Project

100 points

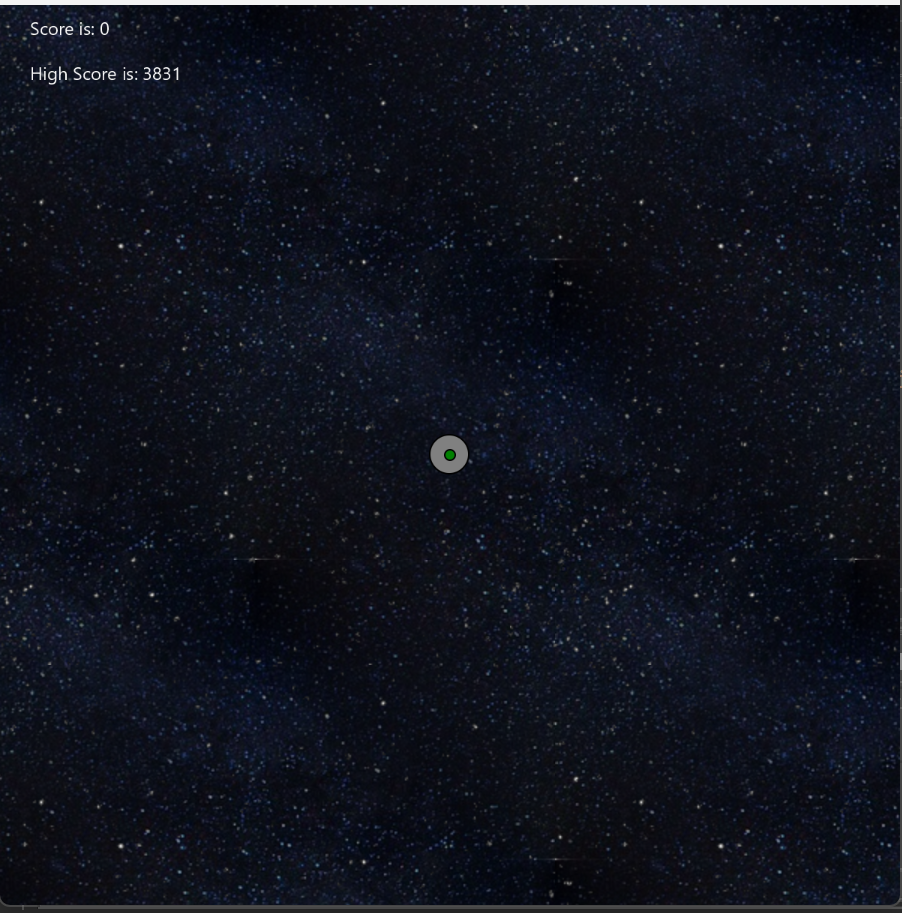
Turn in this project by uploading it to github and then submitting the link to your repository in Canvas.

Please find the starter code on canvas. You will see 3 classes, a Main class, an abstract DrawableObject and an example of inheriting from this class called TestObject and 2 picture files for the background.

I have given you a completed method to draw the background and a couple examples of calls to the draw methods from DrawableObject. There is a call to draw the background initially inside of the start() method, but you will eventually have to use the one in the animation timer once you have a player.

Create the Player

* For the drawing, you may make your own drawing as you see fit that should be about a 35x35 oval. You can make it bigger but the collision is by ovals. Make sure your shape has a black outline on each part to highlight the different parts.
  + This will be implemented in drawMe
* The Player should start at 300,300 (that is, the formulas in the code and elsewhere assume the player started at 300,300).
* You are using WASD to move with some modifications from what you have seen previously.
  + There is a “force” for x and a “force” for y.
  + Each “tick” should add .1 force to the player’s speed in that direction if the key is held (so if up was ‘w’ w as held for 20 ticks, the y speed would be -2). The max is 5 or -5 for each axis.
  + Each “tick” a key is not held down for that particular axis, the force is reduced by .025. (so it takes four times as long to slow down as speed up if a key is not pushed).
    - If a key is not pushed and -.25 < force < .25, set that force to 0.
  + This is suggested to be implemented in act. (you don’t have to use act but I suggest it as the place each object operates on its own functionality apart from draw.)
* Add a score that is to be displayed to the screen. You can also add a highscore placeholder too (completed later on). The score is the distance between where the player started (300,300) and position of player. It should be updated to match the player’s current position each tick.



Right here, you want to make sure you have the player working (moving) and the background moving around right. You shouldn’t have to modify the background method to get it to work (just pass in the player’s position to it).

Mines

* Create a Mine object that inherits from the DrawableObject. Similar to the player, it should have a drawing but a large part of the art should be something that oscillates between red and white overtime (so it should slowly interpolate between them as we did in class). Each mine’s color should effectively start in a random place between red and white.
* Each time the player gets to a new “grid” of 100x100, create up to N mines in each of the surrounding grids (of 3 or 4 away).
  + In the diagram, if P is you, X is the grids you create mines in. Note that the ones to the left/up of the player are 4 away and the ones to the right/down of the player are 3 away.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | X | X | X | X | X | X | X | X | X |
| X |  |  |  |  |  |  |  |  | X |
| X |  |  |  |  |  |  |  |  | X |
| X |  |  |  |  |  |  |  |  | X |
| X |  |  |  |  |  |  |  |  | X |
| X |  |  |  |  | P |  |  |  | X |
| X |  |  |  |  |  |  |  |  | X |
| X |  |  |  |  |  |  |  |  | X |
| X |  |  |  |  |  |  |  |  | X |
| X | X | X | X | X | X | X | X | X | X |

* The grid you are in is defined as  
   int cgridx = ((int)thePlayer.getX())/100;  
   int cgridy = ((int)thePlayer.getY())/100;
* The max amount of mines created at each X square is distance(the coord of gridx, gridy, to the coord of origin,origin)/1000. You have a 30% chance of creating a specific mine. (so if N is 3, then you have a 30% chance of creating each mine).
  + Please note that you will have to multiply cgridx and cgridy by 100 to get its coord. I.e., given a gridx of 10 and a gridy of 5, the position of that grid in the game is 10\*100, 5\*100.
  + This means that every 10 squares you go, you will have the potential of creating one additional mine in the next square. You create a mine 30% of the time.
  + There is an example implementation of the distance formula in the abstract class.
* When thePlayer is 800 units away from a mine, the mine should disappear.

Losing

* Assuming you sized things the same as my implementation (my player was about 14 pixels in radius and the mine is 6 pixels in radius), you would check if the player and a mine were within 20 units. When that happens, the player “hits” the mine. There is an implementation of the distance formula in the DrawableObject class that computes the distance between two DrawableObject.
* The game ends when player hits a mine.
  + Stop the screen from moving anymore and remove the player and the offending mine from being drawn.
* If the score was higher than the saved “highscore” then update the highscore to match the highest achieved score.
  + Use a file to keep track of the highscore; you cannot assume the file will exist the first time the game is played but after you play once you should create it.