released in 2017 by Larian Studios on the Windows platform. After a while, versions were released for other platforms as well. Unlike "The Witcher 3: Wild Hunt", Divinity does not have a large number of awards, but its main difference and advantage is the ability to play in cooperative mode for up to 4 people. For that, the game received the "Best Multiplayer" award in 2018 from the BAFTA Awards [14].

As for the storytelling , the game provides the following features: Cut scenes with the narrator's voice and dialogues with non-player characters. Cut scenes are a set of pictures, with minimal animations. In addition, there are not so many character animations during dialogues, which does not always allow you to get into the emotional part.

On the other hand, the number of NPCs in the game is sufficient, more than 50. The player can interact with each of them and get various tasks. The player can perform any task in different ways, which makes the game less trivial. The main tasks in the game will require 59 hours, and additional quests will require another 153 hours for 100% completion [15].

The game is not linear, the player is free to move around the world and choose what tasks he will perform. The player explores the world with the help of various finds and a journal. Also, when meeting new characters, the player receives information from them, but unfortunately the choice of answers is not always meaningful, which is in my opinion a disadvantage.

The developers took the character development very seriously. In addition to various effects from the selected race and the selected class, the player is provided with the following features: improvement of attributes - basic character statistics (Strength, Finesse, Intelligence, Memory, Constitution, Wits [16]), improvement of abilities - the main characteristics of the character (Presented in two variations Combat and Civilian [17]), as well as talents - passive abilities of the person (46 different talents [18]). With each new level, the player gets one or another development point. And by level 20 already has 15 attribute points, 49 ability points and 7 talent points [19]. All this allows the player to experiment with assembling and building a unique character.

As a result, game received positive ratings from both critics (93 Metacritics [20]) and users (8.8 Metacritics [20]). And the cooperative mode is a rarity for the CRPG. But success is determined by profit, and they have no problems with that. Only on the Steam trading platform the game collected \$147,7m with a budget of \$4.5m [21][22].

2.2.3 Fallout 4

Fallout 4 was developed and released in 2015 by Bethesda Game Studios for Windows, PlayStation 4 and Xbox One. In 2023, Bethesda announced free next-gen update for the game. In 2015 year, Fallout 4 was honored with several awards such as ”Best Role Playing Game”, ”Best PC Game” and ”Best Console Game” by the Game Critics Awards [23]. As per GameInformer, One of the interesting things of the games development was its inspiration from another game, ”The Destiny”, specifically for its locked frame rate across consoles (the same will be done for Fallout 4) and a good feeling of a gunplay. The source also mentions that an essential amount of time from the start to the end of Fallout 4 development was dedicated to enhancement shooting mechanics [24].

The narrative of Fallout 4 is built around the Boston area in the year 2287, with the main character being the sole survivor from Vault 111 [25]. This marks the beginning of the users path in Fallout 4, which culminates after around 81 hours of gameplay including side quests, offering four different endings based on the players choices [26].

The game features an extensive number of NPCs, with over 50 different characters with unique dialogues [27]. These dialogues offer the player the opportunity to influence the vector of the story, making the game non-trivial and dependent on player decisions. In addition to the non-linearity storyline, the player is not limited in world exploration, with the game world size of 9743 square miles [28].

Regarding the character levelling, a simple and intuitive system was chosen, consisting of 7 stats [29]:

1. **Strength** - It affects how much you can carry and the damage of all melee attacks.
2. **Perception** - it directly affects weapon accuracy in V.A.T.S. (Vault-Tec Assisted Targeting System) and successful attempts at stealing items.
3. **Endurance** - It affects your total Health and the Action Point drain from sprinting.
4. **Charisma** - It affects your success to persuade in dialogue.
5. **Intelligence** - It affects the number of Experience Points earned.
6. **Agility** - It affects the number of Action Points in V.A.T.S..
7. **Luck** - it affects the recharge rate of Critical Hits.

Interestingly, each of the seven attributes in Fallout 4 aligns to form the acronym S.P.E.C.I.A.L.. Each stat develops with the help of attribute points.

The game starts the player with 21 attribute points (special points for increasing one level of stat), with an additional point gained with each level. While the game itself does not impose a maximum level, there's a hard cap on the maximum value of 65535 (MAX_UNSIGNED_SHORT) [30]. This scheme provides a clear framework for player progression, while still allowing for variations in the character development. Undoubtedly, Fallaout 4 has achieved great success, garnering numerous awards,and holds a Metacritic rating of 87. In addition to the main game, Bethesda also released a companion mobile game, "Fallout Shelter", which won the "Best Handheld/Mobile Game" award [23]. With its large-scale and detailed story, a variety NPCs with different story direction, impressive graphics, and intuitive character levelling, Bethesda has delivered a high-quality product in the RPG genre.

2.2.4 Conclusion of RPG-based Game Research

After the analysis of various existing solutions, the following key concepts can be noted:

1. Elaborated dialogues, with the influence of the player's choice on the further development of the situation.
2. A kind of quests, which are one of the main things that take time in the game.
3. A story that you want to watch, which depends on the decisions of the player and which is not linear.
4. An open world with the possibility of free exploration for the player.

There are also less key concepts. For example, character development. Although it is important, but as The Witcher 3: Wild Hunt shows, it is not the key, but rather an additional thing that will brighten up and diversify your pastime in the game.

All these concepts will be used later in this work to create the best and more enjoyable prototype for the player.

2.3 Game Engine Definition

Jason Gregory states game engine is a platform for making many games that can be expanded, but also can be used without major modifications [31]. It is impossible not to agree, because people need new games and entertainment, but they need a tool to create them, which is what a game engine is. It should allow different developers to implement their ideas, a place where you can create any world and any role. The at id Software believed it would be engaging for the player to take on the role of a space marine battling demons, and their idea was well-received by players. This game has played a monumental role in game development. At the very least, as The Gamer notes, Doom is specifically responsible for influencing the first person shooter (FPS) genre for many years to come [32]. Furthermore, as Jason Gregory notes, the Doom ”was architected with a reasonably well-defined separation between its core software components (such as the three-dimensional graphics rendering system, the collision detection system or the audio system) and the art assets, game worlds and rules of play that comprised the player’s gaming experience.” [31], which underscores the significance of this game in the gaming industry.

These separated components are part of one large platform, which essentially is an architectural pattern known as the entity-component-system [33], and in addition to the above components (Render component, Physical Component, Sound Component), there are others: Artificial Intelligence Component, Network Component, Scripting Component.

Component rendering
Render component is one of the key parts that is responsible for 3D or 2D images, a good game engine should have a multi-format rendering mechanism. It is also important to work internally with the picture, the material, internal editors, the availability and capabilities of a video editor to create cutscenes. And the elements of the post process (“...the various processes available to give your part a final touch, enhancing visual appeal and/or mechanical properties.” - by The University of Texas [34]) are also important. All this is responsible for the quality of the image and its visual saturation.

Physical component
The physical component is a component related to the design of real actions, movements and reactions in the game world. Speed, Gravity, Projectile Movement, Fluidity - all this and not only gives you the ability to adjust the physical component. This part of the game engine is most often used for games with realistic actions, for a greater sense of a new reality.

Sound component
A sound component is a component that controls sound effects that must be generated and reproduced as a result of in-game interactions. Just like a picture, sounds should be processed in different formats.

Network component
The network component is a component that supports multiplayer games, connecting you with other players to enjoy the game together. This can be either locally or via an Internet connection.

Scripting component
As Jason Gregory highlighted, this platform should be easily extensible if

the developer requires it [31]. This component allows the game developer to configure the game engine at the time of development, and define in-game behavior for moving objects on scene, different events and else.

2.4 Research of existing game engines

We can notice that the quality of game engine components and their availability in general will depend on the capabilities of the developer, how complex will the developer create each part of the game and how qualitatively. All these components allow making specific part of a game more detailed, some genres of games require something to a greater extent, and something to a lesser extent. Combining a more detailed description of the cRPG genre, in this thesis I will focus on such components as Scripting and Rendering, and as examples such engines as: Unreal Engine, Unity and CryEngine will be used. Moreover, an important element is the availability of documentation or training videos.

2.4.1 Unreal Engine

Unreal engine is a game engine developed by Epic Games, Sweeney thought about creating an FPS game called Unreal, and in 1995 began developing a game engine to create this game. And already in 1998 it was released and with it appeared the first version of Unreal Engine. Over time, a large number of game developers started to use that engine, and Unreal Engine continued to develop and received new versions.

Rendering

In 2019, two TV series “The Mandalorian” by Disney and “Love, Death & Robots” by Netflix were released. In both cases, the Unreal Engine was used to create visual effects. Those are not the only cases of course, but some of the most popular and well-known. ILM creative director Rob Brydon said, ”Everything in this volume is designed both to illuminate the actors and to be a backdrop that we can photograph directly. ... So you get the final pixels in the camera in real time.”.[35] It is for a more detailed realistic picture that Unreal Engine is used, the list below will indicate the capabilities of Unreal Engine 4 under different criteria.[36]

Post Process possibilities:

1. Various effects as a part of the engine (Anti-Aliasing, Bloom, Color Grading, Depth of Field, Eye Adaptation, Lens Flare, Scene Fringe, Vignette, Screen Space Reflections)
2. Support for post process materials

Lighting possibilities:

1. Directional, Point, Spot lights
2. SkyLight with accurate representation of daylight and atmospheric scattering
3. IES light profiles for realistic lighting simulations

Shadows possibilities:

1. Real-time hard and soft shadows
2. Cascade Shadows for accurate shadowing of large outdoor environments
3. Distance Field Shadows for accurate shadowing of dynamic objects and terrain

Materials possibilities:

1. Physically based materials for realistic surface appearances and behavior
2. Blueprint Material Editor for creating complex materials and effects with a node-based visual interface
3. Tesselation for adding detail to surfaces

Scripting

As noted earlier, the ability to extend the engine and program the behavior of objects by creating a logical chain for them is one of the key ones and Unreal Engine covers both of these possibilities. The engine is open source and any person or team can expand it for their own purposes. The main language is C++, which is compiled directly into machine code, and the resource-consuming parts of the application can be optimized for maximum performance. It also allows manual memory management. Although all of the above is a benefit, there are disadvantages. As noted by the source circuitstream, ”C++ typically requires a manual approach to memory management with a higher chance of error.”, which will certainly prove to be a problem for many newcomers [37].

Documentation

Unreal engine provides complete documentation for its source code, since it is an open source. All documentation is available on the official website and is displayed in the following format: A short description of the input values and a description of the result. Of course, this is enough to understand the basics, but often this information is not enough for more complex functions. In addition to the basic documentation, there are a large number of training videos on YouTube, as well as a developer forum. All this allows you to better understand how the engine works inside, but often a new user has to figure everything out for himself.

2.4.2 Unity

Unity was released by Unity Technologies in 2005 . According to site statistics itch.io (A service created by Lapham Corcoran for hosting and selling indie games since 2013) this engine is one of the most popular in creating indie games among novice game developers.[38] All this is thanks to a free subscription under a certain condition - ”with less than $100 Of revenue or funds raised in the last 12 months”. Currently has version 2022.1.23.

Rendering

Perhaps Unreal Engine is finding its use in multimedia due to its advancements in rendering, but Unity is not far behind. With each version, Unity continued to improve their work with graphics, giving more and more opportunities to developers, while staying relatively easy to use, which is the main advantage of this engine. At the moment, the rendering part of the engine supports the following functions:[39]

Post Process possibilities:

1. Built-in post-processing stack with various effects (Bloom, Depth of Field, Motion Blur, Ambient Occlusion, Screen Space Reflections, Color Grading, and more)
2. Customizable post-processing via post-processing profile and volume system
3. HDR rendering support

Lighting possibilities:

1. Real-time global illumination with Enlighten technology
2. Support for directional, point, spot, and area lights
3. Lightmapping for baked lighting
4. Real-time light probes for dynamic objects
5. Support for physically based lighting with Unity’s Standard Shader

Shadows possibilities:

1. Real-time hard and soft shadows with shadow mapping
2. Support for shadow masks and shadow cascades
3. Real-time shadowing of point and spot lights
4. Lightmap-based shadowing for baked lighting

Materials possibilities:

1. Physically based materials with Unity’s Standard Shader
2. Customizable materials with Shader Graph and Visual Effect Graph
3. Support for procedural textures and noise generation

4. Real-time reflection and refraction with reflection probes

Scripting

Unfortunately, Unity provides access to certain layers of its source code as reference for educational purposes. However, with Enterprise subscription, you can gain access to additional layers, as well as full access (read/write) to the source code, which allows you to create builds of Unity for use in production.[40] Unity uses C# as its main language, which gives the following advantages: its syntax is less error-prone, since it is a high-level programming language, automatic garbage collector processing and a large number of tutorials and lessons.[37]

Documentation

Different sources can be used to learn Unity. Extensive documentation, tutorials and API references only open up an extensive list. Special attention should be paid to the community forum, where developers can ask questions, share knowledge and collaborate with other Unity users. According to a survey conducted by StackOverflow in 2021, developers highly value these community resources. More than 60% of developers rely on official documentation and community forums for troubleshooting.[41] Which shows the simplicity and usefulness of documentation in Unity.

2.4.3 CryEngine

CryEngine was developed by Crytek in 2002. It empowers game developers with advanced capabilities such as global illumination, and physics simulations. It has a community of over 1.5 million users who provide extensive support. Games such as ”Crysis” and ”Hunt: Showdown” were created based on this engine.[42] Moreover, with its new features such as real-time ray tracing and new terrain system,[43][44] CryEngine continues to push the boundaries of game development.

Rendering

CryEngine is known for its stunning graphics and visual effects, which have been used in various high-end games. For example, Francis Courchinoux from ICE games confirms the statement about the excellent and realistic graphics in CryEngine.[45] And this is true, because developers continue to improve and add new technologies to their engine such as technology ”Vulkan”.[46] The engine offers advanced rendering features:[47]

Post Process possibilities:

1. Real-time Global Illumination for realistic lighting and shadows

2. Advanced Post-Processing effects such as Depth of Field, Motion Blur, HDR, Volumetric Fog, and Color Grading
3. Particle Systems for creating realistic visual effects such as smoke, fire, and explosions

Lighting possibilities:

1. Advanced Dynamic Lighting and Shadows with Real-time Raytracing
2. Support for High Dynamic Range Lighting (HDR) and Screen Space Reflections
3. Advanced Light Propagation Volumes for accurate lighting and shadows in large environments

Shadows possibilities:

1. Real-Time Ray-Traced Shadows for realistic and accurate shadows
2. Advanced Cascade Shadow Maps for accurate and efficient shadowing of large outdoor environments
3. Support for Distance Field Shadows for dynamic objects and terrain

Materials possibilities:

1. Physically-based Rendering for realistic surface appearance and behavior
2. Support for Material Layers and Material Functions for easy creation and customization of materials
3. Advanced Tessellation for adding detailed geometry to surfaces

Scripting

CryEngine provides a visual scripting system called Flow Graph, which allows developers to create complex behaviors and game mechanics without the need for extensive C++ coding. The engine supports two main programming languages: Lua and C++. Lua is a high-level, easy-to-learn scripting language that is commonly used for game scripting, while C++ provides more control and flexibility over the engine's functionality. Crytek developers continue to improve and develop the scripting component. After recent updates, they added a new visual language Schematyc [48], which provides ultimate control over objects within levels.The engine also has a visual editor for working with the landscape, simplifying the work of designers.

Documentation

CryEngine provides extensive documentation, including user manuals, programming guides, and API references. The engine's official website also offers a community forum where developers can ask questions, share knowledge, and collaborate with other CryEngine users. Unfortunately, due to the small community, sometimes this can become a problem. Crytek solves this problem by providing various tutorials and video resources to attract and simplify the integration of new users.[49]

2.4.4 Conclusion of Game Engine Research

Each engine has its own strengths and weaknesses. Unity is great for small projects, but its limitations become apparent in larger projects. CryEngine embodies visually stunning environments well, but is difficult to learn. Unreal Engine offers quite good documentation, provides user-friendly interface and also delivers impressive graphics right out of the box. Furthermore, the engine's toolset and continuous updates make it a convincing choice for developers.
Based on its features and flexibility, I have chosen Unreal Engine as the engine used for realizing RPG prototype.

Chapter 3

Design

In this chapter I will describe important aspects of the game design. I will create a game design document where all the game requirements will be revealed. Game managers and components will also be designed to help solve the identified requirements. The communication of these components will also be described in this chapter. All of this will be done based on the main concepts of RPG-based game from chapter 2.

3.1 Design Of Game

As stated by Julian Gold in his book, the design of a prototype game can be divided into four phases: ”brainstorming” (Description of each concept in different associated words, potentially future mechanics), ”prune the tree” (Cutting off some irrelevant ideas), ”draw the bubbles and lines” and ”validate the design”.[50] Confirmation of his words can be found in Ibrahim Ahmad and Nazreen Abdullasim book, in which they divided the design of a prototype game into three types: Information Design, Interaction Design and Interface Design.[51] And if Wesley assumes that design is a single process, then Ibrahim Ahmad and Nazreen Abdullasim divide design into three varieties. Despite this, in both cases, the result is a document with a complete representation of the game. This allows you to determine the scope of the game, define the main direction of the project and give a unified vision to the entire team.[52] This document is a game design document and could contain the following criteria:

1. Executive summary (game concept, genre, target audience, project scope, etc.)

2. Gameplay (objectives, game progressions, in-game GUI, etc.)

3. Mechanics (rules, combat, physics, etc.)

4. Game elements (worldbuilding, story, characters, locations, level design, etc.)

5. Assets (music, sound effects, 2D/3D models, etc.)

3.2 Game Design Document for Prototype

The purpose of this work is to create a prototype of an rpg game, ,but before I would like to give a definition of the prototype. Will Wright said, "A prototype is a navigation instrument. It's a compass."[53] For MasterClass "Good prototypes should have two main characteristics: they should be fast and cheap to build, and they should help game designers answer questions about their gameplay."[53] Therefore, we will leave some concepts aside and create a small piece of gameplay. He will give us the right direction of our game, which can be expanded in the future.

Target System
The user can run this application using an executable package on his personal computer on the Windows platform.

Target Audience
The target audience is any English-speaking user. who is interested in RPGs, games or is looking for something new.

3.2.1 Specification

The prototype will immerse the player in a new world with not very sociable and friendly characters. In order to achieve something from those characters, a contact should be established.

Setting

The game will be set in a 1870's in American frontier.

Game Structure

The player will be given one settlement in the setting style. Various non-player characters are scattered throughout the game, which are interactive and provide dialogue. The player chooses the answer to the dialogue. The result of the conversion depends on his answer. Some of the NPCs give side quests for the player.

Players

The game is played by a single player on a PC.

Graphics

The graphics in the game are close to realistic. Most of the models and materials will be taken from the default store of the selected engine and from MegaScan.

Objective

The main goal of the game is to investigate and find the sheriff's killer. Explore the settlement, interact with non-player characters, make the right decisions and eventually come to your main goal.

Failing or Winning conditions

The player cannot lose or win, he has a main goal that he must fulfill. What happens in the end depends on the player's decisions.

Story

The player will immerse himself in a half-wandering seeker who lost his family after a raid by bandits. After that ill-fated day, our character Richard went on a journey in search of that gang. Richard has a good sense of justice and is a good shooter, which allows him to enter into contracts to catch villains. Arriving at a new settlement, the character discovers the fresh corpse of the sheriff, who does not leave him indifferent. Richard begins his investigation in search of the killer.
The player interacts with other characters to find out who is behind the murder. Residents of the city do not particularly trust strangers, which complicates the task for the player. Sometimes he has to perform various tasks, like bringing something. The player in some dialogues has a choice on which the continuation depends. For example, instead of completing a task, he can cheat, which will lead to unknown consequences.
After finding out all the information, when the player is ready to make an accusation. The player goes to the potential killer and conducts a final dialogue, after which the ending comes.

Quests

During gameplay, the player is guided by quests. Initially, the player gets the only main task that he has to complete. Over time, he can receive side quests from non-player characters in this world

Main quest

The player automatically receives the main quest at the start of the game. Find the sheriff's body. After examining the body, the player gets a new main task - to find the killer. The player himself conducts the investigation and makes the decision himself. When the player is ready, he must go to the necessary character and charge him.

Side quests

The player receives side quests from certain NPCs. Each character gives their own specific quest, which the player must complete in order to get something on top. This is not mandatory to complete the game. Below is a list of all side quests.

- **Papers** - The player receives this task from Antonio. The player's task is to go to the postman and pick up the necessary documents.

Characters

John

An old friend of Richard's with whom they last met 10 years ago. He meets him on arrival in the city at the corpse. He will tell the player what he is doing in this city and offer help in the investigation.

Joe

Local seller and supplier of goods. Entrepreneur/Businessman. The player can meet him near his warehouse or in his favorite bar near the warehouse. The player will receive an order from him. Bring some papers from Jack, the bank worker. Joe will speak after completing this task or after rough line.

Jack

Jack is a local banker. The player can meet him at the bank or at the opposite bar. He always dreamt of serving the law, becoming the local sheriff. However, his fate was different. He respects the law and rules. Without any tasks, he will give the necessary information to the player. He will also hand over the necessary papers for Joe.

Carl

An elderly bandit who spends a quiet old age on the outskirts of the city. At one time he killed more than 10 people, and was very cruel. When the player finds him, Carl will be very rude and unfriendly. After a certain dialog, it will give the necessary information.

3.2.2 In-game controls

- **W** - Moves the character forward.
- **A** - Moves the character left.
- **S** - Moves the character back.
- **D** - Moves the character right.
- **Q** - Show quest display.
- **E** - Interactions with objects and environments.
- **Spacebar** - Jump.
- **LeftMouseButton** - Interactions with buttons in the user interface widgets.

3.2.3 User interface

The purpose of this work is the inner part of the game, mechanics, storytelling and dialogues. The goal is not the visual part, and therefore the wireframe as in figure 3.1 will be used as the user interface.

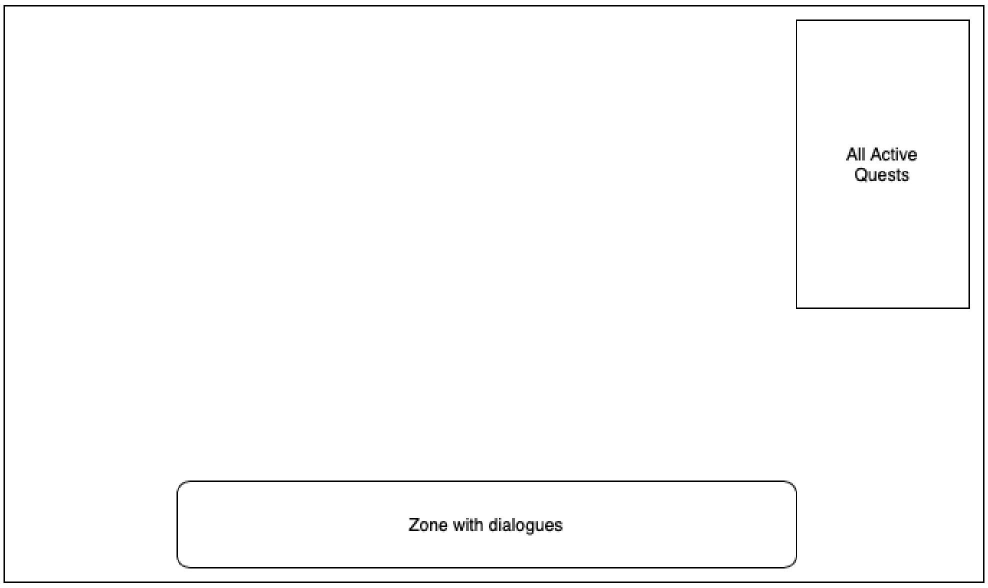


Figure 3.1: UI Layout - wireframe

3.2.4 Quest display

The quest display shows the player’s active quests. Information about quests is updated when the quest status changes. The wireframe as 3.2 will be used as the quest display layout.

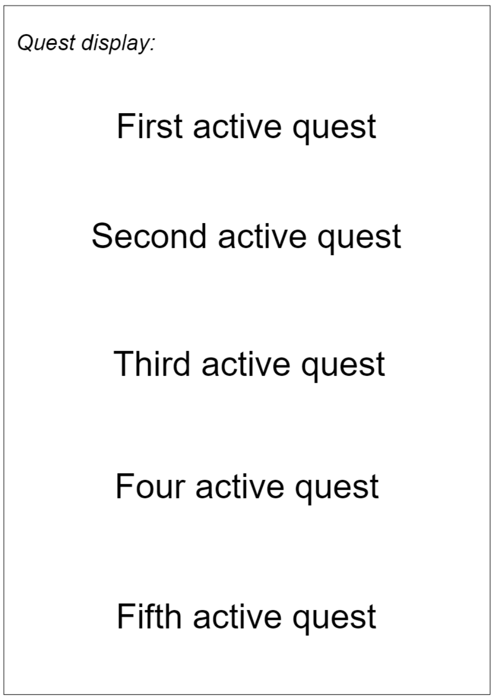


Figure 3.2: Quest display - wireframe

3.2.5 Dialogue Display

The dialog display shows the dialogues of a non-player character with whom the player is having a discussion. The wireframe as 3.3 will be used as the dialogue display layout. Here is up to three interactive buttons, which are used as confirmation of player answer choose or for skipping phrase. And also the main text box with phrases from NPC or Player.

3.3 Conclusion of game design docuemnt

As I can note, this context will give the user the necessary information about the game:

1. The main character and his story.

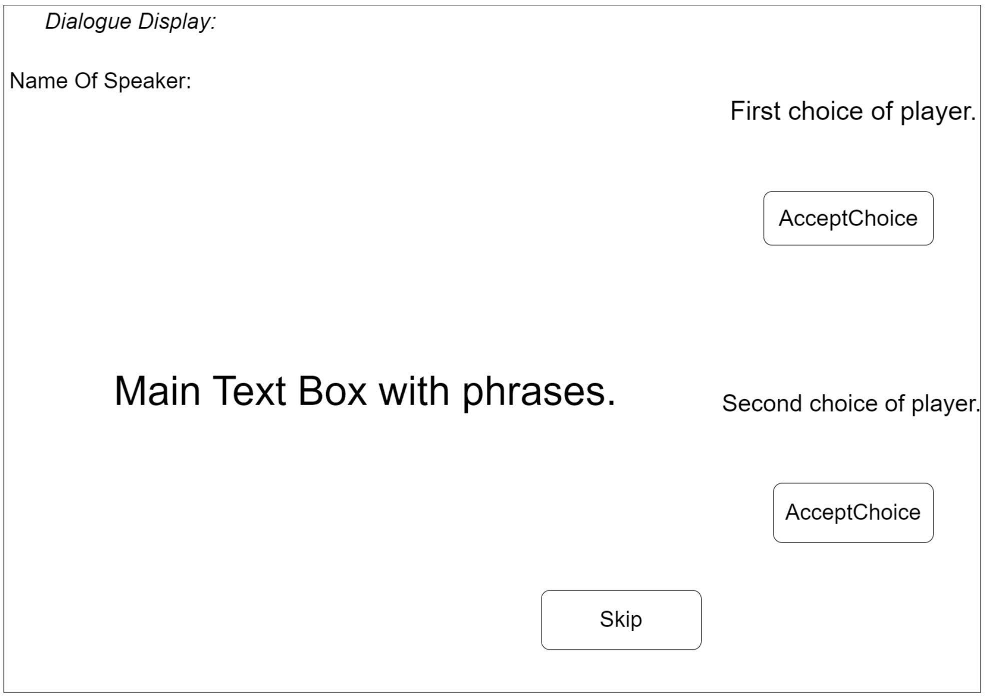


Figure 3.3: Dialogue display - wireframe

2. Non-player characters from whom he will receive information.
3. The killer who needs to be caught.
4. The environment and the filling of the world around which he can explore.

Thanks to this, we have received certain requirements for our system. Which allows us to compare the game design document as a use-case. Since it determines the context and requirements for the system, in our case, the game [54].

3.4 General game design

In this thesis, there will be no delving into the exact processes of work and communication of all components of the game engine. An understanding of the abstract class of game objects is more important. The update thread and draw thread interact with it as needed(One or another thread calls abstract methods for rendering, updating, or other action). Therefore, each new game part should extend from this class. And the game itself consists of all the necessary game objects.[55]
Based on the game design document, the following game objects with their

requirements are derived(Figure 3.4).
Figure 3.5 shows the top view of classes diagram. As you can see, my game

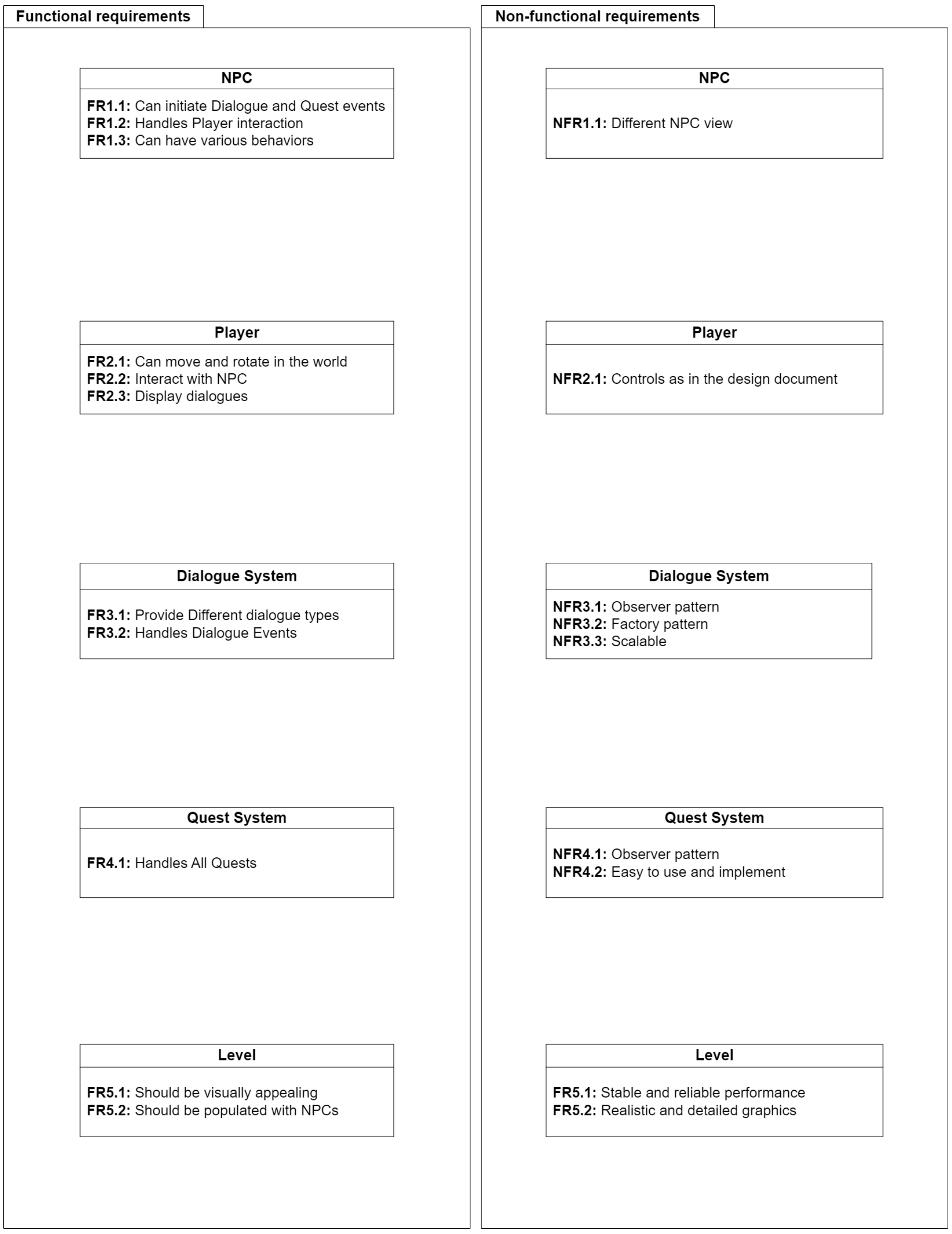

Functional requirements
NPC
FR1.1: Can initiate Dialogue and Quest events
FR1.2: Handles Player interaction
FR1.3: Can have various behaviors
Player
FR2.1: Can move and rotate in the world
FR2.2: Interact with NPC
FR2.3: Display dialogues
Dialogue System
FR3.1: Provide Different dialogue types
FR3.2: Handles Dialogue Events
Quest System
FR4.1: Handles All Quests
Level
FR5.1: Should be visually appealing
FR5.2: Should be populated with NPCs
Non-functional requirements
NPC
NFR1.1: Different NPC view
Player
NFR2.1: Controls as in the design document
Dialogue System
NFR3.1: Observer pattern
NFR3.2: Factory pattern
NFR3.3: Scalable
Quest System
NFR4.1: Observer pattern
NFR4.2: Easy to use and implement
Level
FR5.1: Stable and reliable performance
FR5.2: Realistic and detailed graphics

Figure 3.4: Functional and Non-functional requirements

prototype is built around two objects using both threads. And two systems that only work with the update thread[1]. This is due to the game process. This prototype consists of only two "active" objects. The main interaction takes place between them(NPC and Player). It is described in the sequence diagram in the Figure 3.6. In addition, a component-based architecture was chosen as the main one, which transferred all the computational logic to separate systems(components). In the following sections, these components will be designed according to the system requirements.

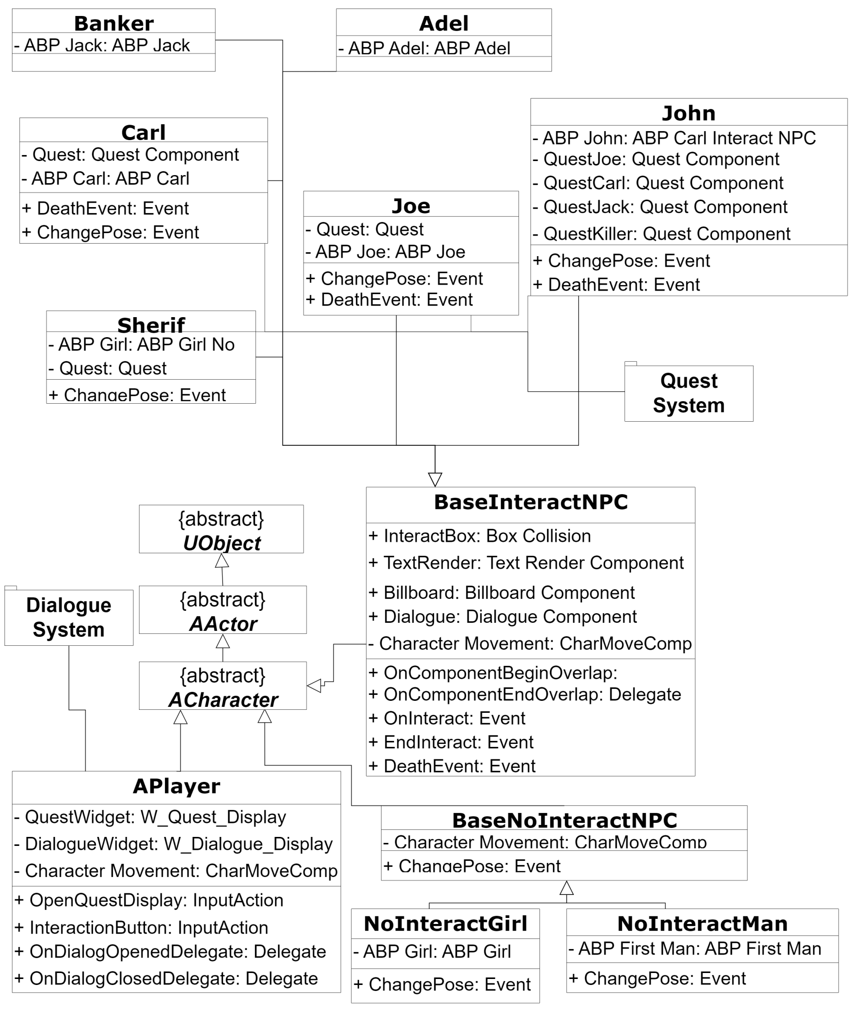

Banker
- ABP Jack: ABP Jack
Adel
- ABP Adel: ABP Adel
Carl
- Quest: Quest Component
- ABP Carl: ABP Carl
+ DeathEvent: Event
+ ChangePose: Event
John
- ABP John: ABP Carl Interact NPC
- QuestJoe: Quest Component
- QuestCarl: Quest Component
- QuestJack: Quest Component
- QuestKiller: Quest Component
+ ChangePose: Event
+ DeathEvent: Event
Joe
- Quest: Quest
- ABP Joe: ABP Joe
+ ChangePose: Event
+ DeathEvent: Event
Sherif
- ABP Girl: ABP Girl No
- Quest: Quest
+ ChangePose: Event
Quest System
BaseInteractNPC
+ InteractBox: Box Collision
+ TextRender: Text Render Component
+ Billboard: Billboard Component
+ Dialogue: Dialogue Component
- Character Movement: CharMoveComp
+ OnComponentBeginOverlap:
+ OnComponentEndOverlap: Delegate
+ OnInteract: Event
+ EndInteract: Event
+ DeathEvent: Event
{abstract} UObject
{abstract} AActor
{abstract} ACharacter
Dialogue System
APlayer
- QuestWidget: W_Quest_Display
- DialogueWidget: W_Dialogue_Display
- Character Movement: CharMoveComp
+ OpenQuestDisplay: InputAction
+ InteractionButton: InputAction
+ OnDialogOpenedDelegate: Delegate
+ OnDialogClosedDelegate: Delegate
BaseNoInteractNPC
- Character Movement: CharMoveComp
+ ChangePose: Event
NoInteractGirl
- ABP Girl: ABP Girl
+ ChangePose: Event
NoInteractMan
- ABP First Man: ABP First Man
+ ChangePose: Event

Figure 3.5: Top View Class Diagram

[1]At the moment, the presence of other objects in the world that also interact with the two thread isn't taken into account. Additionally, the aspect of utilizing two implemented widgets for each system is overlooked, which undoubtedly work with both systems

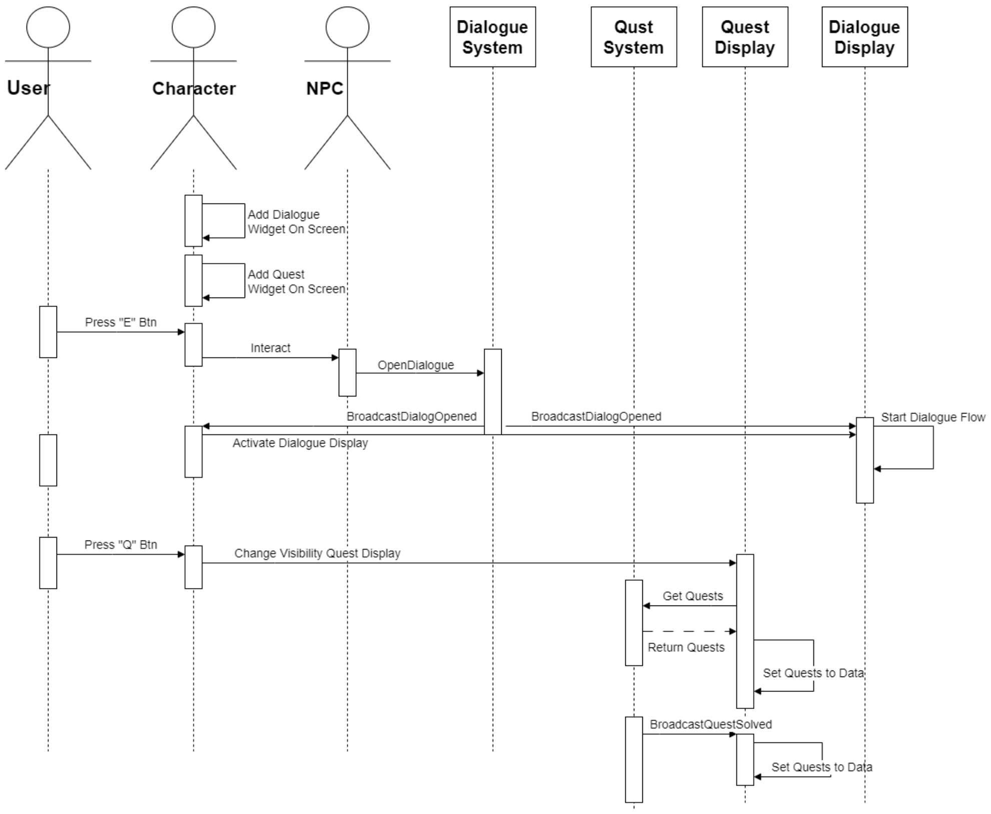


Figure 3.6: Top View Sequence Diagram

3.4.1 Design Dialogue System

The dialog system should have different types of dialogues, which are included in the same dialogue. Therefore, it was decided to divide the dialog into a sequence of lines (in this case, a linked list). Each line has its own internal logic depending on the type, while implementing basic methods.
The system involves creating and configuring dialogues before use. To do this, it was decided to use a UI layer in the form of a Data Asset. The Data Asset was chosen for the following reasons:

Organized and Centralized Data Management
It allows you to create dialogues in a structured form and later work with it in a single place, which makes it easier to interact with them and as a result presents one dialog as one structure. It is important to note that the interaction with this structure takes place in a convenient form, that describes in section 4.3.

Reusability
This concept is quite important in the context of dialogues. One dialogue

can be repeated for different characters. In the case of using the asset date, this is also applicable. One such structure can be used repeatedly, without any difficulties, which gives this asset (in fact, a dialogue) the possibility of reuse.

One character can have an unlimited number of possible dialogues. This means that the system should provide for the possibility of changing the current dialogue. Therefore, the ChangeDialogueDA method was designed, which changes the dialogue of the desired character. This method gets a new Asset date on the intro.
It is important to use the observer pattern. Since the state of the dialog can affect other components of the game. Due to this, it was decided to add two delegates that will store all ”subscribers”. When an event occurs, delegates send notifications to all subscribers about its triggering.

3.4.2 Design Quest System

In the design phase, it was decided to divide the work with quests into two types. The first type is preparatory or the creation of a quest. This will be done in the actor component before creating the packaged version. The quest will be completed and thus tied to the game character. The second type is responsible for interacting with quests during a runtime. Due to this, quest subsystem, that inherited from game instance, was added. This inheritance allowed to initialize subsystem earlier than the quests themselves. And at the time of initialization of each component, the subsystem will already be initialized. Thus, each quest will be able to register itself in the quest container in the subsystem. For further interaction, the following methods were designed: FindQuestByKeyWord 3.2, GetAllActiveQuests 3.1. They sufficiently cover the use cases of the prototype. The implementation of the Observer Pattern within the quest system is also important. Again, because of this method, the necessary other systems can ”subscribe” to quest changes. For example, to display a completed task, or to voice this state without additions and changes in the quest system.

Listing 3.1: GetAllActiveQuests Method

```
TArray<UQuestComponent*> UQuestSubSystem::GetAllActiveQuests()
{
  TArray<UQuestComponent*> ActiveQuests;

  for (UQuestComponent* Iter : QuestStorage)
  {
    if (Iter->GetIsQuestActive())
      ActiveQuests.Add(Iter);

  }
  return ActiveQuests;
}
```

Listing 3.2: FindQuestByKeyWord Method

```cpp
UQuestComponent* UQuestSubSystem::FindQuestByKeyWord(FString
    KeyWord) {
  return (UQuestComponent*) QuestStorage.FindByPredicate([&](
      UQuestComponent* x){ return x->GetKeyWord().Equals(KeyWord
      ); });
}
```

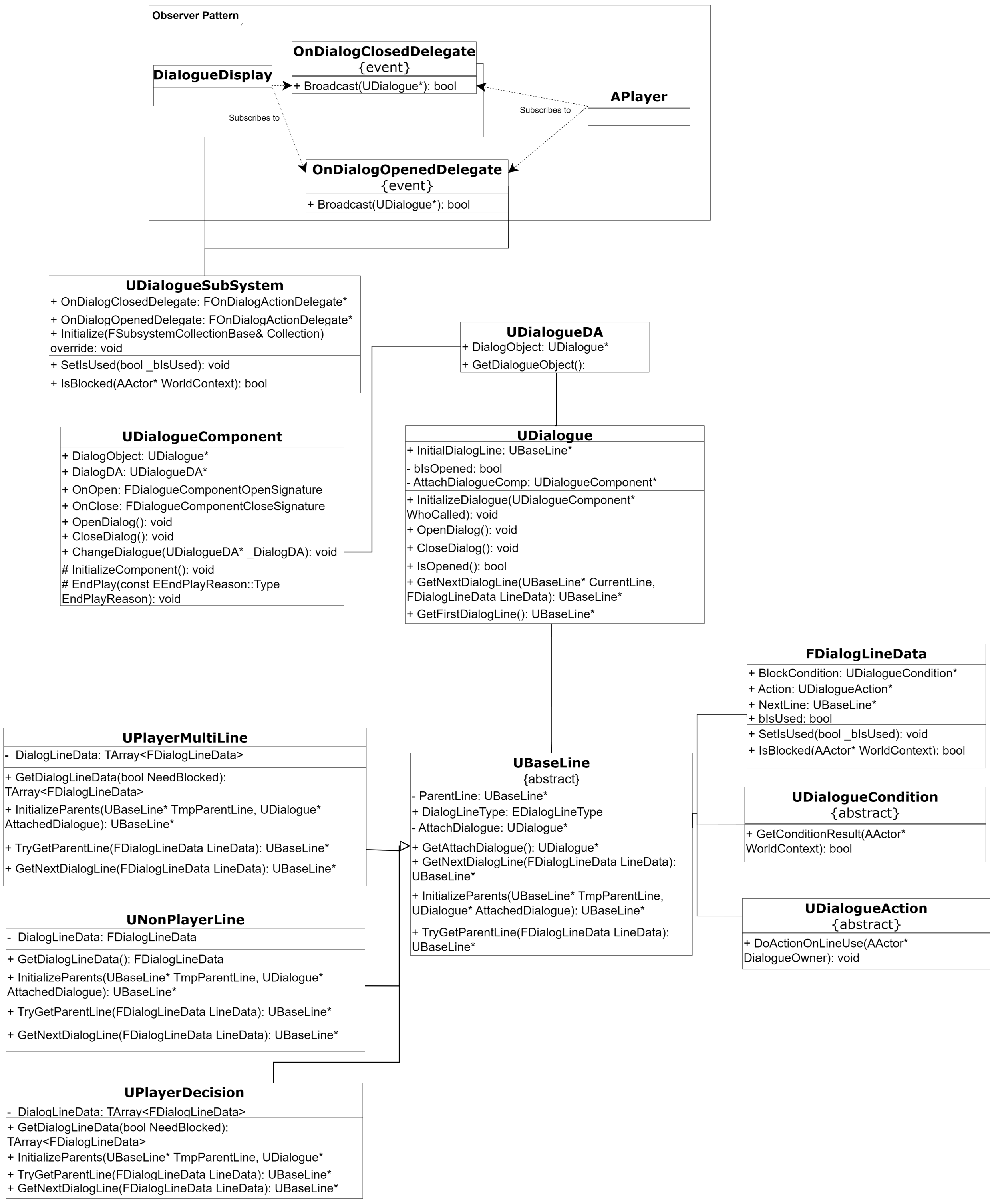

Observer Pattern
DialogueDisplay
OnDialogClosedDelegate
{event}
+ Broadcast(UDialogue*): bool
APlayer
Subscribes to
Subscribes to
OnDialogOpenedDelegate
{event}
+ Broadcast(UDialogue*): bool
UDialogueSubSystem
+ OnDialogClosedDelegate: FOnDialogActionDelegate*
+ OnDialogOpenedDelegate: FOnDialogActionDelegate*
+ Initialize(FSubsystemCollectionBase& Collection) override: void
+ SetIsUsed(bool _bIsUsed): void
+ IsBlocked(AActor* WorldContext): bool
UDialogueDA
+ DialogObject: UDialogue*
+ GetDialogueObject():
UDialogueComponent
+ DialogObject: UDialogue*
+ DialogDA: UDialogueDA*
+ OnOpen: FDialogueComponentOpenSignature
+ OnClose: FDialogueComponentCloseSignature
+ OpenDialog(): void
+ CloseDialog(): void
+ ChangeDialogue(UDialogueDA* _DialogDA): void
# InitializeComponent(): void
# EndPlay(const EEndPlayReason::Type EndPlayReason): void
UDialogue
+ InitialDialogLine: UBaseLine*
- bIsOpened: bool
- AttachDialogueComp: UDialogueComponent*
+ InitializeDialogue(UDialogueComponent* WhoCalled): void
+ OpenDialog(): void
+ CloseDialog(): void
+ IsOpened(): bool
+ GetNextDialogLine(UBaseLine* CurrentLine, FDialogLineData LineData): UBaseLine*
+ GetFirstDialogLine(): UBaseLine*
FDialogLineData
+ BlockCondition: UDialogueCondition*
+ Action: UDialogueAction*
+ NextLine: UBaseLine*
+ bIsUsed: bool
+ SetIsUsed(bool _bIsUsed): void
+ IsBlocked(AActor* WorldContext): bool
UPlayerMultiLine
- DialogLineData: TArray<FDialogLineData>
+ GetDialogLineData(bool NeedBlocked): TArray<FDialogLineData>
+ InitializeParents(UBaseLine* TmpParentLine, UDialogue* AttachedDialogue): UBaseLine*
+ TryGetParentLine(FDialogLineData LineData): UBaseLine*
+ GetNextDialogLine(FDialogLineData LineData): UBaseLine*
UBaseLine
{abstract}
- ParentLine: UBaseLine*
+ DialogLineType: EDialogLineType
- AttachDialogue: UDialogue*
+ GetAttachDialogue(): UDialogue*
+ GetNextDialogLine(FDialogLineData LineData): UBaseLine*
+ InitializeParents(UBaseLine* TmpParentLine, UDialogue* AttachedDialogue): UBaseLine*
+ TryGetParentLine(FDialogLineData LineData): UBaseLine*
UDialogueCondition
{abstract}
+ GetConditionResult(AActor* WorldContext): bool
UDialogueAction
{abstract}
+ DoActionOnLineUse(AActor* DialogueOwner): void
UNonPlayerLine
- DialogLineData: FDialogLineData
+ GetDialogLineData(): FDialogLineData
+ InitializeParents(UBaseLine* TmpParentLine, UDialogue* AttachedDialogue): UBaseLine*
+ TryGetParentLine(FDialogLineData LineData): UBaseLine*
+ GetNextDialogLine(FDialogLineData LineData): UBaseLine*
UPlayerDecision
- DialogLineData: TArray<FDialogLineData>
+ GetDialogLineData(bool NeedBlocked): TArray<FDialogLineData>
+ InitializeParents(UBaseLine* TmpParentLine, UDialogue*
+ TryGetParentLine(FDialogLineData LineData): UBaseLine*
+ GetNextDialogLine(FDialogLineData LineData): UBaseLine*

Figure 3.7: Dialogue System Class Diagram

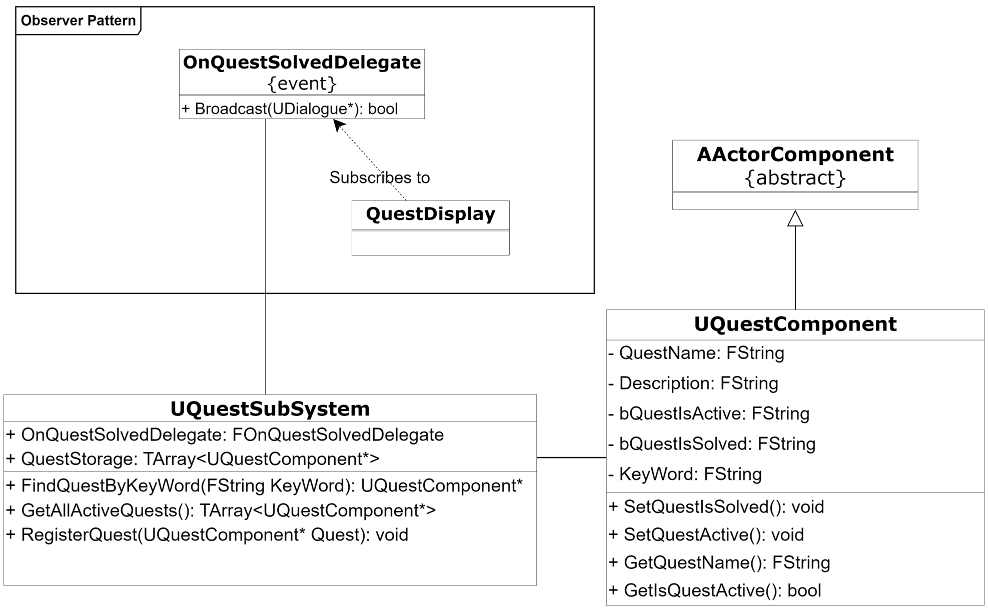

Observer Pattern
OnQuestSolvedDelegate
{event}
+ Broadcast(UDialogue*): bool
Subscribes to
QuestDisplay
AActorComponent
{abstract}
UQuestComponent
- QuestName: FString
- Description: FString
- bQuestIsActive: FString
- bQuestIsSolved: FString
- KeyWord: FString
+ SetQuestIsSolved(): void
+ SetQuestActive(): void
+ GetQuestName(): FString
+ GetIsQuestActive(): bool
UQuestSubSystem
+ OnQuestSolvedDelegate: FOnQuestSolvedDelegate
+ QuestStorage: TArray<UQuestComponent*>
+ FindQuestByKeyWord(FString KeyWord): UQuestComponent*
+ GetAllActiveQuests(): TArray<UQuestComponent*>
+ RegisterQuest(UQuestComponent* Quest): void

Figure 3.8: Quest System Class Diagram

Chapter 4

Implementation

The implementation chapter focuses on the implementation and interaction with implemented systems based on previously designed game components and managers. The chapter is divided into several sections representing the whole implementation phase. Section 4.1 will present all the important elements of the implementation of the designed systems, both Dialog and Quest. Section 4.2 will present the extensions used. Thanks to which the prototype development has achieved greater efficiency. Section 4.3 will present the key works performed within Unreal Engine 4. In the last section, user testing will be conducted.
Before starting, I should mention the game engine version and other basic initial conditions. The prototype was developed on the Unreal Engine version 4.27.2. The code was written in C++. Visual programming (Blueprints) will also be used to speed up prototype development. The prototype is available at Google Drive. It is also worth noting that not all the work done will be presented here. A large number of hours were spent debugging and improving the code.

4.1 System Implementation

This section is dedicated to two systems implemented based on earlier design work. It will describe key aspects of each system's implementation, including important delegate calls that enable the application of the Observer pattern. Possible future extensions of each system will also be discussed.

4.1.1 Quest System

The Quest System was developed based on the previously conducted design phase. A custom Actor Component was created, as shown in Listing 4.1, which serves as the primary location for quest modifications or expansions. It is key to acknowledge that that each quest is identified using a string value. String

value is assigned during the creation or filling data in the quest. This value is later utilized to locate the desired quest through the implemented FindQuest-ByKeyWord event. The same search for this value is executed when ”Solving the quest”. Both of these events, as well as the getter of all active quests, enable external system usage. Since all of them are implemented in a subsystem inherited from the game instance. Here it is worth noting the triggering of the delegate, which notifies everyone ”subscribers” about changes in the quest’s state.
As for future enhancements, the quest search system could be improved by replacing the string key values with tag values, which provide a higher quality of usage. Furthermore, it is possible to expand the variety of quests, for example, to collect something the variety of quests could be extended to include collection-based quests (e.g., reaching a specific value).

Listing 4.1: Quest Component

```cpp
DECLARE_DYNAMIC_MULTICAST_SPARSE_DELEGATE(
    FQuestComponentSolvedSignature, UQuestComponent, OnSolved);

UCLASS( ClassGroup=(Custom), meta=(BlueprintSpawnableComponent)
    )
class SSQUEST_API UQuestComponent : public UActorComponent
{
  GENERATED_BODY()

public:

  UPROPERTY(BlueprintAssignable, Category = "Components|
      OnSolved")
  FQuestComponentSolvedSignature OnSolved;

  UQuestComponent();

  virtual void InitializeComponent() override;

  UFUNCTION(BlueprintCallable)
    void SetQuestActive();

  UFUNCTION(BlueprintCallable)
    void SetQuestIsSolved();

  bool GetIsQuestActive() const { return bQuestIsActive; }

  UFUNCTION(BlueprintCallable)
  bool GetIsQuestSolved() const { return bQuestIsSolved; }
// private:

  UFUNCTION(BlueprintCallable)
    FString GetDescription() const { return Description; }

  UFUNCTION(BlueprintCallable)
    FString GetKeyWord() const { return KeyWord; }
private:
  UPROPERTY(EditAnywhere)
    FString Description;

  UPROPERTY(EditAnywhere)
    bool bQuestIsActive;

  UPROPERTY(EditAnywhere)
    bool bQuestIsSolved;

  UPROPERTY(EditAnywhere);
    FString KeyWord;
};
```

Listing 4.2: Open And Close Dialogue Delegate Call

```cpp
void UDialogue::OpenDialog()
{
  //Call attached component open event
  AttachDialogueComp->OnOpen.Broadcast();

  const UDialogueSubSystem* Subsystem = Cast<UDialogueSubSystem
      >(USubsystemBlueprintLibrary::GetGameInstanceSubsystem(
      this, UDialogueSubSystem::StaticClass()));


  //If subsystem exist at this time, set that dialogue is
      opened and call Delegate about Opening
  if(IsValid(Subsystem))
  {
    bIsOpened = true;
    Subsystem->OnDialogOpenedDelegate.Broadcast(this);
  }

}

void UDialogue::CloseDialog()
{
  //Call attached component close event
  AttachDialogueComp->OnClose.Broadcast();

  const UDialogueSubSystem* Subsystem = Cast<UDialogueSubSystem
      >(USubsystemBlueprintLibrary::GetGameInstanceSubsystem(
      this, UDialogueSubSystem::StaticClass()));

  //If subsystem exist at this time, set that dialogue is
      opened and call Delegate about Opening
  if (IsValid(Subsystem))
  {
    bIsOpened = false;
    Subsystem->OnDialogClosedDelegate.Broadcast(this);
  }
}
```

Listing 4.3: Same Logic for All types of lines in GetNextDialogueLine

```
if (IsValid(LineData.Action))
  {
    if(GetAttachDialogue() && GetAttachDialogue()->
       GetAttachDialogueComp())
    {
      //Call Action that attached to this line.
      LineData.Action->DoActionOnLineUse(GetAttachDialogue()->
         GetAttachDialogueComp()->GetOwner());
      LineData.Action = nullptr;
    }
  }

if(!IsValid(LineData.NextLine))
{
  if(GetParent())
     return GetParent()->TryGetParentLine(LineData);
}

return LineData.NextLine;
```

Listing 4.4: GetDialogLineData Event For PlayerMultiLine line type

```
TArray<FDialogLineData> UPlayerMultiLine::GetDialogLineData(
    bool NeedBlocked) const
{
  TArray<FDialogLineData> NotBlockedDialogues;

  if (NeedBlocked)
    return NotBlockedDialogues = DialogLineData;

  for (int32 Idx = 0; Idx < DialogLineData.Num(); Idx++)
  {
    //Check if Line doesn't contain BlockCondition then add and
         continue
    if (!DialogLineData[Idx].BlockCondition) {
      NotBlockedDialogues.Add(DialogLineData[Idx]);
      continue;
    }

    //If BlockCondition exist, and not blocked then add
    if(DialogLineData[Idx].BlockCondition && !DialogLineData[
       Idx].IsBlocked(GetAttachDialogue()->
       GetAttachDialogueComp()->GetOwner()))
    {
      NotBlockedDialogues.Add(DialogLineData[Idx]);
    }

  }

  return NotBlockedDialogues;
}
```

4.1.2 Dialogue System

The first thing to note is the creation of a custom actor component that connects the character and the dialogue. The current dialogue is stored here as a Data Asset. Further configuration and compilation of the dialogue are implemented within this Data Asset (In the section 4.3 is additional information for Data Asset). During the runtime, this data structure can be changed by utilizing the implemented ChangeDialogue event in the component.

The Dialogue Subsystem is of considerable importance due to its implementation of the observer pattern. This is accomplished by creating two delegates and integrating their calls into the opening and closing of dialogues, as illustrated in Listing 4.2.

Dialogues were designed using different types of lines to cover various use cases. These include:

1. Base lines with plain text.

2. Lines with answer choices that affect the storyline.

3. Multi-lines for choosing a branch without impacting the storyline.

Each line implements both basic parent logic, as in Listing 4.3, and its own unique, as in Listing 4.4. The primary difference is the storage of FDialogLineData, which represents all information about the line. Baselines store only one such structure, the other two types of lines store an array of such structures. This is done to accommodate continuations or dialogue branch divergence. Multi-lines and lines with storyline changes differ in terms of usage recording. The former records only one selected option, allowing the player to return to another continuation, while the latter doesn't permit such a return and marks both as a completed state.

Each line has two actions that are processed when receiving the next line. The first is the BlockCondition responsible for the possibility of transmitting data of this line. The second action is directly the action that should occur at the time of calling this line (both usage are presented in Listings 4.3 and 4.4).

The main interaction with the system occurs through the functionality developed within the dialogue class. Events such as opening and closing originate from a higher level, namely the dialogue component. Further interaction with the lines takes place using the dialogue, which is received when the opening delegate is called or the closing. Next interaction provides by calling the GetNextDialogLine event.

As for future extensions, it is possible to add new dialogue types without making additional changes, as the system allows for it; one simply needs to inherit from the base type line. Additionally, the dialogue storage method could be altered by either changing the storage method inside the Data Asset for optimization or increasing the information stored in the component to enhance its capabilities.

4.2 Used technologies

In addition to the basic functionality of unreal engine 4, additional technologies were used. In order to achieve the best result in graphics, performance and development speed. A list of these technologies is provided below with a description of their use.

Character Creator 4

Character Creator 4 is a tool for creating the most detailed characters developed by Reallusion. From sculpting and working out the details of facial expressions, to applying clothes and animations from the built-in asset library. Also supports exporting to different formats such as .obj, .fbx, .bvh and .goz. In the case of this thesis, the FBX[2] format was selected as preferred. In the implementation phase, four characters were created using Character Creator 4. It took up to 10 hours to create them. While source 3D Ace notes that it will take up to 110 hours to create.[57] Confirmation can be found in the Walla Walla source, which asserts that a high-quality model may require up to a month or more of work[3].[58] Each created character is unique and was created on the basis of pre-created parts. At the same time, the use of this tool did not require deep knowledge of 3D modeling, which in turn significantly saved time and increased development productivity. And this is one of the tasks of the prototype.

The Landscape Tool

The Landscape Tool is a feature that allowed developers to design and sculpt realistic outdoor environment for games or production. The tool provides various possibilities of interaction with the terrain and much more. It also offers manipulation with terrain textures and work with foliage. Thanks to the foliage, a forest, grass or any other surface environment appears on the terrain. It can also be created inside the game engine based on the material and mesh. As a result of using this tool, a landscape was created in accordance with the setting identified in the game design document. Which also made it possible to increase the effectiveness of prototype development and simplified some tasks.

[2]FBX® data exchange technology is a 3D asset exchange format that facilitates higher-fidelity data exchange between 3ds Max, Maya, MotionBuilder, Mudbox and other propriety and third-party software.[56]

[3]In both sources, the focus is on creating highly detailed models for professional production. However, this level of detail is not relevant for this prototype. These sources also mention the average time required to create a common model, which is approximately 10 hours. Nonetheless, using this software has helped to speed up the development process.

Additional Plugins

Since the purpose of this work is not to create various models of additional elements of the environment and it is more aimed at creating a consistent RPG prototype with certain mechanics and other conditions. In order to comply with these conditions, additional plugins and other resources were used. For example, to create animations, as well as some characters were taken from Mixamo.[59] Western plug-in models were used to fill the level.[60]

4.3 Game Widgets

The previous section it was mostly about technologies or functions that are used in the preparation phase. This section focuses on using the implemented systems that were designed in Chapter 4 and Implemented in Chapter 3. In particular, the main place of interaction with the system is the component responsible for presenting the results obtained as a result of this interaction. The User Widget was selected as such a component. In the implementation phase, two main widgets were created for two systems, dialog and quest. The following is a description of the implementation of widgets and the use of the implemented system. A description of the implementation and possible extensions of the system was followed in section 4.1.

Quest Display

To display the relevant information pertaining to quests, as best option a User Widget was chosen. This created widget will be assigned to the player at the time of its creation in the game world. This is done for two reasons:

1. The list of active quests is opened by the player.
2. To ensure the correct representation to the user screen.

The Quest Widget has two primary functions: displaying a list of active quests and providing visual effect for quest completion. To perform the first function, a method for displaying the quest list has been implemented (Figure 4.1). For the second function, the widget subscribes to receive updates on quest states (the delegate mentioned in Section 3.4.2). Consequently, it displays a notification at the appropriate moment and then disables it (Figure 4.2).

Dialogue Display

This widget is responsible for interacting with the dialog system and in particular working with dialogues. For example, their display and switching. At the start of the game, this widget is attached to the player but with the rendering flow turned off.

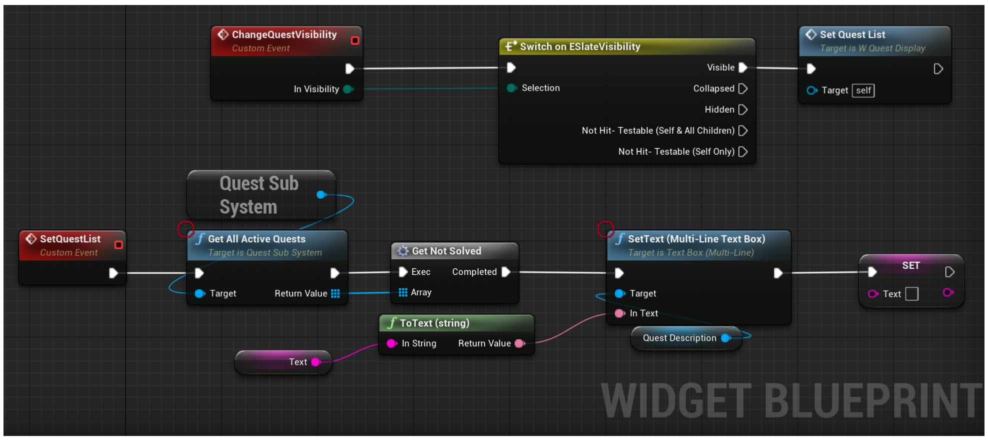


Figure 4.1: Show Active Quests

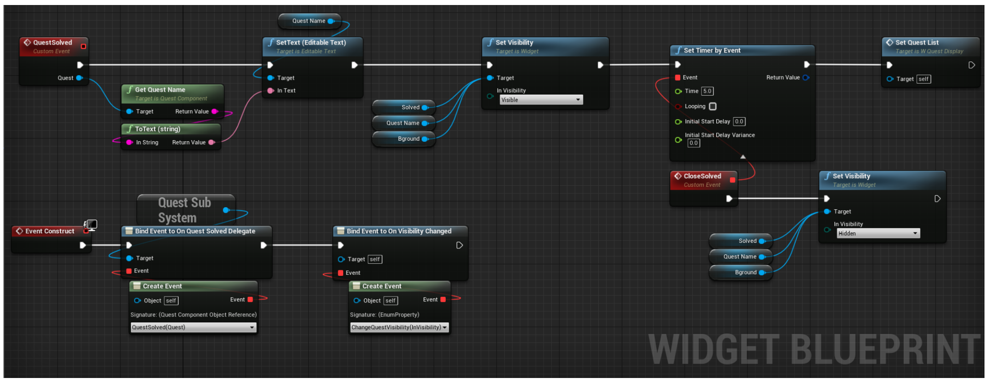


Figure 4.2: Using Observer Pattern in Quest Widget

Upon the initiation of a dialogue, widget draw thread activates and begins its interaction with the open dialogue. This interaction is facilitated by the implementation of the observer pattern in the dialogue system, which allows for ”subscription” to dialogue system events such as Initiate Event. As a result, when a dialogue initiates, it is possible to work with the dialogue and display it to the user screen (Figure 4.4).

It is also important to clarify that dialogues consist of different types of lines following each other. For example, lines like ”variety of answers” or ”plain text”, both of which require individual processing. In the context of implementing the Dialogue Widget in Unreal Engine 4, individual line processing affects as follows. For example, the ”answer options” type should be presented on interactive buttons, and the ”plain text” type should be displayed in a text field. When the next line is received, depending on the type of line, the appropriate processing algorithm is applied. The figure 4.3 illustrates the process of working with different types of lines in the dialog. In addition, this widget is scalable and when new type of line is added to the dialogue, the only thing remains is to add an additional algorithm for processing this line to UI, or use the existing one.

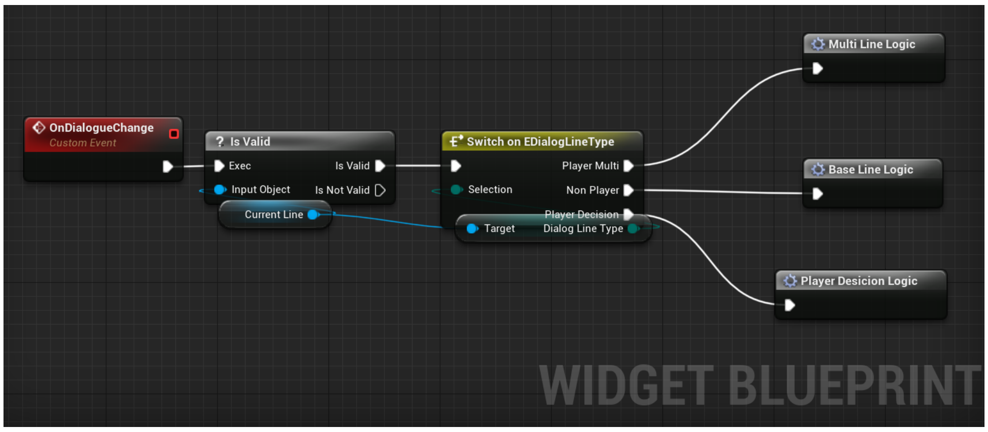


Figure 4.3: Working With Dialogue Lines

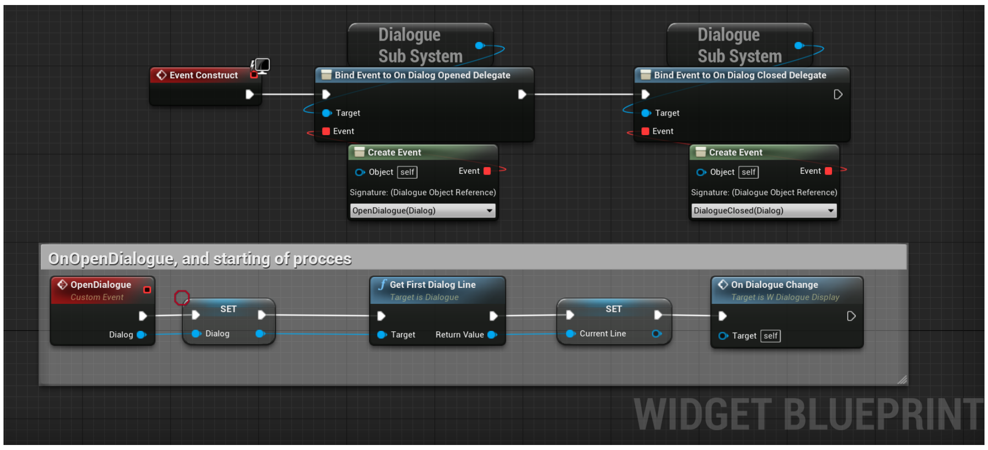


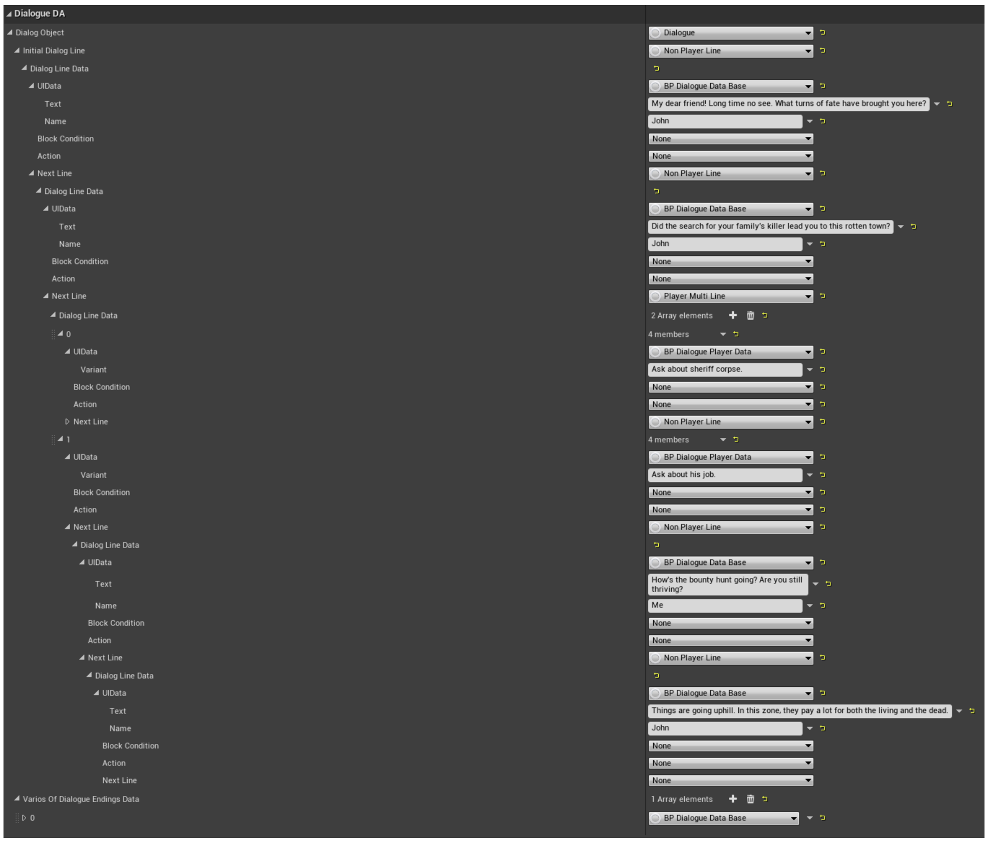
Figure 4.4: Dialogue Observer Pattern Use

Data Asset

It is important to separately note the use of Data Asset. In earlier chapters, its purpose and reasons for use have already been mentioned. In this section, the process of working specifically inside the Data Asset will be described.

The creation process is carried out using the engine's basic functionality; it is only necessary to select the base class of the Data Asset from the dialogue system (Figure 4.6). After opening, work with the lines begins immediately. You need to choose the initial line, after which a list will shown. It needs to be filled in depending on the type of the chosen line. The BlockCondition for the line is also set here, along with the action that should occur at this moment and the next line. More details can be found in Figure 4.7.

As a result of such manipulations, a long list is obtained (Figure 4.5). Branches and sequences of lines are visually visible in it. This simplifies and speeds up the creation of individual dialogues.



```
Dialogue DA
Dialog Object                                   Dialogue
  Initial Dialog Line                           Non Player Line
    Dialog Line Data
      UIData                                    BP Dialogue Data Base
        Text                                    My dear friend! Long time no see. What turns of fate have brought you here?
        Name                                    John
      Block Condition                           None
      Action                                    None
      Next Line                                 Non Player Line
        Dialog Line Data
          UIData                                BP Dialogue Data Base
            Text                                Did the search for your family's killer lead you to this rotten town?
            Name                                John
          Block Condition                       None
          Action                                None
          Next Line                             Player Multi Line
            Dialog Line Data                    2 Array elements
              0                                 4 members
                UIData                          BP Dialogue Player Data
                  Variant                       Ask about sheriff corpse.
                Block Condition                 None
                Action                          None
                Next Line                       Non Player Line
              1                                 4 members
                UIData                          BP Dialogue Player Data
                  Variant                       Ask about his job.
                Block Condition                 None
                Action                          None
                Next Line                       Non Player Line
                  Dialog Line Data
                    UIData                      BP Dialogue Data Base
                      Text                      How's the bounty hunt going? Are you still thriving?
                      Name                      Me
                    Block Condition             None
                    Action                      None
                    Next Line                   Non Player Line
                      Dialog Line Data
                        UIData                  BP Dialogue Data Base
                          Text                  Things are going uphill. In this zone, they pay a lot for both the living and the dead.
                          Name                  John
                        Block Condition         None
                        Action                  None
                        Next Line               None
  Varios Of Dialogue Endings Data               1 Array elements
    0                                           BP Dialogue Data Base
```

Figure 4.5: Data Asset Big View After Creation

4.4 Testing

The main purpose of testing the prototype is to identify the weaknesses of the designed project earlier so that the necessary changes are made before the start of the main development. Relying on the meza source, testing can be divided into two types: low-fidelity and high fidelity.[61] The first stage

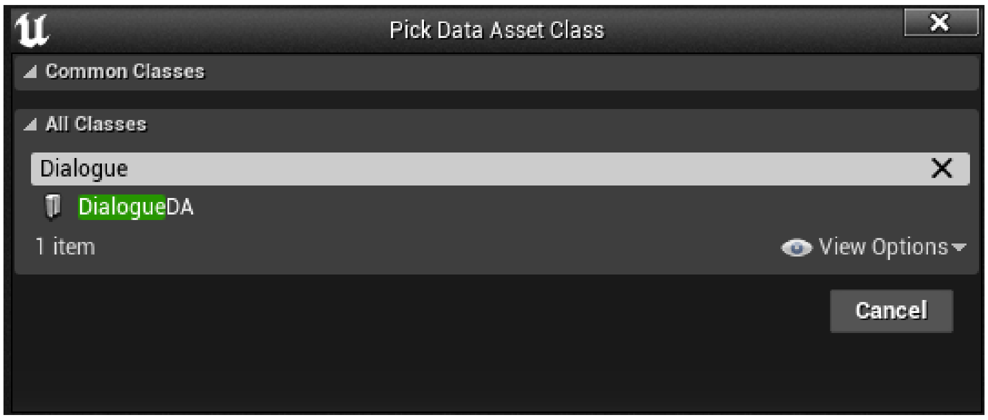
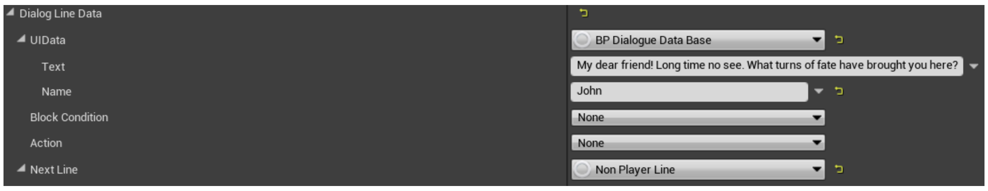


Figure 4.6: Data Asset Creation



Dialog Line Data	
UIData	BP Dialogue Data Base
Text	My dear friend! Long time no see. What turns of fate have brought you here?
Name	John
Block Condition	None
Action	None
Next Line	Non Player Line

Figure 4.7: Data Asset Structure Form

of testing determines the general direction of the prototype, which makes it possible to test different ideas in order to further determine the direction of design. The second stage of testing is more about improving the chosen direction. During this stage, global shortcomings in the following concepts are identified: general design directions, understanding of user flow, quality of graphics and UI components. After that, before the start of the main development, the developer will know about all the critical and crucial parts of the system based on user experience.

4.4.1 Testing preparation

Since testing is based on the user experience, ate the start of the testing, the user must receive certain data on the basis of which he could pass the test. In addition, it is important to determine the target audience of testing. Below is a list that will define the testing task, its main purpose, and more:

What exactly will be tested
: Based on game design document in 3.2, testing goal is - Can the user get acquainted with the storyline through the developed prototype. And how easily the user can do it.

Target audience

Based on the game design document, the target audience is players, there are no other restrictions.

The method of testing

As a testing method, a remote method was chosen using a Google form and a packaged version of the prototype for Windows.

Clear objective for users

Our hero lost his family after being attacked by bandits. In search of revenge, the hero travels around the country and arrives in a new city. At the border of the city he sees a dead body and immediately feels that something is wrong. After viewing the body, the player meets an old friend of John, who brings him up to speed.

Questions that will be analyzed

The questions posed to the user are analyzed in a separate section 4.4.2.

As the result of this preparation for testing, the Google Form appeared with the clear objective to users, with a list of questions and instructions for downloading and launching the prototype.

4.4.2 Targeted questions

First of all, the issues are divided into three categories. First category is about the user. Second is about the process of using the prototype. And the last one is related to the global impression of the prototype.

Identifying user

This section aims to gather user demographic data and gaming preferences to facilitate a better understanding of the target audience and to inform further development.

1. How old are you?
2. How often do you play and on which platforms?
3. What genres of video games do you prefer?

Evaluating comfort

This section focuses on assessing the user's experience with the game's interface and interactions to identify potential improvements that can optimize the gameplay experience.

1. How intuitive and easy-to-use was the game interface?
2. Did you experience any difficulties or inconveniences during interactions with NPCs and quest completion?
3. How would you rate the dialogue system with NPCs in terms of response variety and consequences of choices?

4. Did you have any issues navigating the game world?

Prototype quality

This section solicits user feedback regarding the game's possible issues and new feature suggestions with the objective of refining the game's design and functionality in subsequent development iterations.

1. Were there any flaws or bugs that you noticed during the game?
2. What did you like the most about this game prototype?
3. What would you like to improve or change in this game prototype?

4.4.3 Analyze results

Testing was conducted by providing a list of questions using Google Forms, as well as a link to the game prototype. Due to the limited number of users and the lack of a Windows device on for some, in certain situations, testing was carried out on a single device. The test results are open and can be found at the link. It is worth noting that the analysis is conducted at a specific moment, and the results can be changed in the future. The analysis of the results will be divided into three levels, according to the sections of questions. At each level, the percentage of responses and their interpretation will be described.

Identifying user

The majority of users are aged 16 -26 (71.4%), with remaining 28.6% of testers being 26-36 years old. The results for preferred gaming platform (multi-choice) were predictable, with PCs and consoles enjoying approximately equal popularity (57.1% and 71.4%, respectively). Regarding weekly gaming hours, the results were 14.3% play 0-4 hours, 28.6% play 4-8 hours, 42.9% play 8-20 hours and 14.3% play more than 20 hours. In terms of genre preference (multi-choice), RPG-based games were the most popular with 57.1%, with shooters coming in second with 42.9%.

From the user identification testing part, it was discovered that the tested audience is in the age between 16 and 26 years, with an average 8 hours of gameplay per week. The platform chosen for the implementation is appropriate. The genre chosen for the prototype is correctly selected and in demand.

Evaluating comfort

The first question in this section was concerned the ease of use of the game interface with a maximum rating of 5, and yielded the following results: A majority of respondents, representing 57.1%, gave a score of 3, 14.3% of users rated it at 4 and 28.6% rated it at 2. Evaluating the dialogues conducted with NPCs, the collected data revealed that 57.1%

of users were generally content, indicating above-average satisfaction with a rating of 4. In contrast, 28.6% showed moderate satisfaction with a score of 3, while the remaining 14.3% of testers were dissatisfied with and rated it at 2. Regarding interactions with NPCs and quests, the findings suggested that 71.4% of testers encountered no issues. Two other users indicated enhancements such as sound effects and closing dialogues when moving away from NPCs. When assessing the experience of navigating the game world, the following conclusions were made: A majority of respondents, representing 42.9%, rated it at 2, 28.6% gave a score of 3, and the same percentage assigned a score of 1.

From the evaluating comfort part, initial impression of using the developed prototype were revealed. Overall, the results is above average, with most people not experiencing any difficulties with interacting with the world. The majority of them appreciated the developed dialogues. There first difficulties were associated with the interface, which was found to be not as convenient and easy to use as expected. A critical issue is navigating within the game world, as users had difficulty determining their next location. Such feedback not only provides valuable input for future iterations of development but also demonstrates interest in the project.

Prototype quality

Delving into the initial user experience with the prototype, we find the following patterns. A majority of testers, representing 71.4%, did not encounter any interaction bugs, while the remaining 28.6% reported visual bugs with animations or environment. In response to the question about what users liked the most, 42.9% of respondents were unable to answer, Regarding the remaining 57.1%, summarizing their feedback, people enjoyed the direction of the story, style and overall setting. The final question in this section regarding future changes in the prototype was open-ended. It was expected that answers would later be considered as possible or necessary extensions of the project. Users reacted differently, and I would like to highlight the following conclusions: Addition of combat mechanics, item collection (so-called grinding), and character leveling. Also, there is a need to unify the environment design and expand animations.

As a result of the prototype quality part, it was revealed that users mostly did not encounter bugs. Moreover, some users separately highlighted the setting and style of the prototype. Thanks to the final question in this section of questions, future innovations were discovered, as well as necessary changes in current components.

Summing up the overall results of the testing, the first thing to note is the achievement of the main goal, namely the development of a working prototype.

The conducted testing did not reveal any critical bugs, but on the contrary, it highlighted the strengths and weaknesses of the designed and developed prototype. The good idea of the plot and interesting overall design of the project were noted, and necessary new mechanics were identified that should be added, for example, improving user guidance in the game. Thanks to the testing, the further course of development will be adjusted and expanded, which will undoubtedly lead the project to new successes.

Chapter 5

Conclusion

This thesis aimed to create the basis for various RPG-based games. To achieve this, extensive work has been conducted, divided into several phases.

During the analysis phase, the main concepts of RPG-based games were identified. This was accomplished by researching and analyzing existing games of a similar genre. An analysis of existing solution was also conducted to choose a game engine, ultimately leading to the selection of Unreal Engine 4 as the development tool.

During the design phase, the process of designing a game prototype was studied, resulting in the creation of a game design document. Moreover, based on this document and the basis RPG concepts, the primary game systems such as dialogue and quest were designed.

During the implementation phase, the designed systems were applied to create an RPG-based game prototype based on the game design document. Additionally, to increase development efficiency, additional technologies such as Character Creator 4 and Terrain Tool were explored and utilized. A significant part of this phase is the description of the usage of each developed system, which can be helpful for the development of other RPG-based games. Moreover, potential extensions of each system were presented.

As a final validation of the prototype, user testing was conducted, which demonstrated interest in the project. The testing did not reveal any critical bugs and provide valuable insights. Based on the test results, new concepts (or mechanics) were identified that should be added in future phases to expand the functionality.

After navigating through all stages of the thesis and receiving positive feedback from users interacting with the prototype, I can confidently state that the primary objectives have been achieved. This is because the prototype was developed using systems that were designed and implemented with potential for reuse in future prototypes. Furthermore, my personal motivations have been satisfied. The process of working on this thesis has offered a fresh and engaging insight into the world of game development.

Bibliography

[1] Increase in time spent video gaming during the COVID-19 pandemic worldwide as of June 2020. [online], [cit. 2022-12-11]. Available from: `https://www.statista.com/statistics/1188545/gaming-time-spent-covid/`

[2] Suits, B. What is a Game? *Philosophy of science*, volume 34, no. 2, 1967: pp. 148–156.

[3] Henriksen, T. Hvordan kan man lære gennem fiction. *Teoretiske perspektiver på læring gennem deltagelse i rollespilsformidlet fiktion, specialesamlingen, Det kongelige bibliotek, Institut for psykologi, Københavns Universitet*, 2002.

[4] Simkins, D. *The arts of larp: Design, literacy, learning and community in live-action role play.* McFarland, 2014.

[5] Deterding, S.; Zagal, J. *Role-playing game studies: Transmedia foundations.* Routledge, 2018.

[6] What Is Storytelling? [online], [cit. 2022-12-11]. Available from: `https://storynet.org/what-is-storytelling/`

[7] Fryer, B. Storytelling That Moves People. [online], [cit. 2022-12-11]. Available from: `https://hbr.org/2003/06/storytelling-that-moves-people`

[8] The Witcher 3: Wild Hunt Awards. [online], [cit. 2022-12-12]. Available from: `https://www.imdb.com/title/tt2993508/awards/?ref_=tt_awd`

[9] Purchese, R. The Witcher 3: What is a next-gen RPG? [online], [cit. 2022-12-12]. Available from: `https://www.eurogamer.net/the-witcher-3-what-is-a-next-gen-rpg`

[10] Endings. [online], [cit. 2022-12-12]. Available from: https://www.ign.com/wikis/the-witcher-3-wild-hunt/Endings

[11] NPCs — The Witcher 3 Wiki. [online], [cit. 2022-12-12]. Available from: https://thewitcher3.wiki.fextralife.com/NPCs

[12] THE WITCHER 3: WILD HUNT PC. [online], [cit. 2022-12-12]. Available from: https://www.metacritic.com/game/pc/the-witcher-3-wild-hunt

[13] Annual sales revenue generated by CD Projekt Group worldwide from 2013 to 2021. [online], [cit. 2022-12-12]. Available from: https://www.statista.com/statistics/523934/cd-projekt-annual-revenue/

[14] Divinity Original Sin II Awards. [online], [cit. 2022-12-12]. Available from: https://www.imdb.com/title/tt7410340/awards/?ref_=tt_awd

[15] Divinity: Original Sin II. [online], [cit. 2022-12-12]. Available from: https://howlongtobeat.com/game/39525

[16] Attributes — Divinity Original Sin 2 Wiki. [online], [cit. 2022-12-12]. Available from: https://divinityoriginalsin2.wiki.fextralife.com/Attributes

[17] Abilities — Divinity Original Sin 2 Wiki. [online], [cit. 2022-12-12]. Available from: https://divinityoriginalsin2.wiki.fextralife.com/Abilities

[18] Talents — Divinity Original Sin 2 Wiki. [online], [cit. 2022-12-12]. Available from: https://divinityoriginalsin2.wiki.fextralife.com/Talents

[19] Character Progression (Divinity: Original Sin). [online], [cit. 2022-12-12]. Available from: https://divinity.fandom.com/wiki/Character_Progression_(Divinity:_Original_Sin)

[20] DIVINITY: ORIGINAL SIN II PC. [online], [cit. 2022-12-12]. Available from: https://www.metacritic.com/game/pc/divinity-original-sin-ii

[21] Developer: Larian Studios. [online], [cit. 2022-12-12]. Available from: https://vginsights.com/developer/12015/larian-studios

[22] Fenlon, W. How Divinity: Original Sin almost bankrupted Larian Studios. [online], [cit. 2022-12-12]. Available from: https://www.pcgamer.com/how-divinity-original-sin-almost-bankrupted-larian-studios/

[23] Game Critics 2015 Awards. [online], [cit. 2023-05-09]. Available from: https://www.imdb.com/event/ev0001135/2015/1/?ref_=ev_eh

[24] The Making Of Fallout 4. [online], [cit. 2023-05-09]. Available from: https://www.gameinformer.com/b/features/archive/2015/11/06/the-making-of-fallout-4.aspx

[25] Fallout 4. [online], [cit. 2023-05-09]. Available from: https://fallout.fandom.com/wiki/Fallout_4

[26] Fallout 4. [online], [cit. 2023-05-09]. Available from: https://howlongtobeat.com/game/26729

[27] Fallout 4 characters. [online], [cit. 2023-05-09]. Available from: https://fallout.fandom.com/wiki/Fallout_4_characters

[28] Every Fallout Game, Ranked By Total Map Size. [online], [cit. 2023-05-09]. Available from: https://www.thegamer.com/every-fallout-map-size/#fallout-brotherhood-of-steel-15-000-sq-mi

[29] Fallout 4 primary statistics. [online], [cit. 2023-05-09]. Available from: https://fallout.fandom.com/wiki/Fallout_4_primary_statistics

[30] Level. [online], [cit. 2023-05-09]. Available from: https://fallout.fandom.com/wiki/Level#cite_note-2

[31] Gregory, J. *Game engine architecture*. AK Peters/CRC Press, 2018.

[32] Nordtveit, D. 10 Undeniable Ways Doom (1993) Shaped The FPS Genre. [online], [cit. 2022-12-11]. Available from: https://www.thegamer.com/doom-1993-ways-shaped-fps-genre/

[33] Entity Component System: An Introductory Guide. [online], [cit. 2022-12-12]. Available from: https://www.simplilearn.com/entity-component-system-introductory-guide-article#what_is_an_entity_component_system

[34] Post-Processing. [online], [cit. 2022-12-11]. Available from: https://www.utep.edu/keck/services/post-processing.html

[35] ILM reveals how it used Unreal Engine for 'The Mandalorian'. [online], [cit. 2023-03-17]. Available from: https://venturebeat.com/pc-gaming/ilm-reveals-how-it-used-unreal-engine-for-the-mandalorian/

[36] Designing visuals, rendering, and graphics. [online], [cit. 2023-03-17]. Available from: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/

[37] Gajsek, D. C vs C++: Complete Comparison Between Unity and Unreal Programming Language. [online], [cit. 2022-12-11]. Available from: `https://circuitstream.com/blog/c-vs-c-complete-comparison-between-unity-and-unreal-programming-languages/`

[38] Most used Engines. [online], [cit. 2022-12-12]. Available from: `https://https://itch.io/game-development/engines/most-projects`

[39] Graphics. [online], [cit. 2023-03-17]. Available from: `https://docs.unity3d.com/Manual/Graphics.html`

[40] How do I purchase read-only source code? [online], [cit. 2022-12-12]. Available from: `https://support.unity.com/hc/en-us/articles/10033669359124-How-do-I-purchase-read-only-source-code-`

[41] 2021 Developer Survey. [online], [cit. 2023-03-17]. Available from: `https://insights.stackoverflow.com/survey/2021#worked-with-vs-want-to-work-with-tools-tech-worked-want`

[42] Crytek (Company). [online], [cit. 2023-03-17]. Available from: `https://www.igdb.com/companies/crytek`

[43] CRYENGINE 5.5.0 Preview 1. [online], [cit. 2023-03-17]. Available from: `https://docs.cryengine.com/display/RN/CRYENGINE+5.5.0+Preview+1`

[44] Terrain Editor. [online], [cit. 2023-03-17]. Available from: `https://docs.cryengine.com/display/CEMANUAL/Terrain+Editor`

[45] Why developers choose CRYENGINE. [online], [cit. 2023-03-17]. Available from: `https://www.cryengine.com/news/view/why-developers-choose-cryengine`

[46] Vulkan Support in CRYENGINE. [online], [cit. 2023-03-17]. Available from: `https://docs.cryengine.com/display/CEMANUAL/Vulkan+Support+in+CRYENGINE`

[47] CRYENGINE V Manual. [online], [cit. 2023-03-17]. Available from: `https://docs.cryengine.com/`

[48] Schematyc. [online], [cit. 2023-03-17]. Available from: `https://docs.cryengine.com/display/CEMANUAL/Schematyc`

[49] TUTORIALS. [online], [cit. 2023-03-17]. Available from: `https://www.cryengine.com/tutorials`

[50] Gold, J. *Object-oriented Game Development*. Addison Wesley, 2004, ISBN 9780321176608. Available from: `https://books.google.cz/books?id=WRwFBG93tJMC`

[51] Ahmad, I.; Abdullasim, N. *Game Development from Idea to Prototype (UTeM Press)*. UTeM Press, 2020, ISBN 9789672145899. Available from: https://books.google.cz/books?id=8EYyEAAAQBAJ

[52] Game Design Document Template and Examples. [online], [cit. 2022-12-11]. Available from: https://www.nuclino.com/articles/game-design-document-template

[53] Will Wright's 3 Tips for Successful Prototyping in Video Games. [online], [cit. 2022-12-11]. Available from: https://www.masterclass.com/articles/will-wrights-tips-for-successful-prototyping-in-video-games

[54] Use-case diagrams. [online], [cit. 2022-12-11]. Available from: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

[55] Game architecture. [online], [cit. 2023-05-05]. Available from: https://subscription.packtpub.com/book/web-development/9781783551774/1/ch01lvl1sec10/game-architecture

[56] FBX. [online], [cit. 2022-12-12]. Available from: https://www.autodesk.com/products/fbx/overview

[57] HOW LONG DOES IT TAKE TO MAKE A 3D CHARACTER MODEL? [online], [cit. 2023-05-05]. Available from: https://wallawallastudio.com/article/how-long-does-it-take-to-make-a-3d-character-model/

[58] HOW LONG DOES IT TAKE TO CREATE A 3D MODEL? [online], [cit. 2023-05-05]. Available from: https://3d-ace.com/blog/how-long-does-it-take-to-create-a-3d-model/

[59] Mixamo - Get animated. [online], [cit. 2023-05-09]. Available from: https://www.mixamo.com/#/

[60] Western Desert Town. [online], [cit. 2023-05-09]. Available from: https://www.unrealengine.com/marketplace/en-US/product/western-desert-town

[61] Prototype testing: How to nail your next product launch. [online], [cit. 2023-05-09]. Available from: https://maze.co/blog/prototype-testing/#how-to-test-your-prototype

Appendix A

Acronyms

RPG	Role play game
TRPG	Tabletop role play game
LARP	Live action role play game
CRPG	Computer role play game
MORPG	Multiplayer online role play game
GE	Game Engine
NPC	Non-player character

Appendix B

Contents of enclosed CD

- `implementation` .... the implementation directory
  - `prototype.zip` .... an archive of the unreal project
  - `game` .... package version of prototype
- `text` .... the thesis text directory
  - `thesis` .... the directory of LaTeX source codes of the thesis
  - `thesis.pdf` .... the thesis text in PDF format