Term Project Proposal

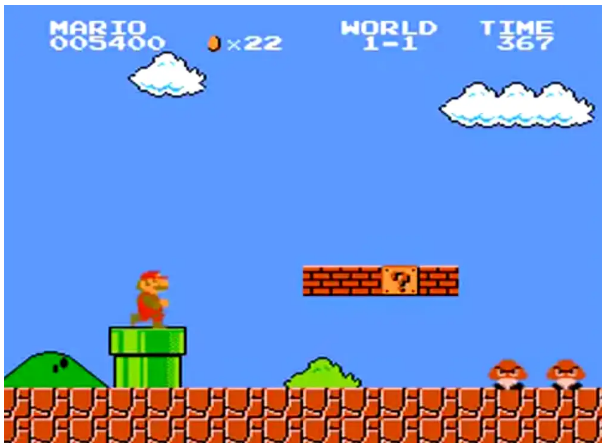
Benjamin Chen

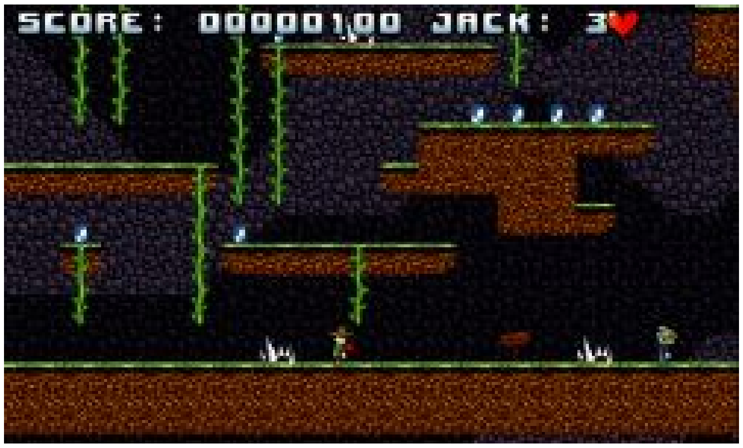
Project Description:

The name of my term project will be 2d Runner. 2D Runner is essentially a 2D platformer game with the main feature of the game being an auto sidescroller that threatens to catch up with the player. In order to progress through the game, the player must maneuver the character through a series of platforms and various other obstacles all while collecting many power ups and coins along the way. Each coin collected will contribute to the player total's score which will be shown at the top the screen. The game is designed to be a 2D infinite runner, meaning that the game only ends when the screen catches up to the player. Assessment of performance will solely be decided on the score the player has attained at the end of his or her run.

Competitive Analysis:

While 2D platformer games have been done in the past, there are some features that 2D Runner has that makes not only more algorithmically complex but also delivers an unique player experience that **fosters competitiveness in a potential speedrunning community.**
The first feature aimed towards this goal is **procedural terrain generation**. Instead of hardcoding certain maps or paths for a character to navigate, 2D Runner adds in a random aspect to keep players on their toes. With procedural terrain generation, faster and cleaner runs are all dependent on the player's skill and mechanics instead of practiced muscle memory. This in turn makes 2D Runner stand out from other games like Super Mario Bros which award players not on skill level but how accurate their muscle memory is when speedrunning.





SUPER MARIO BROS (HARDCODED) PROCEDURAL GENERATION

However, 2D Runner also offers a more **casual** experience for those who don't intend to take this to a hardcore gamer level. Its **powerup generation feature** is designed to make the gaming experience overall more enjoyable with its wide collection of powerups like double jumps and even time control. Moreover, **a difficulty option** at the start screen will allow the game to be geared towards certain types of players.

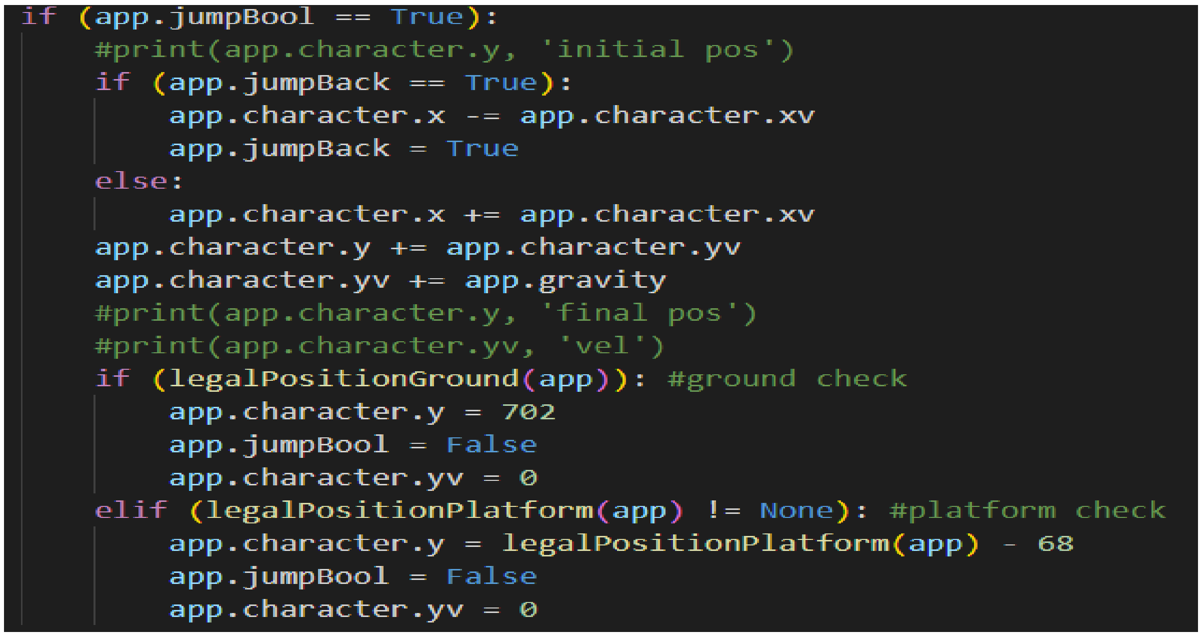
Structural Plan:

At the top level, the game is separated into a wide array of classes. The main class, game.py, is designed to have all the MVC functions and will run the game when executed. Other classes including character and powerup classes store information regarding the location, velocities (used for character physics), and power types. By separating each of the entities in the game into different classes, the process of controlling them in the game file becomes a lot more simplified.
Current functions in the main file include:

- **Hitbox Functions (legals):** These functions will be run continuously or at certain times to check for collisions. As the game relies on a character running through a series of obstacles, it is essential that hitbox collisions are accurate to ensure a consistent playthrough
- **Event Functions**: These functions will take input directly from the keyboard to control the character. As the character can be moved around freely, many inputs must be be taken in to account for this/
- **Continuous Functions:** Timerfired is heavily relied on. As an overview, it contains the jumping physics, gravity, platform/powerup/coin generation, auto-side scrolling etc.
-

<u>Algorithmic Plan:</u>
Algorithmic complexity will be tackled in 3 ways:

1) Basic Gameplay: Gravity, powerup generation, and coins are all part of the basic gameplay. Gravity is especially important as it controls how high and far the character can jump, greatly affecting the difficulty of the game. Kinematic equations and continuous math calculations will be used to determine character arcs when jumping. Gravity is also constantly applied to the character at all times. This is done by running the downward arc equation in projectile motion when the game detects that the character is not on solid ground. Power generation will also add another degree of algorithmic complexity by directly modifying the character's abilities. It's random spawn location combined with how it affects the game will be tough to implement but if done correctly, will most certainly bring in a lot of algorithmic complexity into the project. Lastly, coins are added in to incentivize players to take the correct path in achieving the highest score.

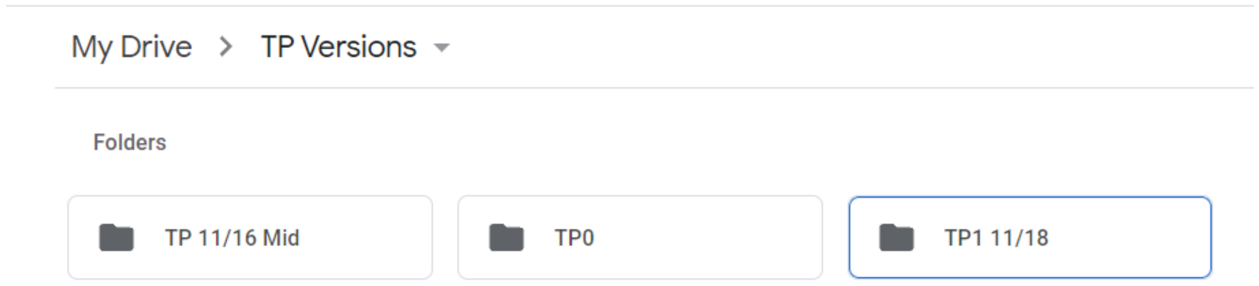


```
if (app.jumpBool == True):
    #print(app.character.y, 'initial pos')
    if (app.jumpBack == True):
        app.character.x -= app.character.xv
        app.jumpBack = True
    else:
        app.character.x += app.character.xv
    app.character.y += app.character.yv
    app.character.yv += app.gravity
    #print(app.character.y, 'final pos')
    #print(app.character.yv, 'vel')
    if (legalPositionGround(app)): #ground check
        app.character.y = 702
        app.jumpBool = False
        app.character.yv = 0
    elif (legalPositionPlatform(app) != None): #platform check
        app.character.y = legalPositionPlatform(app) - 68
        app.jumpBool = False
        app.character.yv = 0
```

2) Procedurally Generated Terrain: Platforms will be spawned randomly. This will be done using a random number generator to determine at what height level (reachable by the character at the moment) that a new platform will spawn in.
3) Advanced Algorithmically Complex Generation Based on Player Performance: As the player achieves higher and higher scores, the game will correct for this and make it harder for the player to jump to certain platforms and their respective coins. Various debuffs will be implemented as another obstacle players must overcome to get higher scores.

Timeline Plan:
TP1: Platform, powerup, and coin infrastructure laid down. Start screen and game over conditions implemented. Basic character physics and hitboxes working (small margin of pixel errors is permissible)
TP 2: Auto-side scroller fully working. Powerup modification fully working.
TP 3: Fully working game. Graphics fully implemented. Possible background soundtrack and sound effects

Version Control Plan:



Google Drive is being used to store versions of each date I am working on. More specific copies are stored within each folder.

Module List:
No additional modules will be used.

Storyboard (in zip file, labeled 'storyboard.jpg')

TP2 Updates
- Modified MVP Definition (Feature 3) to terrain generation based on player performance
- Revised project description to solely focus on ending the game when the screen catches up
- More emphasis on the coin collecting feature to incentivize higher scores

TP 3 Updates

- No updates made