Subject: 15-112 Term Project Proposal
From: Richelle Guice

Project: Mini Dimensional Traveler

Description

In this game, the user will navigate through a randomized maze to reach an exit. However, while traversing there will be obstacles the player must be aware of and be able to pass. Obstacles include spikes, refiring missiles, and a chasing lava. As the player is traversing, there will be coins to collect to up their score as well different levels with increasing difficulty. The objective of the game is to reach a portal that will take the player to safety.

Competitive Analysis

This game is similar to some other projects I have seen online. For example, Adhvik's Primerunner game. He also had what seemed like a randomized maze where a player navigates through and picks up things and also avoids obstacles. Furthermore, his project also implemented side-scrolling which will be a big component in my project.

Another example that I have seen which I feel is a little similar to mine, is Anesha's 'Survival of the Fittest' game. Every time the player plays, there is also a randomized maze that the user must traverse through. Here, the user has to pick up survival items to keep their health up or else they die. While a completely different function, it seems to be a very similar implication to mine where I have to put obstacles the user must avoid.

Structural Plan

One function will be the appStarted that sets all the base values. One main function for generating the randomized maze. Another main function for sidescrolling with the player. For allowing the player to actually move, there will be multiple functions for: moving the playerPiece, seeing if the move is valid, and then drawing where the player currently is. There is obviously a keyPressed function for allowing the player to move as they wish. There will be another function for generating obstacles and another function for obstacle collision.

Algorithmic Plan

The first trickiest part of my game is implementing a randomized maze that will account for obstacles and allow for my character to move through in a timely manner. This is not just a game where the object is to be able to simply navigate through a randomized maze and find an exit. Instead, it must be playable and fun and also have an "action" feel. The user moves in a "hard" direction. For example, if pressed down, it moves immediately to the nearest wall that is eligible. To tackle this problem I first set up the board as rows and cols. For now, I set a baseline number of columns (20) just for easier debugging purposes.

1. Board is set up as rows and cols
2. Randomize by how many rows there will be a row of wall
    a. Randomized a number between (2, 3)
    b. For every row, there has to be at least one cell space to allow the character to move thus randomize an exitSpace(1, 3)
    c. Randomize by how many cols the row of wall will reach while accounting for space for the character to move
3. Right side of wall is just from start to randomized length of col
4. Left side of wall is ^ + randomized exitSpace
5. Issue of allowing player to be able to bounce off of walls: Using an imaginary player, set its column (iCol) always at the middle of board to play through the randomized maze of rows then to draw additional vertical walls
6. Figure out where there is gap: wallCol → start of gap, gapSpace → app.exitSpace
7. Three cases:
    a. If iCol on gap → nothing, imaginary player can fall through
    b. If gap is on the left of imaginary player→ add block above (row-1) the left wall
    c. if gap is on right of imaginary player→ add block above right wall ONLY if in bounds
8. At end row+=1 to keep iterating through entirety of board

Second most trickiest part in coding my game was placing spikes that would make the game harder but also ensure that it was not unplayable. Taking a look at my code for my randomized maze, I decided to place spikes where the imaginary player could not go.

So:

1. Calculate the length of the opposite side of wallRow
    a. oppSideD = app.cols - (app.wallCol[i] - app.exitSpace[i])
2. If it was != 0, place a spike above that wallRow (row-1) on the opposite side of vertical maze wallCol

Moving forward: I had to figure out how to make the maze more sophisticated by adding free-floating maze walls. To place them, the algorithm had to understand when the was an empy space of 3x3 cells at least to be able to place a floating maze wall.

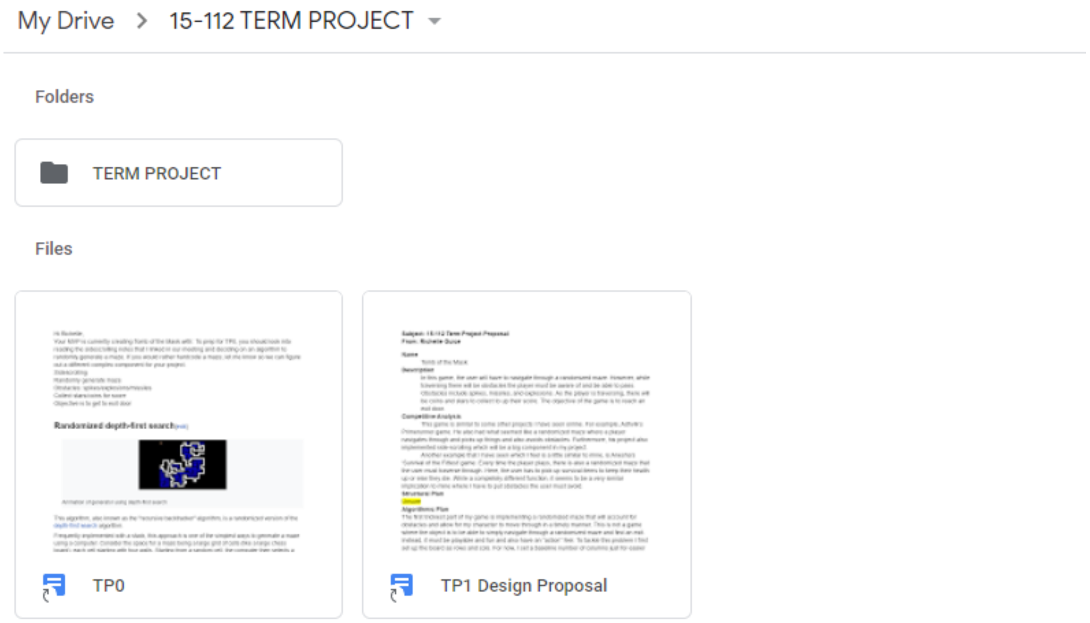
1. Using oppSideD, if it was >= 3 and on that iteration of the loop from app.wallRow (it is a list of the distance between every row iteration) if app.wallRow[i] == 3:
2. I used a while a while loop to draw floating maze walls in the case the empty space was 3x4 or 3x5 or 3x6 etc...

Timeline Plan

Feature:	Date:
Randomized maze	4-26-2021
Character on-screen and movable	4-26-2021
Obstacles and obstacle collision	5-01-2021
Coins and stars and show score	5-01-2021
On-screen design improvement	5-01-2021
Sidescrolling	5-01-2021
Additional design changes	5-03-2021

Version Control Plan

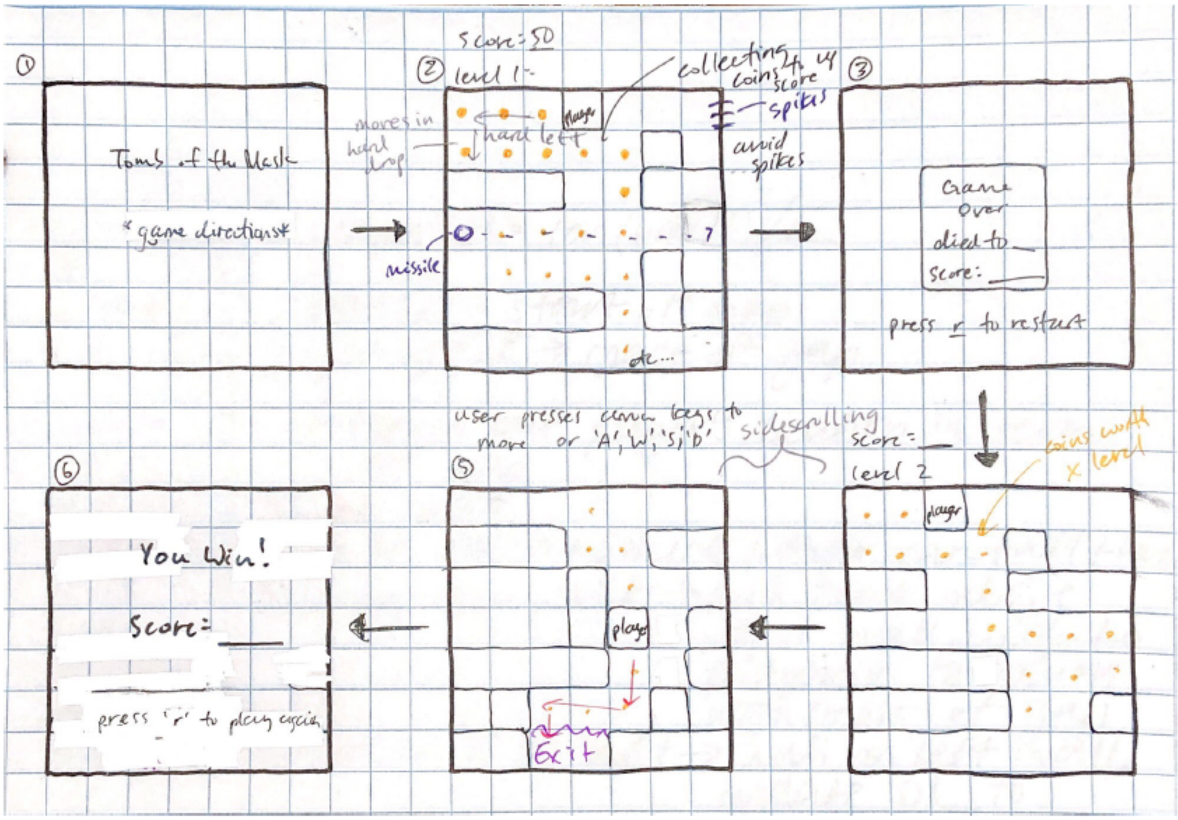
I copy all my files to google drive after every change.



Module List

- I do not need external modules/hardware/technologies

Storyboard


①
Tomb of the Mask
game directions
Score: 50
② level 1:
player
moves in hard left
hard drop
missile
collecting coins 4 score
spikes
avoid spikes
etc...
③
Game Over
died to ___
Score: ___
press r to restart
user presses arrow keys to move or 'A', 'W', 'S', 'D'
sidescrolling
Score = ___
level 2
player
coins worth x level
⑤
player
Exit
⑥
You Win!
Score = ___
press 'r' to play again

TP2 Update

Regarding design changes made, I switched up the colors so the game seems more interesting and also make the maze appear as obvious. Compared to TP1, my randomized maze now has free-floating maze walls to not only sophisticate my maze but also make the user experience more fun and competitive. Furthermore, in addition to different levels, the game now has increasing levels of difficulty with every progression in level. For example, the missiles will shoot in a random order after the user successfully passes Level 1.

TP 3 Update

Further design changes I made were implementing a pause/play button for better flow in user experience. This way, the user could pause the game in case they were busy or wanted to come back to it. I also changed up the colors once the player won or died. Regarding additional changes to the fundamentals of the game, I incorporated a chasing lava that would force the user to have to think quick, making the game a little more difficult while still keeping the game enjoyable. For every increase in level, coins became worth that many times more per coin collected as well as missiles shot faster. So for example, if it is level 2: the coin is now worth 2 points instead of just 1 point. In addition, as the levels increased, the missiles refired at a faster rate per level.