Assignment 3 – Zeid Aldaas

New Additions for Assignment 3 Bolded

Game Name – Knight's Quest.

Number of Scripts: **29 (16 on Assignment 2)**

Number of Game Objects using created scripts: **70 (53 on Assignment 2)** if including multiple objects of same functionality and type, **18 (11 on Assignment 2)** unique objects utilizing different scripts in different ways

Number of Scenes: **7** (MainMenu, Intro, Level, Corruptor Intro, Level 2, **Level 3** Outro)

Game Description:

Knight's Quest is a platformer game where the goal is to make it to the top to reach the tree of life to save the land from treachery. Dodge dangerous fire or make difficult jumps in order to make it to the top and achieve victory!

Now with a new level where you face off against the Corruptor and try to avoid his fiery wrath to have him consume himself in his own flame! Complete Level 1 to unlock the ability to load this new level from the main menu screen and skip level 1 whenever you like.

Directly after Level 2, face off against the Corruptor in a final battle where you use combat prowess to slay him and save the land once and for all! Complete Level 2 to unlock the ability to load level 3.

Controls: Arrows/AD to move, W/Space to jump, 1/Left-Click to attack, 2/Right-Click to heal, ESC to pause during level.

Scripts Description:

Here is a brief description of the purpose and function of the 12 scripts used for the development of Knight's Quest.

- MainMenu.cs: This handles the black screen and audio fading, as well as the PLAY and QUIT buttons in order to transition to the Intro scene or quit the game.
- TypeWriter.cs: This gives the text in the Intro and Outro scenes a typewriter effect, displaying it as if it was being typed on the screen.
- NextScene.cs: This handles the same black screen fading, as well as takes into account the input of any key in order to transition to the Level scene.
- FadeInOut.cs: This handles the same black screen fading for the Level scene.
- PlayerMovement.cs: The most essential script, this handles all the functionalities of controlling the Knight in the game, from animation triggers to physics, it handles the running, jumping, and turning animations.

- MovingPlatform.cs: This essential script enables certain platforms to move back and forth between two points.
- PlatformOnce.cs: This variation of the MovingPlatform.cs makes it so that when the player steps on this particular platform, it makes its movement to the other point once, and never moves again.
- PlatformTrigger.cs: Similar to PlatformOnce.cs, this waits for the player to step onto the platform, and once they do, it will function identically to the regular MovingPlatform.cs.
- DeathZone.cs: This very important script creates the functionality of the various Death Zones on the map, whether it is from falling out of bounds or touching a flame. It will prompt the player to play again after the death animation plays out.
- Fireball.cs: This essentially intertwines the Platform scripts and DeathZone script to create a moving hazard.
- GameVictory.cs: This is a variation of the DeathZone.cs script, where instead of prompting to play again, it transitions to the Outro scene using similar logic to the MainMenu.cs and NextScene.cs scripts.
- FinalScene.cs: This handles the same black screen fading, and takes the inputs ESC or Enter to either exit the game or go back to the main menu to play again.
- CorruptorScene.cs: This handles the same black screen fading, and takes the inputs ESC or Enter to either exit the game and save or go back to the main menu to play again.
- PauseGame.cs: Adds pausing and resuming game functionality by hitting ESC.
- MovingPlatformSync.cs: Makes designated platforms that are desired to be synced complete their route in the same time, making them synced.
- CorruptorVictory.cs: Variation of GameVictory.cs, where the victory is achieved after 30 seconds of game time without dying in the level.
- **BossHealth.cs: This handles the Corruptor's health pool and taking damage to ultimately allow the player to win once the health pool is depleted.**
- **PlayerHealth.cs: This handles the Player's health pool and taking damage to ultimately cause the player to lose once the health pool is depleted.**
- **BossHealthBar.cs: Integrates the healthbar script for the Corruptor's healthbar.**
- **PlayerHealthBar.cs: Integrates the healthbar script for the Player's healthbar.**
- **Checkpoint.cs: This creates the functionality of Checkpoints on any platform of my choosing, both moving and still. It also announces the Checkpoint Unlock message, and handles clearing checkpoint data in conjunction with GameSession.cs**
- **GameSession.cs: This script is used to hinge onto a persistent GameObject in the Level 1 scene in order to store and clear checkpoint data.**
- **FireballController.cs: This deals with having the Fireballs in Level 2 follow the character rather than have a fixed direction as it was in Assignment 2. The Fireballs shoot towards where the player was at during their initial trajectory and continue in a straight line towards the player.**
- **CorruptorFollow.cs: This essentially builds off the FireballController.cs, with some tweaks to account for differences in rotation**
- **FinalBossDefeat.cs: This script is essentially a combination of CorruptorVictory.cs (which know has improved logic to display a timer) and DeathZone.cs, in that it uses**

the CorruptorVictory.cs logic of the timer, but in the opposite way in that the timer expiring is defeat rather than victory.

- **FinalBossVictory.cs: This script uses similar logic as the victory scripts, but hinges on the states of the Player and Corruptor from the Health.cs files.**
- **PlayerCombat.cs: Handles the attack animation for the player.**
- **TipsManager.cs: This script uses an algorithm to shuffle between main menu screen tips in a neat and proper fashion.**
- **TipTypeWriter.cs: This script applies the TypeWriter.cs logic to the Tips.**

Game Object Composition:

The Game Object Composition for the MainMenu, Intro, Corruptor Intro, and Outro are essentially the same. They all have the Canvas objects for either text or buttons, as well as the Background object(s). The MainMenu scene also has the MenuMusic. These are all that is necessary to create these screens.

The Level scene contains the more complex Game Object Composition. It has the Player_Knight prefab which serves as the Player tag and is where the animations are handled. The most fundamental group of other objects after the Player_Knight object in the Level scene are the NonMovingObjects/MovingObjects which consist of the platforms both moving and non-moving, and the tree of life. Next, is the DeathZones objects which consist of the game boundaries and both stationary and moving fire. Following these are the same standard things we see in all the other scenes, which is the Background, Music, and the Play Again prompt on death with the Play and Quit buttons. Level 2 contains the same essential structure in terms of Game Objects and scripts, but is focused more on avoiding hazards rather than jumping on platforms. It also has different music. It features an evil wizard atop the tree summoning fire balls and destruction for the player to avoid. **Level 2's win condition is also different in that you must beat a timer to beat it. In Level 3, the Game Object Composition is again relatively similar, with the same Player_Knight prefab, Background Music object, and most of the same composition of other objects. However, the gameplay is significantly different, since it is now focused on attacking an enemy, healing, and using jumps and timing to your advantage to take down the Corruptor's health before he takes down yours. The Corruptor is a new Game Object here, although it existed in Level 2, it was treated just like all the other DeathZones in Level 2. There are certain mechanics to be aware of, such as doing more damage by spinning back and forth while attacking, doing more damage by charging ontop of the Corruptor (while also taking more damage), and reflection of damage by the Corruptor. The player must also defeat the Corruptor before the 1 minute timer expires, and the timer exists in the Canvas group.**

Scene Descriptions:

GameObject references denoted with parentheses, and scene references denoted with brackets:

- MainMenu: This scene displays a forest background (Background) with the title of the game (Title), the Menu Music (MenuMusic) and the Play and Quit buttons (Canvas). Play will transition to the next [Intro] scene, and Quit will close the client.
- Intro: This scene displays a cloudy background with a hill with a stone on it (Background), and displays some story writing with the typewriter effect (Canvas -> TMP). It prompts to Press any key to continue…, upon which pressing any key will transition to the next [Level] scene.
- Level: This scene starts off the Knight (Player_Knight) on a platform, and the Knight is to jump on the upper platforms (NonMovingObjects/MovingObjects) to try and maneuver to the level and make it to the top to get to the tree of life to win. There are flame hazards (DeathZones) on the way as well as moving platforms and small platforms to make the quest difficult. Reaching the tree of life will transition to the next scene [CorruptorIntro].
- CorruptorIntro: This scene displays a dark forest background (Background), with the same typewriter effect as Intro (Canvas -> TMP) displaying the news that the Corruptor is there. It prompts the player to Press ESC to save and quit or Enter to continue to the next scene [Level 2].
- Level 2: This scene places the Knight (Player_Knight) on the final platform with the tree (NonMovingObjects), with flames (DeathZones) attacking from all angles. The Knight is meant avoid these flames as long as he can until the Corruptor comes down. Successfully avoiding the flames will load the next scene [Level 3]
- **Level 3: Immediately after Level 2, the scene begins with the Corruptor (Corruptor) on the left side, and the Knight (Player_Knight) facing him. The health bar of the Corruptor (Canvas -> Corruptor Health) and the health bar of the Knight (Canvas -> PlayerHealth) are displayed, alongside the timer (Canvas -> Pause -> Timer) and information about healing cooldowns and usage (Canvas -> HealUsed/Canvas -> HealAvailable). Attacking the Corruptor and slaying him before he slays the Knight within the timer achieves victory for the Knight. This will transition to the next scene [Outro]**
- Outro: With a very similar format to the Intro scene, this scene displays an enchanted forest background (Background), where the same typewriter effect (Canvas -> TMP) is displayed to congratulate the player on winning the game. It prompts to Press ESC to quit or Enter to play again…

Source Code:

Included in .zip

General Discussion

Here is a list of features that were added that haven't been discussed already:

- **Volume settings made to persist across all scenes rather than main menu only**
- **Controls and How to Play information added on a new info tab on main menu**

- **Added indicators for which Level is selected at main menu**
- **Various sound effects added**
- **Added ambience to all gameplay levels**
- **Added menu sound effects for pressing buttons**
- **Added sound effect for dying by fire**
- **Hitboxes refined and animated properly on all moving objects**
- **Tips (Many of these are very valuable to know)**
- **To activate a moving platform after respawning at its checkpoint, jump.**
- **The first checkpoint in Level 1 is a crossroads… Pick your poison.**
- **Your checkpoints don't save when you quit to main menu or close the game.**
- **Hope you're not scared of heights.**
- **Beat Level 1 to unlock Level 2, and so on...**
- **Always be on your toes when dealing with fireballs.**
- **No one knows the origins of the Corruptor. There are mere myths and fairytales.**
- **The Corruptor becomes exposed while casting.**
- **The Corruptor reflects damage.**
- **Charging an enemy head-on does more damage, but leaves you more vulnerable.**
- **Spinning back and forth while attacking does more damage.**
- **Heal has a 10 second cooldown.**
- **The Corruptor can only damage you once every second... Strike fast!**
- **Brief Summary on How Requirements were Satisfied:**

1. **Instructions on how to play are on info tab of main menu as well as introduction screens before levels.**
2. **There are many interesting decisions for the player to make, from figuring out whether to take the parkour route or fire route on Level 1, to discovering different strategies on different areas of the platform on Level 2, to utilizing different types of attack and dodge strategies on Level 3.**
3. **This document and especially this General Discussion outlines very well the polishing that has been done for this iteration.**
4. **There is a clear win/loss condition, whenever the player dies, they are prompted to play again or quit, whenever they succeed, they advance and unlock the next level.**
5. **Saving and Loading works with the settings such as saving volume and fullscreen preferences, but also applies to checkpoints on a small scale, and level unlocks on a large scale.**
6. **There is a robust sound system in the game, with various sound effects, ambiences, and music tracks.**
7. **There is a volume slider and fullscreen opt in the settings.**
8. **There is a Main Menu screen with play and quit buttons, 3 levels of gameplay (with pausing possible on all 3), and smooth fade to black scene transitions between all scenes.**
9. **There is emergent gameplay in that there are multiple levels with very different playstyles, from parkour, to dodging, to combat.**

Analysis

1. Be a rule-based system

Knight's Quest qualifies strongly as a rule-based system. The game operates under a well-defined set of rules scripted through various functional scripts such as PlayerMovement.cs, DeathZone.cs, and BossHealth.cs, which govern player interactions, movement, combat dynamics, and consequences (e.g., death or victory). These scripts ensure that interactions within the game are consistent and predictable according to the rules coded into the game. This rigidity in behavior aligns perfectly with the definition of a rule-based system, where player actions are tightly governed by predefined logic and structure.

2. Have variable and quantifiable outcomes

The game exhibits variable and quantifiable outcomes notably through its level progression and combat results. Outcomes in Knight's Quest are dictated by player performance and choices. For instance, success in navigating through hazards or defeating the Corruptor directly impacts the game's progression—unlocking new levels and altering the game state. Additionally, the game scripts like CorruptorVictory.cs and FinalBossVictory.cs quantify victory conditions (e.g., surviving for a specific duration or depleting an enemy's health), making outcomes both variable and measurable based on player interaction.

3. Different outcomes have different values

Outcomes in Knight's Quest are hierarchically valuable and contribute differently to the overall game experience. For example, defeating the Corruptor in Level 3 is of higher value compared to merely avoiding obstacles in Level 1, as it leads to the game's climax and resolution. Moreover, the game design employs different rewards and consequences for varying outcomes—unlocking levels, affecting game narrative through intro and outro scenes, and changes in the gameplay environment. This differentiation in outcome importance enhances the strategic depth and player engagement.

4. Player effort is required

Player effort is fundamental to progressing in Knight's Quest. The game demands active engagement in forms of strategic movement, timing, combat, and resource management (e.g., health and healing). The scripts such as PlayerMovement.cs and PlayerCombat.cs require direct input and skill from the player to effectively navigate and interact within the game world. This requisite for active participation ensures that player effort is directly proportional to game progression and success.

5. The player feels emotionally attached to the outcome

Knight's Quest fosters emotional attachment through its narrative structure, character development, and gameplay challenges. The narrative drive to save the land from treachery, combined with the escalating difficulty and complexity of challenges, creates a compelling motive for players. Emotional engagement is further amplified by the game's feedback mechanisms and aesthetic elements (typewriter effects in storytelling, sound

design, and visual cues) that enrich the player's immersion and emotional investment in the outcomes.

6. The consequences of the activity are negotiable

While the game’s core mechanics and narrative progression follow a predetermined path, certain elements of Knight’s Quest offer negotiable outcomes through player choices and interactions. For example, the options to select different paths or strategies in combat allow players to influence immediate gameplay consequences and experiences. However, the game's primary storyline and outcomes remain largely invariable, providing a structured yet partially flexible gameplay environment.

References

- Unity Assets
- Character Model: 2D Simple Character : Swordman – Black Hammer
- Platforms: Magic Cliffs Pixel Art Environment - Ansimuz
- Backgrounds
    - Animated Pixel-Art Backgrounds – Feony
    - Painted HQ 2D Forest Medieval Background – Super Brutal Assets
- Flames: Pixel FX – Fire – Admurin
- Intro and Outro Fonts: Hana Pixel Font – Mystery Corgi
- Music: Medieval Music Pack Vol. 2 – Alkakrab
- Cinemachine Camera Follower
- Evil Wizard – Luiz Melo
- Simple Fantasy GUI – Nayrissa
- Stone UI – Silena_Art
- Fantasy Menu SFX – Chris M Audio
- **Free Cinematic Sound Effects – Gregor Quendel**
- Non Unity Assets
- Game Over and Victory Sound Effects: https://www.fesliyanstudios.com/royalty-free-music/downloads-c/8-bit-music/6
- https://mixkit.co/free-sound-effects/game-over/
- **https://pixabay.com/sound-effects/evil-laugh-49831/**
- Script Inspiration
- MovingPlatform.cs: https://www.youtube.com/watch?v=GtX1p4cwYOc
- PlayerMovement.cs: https://www.youtube.com/watch?v=EVB1BAQIBxE
- MainMenu.cs Settings: https://www.youtube.com/watch?v=YOaYQrN1oYQ
- Script Usage
- TypeWriter.cs: https://github.com/rioter00/UnityExamples/blob/master/typewriterUI.cs
- **Player/BossHealthBar.cs:** https://www.youtube.com/watch?v=BLfNP4Sc_iA