Term project Proposal

Project Description

This project is a side scroller game called Go Disney Princess.

Background story
The world is fighting a scary virus that caused a lot of death. To save the world, three brave Disney princesses formed a squad to get the antidote in a castle guarded by a ferocious dragon. On the adventurous way to the castle, they have to fight different enemies and overcome many obstacles; but they work unitedly as a team using the different abilities that they have, and finally reach the destination and beat the dragon.

How it works
In the game, players can play the roles of three Disney princesses: Mulan, Elsa, and Jasmine. Each princess has her own actions: Mulan can attack the enemies with the sword, Elsa can freeze the enemies with her magic, and Jasmine can fly over the obstacles with her magic carpet. All of the princesses can walk, run and jump, and shoot balls to enemies. Players can switch characters at any point, they have to strategically decide which princess to choose given the different challenges that they encounter. The worlds are full of enemies and platforms, and obstacles. The players have three lives, they can collect lives along the way as makeups for the lost ones, they will lose the game if they lose all of their lives. At the end of the game when they are facing the Axe enemy, the different powers will combine as one, so players can use the different superpowers without switching characters.

Competitive Analysis

I explored some side scroller games online including Super Mario, DONKEY KONG COUNTRY Number 10, and also looked at some 112 projects that use pygame.

Similarity: My game will involve some basic features in traditional platformers and side scroller games such as jumping between platformers, collecting rewards, and fighting enemies.

Difference: My game has a special theme about Disney princesses that no one has done before and the background story is interesting. There is also a special feature, which is that players can switch to monitor different Disney princesses that have different powers at any time during the game. What's more, unlike most games that have different modes, this game has only one mode and the difficulty level will increase as the player moves right.

Structural Plan

My game will include three levels. As the player moves right, the game will become more and more difficult.

I will use three files to organize the code and link them together by importing one from another.

1. **Settings.py** is used to store fundamental information such as width and height of the window, width and height of the objects, colors, , images, etc.
2. **main.py** that contains the main game code
   Game Class
   # initialize the game
   new()
   run()
   # update the screen
   update()
   # loop through all the events
   event()
   # draw the sprites
   draw()

3. **Sprites.py** that contains all the sprites classes
   Princesses
   Upper class: Princess
   Inherited class 1: Mulan
   Inherited class 2: Elsa
   Inherited class 3: Jasmine
   Enemies
   Upper class: Enemy
   Inherited class 1: QuickEnemy
   Inherited class 2: FlyingEnemy
   Inherited class 3: Dragon

Object
Platform
Award
Sword
Carpet
Ice
Fireball
Ball
Axe

I will also create a folder to put all the images, and a folder to put all the sound files.

Algorithmic Plan

Level 1: Jumping, collecting rewards, and fighting enemies, moving forward, platforms will be shorter and the player will die if fall down.
Level 2: Flying enemies that will attack the player with fire, and can dodge attack
Boss: Fiery dragon, can chase the player, can fly, can heal itself, can shoot fireball.

More complex parts:

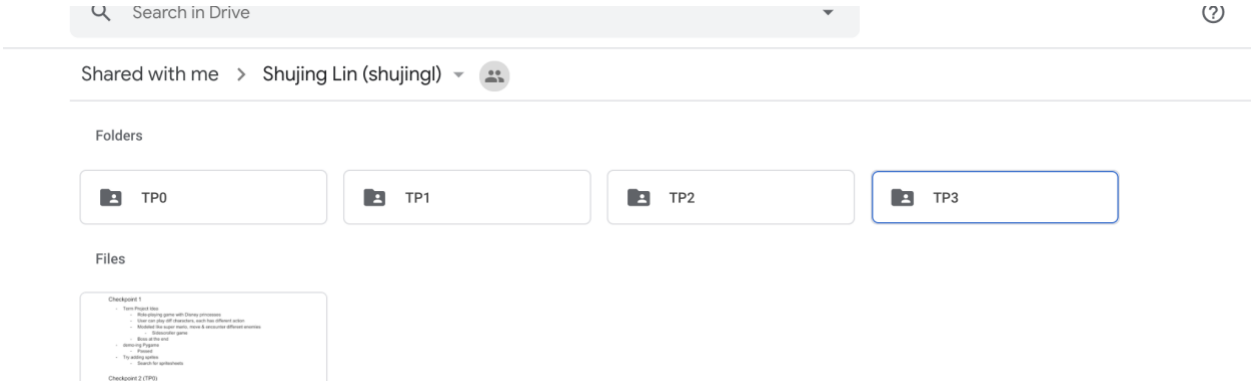
Boss finding the shortest path to reach the player
Boss predicting player movement
Boss shooting fireball to the player in different angles

Timeline Plan

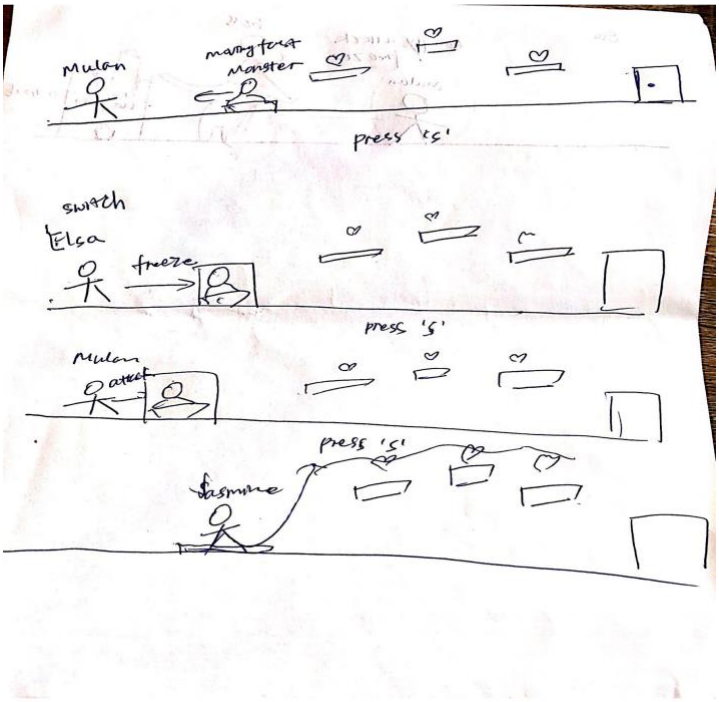
Time	CP4/TP1 April 16	CP5 April 20	CP6/TP2 April 23	CP7 April 27	TP3 April 29
Due	Jumping Collecting Fighting Switching roles	Side scrolling Include All the levels	Finalize all the levels	Add sounds More Intelligent Game AI	Video Demo Final product

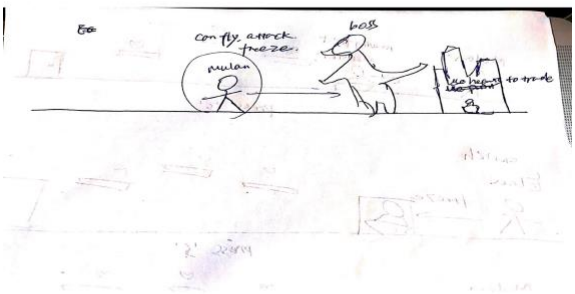
Version Control Plan

I will save all my work for each checkpoint into google drive.



Storyboardx4





TP3 Update

Version Control Plan

I decided to use Github to synchronize my code as I make progress.

https://github.com/shujingl-2020/go_disney_princess

Structural Plan

Level 1: Jumping, collecting rewards, bypassing obstacles, and fighting soldier enemies.

Level 2: Flying enemies that will attack the player at any angles with fire. Dragon enemy that can spit fire to the player and heal itself.

Level 3: The Axe Enemy can jump and move between platforms to and find the shortest path to chase the player. Platform maps change constantly. Player should beat the final boss (Axe Enemy) to win the game.

Algorithmic Plan

Most complex algorithms:

Breadth First Search Algorithm to find the shortest path:

Purpose: enemy can detect where the platform is and move between platforms to find the shortest path to reach the player.

Create a matrix with 1 being the path and 0 being the obstacles

```
# get the starting point
getSource(matrix)

# get the target point
getDestination(matrix):

# get all the points that are adjacent to a node using the direction list
getNeighbors(currentCell, matrix, visited):
direction = [(-1, -1), (0, -1), (1, -1),
             (-1, 0), (0, 0), (1, 0),
             (-1, 1), (0, -1), (1, 1)]

# avoid a cell being visited twice
getVisited(r, c):

# using queue to do the breadth first search
solve(matrix):

# keep track of the parent node
getPrev (r, c):


# get the shortest path from start to end by reversing the path get from prev
constructPath (prev, matrix, s):


# update the destination point of the path searching constantly as the player moves
def updateMatrixE(matrix):
```

# update the starting point of the path searching constantly as the player moves
def updateMatrixS (matrix):

Angle finding:

Purpose: make sure that both player and enemies can shoot at any angle.

1. Get the coordinate of two points (starting point and end point)
2. Calculate the slope and y intercept of the equation constructed by the two points.
3. Give accelerations of different directions according to the relative positions of the two points. For example, give an acceleration on x, calculate the y position as x move with the equation that we get.

Background scrolling:

Purpose: the background will start and stop scrolling at certain points, and all different objects will occur at different points.

Have a stage width, display screen width, and background picture width.
Get the relative X to decide the starting point of background drawing
Self.relX = self.x % bgwidth
This will get a poisitve value and make sure the background is constantly drawn.

Draw all the objects on the stage, when the player reaches the start scrolling point, the player will stop at the middle at the screen and its displacement will be given reversely to all the objects, so the stage will constantly move to left as the princess moves to right, and the player can only see the part of the stage that is drawn on the screen.

Constantly changing platform maps:

Purpose: to show that the path finding algorithm can apply to any maps.

Using the get clock tick method in pygame to get the time elapsed, and the platforms will change every few seconds . To make sure that the map is refreshed and not cumulatively drawn, every few seconds the platforms will be killed and new platforms in the new map will be added to the sprite group.

Switching roles:

Three princesses will share some attributes such as blood, life, and score. But they have some independent actions such as attacking with sword, freezing, and flying.
Make sure that user can switch role easily at any time.
Make good use of OOP to realize it.

Animations:

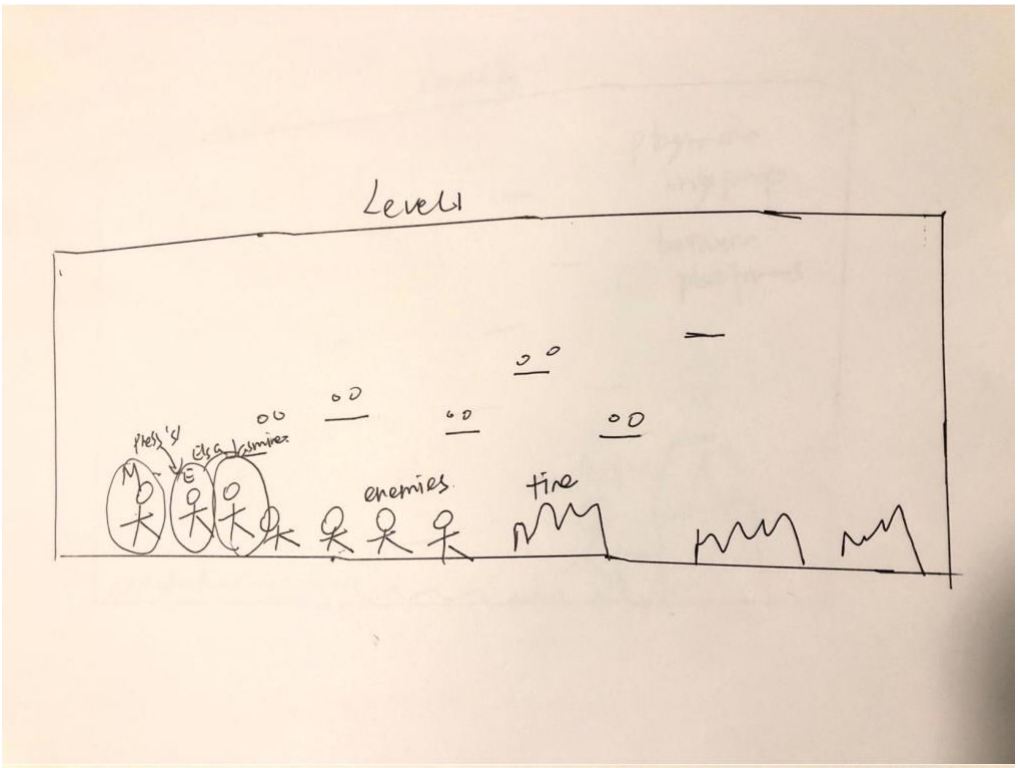
1 princess animation

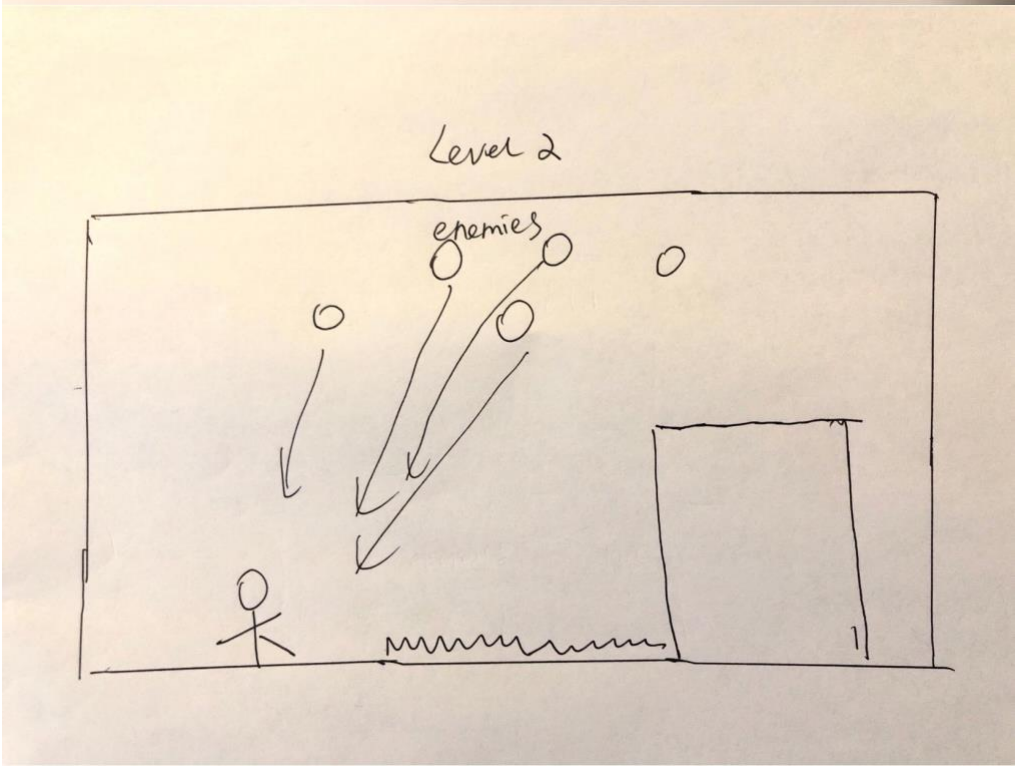
Count the movement according to time per frame and loop through the sprite sheets to create running and jumping animations using the counts.

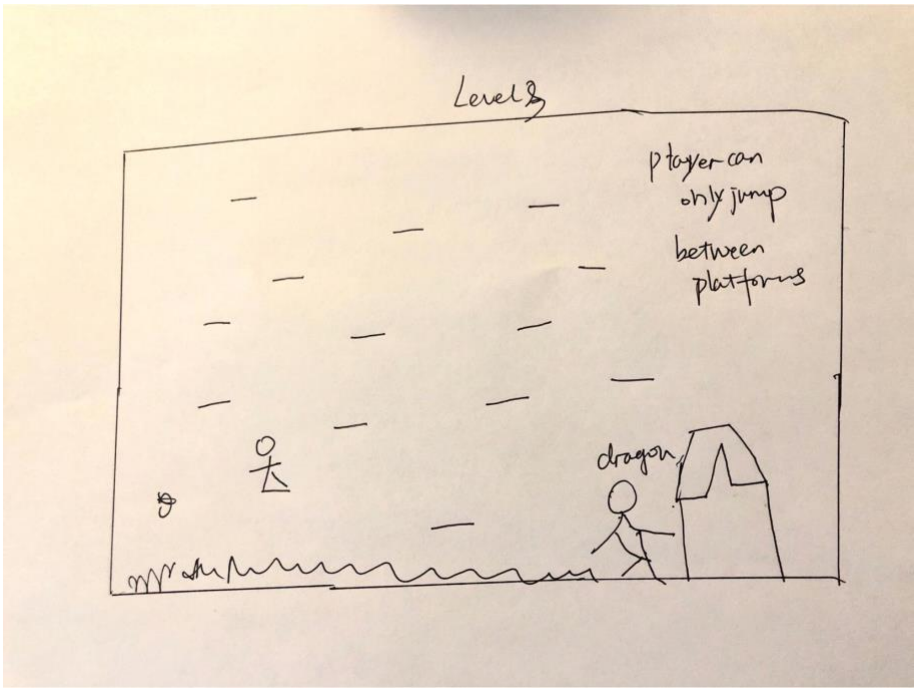
2 soldier and princess sword animation

Use the model to control the display status of the swords, it will appear and disappear when the princess or the soldiers are attacking.

Storyboard


Level 1
enemies
fire


Level 2
enemies


Level 3
player can only jump between platforms
dragon