The binding of shapes

A 2D roguelike shooter that is inspired by The Binding of Isaac. It would include randomly generated floors that the player can navigate and monsters for the player to fight against.

**Structural plan:**

Files:

1. Main.py
   1. Calls appStarted, redrawAll, timerFired, etc
2. Characters.py
   1. Class Player
   2. Class Monster for other monsters and bosses to inherit from
3. Util.py
   1. Includes utility functions/classes/constants that most file would use, such as:
   2. Class Timer and Vector2
   3. Functions that convert between row column to x, y pos
   4. Constants such as APPWIDTH, APPHEIGHT, etc
4. Shapes.py
   1. Includes basic shapes and is responsible for collision detection
   2. Class Rect and Circle
5. Controller.py
   1. Handles key and mouse presses
   2. Attempts to detect when the keys are actually released instead of key repetition
6. Generation.py
   1. Class Grid
   2. Responsible for generating a maze-like structure for player to navigate
7. Rooms.py
   1. Class DungeonRoom
   2. Each room is responsible for updating and drawing the monsters and obstacles in it
8. Projectiles.py
   1. Class Bullet that player and monsters shoot out
9. Obstacles.py
   1. Includes different types of obstacles such as rocks

**Algorithmic Plan**

* Generating the rooms
  + The rooms would be generated using a maze generation algorithm like Wilson in a NxN grid
  + Then choose a random point of maze and make it the start
  + Use BFS to traverse a fixed number of cells of the grids and make them into rooms
  + Pick furthest room (last room traversed) to be the boss room
* Drawing minimap
  + Use DFS or BFS to go to every room that the player have visited and draw them on minimap
* Pathfinding
  + Separate room into a grid to reduce run time
  + Every time player moves to new cell
    - Run djikstra and store each cell’s distance from player
  + Make enemies that follow player move towards nearby cell with min distance
* Circle to Rect collision detections
  + Find point on rect closes to center of circle
    - x = Median of r.x, c.x, r.x+r.w
    - y = median of r.y, c.y, r.y+r.h
  + check if point is in circle

**Version Control: Github**

Graphical user interface, application

Description automatically generated

**TP 2 Update**

No change