Name: Jarel Tey

Project Proposal

*Project Description*

Super Bump Sheep. A game between 2 players (or 1 player and an AI) where the aim is to get sheep of your color to the opponent’s side of the board, while preventing your opponent’s sheep from doing the same. Whoever scores a certain amount of points first wins.

*Competitive Analysis*

One similar project is “New Plants vs. Zombies” by Jack Lyu. This is similar because both are games which are played out on a grid with rows and columns. Also, these fall into the same category of resource management where decisions are made regarding the use of limited resources to try to win.

One difference is the options available to players – while in “New Plants vs. Zombies” there are many plants with different functions, my intended game only has variation in the sizes of sheep available. To make up for this my game presents more options for which rows and columns to send sheep in. Having foresight and good timing play a larger role in doing well.

*Structural Plan*

**Files:**

* 1 file (TP\_main.py) handling main functions of the game (~1000 lines of code)
* 1 file (TP\_AI.py) containing code for AI (~450 lines of code)

**Objects:**

* Sheep()
* Row()
* Col()
* Player()
* Bump() – these are individual collision objects
* MyApp(App) – handling animations

**Functions:**

Main functions listed below, many helper and draw functions not listed

* generateNextSheep(): randomly generate new sheep for players
* checkSheepReady(): checks if sheep can be sent
* moveActiveSheep(): move all sheep on grid
* checkCollision(): resolve any kind of collision
* addPoints(): add points for sheep that reach opponent’s side of board and remove them

*Algorithmic Plan*

Parts of the project I anticipate will be the trickiest are handling collisions (row-row, col-col, row-col) and building the AI.

**Row-row & Col-col collisions**

Handling of row-row & col-col collisions will be the same except in different axes, so I’ll explain for just row-row. First I’ll need to handle collisions between sheeps of opposing color. This will be done by having a row object, and when a collision is detected (distance-wise), calculating rowCollsionNetPower, rowCollisionTotalPower and speedFactor attributes for the row object. The speedFactor will then be used to adjust the speeds of the colliding sheep in that row.

Since the game allows sending multiple sheep on the same row to combine their strengths and push back opposing sheep, I’ll need to handle sheep supporting sheep of the same color (same color collision). When such collision is detected, I’ll add on the new sheep’s attributes to the row object and update the speedFactor accordingly.

**Row-col collisions**

Row-col collisions will only occur for sheep of different colors. When such a collision is detected, I intend to move the sheep on the column so that it is on the same row but in an opposing direction from the other sheep (row-row collision). This is done by removing and adding to my list of activeSheep.

**AI**

My AI considers a few factors when deciding which rows/cols to send sheep on.

* Current points and board state (play defensively if opponent about to win, aggressively if AI about to win)
* Importance of winning particular row/col (send sheep in row/col where points gained can be maximized or points gained by opponent minimized)
* Size of upcoming sheep (optimize plays based on size of upcoming sheep)

*Timeline Plan*

Board layout – Completed

Working normal version of Bump Sheep – Completed

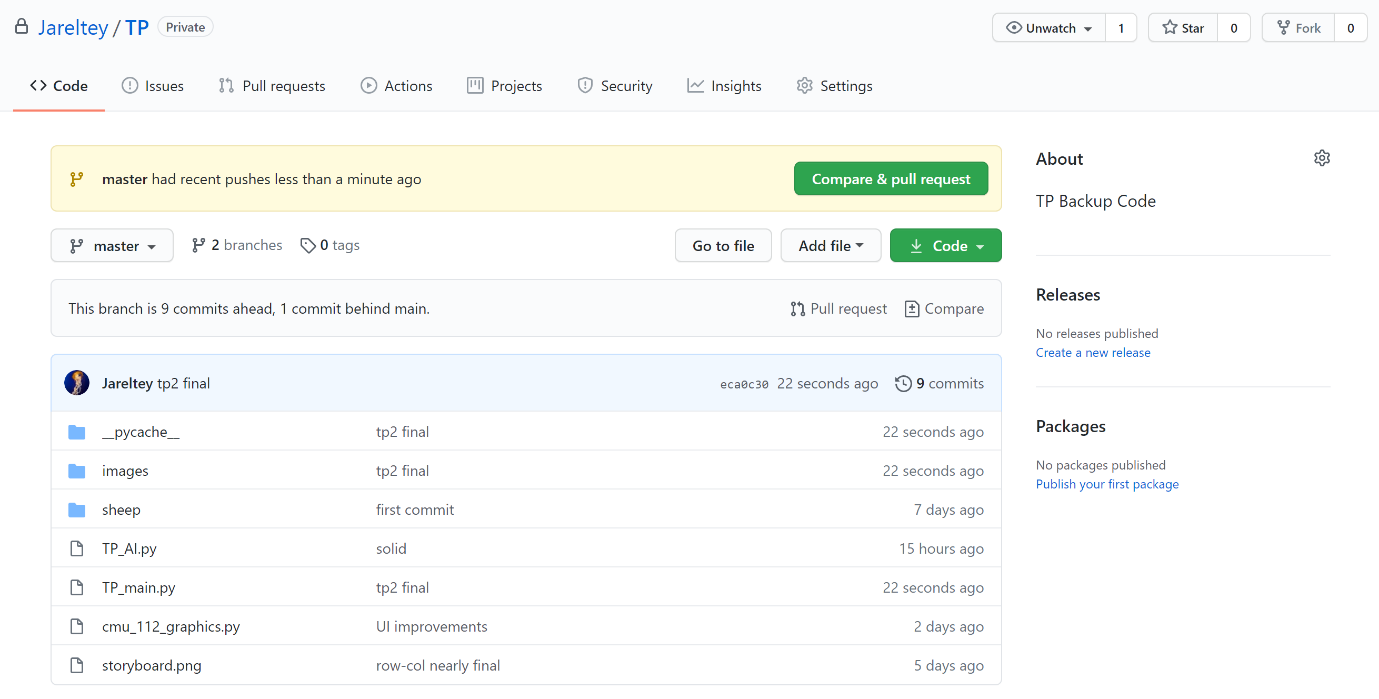
Expanded to columns (col-col, row-col collisions) – Completed

AI – Completed, making adjustments

UI Improvements - Ongoing

*Version Control Plan*

I will be backing up my code on GitHub, pushing each time I make changes.



*Module List*

I will not be using any external modules.

*Citations*

I obtained the following images online:

(Grass Grid) <https://www.904custom.com/media/catalog/product/cache/7/image/600x600/9df78eab33525d08d6e5fb8d27136e95/1/6/16403-grassy-field-grid-gaming-mat-hcb.png>

(2 Sheeps & Logo on Front Page)

<https://i.ytimg.com/vi/LcAt5S9nGio/maxresdefault.jpg>

(Arrow)

<https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRh96iifammcJ_PQSRh_Sj2Uf-VPdM8-WD7Lw&usqp=CAU>

I obtained images of sheep and buttons from screenshots in the “Bump Sheep” game.

I obtained code for getCachedPhotoImage function from

<https://www.cs.cmu.edu/~112/notes/notes-animations-part3.html>

*TP2 Update*

* Added TP\_AI.py file
* Small change to AI: due to steep difficulty of trying to optimize sheep sending based on more than the next sheep (originally planned to consider next 3 sheep), I modified AI to just consider the next sheep. Still a very strong AI based on feedback given.