USING GRAPHS IN GAMES

# High Concept

This is a game implementation to how graphs are used in the stealth games. There will be 2 levels(rooms). The player will move across the room from the entry to exit without being detected. There are guard around in the room that have a detection zone. If you are to cross that, game over.

The guards move towards the noise so the player can use this to distract them to make its way

# Feature Set

* 2 levels that are pre generated
* A player that moves from point a to b
* Enemy
  + One or more
  + Have a detecting view that would be visible
  + Will note the point of noise and walk to it
* Game over if you are in the detection zone
* Player can create sound distractions to lure enemies away
* (optional) only the enemy closest to the noise will be distracted
* (optional) you have to collect some stuff in respective order and then exit the room

# Workflow

We need a room where the player and the enemy move.

The room needs to have an entry and an exit

The room will most probably need to be generated rather than created. As in the room will be generated based on some data in a file which the code will execute (saumya)

We need a player

The player either moves with the help of movement keys or click and path to that point (Kuntal)

We need an enemy (mudra and asavari)

1. The enemy moves on a set path. This path is decided when the game starts. The path updates to the point of sound. Once the path is complete the original path is restored
2. There is a detection zone attached to the enemy. If the player collides with the zone then the game is over/restarts.

We need a collectables system (shama)

4 to 5 things the player has to collect in order to unlock the exit

# Mood board

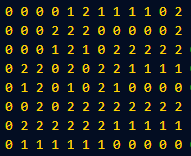
<https://www.youtube.com/watch?v=JOsolUFmB3c&app=desktop>



Logic and Guidelines

# Saumya

Create a map.py or just a normal .txt document and fill it with the grid for the level

example

Then create another file that will read this data(main.py for eg). Setup the pygame window. Use an image or just a plain square but make sure you know the dimensions. Create a method where you read all the grid data and save it in a list/array. Create another method where you use this list to place the images/squares in the location you have marked(eg load image on all the 0s in the above grid). We need the dimensions of the square because we will be creating a collision rectangle for each tile created. Or else the player wont be able to register the walls.

Make sure all the files are in the same directory and have been correctly included in the code.

# Kuntal

You have to do the movement of the player. For now just use a rectangle as a player and make sure the dimensions are smaller than the tile Saumya would be using for the map.

Setup the pygame window. Create a rectangle on the screen. Define a method that would take keyboard input and move the player accordingly. You window is like a quadrant in a graph so to move it would mean you add/subtract a defined speed value to the x/y cods. And that is the basic player movement setup.

Then you create a method or something idk. What you need to do now is take those collision rectangles of the map and make sure the player cannot walk over them. Create another rectangle/tile for testing purposes. Add a collision rectangle to it. And then make sure the player is stopped when trying to cross it.

# Mudra

You will have to create a detection zone which will return true if the player collides with it. Create a pygame window. Create a rectangle and call it enemy. Attach another long rectangle to it and let this be the detection zone. Note down its dimensions and create a collision rectangle to the detection rectangle. Create another box or circle and let that be the player. Use the movement script Kuntal would have created. Create a method where it checks if the player is colliding with the collision rectangle you had previously created. If it does, return true. (we will use this output to stop the game if this method returns true)

# Shama

Your work is very similar to mudra. Like we had done previously, we had made a portal for the player to reach and the game gets over. You have to do it almost like. Once that is done you will make a method that will check the order of the items collected and compare it to an existing list. If it matches return true.

# Asavari

Your part isn’t really easy to type and explain. I shall explain it to you on call. Basically all we have to use is the logic of the maze generation where it unloads from the stack to find the shortest distance to the end point.