Design Doc

***TP3 Update**: design changes*

- Updated user interface: sprites of characters, background, and floor
    - Took screenshots from https://www.coolmathgames.com/0-double-panda
    - Cleaned background of screenshots, rescaled images, and stitched together spritesheets on GIMP
- Added intro, instructions, and game over screens
    - Images were also created on GIMP
    - Made these modes of a modal app
- Fixed bugs in candy collisions and enemy collisions (neither was accurate before)

***TP2 Update**: design changes*

- Each group of platforms can only skip one level
- Eliminated the box because it doesn't affect platform generation much (just shifts up the bottomLevel)
- Enemies only have one life because otherwise it would too hard to play
- Only BasicEnemy can walk towards the main players (at a high score) if they are on the same platform

Project Proposal

***Project Description** [2.5 pts]: The name of the term project and a short description of what it will be.*

Name of TP: Endless Double Panda
Based on the original Double Panda (found at https://www.coolmathgames.com/0-double-panda), this single-player game is one endless level featuring random platform generation and enemy AI. The goal is to control the two pandas to score as many points as possible.

***Competitive Analysis** [2.5 pts]: A 1-2 paragraph analysis of similar projects you've seen online, and how your project will be similar or different to those.*

Since my project is based off of the original Double Panda game, it will have similarities and differences. My game will have some of the same components, such as two main players (pandas) that the user can switch between, bamboo that only the red panda can climb, enemies that only the giant panda can climb, and different heights of platforms. The original game is structured as 20 different hard-coded levels that are essentially puzzles, and the goal in each level is to use the different functions of each panda to get to the finish line. In my game, the goal will be to score as many points as possible, so the user cannot just keep running in a straight line. There will still be some problem-solving involved in figuring out how to get onto higher platforms with the two pandas. In my game, it will also be easier to die because there will be more enemies, and it will be harder to kill some of them.

Another similar project I have seen online is the game Doodle Jump. The goal of the game is to jump on different randomly generated platforms to get as high as possible and defeat enemies along the way. My game will be similar in that it will also have randomly generated platforms and enemies. The main differences are that moving is not as simple since the user has two pandas to control and the goal is to score points by eating candy instead of running as far as possible.

Structural Plan *[2.5 pts]: A structural plan for how the finalized project will be organized in different functions, files and/or objects.*

- Parent class: Character
    - Child class: GiantPanda
    - Child class: RedPanda
    - Child class: BasicEnemy
    - Child class: ArcherEnemy
- Class: Platform
- Class: Bamboo
- Class: Candy
- Class: MyApp
    - Creates one instance of GiantPanda and RedPanda each
    - Has move methods that control the movements of the two main players (pandas)
    - Has methods to generate (instances of) platforms, enemies, and candy based on probability and where the player is right now

Every class will be in its own file. (The Character child classes will all be in the same file as the Character parent class.)

Algorithmic Plan *[2.5 pts]: A detailed algorithmic plan for how you will approach the trickiest part of the project. Be sure to clearly highlight which part(s) of your project are algorithmically most complex, and include details of the algorithm(s) you are using in those cases.*

- Generating “terrain”
    - Platforms
        - Randomly choose what the bottom level will be (1, 2, 2.5, 3, 4) using probability
        - Randomly choose what the top level will be (3, 4, or 5) using probability and the current score
        - Get the last 5 or less platforms in the list and find the furthest platform (to make sure the new generated platforms do not collide with it)
        - Loop from the bottom to top level to generate new platforms with random lengths and set off distances from the last platform
        - Randomly skip a platform during the loop in order to require one panda to jump on the other’s head to reach it
    - Bamboo
        - Generate next to platforms that

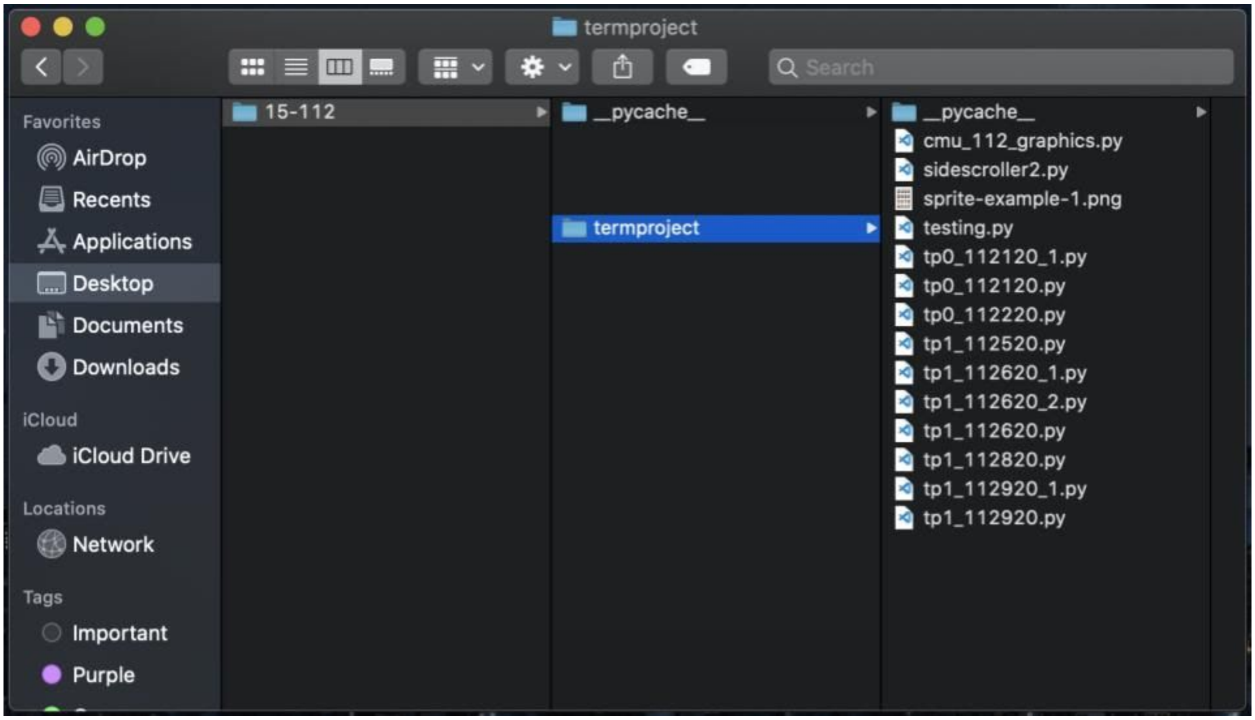
            - Are the bottom level and at level 3+
            - Are not the bottom level but there is a >2 space between lower levels or if the bottom level is 1.5 and there is a 2 level difference from the bottom to next higher level
    - Box
        - Generate next to bottom levels that are 2+
- Generating enemies
    - Basic
        - Wanders on platform or on the ground, or walks towards currMain or otherMain
        - 0, 1, or 2 lives
    - Archer
        - Attacks using arrow
        - Takes away one life
        - Wanders on platform or on the ground, or walks towards currMain or otherMain
        - 0, 1, or 2 lives
    - Randomly generate on platforms or on the ground
        - The higher the score → the more likely the enemy has more lives, the more likely to get an archer enemy, the more likely the enemy is to walk towards the players
- Generating candy
    - Randomly generate on platforms
    - Randomly choose what score gain it will have

Timeline Plan *[2.5 pts]: A timeline for when you intend to complete the major features of the project.*

- Movements of players — 11/26
- Basic platform generation — 11/28
- Bamboo generation — 11/30
- Candy and box generation — 12/1
- Enemy generation — 12/4
- Fixing the appearance — 12/5

Version Control Plan *[1.5 pts]: A short description* ***and image*** *demonstrating how you are using version control to back up your code.*

I am saving versions of the file in a folder on my desktop. I usually create a new copy if I am making larger changes that I might want to undo.



15-112

__pycache__
termproject

__pycache__
cmu_112_graphics.py
sidescroller2.py
sprite-example-1.png
testing.py
tp0_112120_1.py
tp0_112120.py
tp0_112220.py
tp1_112520.py
tp1_112620_1.py
tp1_112620_2.py
tp1_112620.py
tp1_112820.py
tp1_112920_1.py
tp1_112920.py

Module List *[1 pts]: A list of all external modules/hardware/technologies you are planning to use in your project. Note that any such modules must be approved by a tech demo. If you are not planning to use any additional modules, that's okay, just say so!*

No external modules/hardware/technologies