Functional

**The game will be an rpg**

In Rpg games The main character usually has a level up system.

Leveling up in our game will increase various stats of the main characters like attack and defense. The main character will be able to level up by killing enemies and collecting experience in chests and in hidden items.

**The main character will be able to discover and use equipment.**

This equipment will be things like weapons armor or shields.

• and each form of equipment will have different stats or abilities.

**The main character will have a cone of vision.**

The main character will only react to objects within this visual cone.

And its up to the player to alter whether or not the character should continue his interaction.

**The player will use a secondary character**

The secondary character will be able to ﬂy freely around levels And use ﬂight to reach objects and gain a better view of the environments. The secondary character will also be able to shine a light.

And This light will alert the main character to objects or enemies. This light can also be used to support the main character in combat

how a few button commands would correspond with how the Light knight will act accordingly.

Non-Functional Requirements

Platform/Must run in Windows 10. Installing on a computer with a Windows 10 OS and running simple tests helps test this.

Minimum Framerate/ needs to be an average of 30 FPS, no less than 20 FPS. Can be monitored directly from Unity engine.

Response Time/ The average response time between click and reaction needs be less than 5 milliseconds. The maximum response time between click and reaction must be 2 seconds. A quick response time is needed to help keep the main character away from any danger that may need to be averted quickly, such as steering the character away from traps or properly aiding him in combat.

Required Resources/ The game should be able to run with minimum of 2 GB of RAM. The game must use less than 2 GB of hard disk space. Checking the total size of the folder in which the game was installed, for the hard disk space can test this requirement. To monitor the amount of RAM being used when playing the game, the physical memory in the Windows Task Manager can be used.

User Interface/ Needs to be clean and presentable, minimizing any confusion regarding health or locations on the map. All of the UI must be compatible on the Oculus screen

Characters’ portion

system

**(Discuss player’s character)**

**(Discuss and describe main character)**

Main Character AI character named Light Knight

Floating ball named Too that is the player controlled character that essentially acts as a cursor and has two commands: shine or yell.

Will not be able to move outside of certain range of main character.

Currently the only Enemy Character type we have so far is a

the actual visual design of these characters will be simplistic and cartoony. We made this decision to make the game more accessible to all ages and to the casual audience and also as a way to decrease overall graphical demand for most computers as a more realistic setting may require much more processing power.

Main Character AI will be based on a learning system that has set percentage values that fluctuate based on whether or not the consequence of an action is beneficial or not.

for example: If the ai starts with a percent value of 50 that decides whether or not it will open a chest. when the ai next opens a chest, if its not a trapped chest, the percentage chance of the ai opening another chest may later increase from 50 to 55. the plalyer will be able to perform 2 basic actions with their character in order to influence what the ai does and how these percentage chances increase or not, shine a light or yell. the light that the player can shine will call the knight to the player. if there is an object under the light the knight will interact with it. this is how the player teaches the ai what to do, or in other words this is how the player ensures that beneficial percentages rise. the yell function of the player character will do the oposite, yelling will tell the ai to stay away. this is how the player can teach the ai what not to do, for instance the player can yell when near a trapped chest and this will teach the ai not to open chests like the trapped one. this teaching is the main mechanic of the game, but if done improperly it can mess up the ais reasoning. for instance if the player fails to notice that a chest is trapped and calls the ai to it then the ai will start to think chests are bad, in other words the percentage chance that the ai will open chests, whether beneficial or not, will decrease. so far we have the working chests and doors with switches that the ai can interact with and learn to use.

**(Discuss enemy characters)**

Vr portion

(**Porting into the Vr integration)**

Will be using unity as well

**(Go into explaining what VR is and the different Vr devices)**

Google cardboard, GearVr, PlaystationVr, Oculus Rift, HTC Vive.

**(Explain that you’ll be using Oculus and why)**

No choice

**(Explain how the controls will work and why the decision was made for the control scheme chosen)**

**Normal locomotion vs teleportation method.**

In the Oculus VR version, the game will be virtually identical to the standard game where the game will be played in the same top-down, isometric view and the player will give the shine and yell commands in the same manner with the main exception being the movement of the player-controlled character. We’ve made a decision to make the player move via teleportation method, which is a control scheme where the player aims a beacon of light at a desired area with the controllers analog stick and releases the stick to teleport to that location. The reason that we’ve chosen this method of movement instead of the traditional locomotion that’s traditionally used in games is because currently, Vr as a whole is still in its early stages and normal locomotion has yet to be properly conﬁgured and gives most users motion sickness, so the teleportation method would be a safer, more convenient method. And its up to me to create proper arc angles to prevent any potential collision problems and to give proper warp distances and angles in order to help to game’s pacing on par with the non VR version’s speed.

All of the character models and objects designed in the base game are fully rendered 3D objects