Project Proposal - Antoine Assaf

Project Description

Name: Metro Masters

Description: Metro Masters is going to be an endless subway runner where the player can collect coins and power ups while sprinting through a metro. In this metro, trains will come towards the player, and the player needs to jump/switch lanes in order to avoid them. Players can level up their power ups to increase their duration with the coins they receive, and they can see their high scores against calculated bot scores.

Competitive Analysis

In accordance with similar projects online, projects that are similar are the mobile app Subway Surfers (by Kiloo and SYBO Games) and Subway Sprinters (by Snazzy Snacks). In terms of similarities, all games will have the ability to switch lanes to avoid trains, collect coins, and utilize/upgrade power ups. Both Subway Surfers and Subway Sprinters are different because their codes use a premade 3D engine while Metro Masters has a 2.5D graphic programmed from scratch that contains one point perspective. Metro Masters will also have a different powerup, the speed boots, that will allow players to expedite their travels and coin collections mid game.

	Graphics	Unique Power ups	Hoverboard	Unique Obstacles	Coin Usage
Metro Masters	2.5 D engine, better in performance yet lacks cross-depth	Speed Boots, Forcefield	This game will not have a protective hoverboard	Deadly Train Plank Spikes	Upgrade Power ups, Upgrade Multiplier
Subway Surfers	3D - uses a lot of space/storage	Coin Jetpack (flying through the sky)	Hoverboards are defensive and serve as a forcefield	Tunnels (blocks two lanes)	Upgrade Power ups, buy characters
Subway Sprinters	3D - uses a lot of space/storage	None	Has a default hoverboard (double tap)	Springboard	Upgrade Power ups, buy characters

On the other hand, Metro Masters differs from Subway Surfers and Subway Sprinters because they have a jetpack power up that allows the players to fly through the sky and collect coins for a certain number of seconds. Although Metro Masters does not have the jetpack powerup, it instead has the speed boots as described previously. Another component that Subway

Surfers has that Metro Masters doesn't is the ability to put a protective hoverboard at any time for 30 seconds. Although Metro Masters does not have that component, the lack of protective hoverboards will be compensated with protective items (cannot get hit by a metro while having the speed boots equipped).

Structural Plan

The finalized project is planned to have files containing different classes as listed below:

CMU tkinter File (from https://www.diderot.one/course/34/chapters/2808/#anchor-segment-214904)

→ Includes Modal App, App, and TKinter graphics

geometry File

→ Contains subclasses for points, lines, surfaces, objects that can be rendered accurately inside (includes Z-fighting in terms of layering). All of these inert from the Geometry class

→ Each of these subclasse will attribute location, absolute location (on canvas), color, points, a move function (called by Timer Fired in the Play file)

→ References CMU tkinter File

__init__ File

→ is a MAIN file and will heavily rely on the Modal App

→ Modal App will include 6 Modes

→ start, play, shop, pause, end, highscores

→ Since play is the extensive bulk of the code, this code will import the code from there (below). __init__'s purpose is just to structure all of the module together and deal with button interaction

→ Runs all of the modes and deals with any backtracking algorithms and preparation algorithms to make the game interactable

→ References CMU tkinter File

→ References geometry Class

play File

→ attributes of score, multiplier, coins collected

→ references the runner Class to fetch data there

→ backtracking generator function

→ standard input functions: keypressed, mouse clicked (from CMU file) which will detect input and move the runner accordingly

→ References CMU tkinter File

→ References geometry Class

runner File

→ Runner class

→ contains attributes including, position, lane, speed, height, level (on train or not)

→ move function will let the runner switch lanes according to input from the play File

highScore File

→ A local file that is updated every time a player finishes a game. This file will be updated and read. Although no encryption will be protected from direct editing, these numbers will be local, meaning that any direct file editing will render as a mere visual difference.

→ References CMU tkinter File

Algorithmic Plan

One of the most mathematical challenges will be the calculation of which objects to generate (in terms of trains and obstacles). To determine what is generated, there will be a variable to determine how many obstacles will spawn at a given portion of time (default param. @1). Given the number of obstacles, the generating algorithm will utilize backtracking to ensure that the trains and obstacles generated are both possible for the player and reasonable. In order to include variety in what obstacle sets are generated, there will be a random factor when looking through the possible moves (if not, every obstacle set will be generated the same).

```
First: create a recursive function with a state parameter
(containing the obstacle moves) and a target for the number of
obstacles
     -obstacles will be stored as a string and their location as
an index depending on the height they affect (this is necessary
to ensure that collisions are valid later)
     -when target number of obstacles = the number of obstacles
(base case) then return the state

Second: loop through all of the moves using a random generation
to approach the list (perhaps a random sort then go through the
choices)
     → random indexing through the choices of all moves will
ensure that each set of obstacles will be different (rare for
exact copies)

Third: Checking legality of the move
```

→ check if the state of moves (where the trains and obstacles are) are actually possible for a player by checking a bool array that each obstacle is assigned to in a dictionary

→ need to consider ramps (allow players to go onto a higher level)

Fourth: recurse if valid and not None and if not return None.

This extensive look at backtracking, especially in terms of storing the possible moves and checking validity of those moves, will require a recursive function with a base case where the number of obstacles placed equals the target number of obstacles desired to spawn. This desired number will fluctuate depending on the difficulty (which is dependent on the number of seconds elapsed since the start of the game).

Timeline Plan

For Monday:

→ create the Geometry class along with all of its points (2.5 D 1 pt perspective)

→ allow and keep track of player movement using WASD or arrow key inputs

→ make sure spawned trains will disappear (no reason to whole more data than necessary)

(after submission)

→ deal with collisions for trains and obstacles

→ implement a ducking mechanic

→ add a score and a base multiplier

For Tuesday:

→ allow for coins to be collected

→ allow for the player to ramp onto metros and duck below (this includes jumping physics)

For Wednesday:

→ create the backtracking generator for trains and obstacles to spawn

→ program the skeleton of powerups and implement their visuals

→ MAYBE create the cutscenes (intro, shop, high scores, paused, final)

For Thursday (EXTRA):

→ let the player control their character via clicking

→ add an animated gif for the player and certain moves

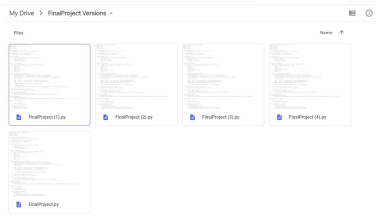
→ with that, perhaps allow players to buy

→ add more visuals for the main screen

→ implement a multiplier system (where players can purchase a multiplier item in the shop to help their overall base multiplier)

Version Control Plan

All versions of the game will be stored on Google Drive using my Andrew CMU email to ensure that the account is protected. Individual Python files will be stored on there as much as possible to ensure that minimal progress is lost if a technical error or mishap occurs.



TP2 Update

Here are all of the main features added in TP2:

- Coins
- Score (accelerates as speed increases, and multiplier caps out)
- Collisions (die on hit unless forcefield is present)
- Ladders
    - Currently bug if a play tries to jump on a ladder from a train
- Jumping/Ducking
- Duck Barrier (requires players to duck)
- Jump Barrier (requires players to jump)
- Second Level (on top of trains)
    - This includes jumping from train to train
    - Falling (and dealing with that collision detection in of itself)
- Powerups (all currently last for 12 seconds; time left is displayed)
    - Magnet (auto collect coins)
    - Speedy Boots
        - Provides a Forcefield
        - Speed is set to 133%
    - Double Score Multiplier
- Force Field (all objects disappear if player gets hit with force field on)
- Z-Index Layering (zDepth)
    - Currently does NOT decipher objects between lanes (TP3 goal)
- Progressive speed (caps out at 100%)
- Backtracking/advanced recursion Object generator
    - Can only generate objects that are physically possible for the player
    - More obstacles are present as the speed increases

(tp3 on next page)

TP3 Update

Here are all of the main features added in TP3:

- Different screens including
    - Home screen
    - Instructions screen (ONLY FOR FIRST-TIME PLAYERS)
    - Help screen (different from screen above)
    - High Scores screen
    - Shop screen
    - Congratulations screen
        - High score is displayed on this screen
- Custom-made visuals and images
    - With all of these screens includes interactive buttons to navigate around
- Transition effect (trigonometric) when switching screens
- Coins/scores/power up levels now save
- Fixed rendering issue where trains would overlap with each other
- Adjusted and added more power ups
    - AI Powerup
        - Automatically plays the game for you during the power up with color animation by predicting/adjusting to collision detection
    - Magnet
        - Automatically collects the coins for you with color animation
        - AND makes every coin worth 5 points
    - Double Points
        - Doubles the multiplier with shining floor effect
    - Speedy Boots
        - Speeds the player with an attached forcefield with a green oozed effect
    - All power ups now have an improved visual effect
- When grabbing a powerup, a progress bar now is displayed at the top of the screen for how long left the powerup will last
- Shop screen:
    - Players can purchase for their power ups to have a longer duration (2 extra seconds per level)
        - This includes a progression bar for each item
    - Max level is 6 which is when the item will last for 22 seconds
    - Ensures that gold is never negative and players has sufficient gold
- High score screen:
    - Saves all of your scores and only displays the top 10
    - Automatically puts in bot scores with threshold difficulties
    - Automatically sorts and displays scores

- Data now SAVES in the txt files
- Play screen now displays the score, the coins, the multiplier, and the map in an aesthetic way
- Added two new maps (themes):
    - Malibu
    - Golden Gate Bridge
        - Includes generation of bridge posts
    - NOTE: these maps are chosen randomly/ map is displayed at bottom of screen
- Tunnels now generate during the map for decorations (z-fighting is account for)
- Fixed major bug where the backtracking algorithm would sometimes generate an impossible train arrangement