By: Akash Arun (Andrew i.d: akasha)

TP1 Project Proposal: “Escape from Ju-On 112 edition”

Contents:

Project Description:

My project idea is that of a quasi-3d horror game based on the movie grudge and I call it ‘’Escape from Ju-On 112-edition’’. The game starts with the player randomly spawned in a maze-like dungeon system without an exit which lacks an exit. Surrounded by creepy background music, the player’s only hope of escape from this hellscape is to collect all the paintings/ rings randomly found throughout the maze. Only after this may the player escape the maze. However, escaping the maze isn’t the player’s only problem. They must not be caught by Kayako (the grudge lady) who cannot be fought. Kayako randomly spawns every 50 seconds away from the player and moves towards the player and de-spawns after 40 seconds. With each successive ring/ painting collection, Kayako gets faster in terms of how she moves. During the initial parts of the game, the player may find that they can outrun Kayako but eventually running no longer remains an option. In such scenarios, there are hiding spots spread throughout the map, where Kayako will be unable to reach the player. These spots will become the player’s only haven during times when Kayako is searching for them. I may include various difficulty levels depending on which the number of items to collect, the size of the dungeon system, the spread of the items to collect as well as Kayako spawn interval would be appropriately changed.

Additional features that may be implemented include projectiles the player can shoot at alternate mob types such as cockroaches which could be spawned throughout the map, A Leader board of the highest number of rings/ paintings different players collected and a customizable “You’re Dead Screen” which can be altered to match the players own face.

Competitive Analysis:

Similar projects I've seen on the internet include a ray casted first-person shooter game resembling Doom. In which players have to shoot different enemy types and defeat all of the enemies to win the game. I aim to employ a similar type of ray casting techniques in my game for the walls as well as for sprites. I may also consider implementing projectiles for shooting mob enemy types such as cockroaches in a similar manner Post MVP.

I've also seen an implementation of Pacman in a 3d manner using ray casting with AI ghosts. I aim to use an A* pathfinding algorithm instead to make my Ju-On (Grudge lady) AI find the player and walk closer to them. The project also had a debugging mode which had a bird's eye view which I'd also like to implement.

Both of the projects I've mentioned above had static maps and enemy mob type positioning and to create an element of randomness I plan to use a modified version of a maze generation algorithm to generate changing dungeon maps.

Structural Plan:

The final product game will be in a folder containing different python files containing the various aspects of the game including the ray-casting algorithm, the dungeon-generation algorithm, the grudge pathfinding algorithm as well as the main game file which integrate each of these aspects. The folder would also contain the sprite images in a png/jpeg format for use in ray-casting as well as the various audio-tracks which make-up the background music, the grudge death rattle sound, sounds of the enemy mob's such as cockroaches as well as the sound of projectiles if used.

Algorithmic Plan:

There are 3 primary aspects of my project through which complexity may be achieved. The first is the ray-casting algorithm which is based on an algorithm called the DDA or digital differential analyzer. In this algorithm, I draw rays of different lengths such that they intersect the imaginary grid lines bordering cells in my dungeons. The ray length of the rays progressively increases until they hit a wall. Upon hitting a wall the length of the ray is used to calculate the distance the wall is away from the source of rays or in my case the player. This can be used to render the game. My algorithm is different from certain similar raycasting algorithms which use vectors as the main way of calculation meanwhile I use trigonometry primarily. I will use a similar approach to ray cast and create sprites and collectible items.

The second aspect is the dungeon generation which I achieve with a 4 step approach. First I generate a perfect maze using Prim's algorithm. But to create a proper dungeon structure I need an imperfect maze. To achieve this depending on the size of the maze I fill in a certain amount of dead ends. Then to create connections and loops I breakdown the remaining dead ends until they loop back with the maze. This leads to the successful creation of an imperfect maze. Then I spawn a set number of relatively large dungeon room templates throughout the maze at intervals such that it links up with the maze corridors. These room templates contain the hiding spots for the player to seek refuge from Ju-On. After all of this depending on whether I implement rings of painting I turn some of the pathway cells into rings or some of the wall cells bordering pathways into paintings. I will also randomly create a timed behavior of the maze to randomly spawn and despawn the grudge in a set time interval in a path far enough away from the player.

The third part of complexity comes from the grudge pathfinding algorithm which is implemented using an A* search algorithm to find the shortest path to the player and consequently shift the grudge AI towards the player in the direction of that shortest path until it reaches the player's cell which results in a game over sequence.

Post MVP, I plan on implementing a database for storing player information and user logins as well as a photo of the player. I may also implement a collision detection system if I plan on implementing projectiles to hit the enemies. This may not be too algorithmically complicated but I will compute the distance of the player away from the grudge AI and depending on how close she is to the player I will increase the loudness of the grudge track will simultaneously decreasing the background music and vice versa if the grudge goes away.

Timeline Plan:

I plan to have completed the raycasting as well as dungeon generation parts of my algorithm by the tp1 deadline and ideally integrate it with the collectibles and the grudge AI by the tp2 deadline to reach my MVP + basic UI. Post reaching MVP I will better implement the music as well as other parts such as projectile collisions, player databases, and image alteration.

Version Control Plan:

My code will be backed up to my google drive routinely with each version of files saved under a different folder as seen in the image below:

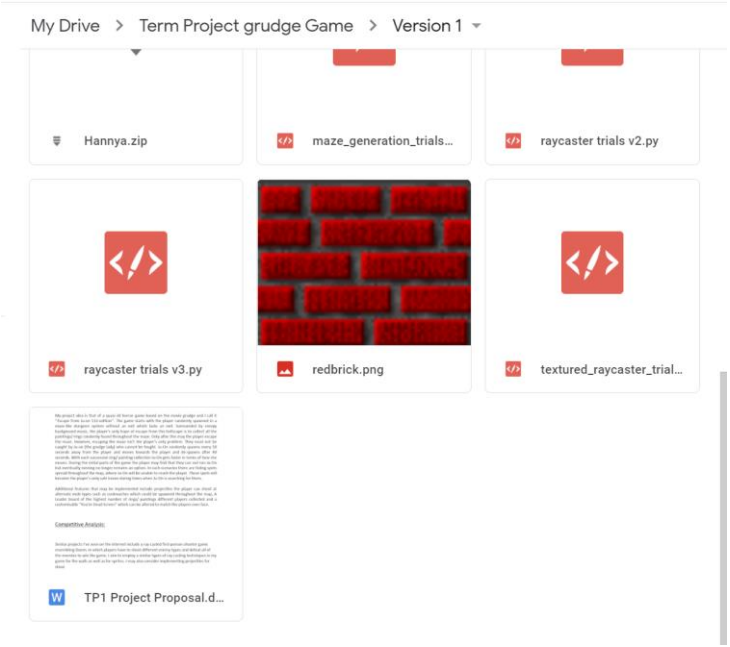

Figure 1: Version 1 files of tp in Google Drive

Module List:

Ideally, I don't plan on using any additional modules apart from pygame for audio (approved with tech demo) until MVP but I may consider adding other modules to improve parts of my project post MVP.

I also use math, random and time throughout the project.

Post MVP Update: I am also using the mysql.connector python module along with mysql.

Sources used so far:

Sound credits:

Hannya.wav from: https://www.youtube.com/watch?v=BZD9ZBQxusw&ab_channel=Kraosando

Grudge death rattle sound from: https://www.youtube.com/watch?v=-0v24Af1KhU&ab_channel=GhostlyHighway

mysql.connector from https://dev.mysql.com/downloads/connector/python/

pygame from https://www.pygame.org/wiki/GettingStarted

Cmu_112_graphics from: https://www.cs.cmu.edu/~112/notes/notes-animations-part2.html

Creepy Grudge background image:

https://www.google.com/url?sa=i&url=https%3A%2F%2Fhipwallpaper.com%2Fthe-grudge-desktop-backgrounds%2F&psig=AOvVaw3T_cZosjx8vwXled8Av8j3&ust=1607280213351000&source=images&cd=vfe&ved=0CA0QjhxqFwoTCNCU4cO_t-0CFQAAAAAdAAAAABAD

Creepy You Died Image:

https://www.google.com/url?sa=i&url=https%3A%2F%2Fbloody-disgusting.com%2Fmovie%2F3599114%2Fgrudge-sequels-explore-time-periods-exclusive%2F&psig=AOvVaw2p-EevFtHfCaAnrM0u49w6&ust=1607280188211000&source=images&cd=vfe&ved=0CAMQjB1qFwoTCNjPkum_t-0CFQAAAAAdAAAAABAJ

Cute you Win puppy image:

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.wallpaperflare.com%2Fthree-puppies-basket-dogs-pets-canine-domestic-domestic-animals-wallpaper-smqps&psig=AOvVaw3OHD-aUr66qabqjVohaQau&ust=1607280752429000&source=images&cd=vfe&ved=0CAMQjB1qFwoTCMD8p8rBt-0CFQAAAAAdAAAAABAD

Grudge sprite Image creds:

https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.vippng.com%2Fmaxp%2FhiRRomb%2F&psig=AOvVaw0pBS2Eh_4m-69bEIM96QN2&ust=1607280931144000&source=images&cd=vfe&ved=0CAMQjB1qFwoTCOjk-prCt-0CFQAAAAAdAAAAABAD

Ring Image creds:

http://pixelartmaker.com/art/5bbab14010c0068

Cockroach Image creds:

https://www.clipartmax.com/middle/m2H7N4A0Z5b1m2G6_this-high-quality-free-png-image-without-any-background-cockroach-png/

A star algorithm inspiration:

https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2

Ray casting inspiration:

3d sage's explanation in C: https://www.youtube.com/watch?v=gYRrGTC7GtA&t=423s

And https://lodev.org/cgtutor/raycasting.html

Sprite casting inspiration present TA Terry Feng's raycaster:

https://www.youtube.com/watch?v=PM4WyIsWJ_8&ab_channel=TerryFeng

Prim's algorithm inspired from:

https://medium.com/swlh/fun-with-python-1-maze-generator-931639b4fb7e#id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjJlMzAyNWYyNmI1OTVmOTZlYWM5MDdjYzJiOTQ3MTQyMmJjYWViOTMiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCJuYmYiOjE2MDY2NTg4ODEsImF1ZCI6IjIxNjI5NjAzNTgzNC1rMWs2cWUwNjBzMnRwMmEyamFtNGxqZGNtczAwc3R0Zy5hcHBzLmdvb2dsZXVzZXJjb250ZW50LmNvbSIsInN1YiI6IjEwMDc2NzkxODg2OTU3Nzc5ODI4NCIsImVtYWlsIjoiYWthc2guYXJ1bjIxMDFAZ21haWwuY29tIiwiZW1haWxfdmVyaWZpZWQiOnRydWUsImF6cCI6IjIxNjI5NjAzNTgzNC1rMWs2cWUwNjBzMnRwMmEyamFtNGxqZGNtczAwc3R0Zy5hcHBzLmdvb2dsZXVzZXJjb250ZW50LmNvbSIsIm5hbWUiOiJBa2FzaCBBcnVuIiwicGljdHVyZSI6Imh0dHBzOi8vbGg1Lmdvb2dsZXVzZXJjb250ZW50LmNvbS8tNlFZNTNsQkVLWjQvQUFBQUFBQUFBQUkvQUFBQUFBQUFBQUEvQU1adXVjbXVkSlh3dEc1c193dkRyNlNJdUVTczM2WjFqQS9zOTYtYy9waG90by5qcGciLCJnaXZlbl9uYW1lIjoiQWthc2giLCJmYW1pbHlfbmFtZSI6IkFydW4iLCJpYXQiOjE2MDY2NTkxODEsImV4cCI6MTYwNjY2Mjc4MSwianRpIjoiMWNiMTNlNDBmZGI5NmY2OTlmYmVjNmFiMDQxYTI1ZGVlMmU3OTQzNyJ9.l2HWvJBR3ewDeUKydefLwnsItvObc98vgHBwLJJc7Os8-58yceYpP1vcYMMWalXm3607bqgch2CSsK4mfN69W5yoFQhdkoqBXA4dHJXFESVT8WLqHJLmFmjyzyZgDJaOuR5geJWZCS3pvUKEgzip6NhiGWO59nNNZZe7Vy-72gW0GMjKfzYArV7eIGsd5TSskfDPznkUoWl7bYnM-q2_S2DPvI1SlVAxTqPAPfsL7brlHDbWVB9WwCC3ORnaFsxjC5Elye-B-VnbvuTcZzztb2VmCN_qMaffPoEzfGBgY2_rotRJWz9eftqUf7-J3oSvQ59eu4E6Zi3uA9K-WAk2XA

the general idea of how to create the dungeonified maze comes from:

http://www.brainycode.com/downloads/RandomDungeonGenerator.pdf