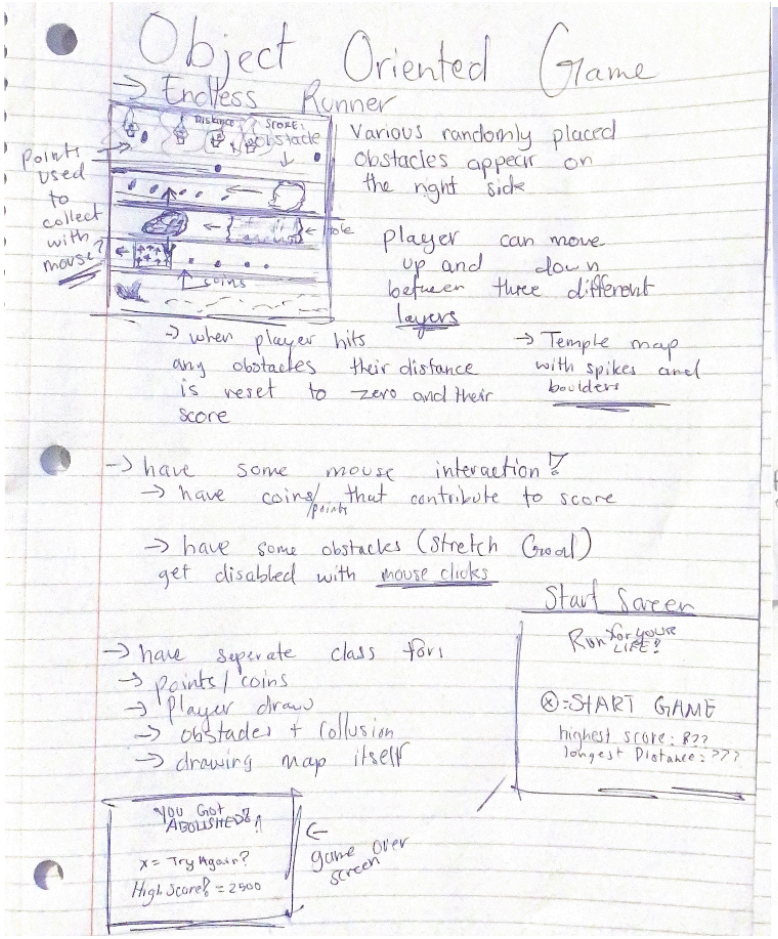
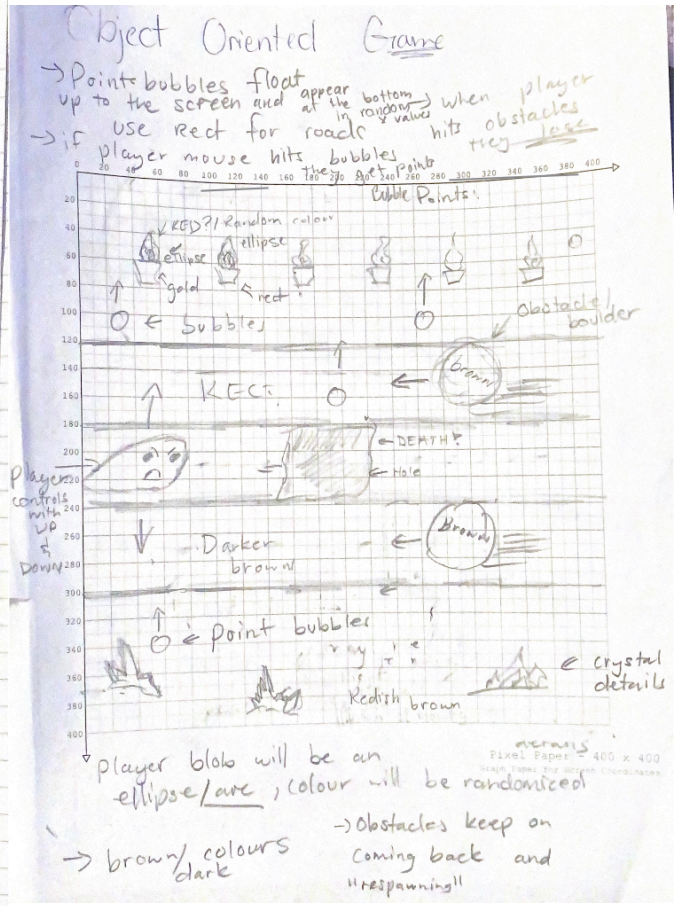
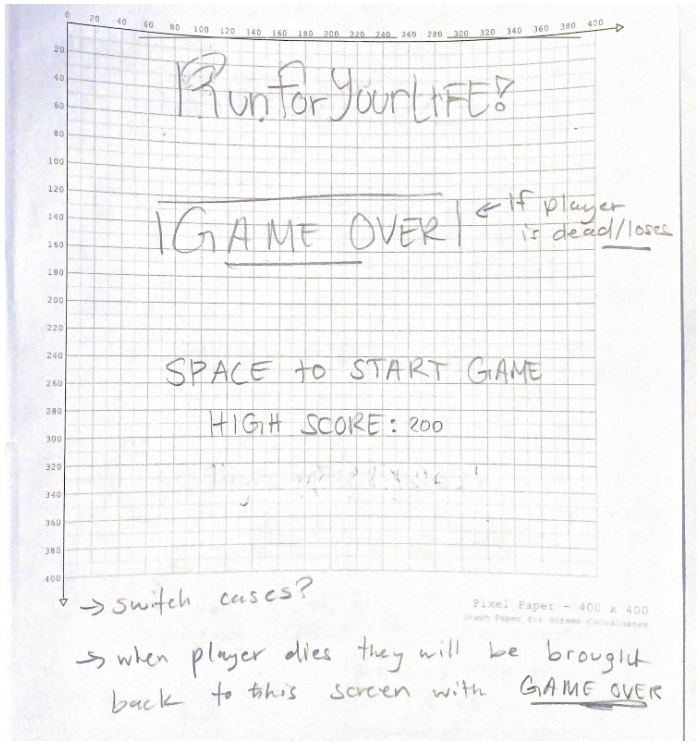
Abbas Haidari Object Oriented Plan

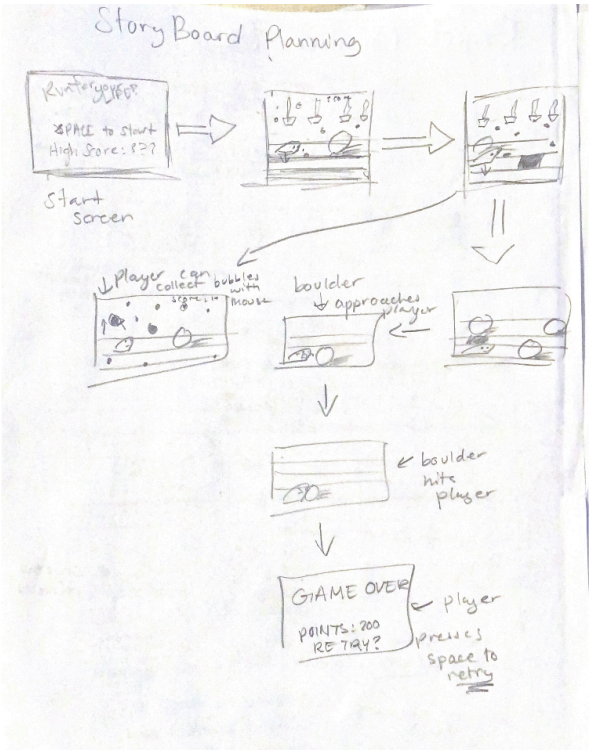
Use this to summarize your idea, plan it using sketches, notes and pseudocode as needed

Sketches and Notes









Pseudocode

Main Class:

- Create object class array for death pits
- Create object class array for boulders
- Create object class arraylist for bubbles
- Create object class for torches
- Create object class for lanes/layers
- Create object class for start screen
- Create object class for game over screen
- Create object class for the player blob avatar
- Make an object array for the obstacle class and set the max value as maybe 6 or 10?
- Make an object arraylist for the bubbles class

Stretch Goal: Make an array for the torches in the back and amplify the illusion of moving so it makes it look like the player themselves is moving left to right but they're just static

Setup:

Set size (400,400)

Have array for boulders in a for loop (set the amount)

Have array for death pits in a for loop (set the amount)

Draw:

Set background color here (brown?)

Call the road object and display and update the road

Call the player object to display and update the player

Have for loop for the arraylist to update and show the bubbles

Have for loop for the array to update and show boulders

Have for loop for the array to update and show pits

KeyPressed()

If the player presses UP

Update the Y axis of the player class up +1 ex. blob.updateLocation();

If the player presses DOWN

Update the X axis of the player class down -1. Ex. blob.updateLocation();

Start Screen class

Display start screen

Update start screen so that when the player presses X or starts the game starts.

Use switch cases for the start screen and break();

Game over class:

If player has died display game over screen

Use boolean to detect whether the player is alive or not

Player/blob Class

Create a constructor and have parameters for the color of the blob to set color in main class

Have a Pvector for the X and Y position of the blob

Have a float variables for both the width and height

Function to display blob

Center MODE
Ellipses for the blob
Ellipses for the eyes
Randomize color
Function to update blob
if (position.y>140) player cannot move more than the three layers
    Position.y+50?
if (position.y<2400)
    Position.y-50?
OR constrain position.y to (120, 300)

Have methods for **width, height, x and y positions** of the blob/player to calculate collision detection for the death pits and boulders in both classes

Boulder class
Call and create an object for the player class to use to detect player colliding with boulders
Use PVECTOR for the x and y positions and their velocity of the obstacles
Create Displays boulder function
Use ellipse and fill with brown
update functions for the obstacles
Get the x and y positions, height and width of the player. **If the player collides with the boulders, display the game over screen.**
If the boulders x position is less than -1 reset the x position from 400-490 randomly
Death Pits class
Call and create an object for the player class to use to detect player colliding with death pits
Use PVECTOR for the x and y positions and their velocity of the obstacles
Create Displays boulder function
Use rect and black to make display/visuals
update functions for the obstacles
Get the x and y positions, height and width of the player. If the player collides with the death pits, display the game over screen.

If the death pits X axis is less than -1 reset the x position from 400-490 randomly

Bubbles class
Use pvector for the x and y positions and velocity x and y of the obstacles
Have a variable for bubble points
Randomize and set the y positions from 400-440
Randomize x position from 1-400
Create display bubbles function
Text display for “Bubble points”
Use ellipse and no fill, with a bit of stroke()?
Update function for the bubbles
If the players mouse collides with the bubbles give the player one point to points variable and Reset the bubbles back to the bottom and randomize them again
If the bubbles Y axis number are less than 1 reset them back to 400

Torches class
Create constructor
Have parameters for color of torches and have color set in the main class parameter for torches class
Display function

- Use fill() for torch light color and with the same variable used from the constructor
- Have ellipses for torch light
- use fill() for the torch holder with a golden color
- Have rect for torch holder

Rocks/Crystal class
Create constructor
Have parameters for the color, width and height of the crystals
Have display function
Use fill() with the same variable for the constructor
Use rect() for the crystals and mash them up together

Where will the inventory skills be demonstrated? List every one to be sure you've included them.
Shapes:

1. Lines, Ellipses, Rectangles are probably going to be used to make the shape for each of the static images, blob avatar, and obstacles (boulders, spikes, holes)
2. Fill, stroke for the obstacles, blob avatar and background and other details for the main
3. CORNER (default) will probably be used for the whole assignment

System:

4. Setup and draw functions/methods in the main class
5. Background() for the background
6. Constrain maybe to constrain the spaces the player can move (they can only move between three layers)
7. Keypressed() for the player moving up and down
8. Incrementing and decrement variable for when player moves up and down between layers
9. For the player moving in the Y axis (up and down)
10. A variable for where the obstacles will be placed randomly maybe?
11. Debug whether the array list and arrays are working as intended
12. Collusion detection, if the player is detected in the space of the obstacles they lose and collusion detection for the bubbles to obtain points
13. For the collusion detection, if the players width and x are greater and less than the width and x of the obstacle they are reset
14. Again for collusion detection
15. Start screen/retry screen
16. To loop the obstacles (for loop) with an array possibly (in main class)
17. Possibly nested loop for the obstacles and array

18. Break(); for the starting screen

19. Will be answered in the repository text file

20. Different functions for drawing player avatar (eyes, body etc.) different functions for obstacles and the collusion

21. Function that gets/returns the player x and y, width etc. to use to detect when player has hit obstacles

22. The parameter is a variable that is listed in the parenthesis of the function, and the argument is where you initialize the value of the variables

23. Passing the position x and y, height and width

24. Taking the torches object and passing it through to move it throughout the screen

25. A object is an instance of a class, and a class is basically the blueprint for the objects. Objects have a certain state, behaviour and identity while classes are composed of methods and attributes.

26. It is a function that creates and initializes an object instance in a class. This basically allows you to initialize certain variables from another class for that object.

27. It keeps classes way more organized, rather then keeping them in the same one does not make sense and seems very messy. Furthermore, it does not make sense for a water class code to be in something like a wall, as they are two completely different things and have different behaviours.

28. Player class will have a constructor function, probably majority of the classes

29. This will be done for each of the classes

30. We can have parameters for the static images such as the road or the little rocks on the bottom of the screen to set the width or color of them, or even have it randomized. Maybe even have parameters for the blob character itself and randomize the color of the blob or the eyes.

31. Arrays are a fixed size that need to be set, while ArrayLists can be increased or shrink

32. It could possibly be used to reverse something, like if you have an image going from left to right, maybe decrementing it will make it go right to left.

33. ArrayList for the bubble points will be used and an array will be used for the obstacles

34. ArrayList for the bubble points

35. Bubble object will probably be managed in an arrayList

36. I will need to use an array list method for the obstacles, and since it needs to be in a loop, size or get() will be important

37. Pvectors are better to use when you want to use velocity or acceleration for a game or project, furthermore they can store two numbers in that one class, for example for a float for a triangle, you will need to initialize a variable for both the x and y coordinates for that triangle which is just extra lines of code.

38. The PVector class will be used for the x and y positions of the obstacles and their velocity

39. Position and Velocity for the obstacles and bubbles, how fast they move and the direction (right to left) they will go etc.

40. Collision has to find the distance between two points for it to work efficiently

41. Randomizing the bubbles and the obstacles where they are positioned.

Milestone 1	**Milestone 2**	**Milestone 3**	**Milestone 4**

What will I deliver? Make the three lanes for the game Make the player movement for the game Create Start screen	Bubbles and Boulder and Pits ArrayLists and Arrays Player detection/collision	You are strongly encouraged to deliver your finished game at Milestone 3. Finish Game off, and test and see whether it meets all requirements, efficient code and works as intended	

Which inventory skills will this demonstrate? List them. 1. Lines, Ellipses, Rectangles are probably going to be used to make the shape for each of the static images, blob avatar, and obstacles (boulders, pits)	16. To loop the obstacles (for loop) with an array possibly (in main class) 17. Possibly nested loop for the obstacles and array 18. Break(); for the starting screen 20. Different functions for drawing player avatar (eyes, body etc.) different functions for obstacles and the collusion 21. Function that gets/returns the player x and y, width etc. to use to detect when player has hit obstacles 22. The parameter is a variable that is listed in the parenthesis of the function, and the argument is where you initialize the value of the variables 28. Player class will have a constructor function, probably majority of the classes 29. This will be done for each of the classes 30. We can have parameters for the static images such as the road or the little rocks on the bottom of the screen to set the width or color of them, or even have it	35. Bubble object will probably be managed in an arrayList 36. I will need to use an array list method for the obstacles, and since it needs to be in a loop, size or get() will be important 37. Pvectors are better to use when you want to use velocity or acceleration for a game or project, furthermore they can store two numbers in that one class, for example for a float for a triangle, you will need to initialize a variable for both the x and y coordinates for that triangle which is just extra lines of code. 38. The PVector class will be used for the x and y positions of the obstacles and their velocity 39. Position and Velocity for the obstacles and bubbles, how fast they move and the direction (right to left) they will go etc. 40. Collision has to find the distance between two points for it to work efficiently 41. Randomizing the bubbles and the obstacles where they are positioned.	

2. Fill, stroke for the obstacles, blob avatar and background and other details for the main			
3 CORNER (default) will probably be used for the whole assignment			
4. Setup and draw functions/ methods in the main class			
5. Background() for the background			
6 Constrain maybe to constrain the spaces the player can move (they can only move between three layers)			
7. Keypressed() for the player moving up and down			

8. Incrementing and decrement variable for when player moves up and down between layers 9. For the player moving in the Y axis (up and down) 10. A variable for where the obstacles will be placed randomly maybe? 11. Debug whether the array list and arrays are working as intended 12. Collusion detection, if the player is detected in the space of the obstacles they lose and collusion detection for the bubbles to obtain points 13. For the collusion detection, if the players width and x are greater and less than the width and x of the obstacle they are reset 14. Again for collusion detection 15. Start screen/retry screen			

	You should deliver approx. 10 skills at this milestone	**You must deliver 30 inventory skills by this milestone.**	