Name: Oliver Reekie Student Number: 190000981

IN3036 Project Report

The Last Stand

Overview

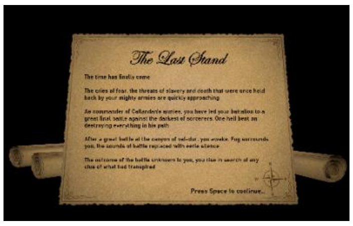
My game is a 3rd person fantasy RPG taking place in a medieval fantasy world. The game takes place after a great battle between the heroic armies of Calladan (led by our player character) and an army of monsters led by an evil sorcerer hell bent on destroying the world. The game begins with our hero waking up after the destruction of the battle not knowing what is going on. The player must first find a solider and after approaching one will be told that the evil sorcerer is hiding behind a wall of flames, casting a spell that will allow him to take over the world. This starts a timer, giving the player a matter of minutes to kill the giant NPC rats that litter the battlefield. Upon their deaths, the rats will drop souls that the player must quickly take. After collecting seven the player can run through the cursed flames to kill the sorcerer, which if the player runs through without enough souls, will set the player on fire. The player must now destroy the boss whilst dodging the boss's attacks before the time runs out.

To control the character, the player will use the WASD keys which allow the user to move forward, turn and strafe backwards when held down. The SHIFT key can also be used to sprint and escape enemies. The player will use the SPACE BAR to shoot plasma balls to damage his enemies, whilst dodging the rat's jumping attacks and the boss's own plasma balls. To help in this quest, there are health packs littered throughout the level, but to add to the challenge, the boss can also pick up these health packs and gain the health for himself.

There are three possible endings to the game: Victory, Defeat and Death. To win the game, the player must slay the boss before the time runs out. This will cause the screen to fill with a bright light which covers the screen, after which the game will become happier and sunnier, with the skybox being changed, the battle music replaced with birds tweeting and the flames and spell circle being removed. To be defeated and lose the game, the player will run out of time. In which case the spell circle will expand to cover the map and the screen will fade to a black lose screen with the ominous laughter of the sorcerer playing in the background. A death ending will occur if the player runs out of health, in which case the player will fall to the ground and the screen will fade to a black death screen, again with the sorcerers' laughter playing in the background.

Basic Game Modelling

Intro screen


The Last Stand

The time has finally come

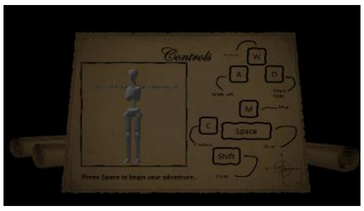
The cries of fear, the threats of slavery and death that were once held back by your mighty armies are quickly approaching

As commander of Callandan's armies, you have led your battalion to a great final battle against the darkest of sorcerers. One hell bent on destroying everything in his path

After a great battle at the canyon of sel-dar, you awake. Fog surrounds you, the sounds of battle replaced with eerie silence

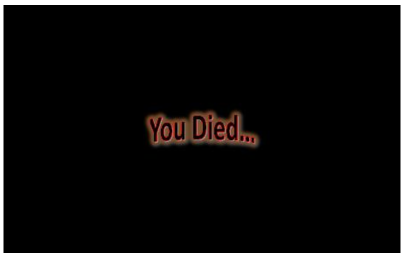
The outcome of the battle unknown to you, you rise in search of any clue of what had transpired

Press Space to continue...



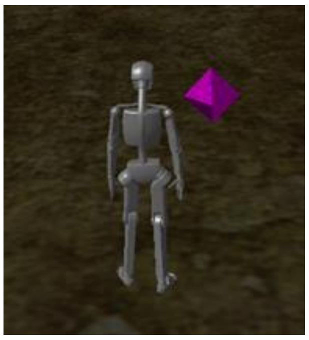
Press Space to begin your adventure...

My intro screen has been created using quad screen rendering, texturing the quad with a scroll asset that I edited with PowerPoint to add the relevant information. When the game begins the controls screen is showed (this can be shown throughout the game by pressing the 'C' key). This controls screen uses squiggly lines when drawing the controls to appear as if drawn on the map by hand, fitting the fantasy theme. The controls screen also contains a model of the mannequin acting as the player character. This is rendered using the orthographic camera and sits on top of the controls screen. The model is scaled to fit the map properly and continually rotates, showing the player who they are playing as.

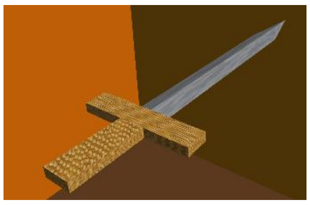


After closing the initial controls screen, an intro screen is shown. This was created with the same quad screen rendering and map texture to fit with the theme. This page gives details of the story and world to the player. This then cross fades out to present the idea of the commander waking up. The next screen is a quest screen which appears when the player approaches the fallen warrior. Like the screens before it, this uses quad screen rendering and a map texture, detailing the events of the battle and giving the player the first quest. Finally, I have created a death and lose screen which has a black screen with red writing stating that the loser has either lost or died. This is again created with quad screen rendering with this black screen textured on to it. All these screens are closed on a space bar press.

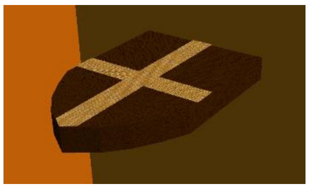
Primitive Objects

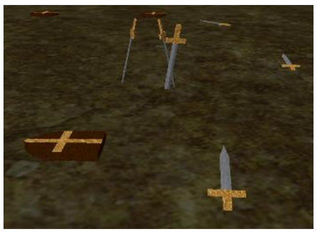


I have created three primitive objects using opengl in my game: Souls, Swords and Shields. All of these have correctly calculated normals. The souls are diamonds created by defining 6 coordinates and 8 triangles and will be dropped by monsters when killed by the player. They are spawned as pickups at specific locations with the function ‘soulSpawn()’.



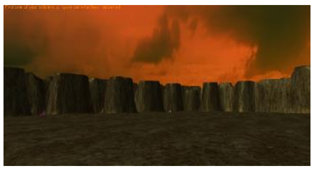
The swords are made of two parts, the blade and the hilt. The blade is made of 9 coordinates and 12 sides, creating a 3d blade shape. This is textured with a metal texture. The second section is the hilt which is made of 16 coordinates and 33 triangles. The hilt is textured with a gold texture and sits at the base of the blade. The swords are spawned in two different ways across the map. The first is flat which sees lots of blades scattered at different rotations and positions around the map. The second is upright, with the blades stabbed into the ground in sets of threes as if randomly struck into the ground after the battle. This was made to exemplify the post battle design.



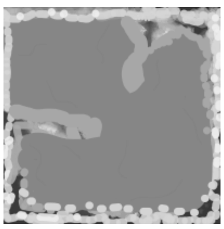


The final primitive is the shield, again made of two different sections. The base of the shield is made of 26 coordinates and 46 triangles and is textured with a wooden texture. The shield is curved towards a point at the bottom. To make the design more detailed, I have also created a trim. This trim was created with 12 coordinates and 8 triangles and creates a cross design on the shield textured with the same gold pattern as the sword hilt. This is a very famous shield design and makes the shield more distinct and recognisable to the player. The shields are littered around the map, like the flat shields, at random positions and rotations to add to the post battle atmosphere.

Skybox and Terrain



In my game I use two skyboxes, a gloomy sky that exists for most of the game to create an apocalyptic atmosphere. This skybox was extended to prevent clipping of my terrain. Later in my game, if the player wins and defeats the sorcerer, a blue skybox with white clouds is rendered to create a happy atmosphere of winning the game.



The terrain texture I use is of dirt and is used as the game is set in a canyon after a destructive battle. In order to create the two playable areas and cover the sides of the terrain in cliffs (to stop the player running off the bounds of the map) I have created a heightmap. I did this in Paint 3D and mapped it to the terrain. This not only marks the outer bounds but prevents the player from reaching the boss area whilst fitting with the setting. As well as creating cliffs the heightmap allowed me to create cracks in the ground to increase the detail in the environment. In order to achieve this, I prevented the player from walking through the walls (discussed in the physics section of this report).

Audio

My game contains 2 tracks, 5 events and 1 spatialised audio sounds. The two tracks are 'Background Music' which is a fantasy battle track to fit the theme and plays throughout most of the game, and 'Summer' which is a track filled with birds tweeting. On a victory condition the background music will pause, and summer will begin, to mark the end of the battle.

My 5 event sounds are split into the 'Zap', 'Laugh' and attack sounds. Zap is the electric sound that plays when the user fire as a plasma ball and lasts for the same amount of time as the plasma does. The laugh is played on either a defeat or death. This see's the background music paused, and a menacing laugh played for the user. The final event sound is the attack sounds which are played when the rat collides with the player, attacking it. To prevent the same monotonous sound being played again and again, annoying the player, I have included three attack sounds. When the player and rat collide, a random number is generated which is used to select one of the three audio events. All attack noises simulate growls to suit with the theme of giant rats. Having all three tracks as options like this make the game audio immersive, yet not annoying, to the player.

Finally, I have used a spatialised audio track called 'Ghost' which plays ghostly moans and wails. This event is looped every 6 seconds in the update method using a timer and the location of the audio is synced to the location of the spell circle. As the user approaches the spell circle, the ghostly moans will play which will create the idea that the sorcerer is using trapped souls to fuel his evil spell and circle. This will make the boss level spookier and more immersive. The ghost sound effect is stopped on a victory so as to not interfere with the summer sound effect.

Heads Up Display

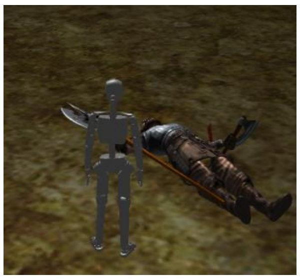
I have used four HUD elements in my game, two rendered with text and two rendered with images.

The two text rendered elements are the quests and the timer. Both are rendered with a fantasy font called 'The last kingdom font' in order to fit with the theme of the game. The quests are in orange and are along the top of the screen. The text is rendered at different positions for each quest to ensure they are always centred and is updated as the player completes objectives. The timer is in purple and sits on the right-hand side of the screen counting down the rest of the game. This appears after the player finds the fallen warrior and the first quest is completed.

The two image elements are the health bar and the soul holder. These are both created in PowerPoint, with each incremented part taken as an individual image. When the game begins, each of these images is created as its own instance of the soul holder or health bar class. On each update cycle the number of souls or current player health is examined, and the relevant HUD element is activated with the others being deactivated. The soul holder has images for each of the possible amounts, from 0 to 7 and the health bar has incremented images for every 5% of health. This allows for a quick and seamless transition of soul number and health (as the health decreases/increases in multiples of 5).

Camera, Meshes, Lighting and FX

Camera Motion Technique



After lots of testing with different motion techniques such as first person, or third person controls with a mouse (such as the type found in World of Warcraft) I have settled for a 3rd person experience with the camera set behind the head of the player. This camera movement is controlled by the player walking.

The player movement is activated when the player holds down a key, stopping the function when the key is released. The player moves forward when the W key is held and if the shift button is held down at the same time, sprinting begins, changing the player animation to the run animation and increasing the speed. The A and D keys are used to rotate the player around the y axis whilst the S is used to strafe backwards. This causes the player to sprint backwards very quickly for one second, counted down with a timer.

I have also added a map function that the player can use at any time by pressing the M key. This will change the camera to show a top-down view of the battlefield allowing the user a greater field of view to see objectives.

If the player dies, the player is rotated to fall to the ground and the camera moves around it whilst the screen fades to black.

Mesh Based Objects

In my game I have used 6 different mesh-based objects. The first is the mannequin model used for the player. This was found in the template code. The second is a fallen warrior model which has been scaled to look the best from the camera perspective. This is important to the game as the player must approach the warrior to be given the first quest. The third is the throne which is scaled and rotated to be looking at the player when it enters the boss area as if the boss were standing on it.

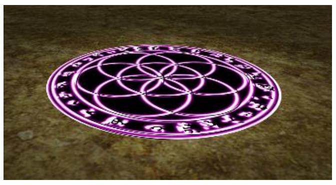
The fourth is the crosses which line the outside of the boss area, these are all scaled to the best size and then spawned using the 'crossSpawn()' method. This method transforms the crosses to given locations around the area as well as specific rotations meaning that the crosses are always pointed inwards, encircling the player.

The final two mesh-based objects are used for my NPC's. The boss mesh object is a skeletal creature (after testing multiple sorcerer models, this was the only one that worked in the software). The boss is scaled to be slightly bigger than the player and is spawned in the boss area in front of the throne. Its rotation varies on its actions, typically looking at the player unless it is running towards health packs.

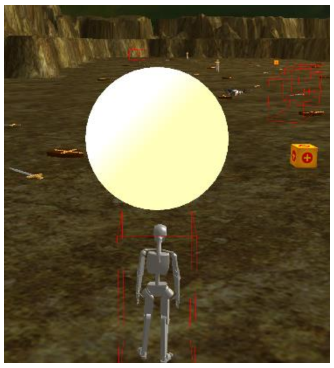
Finally, the rat NPC has been scaled to be giant compared to the player, with its rotation and position again being variable depending on its current actions. The model for the rat NPC has been edited in blender to look browner like a rat, increasing its realism. These rats are spawned multiple times at the beginning of the game and then are 'respawned' upon their death at one of multiple set spawn points.

Lighting Effects

My game contains three light effects as well as a slight dimming of the general lighting to make the game appear darker to fit the post battle theme of the game. I also found that the dimmer light made it easier to see the sides of the swords and shields making them appear more realistic.



The first lighting effect is the spotlight effect used on my spell circle. I did this to illuminate it more, to make the colour more prominent and for it to appear more magical. This spotlight sits above the spotlight, pointing towards the ground and was moved so that the light covers the whole spell circle.

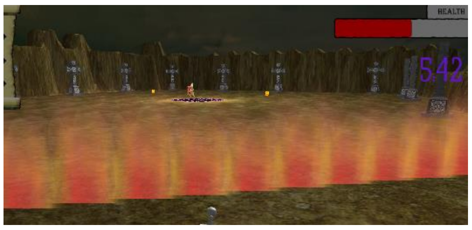


The second lighting effect is the point light used in the victory effect. When the player kills the boss, winning the game, a sphere is spawned in the centre of the screen with a point light at the centre. This has been set to have a slightly yellowed white light to give the sphere a glowing sun effect. This glowing sphere grows continuously to fill the screen. This light stays on afterwards making the environment more lit and sunny, which fits the then theme of a bright blue sky with birds singing.

The final effect is the point lights that I have used in my plasma projectiles. These plasma lights sit within the plasma and glow with a purple colour, moving with the plasma along its trajectory. Due to the limitations of the shaders to handle more than 8 point lights, I have reused the point lights. To do this, I created a vector holding timers for the point lights each set to 2 seconds (the amount of time for the plasma to fade out). When the timer for each light runs out, the colour is turned black (i.e., the light turns off) and the light is stored as able to be reused. The lights can then be reused and when a plasma is fired, rather than a new light then being created, the next available light is moved to the current plasma position and then switched back on. This allows the user to continuously fire glowing plasma balls despite the shader limitations.

Special Effects

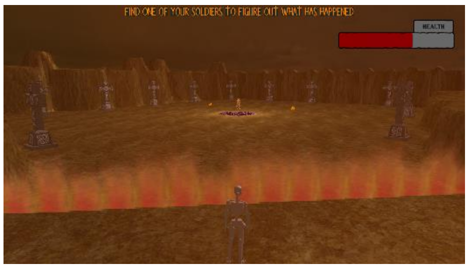
My game contains four primary special effects, the first and most important of which is my fire particle effect. The fire effect acts as a barrier at the entrance of the boss area and provides the difference between the battle with the rats and the boss. If the player attempts to cross the fire without collecting enough souls from the rats, the player will get shot back and set on fire (discussed more below). The fire is created with an emitter, creating particles along the line of the entrance. These particles are then created with a fire image that I created



myself. Each of these fire particles has a randomly generated height and, using a timer, the height increases and decreases, creating the effect of a wall of fire. The image is set to always be angled at the camera. Within the wall of fire is a series of blank balls with a transparency set to zero, these balls are used to add collision detection to the wall of fire, discussed in further detail in the physics section.

The second effect is a lightning effect used on the plasma ball that the player can fire. This effect creates a sphere with multiple randomly moving lines along its surface. This effect is used twice; to create a purple sphere that the player shoots and a red sphere that the boss shoots.

The third effect is a shockwave effect that is activated when the main game timer reaches 0:00. This has the radius of the spell circle increase over time, having the spell circle cover the ground, ending the game.

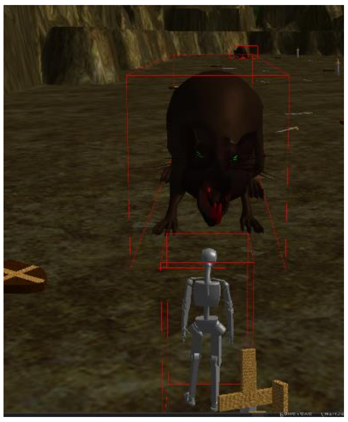


The fourth effect is a cross fade effect which is used multiple times in the game. Using a timer (considering timestep), the transparency of a quad can be increased or decreased consistently over time. This effect is first used when closing the intro screen, with the screen fading out to look as if the player is slowly waking up after the battle. Cross fade is then used on the death and lose screen, where the transparency increases slowly to have the screens slowly appear on the screen. Finally, cross fade is used when the player is on fire, with the transparency of the orange screen fading in and out as if the player is on fire. This effect is triggered when the player attempts to run through the wall of fire without having collected enough souls and lasts for 3 seconds, with the player slowly losing health over that time.

Physics, AI and Gameplay

Game Physics

In my game I have used three types of collision detection and physics. The first is used for the terrain, preventing the player from moving through cliffs, locking the player to the main game area. To accomplish this, I used the built-in method to return the height of a terrain at any given point. When the player attempts to move through a terrain that is too high or low, the movement is stopped, and the animation set to still. This creates a more believable environment.



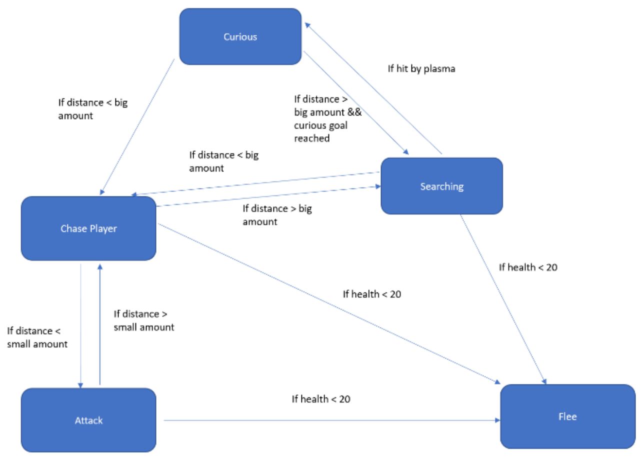
Secondly, is the collision detection for the health and soul powerups as well as the collision with the fallen warrior. Due to the limited number of powerups at any given time, I have had the program calculate the distance between the player and each power up on each update cycle. This is more efficient than using a physics engine as we do not need to know where or how the collision occurred, simply that the player is close enough to them. If this happens their effect can take place. This is also how the boss NPC interacts with health packs.

Finally, for the remainder of my physics I have used the bullet physics engine. This has the player, rats, boss, blank balls, plasma balls, boss plasma balls and terrain acting as game objects with physical properties and collision detection. The player, rats and boss all have a bounding box around them with the red lined box to display this collision area. Whilst the boss and player have no direct collision methods, the rats are considered melee opponents. Therefore, there is a collision method defined if a rat hits a player. When the rat is close enough to the player, it will jump in the air (pulled down by gravity) and will collide with the player, playing one of three rat attack sounds and damaging the player. To ensure this collision only happens once, the player can only take damage from a rat attack once every 0.25 seconds, I have found this provides the best experience both from a difficulty and realism point of view. One issue I encountered with the rats jumping at the player, was that the player was often knocked back off the map or is crushed into the ground. To prevent this, the mass of the rats has been set to 0.1 so they do not move the player unintentionally.

The fire effect that divides the main and boss area is lined with blank balls which act as collision detection for the fire. To stop them being moved on impact their mass is set to be extremely high. On a collision with the player, the player is shot back and into the air (falling due to gravity) and is set on fire. Upon the player collecting 7 souls however, the blank balls are removed so the player is free to cross the flames.

The most essential form of collision detection is in the plasma and boss plasma balls which act as the main weapons in the game. These balls are sphere game objects that are created with the same radius and position as the plasma effect, and they move in unison. As the plasma balls are energy they should not be affected by gravity. As such I have set their mass to zero meaning they are able to maintain their y position as they travel. However, the player will fire a lot of plasma balls over the game which eventually will cause the game to slow down as the number of collision tests per second increases. The plasma effect also fades and disappears after two seconds. To combat this, I have used the same idea as the lights in my game, where I use a timer to count down two seconds for each plasma ball, after which it is hidden below the map and is considered ready to be used again. When another plasma ball is required these available plasma balls are then returned to the player position and reactivated. This means I am reusing all my game objects and am not slowing down my game. There are collision methods set up for both plasma ball types, with the plasma ball causing damage to the rats and boss when they collide and the boss plasma ball causing damage to the player. One issue with collision is that the collisions are detected multiple times a second, this makes damage inconsistent and could even ruin the game leading to an instant player death. To ensure this doesn't happen, I have set each plasma ball to only cause a collision once (this is reset when the plasma ball is respawned).

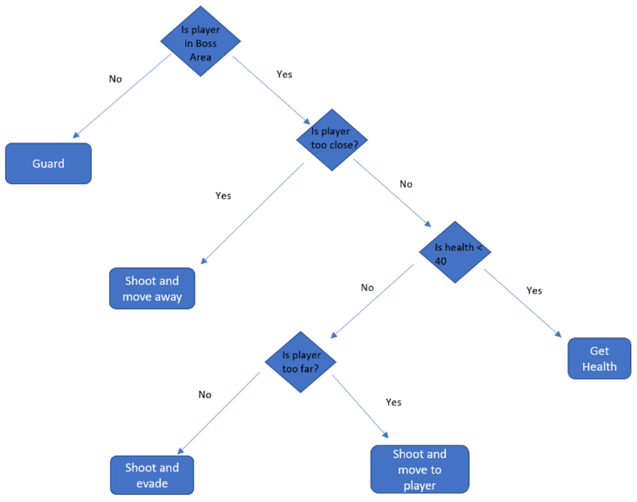
Non-Player Characters



My game contains two NPC types, the rats and the boss. The rats are encountered in the main game area and are controlled by the state machine shown below. They initially are set to search around the general area. This involves detecting the general area the rats are in and picking a random coordinate in that area for the rat to move towards. The rat will remain in this state until the player comes into a visible distance, or the rat is hit by a plasma ball. If the rat is hit, but the player is still not visible, the rat will enter the curious state and will slowly move towards the origin of the plasma that was shot. If during this or the searching state, the player comes into view, the rat will enter the chasing state and will towards the player. If the rat comes within an attacking distance of the player, it will enter the attack state. This has the rat jump into the air towards the player, colliding with the player. When the player and rat collide, a rat attack sound is played (see audio section) and the player loses health. As collisions can occur multiple times a second, I have used a delay timer to ensure damage is dealt slowly but consistently, without breaking the game.

At any point, if the rat's health falls below 20, the rat will enter the flee state and will run towards the closest of a set of flee points. If the rat reaches one of these points it will be removed without a soul being produced. This flee mechanic, combined with the fact that the rats run very quickly to the flee area, I have found, provides a more difficult but fun experience for the player.

My boss NPC is found in the second area and is therefore the second challenge the player must face. This NPC is created using the skeletal mesh object loaded in and is controlled using the decision tree shown below. The initial state of the boss is the guard state which see's the



boss standing in front of its throne in the boss area. Should the player leave the boss area, the boss will return to this state to guard and keep the fighting in this boss area. If the player is too close to the boss (a predefined distance), the boss will continue to look at the player whilst moving away. During this, the boss will shoot plasma balls at the player once every 2 seconds. If the player is not too close but the boss' health is less than 40, the boss will run to a health pack. It has been created this way so the boss will prioritise getting away from the player before running for health. If the boss has enough health and the player is quite far away, the boss will move towards the player while continually shooting. Finally, if the player is not to close or too far, the boss will maintain its distance, whilst evading to the left or right to make it harder for the player to hit it. This left or right distance is found using the cross product of the forward and up vector. The boss will randomly pick to run left or right and will do so for 2 seconds before deciding again. The boss will continuously shoot in this state. I have found that this design provides multiple types of boss movement and provides multiple challenges in this final fight.

Power-ups

My game level contains two types of powerups, souls and health packs. The health packs are created using the built-in quad method and textured using a texture I designed myself. The health packs are then spawned randomly throughout the map. In order to do this, I split the first area into two rectangles and used a random number generator to pick one of the areas (this was done due to the irregular shape of the first area). I then used a random number generator to pick random coordinates in the given area to spawn the health pack in. The health packs are set to rotate to look more interesting to the player. On each update cycle the distance to the player is calculated to detect collisions. This is discussed in the physics section. When the player reaches a health pack, the player's health increases by 20 and the health pack disappears, with a new one being created using the same random technique as with the initial spawning. To ensure there are always 3 health packs in the main game area and 2 in the boss area, I have also created boss health packs. These work in the same way as regular health packs; except they spawn at random coordinates in the boss area and the boss is also able to collect them. The distance to the boss is also calculated on each update and if the boss collects the powerup, it is given 40 more health and the pack disappears.

The soul powerups are integral to the first section of the game, where the player must collect 7 in order to progress to the boss area. The soul powerups are created with the diamond primitive shape and spawn when a rat is killed. When a rat dies a random number between 0 and 1 is picked. If this number is 0, a soul will spawn at the location where the rat died. This means a soul will spawn only half the time, increasing the game difficulty. Each soul also uses a timer meaning that they only exist in the game world for 1.5 seconds. This adds another extra challenge for the player and means the player cannot simply attack the rat from a great distance away. Like the health packs, the souls use a simple distance calculation to work out whether the player has hit them. If this is the case the soul is removed, and the player's number of souls is increased by one.

Timers

My game includes three main forms of timer, all created considering processor speed to ensure they all run correctly on any computer. These timers consist of: The main timer, The soul timer and the fire timer. The main timer begins after the player speaks to the fallen soldier and acts as our game's score. It counts down the rest of the game and if the player has not killed the sorcerer before the time runs out, a defeat animation will be called. This causes the spell circle that the sorcerer is casting the explode outwards, filling the map as the screen fades to black displaying a losing screen with the sorcerer laughing in the background. This timer is shown on the HUD and counts down by seconds.

The soul timer acts as a second challenge to the player and sees the souls only existing for 1.5 seconds after being dropped by the monsters. When a soul is created, it is added to a vector which holds references to them. At the same time, a vector holding the time remaining for each soul is created and updated on each update call meaning that each soul has its own timer which will remove the soul after a few seconds.

The fire timer is part of the set on fire effect detailed above. As part of this effect, the user needs to lose health spread proportionally over the effect's time to create the idea that the user is on fire. This timer is created when the effect begins and has the user lose 5 health at the end of each run. This timer will create another instance of itself when it runs out until 3 timers have been created. This means the user will lose 15 health over the course of the effect.

Smaller timers have also been created for use in the strafing method, which cause the player to strafe backwards for 1 second, and the boss shooting method. This means the boss is able to shoot a plasma ball once every second.

Evaluation

Overall, I am very proud with what my game level has achieved. I have worked to give the game multiple special effects, loaded in multiple meshes and have created detailed primitives to increase the realism of the game. I believe the game sits very successfully as a stand-alone level, with both the timer and two areas, with different NPC enemies, creating a lengthy and fun experience for the player.

One of the most successful parts of the game, in my opinion, is the sword and shield primitives that I created, which whilst taking a very long period of time, I believe make the game look more realistic. I also believe that my inclusion of effects on the victory or defeat of the player make the game more of a complete experience than just having the time run out or the boss killed with no follow up. Another key part of my game that make it more polished is the two types of enemy NPC's which provide two different challenges within the time that the game is played for. This also allows for two areas of different difficulty, with enemies that act in very different ways. Finally, I believe my use of audio has greatly increased the realism in the game. After adding thematically appropriate background music and special effects for the events in the game (especially multiple attack sounds for the rat NPC), I believe the game has become very immersive, improving the experience.

The two biggest challenges I faced in this project was in loading mesh assets and the time frame of the project. When downloading mesh assets, I found that many were in a .blend file format and so a great many of them I could not get to texture correctly. Due to my lack of experience with blender I was unable to convert most of the models correctly, keeping their texture. It therefore took a long time to find models that did work, changing slightly how the game ended up looking. The biggest change was that I was not able to get a sorcerer model to work so instead had to use a skeletal model for the evil sorcerer. This also proved to be an issue when attempting to load in animated files which did not work in the software. My game is set up such that it would be very easy to switch the mannequin out for another animated mesh, however I was unable to find one that worked. This also means that my rats and sorcerer are static models and so 'float' across the ground rather than moving in an animated and more realistic manner. In total this wasted a lot of time and therefore slowed the development of my project.

The other big problem was the time commitment. With the deadline being as soon as it is, and my having other coursework during this time, I was not able to have the game polished exactly as intended. If I had more time to work on the project, I would have liked to have implemented flocking in my rat NPC's. My idea was that if the player came into a rat's vision (where now the rat chases the player) and there are rats in the vicinity of the specific rat, then the rat in question would ‘call’ the others. These rats would then come together and flock towards the player, surrounding the player and attacking in a coordinated manner.

Another change I think would make a big improvement to the game given more time, would be to switch from the player shooting plasma balls to using a sword and shield. This would have the player carrying the sword primitive that I created and slashing it along the screen, with the sword using bullet physics to detect collision. The player could then hold down E which would cause the shield to appear in front of the player to block from enemy attacks. Whilst this would have been a great addition to the game, it would also take a very long time to implement as I would have to create the swords animations by hand using rotation of the object. Making this look realistic would be very difficult. Given the time commitments this was not possible to do, and whilst I am very proud of the mechanics my game uses, I also believe this would be a fun aspect to add to the game.

Finally, there is a limitation in the current software with the number of lights that can be used at any one time. I initially intended to place point lights in each of my souls to have them glow and look more magical. I did test this and whilst I was very happy with the results, the shader limitations meant having lights in each of the souls would mean I could not have them in all the plasma. This is a limitation with the software, but had it been able to handle more lights at once, I would have elected to have lights in my souls.

My level sits very well as the final level in a game, should it be expanded into a full game. This longer game experience can use much the same mechanics as this final level, simply in different environments and with different NPC’s. It could follow the war up until this point, with each level having multiple objects like this level has. Given more time and animated models that can run successfully, I believe I could use this level as the base to create a full game with multiple levels, even introducing multiple classes of characters such as mages (who shoot plasma balls) and warriors (who use the sword and shield mentioned above). This game would fallow the protagonist defending Calladan from the rise of the sorcerer, fighting hordes of his armies, culminating in his death in this final level.

Overall, in the time provided, I believe I have made a full and detailed game level which provides a challenging and immersive experience to the player. Whilst with more time I could have added more elements to the game, I believe the level sits firmly and polished as its own complete game.

Assets

Downloaded

Name	Date of Download	License	URL
Gloomy skybox	19/10/21	CC-BY 3.0	https://opengameart.org/content/gloomy-skybox
Blue skybox	11/10/21	Provided in template	http://www.vwall.it/wp-content/plugins/canvasio3dpro/inc/resource/cubeMaps/
Dirt Terrain	19/10/21	GPL 2.0	https://opengameart.org/content/filth-texture-set-trakbasegjpg
Soul texture	22/10/21	CC0	https://opengameart.org/content/seamless-looping-magicforcefield-effect-0
Spell circle texture	23/11/21	CC-BY 3.0	https://opengameart.org/content/colored-summoning-circles
Map texture	9/11/21	CC0	https://opengameart.org/content/map
Blade texture	15/11/21	CC0	https://opengameart.org/content/grey-metal
Hilt texture	15/11/21	CC-BY 3.0	https://opengameart.org/content/fabric-raised-gold-seamless-texture-with-normalmap
Soul holder texture	24/11/21	CC-BY-SA 3.0	https://opengameart.org/content/burnt-scroll
Throne mesh	24/11/21	Non-Commercial Use	https://sharecg.com/v/92659/browse/5/£D-Model/Birdmans-Chair-Throne
Shield wood texture	22/11/21	CC0	https://opengameart.org/content/dark-wood-seamless-handcrafted-texture
Fallen Warrior mesh	10/11/21	CC-BY 3.0	https://opengameart.org/content/brigand
Cross mesh	18/11/21	CC-BY-SA 3.0	https://opengameart.org/content/Celtic-cross
HUD fonts	20/11/21	Non-Commercial Use	https://1001freefonts.com/the-last-kingdom.font
Rat Mesh	30/11/21	CC-BY-SA 4.0	https://opengameart.org/content/rat-1
Boss mesh	30/11/21	Royalty Free Lighting	https://www.cgtrader.com/free-3d-models/character/fantasy/undead-anthropomorphic-lich
Background Music	7/12/21	CC0 1.0	https://freesound.org/people/Airwolf89/sounds/346455/
Attack1	7/12/21	CC0 1.0	https://freesound.org/people/angelkunev/sounds/560587/

Attack2	7/12/21	CC0 1.0	https://freesound.org/people/angelkunev/sounds/560589/
Attack3	7/12/21	CC0 1.0	https://freesound.org/people/angelkunev/sounds/560591/
Zap	7/12/21	Sampling Plus 1.0	https://freesound.org/people/DeadlyMustard/sounds/79643/
Ghost	7/12/21	CC0 1.0	https://freesound.org/people/Litruv/sounds/175944/
Summer	7/12/21	CC0 1.0	https://freesound.org/people/straget/sounds/403318/
Laugh	7/12/21	CC-BY 3.0	https://freesound.org/people/lalazzylee1/sounds/322459/

Produced

Name	Description	Software Used
Health pack texture	Used the built in PowerPoint shapes to draw a gradient square and a plus symbol drawn on the centre	Microsoft PowerPoint
Health bar	Drawn with PowerPoint squiggly lines and multiple built in red rectangles to build each amount of health in the bar	Microsoft PowerPoint
Fire texture	Created by setting a .png background to a shade of orange	Paint 3D
Heightmap	Created in paint 3D through drawing with different shades of grey to specify where cliffs should be to mark game areas	Paint 3D
Lose screen	Created a black screen and then used word style effects to create the 'You Lose' in the centre	Microsoft PowerPoint
Died screen	Created a black screen and then used word style effects to create the 'You Died' in the centre	Microsoft PowerPoint
Flame texture	Created using a blank .png in Paint 3D and then using a spray paint effect to draw reds, oranges and yellows to create a flame effect	Paint 3D
Soul holder	Created using the soul holder texture (as specified above) as a background and then placing screenshots of my in-game diamonds onto the screen with text in PowerPoint to create each stage image	Microsoft PowerPoint
Rat mesh	Using the rat mesh, I downloaded (see downloaded assets), I modified the fur colour of the rat from black to brown to make it look more realistic. I also changed the eye colour from green to red	Blender
Blank texture	Created by making a blank .png in Paint 3D	Paint 3D
Intro, Controls and Quest screens	Created using the map texture (as specified above) as a background and then using squiggly lines and text boxes to draw details on the screen	Microsoft PowerPoint