Game Development
COMP30540
Assignment 1
James Dorrian
13369451

Story Line:
I achieved all of the basic components of the game insofar as: The wizard has a shield and a magic power weapon, enemies spawn and fire TNT at the wizard and every now and again an enemy becomes mutated/goes berserk and charges the wizard.

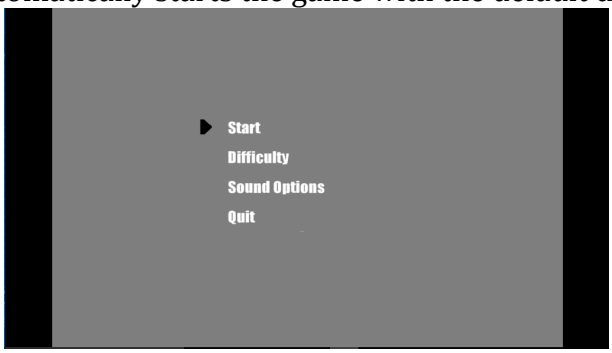
Controls:
Rotation of the wizard occurs when the left/right arrow keys or when the 'A'/'D' keys are applied. The wizard shoots using the spacebar and the magic weapon needs to be charged before use (sound effect included). I chose a fireball as my weapon of choice and added an appropriate explosion sound and animation on impact with an enemy.

Difficulty:
I offered 4 levels of difficulty from beginner to expert. The difficulty effects: spawn rate of enemies, speed of grenades, size of shield, magic weapon charge time and final point score. I made sure that I could pass all difficulties of the game before I finalized the weighting of these variables.

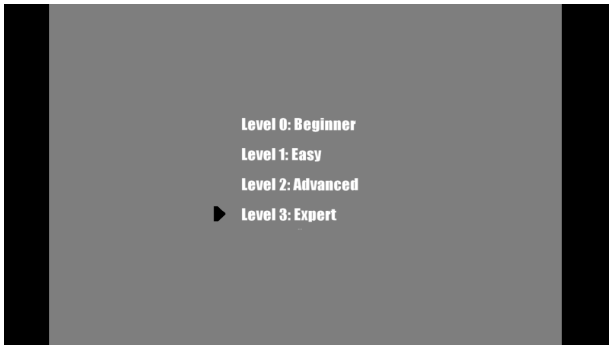
Embellishments:
I found nice creep animations online and a particularly menacing berserker (through some kind of mutation) but didn't concentrate too much time on the aesthetics, as I knew this was not the goal of the assignment. I added explosion and weapon charging sound effect. I added some simple crates as obstructions and configured my creeps to reposition themselves if their view of the hero is obstructed. I also configured the Berserkers to avoid walking into the crates on their way to the hero.

I also added a few embellishments of my own including: A score, a healthbar, a weapon powerup bar, a pause menu and a final points review menu that summarises your game once it is finished.

The menu system
When you start the game you are presented with a clean clear menu system depicted below: The 'Start' button automatically starts the game with the default difficulty beginner.



The 'Difficulty' option you into another menu where you can choose between 4 difficulties as depicted above.

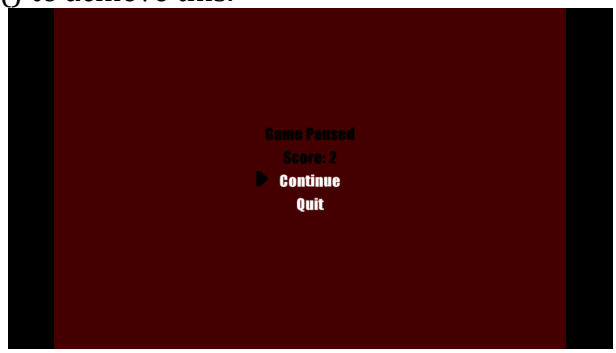


The ‘Options’ tab offers the user the option to turn on/off sound.

The game structure

I started with only two levels of difficulty and unlimited waves of zombies attacking the castle, but this quickly became boring, as the only way to end the game was to die. I wanted there to be some kind of ending but still make the game challenging. By offering 4 levels of difficulty and a small HP bar my aim was to avoid making the game too repetitive by keeping it challenging and using a point system to rank players. My aim was to make beginner easy enough for a young child and expert engaging and challenging enough for an adult.

All good single player games can be paused. I used instance_deactivate_all() and instance_activate_all() to achieve this.



Experiments:

I played around with a lot with things to do with the rate of spawn, number range of creeps, how often they go berserk and how often they fire grenades. I scaled the game in such a way that I could apply mathematical operations to these variables as the difficulty changes. For example: the rate of a creep going berserk is approximately 1/(3000/global.round_number) when compare this to the constant rate of TNT fire which is 1/350 it is easy to see how the early game is slow and controlled mainly by TNT fire while late game is all about avoiding berzerkers and having your weapon charged and at the ready.

The rate of spawn for creeps is monitored solely by the difficulty the user selects a counter determines how often a random number between 1 and 10 is called which in turns spawns a creep. In the beginner difficulty this counter is activated every 120 frames or 2 seconds. This is denoted in code by counter >= 240/(global.difficulty+1), as difficulty begins at 0.

Another interesting experiment was that I tried to implement a ‘boss’ system where after a random number of enemies were killed, a ‘boss’ was spawned. However, this made the game predictable so I opted instead for an algorithm which governs how often enemies turn into these bosses or ‘Berserkers’.

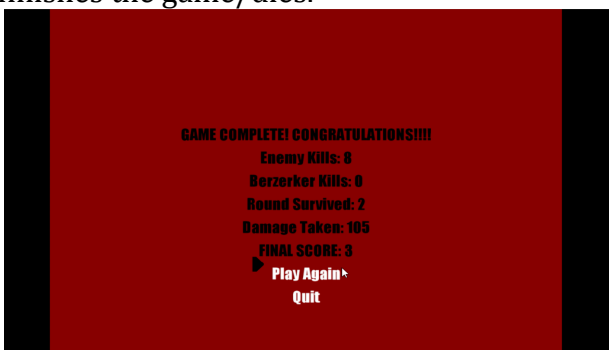
Obstacles:
One of the more challenging aspects of the game was implementing the obstacles that the creeps had to avoid. I made sure that enemies did not shoot unless their view was not obstructed. This was done using ray Tracing. When an enemies view was obstructed and they were in the outer ring, I sent them along the x-axis until they could fire with no obstruction.

To stop the berserkers from running through the crates I used mp_potential_step() which allows the object moving to avoid solid objects, in this case the crates and the walls of the castle.

Collisions, Inner & Outer Ring:
Originally I tried to use paths to create the outer ring, I had about 20 points that creeps spawned at (at the top of the screen) and then had them move down until their y co-ordinate matched one of the points on the path. The creep spawning looked too similar and the code was messy.

Instead I spawned the creeps and saved their x co-ordinate to a global array. Until the round is over no 2 creeps can be spawned within 32 pixels either side (64 pixel image). This allows a free and unpredictable spawn locations. When the objects moved down and reached a certain distance from the wizard (550px) they stopped, this formed the outer ring. An algorithm determining how often creeps go berserk could change an enemy's distance from wizard to 350, once they reach 350px the enemy mutated/went berserk and charges the hero.

How does it all end:
It ends in one of 2 ways, either the player is killed or finishes round 10. I based the game around and spawn mechanics on my favourite game as a child (call of duty world at war - Nazi zombies). It is a points based game where there is only an end when the nazi's have been killed, I decided upon 5 rounds + global.difficulty (so up to 8 rounds). This is the screen that is displayed if a player finishes the game/dies.



Conclusions:
There were many design considerations to keep in mind including spawn rates, round enemy algorithms, fire rates, shield sizes, health bars, score calculation etc. No single one of these things makes a great game but a particularly poor decision in any of these areas could make a poor game.

I endeavored to make the best game possible implementing the design considerations. I feel that I have learned the necessary skills to apply myself to the development of any game maker game. I concentrated my main efforts on the programming side of the game, rather than the graphics, sounds and animations, as they are all very simply implemented.