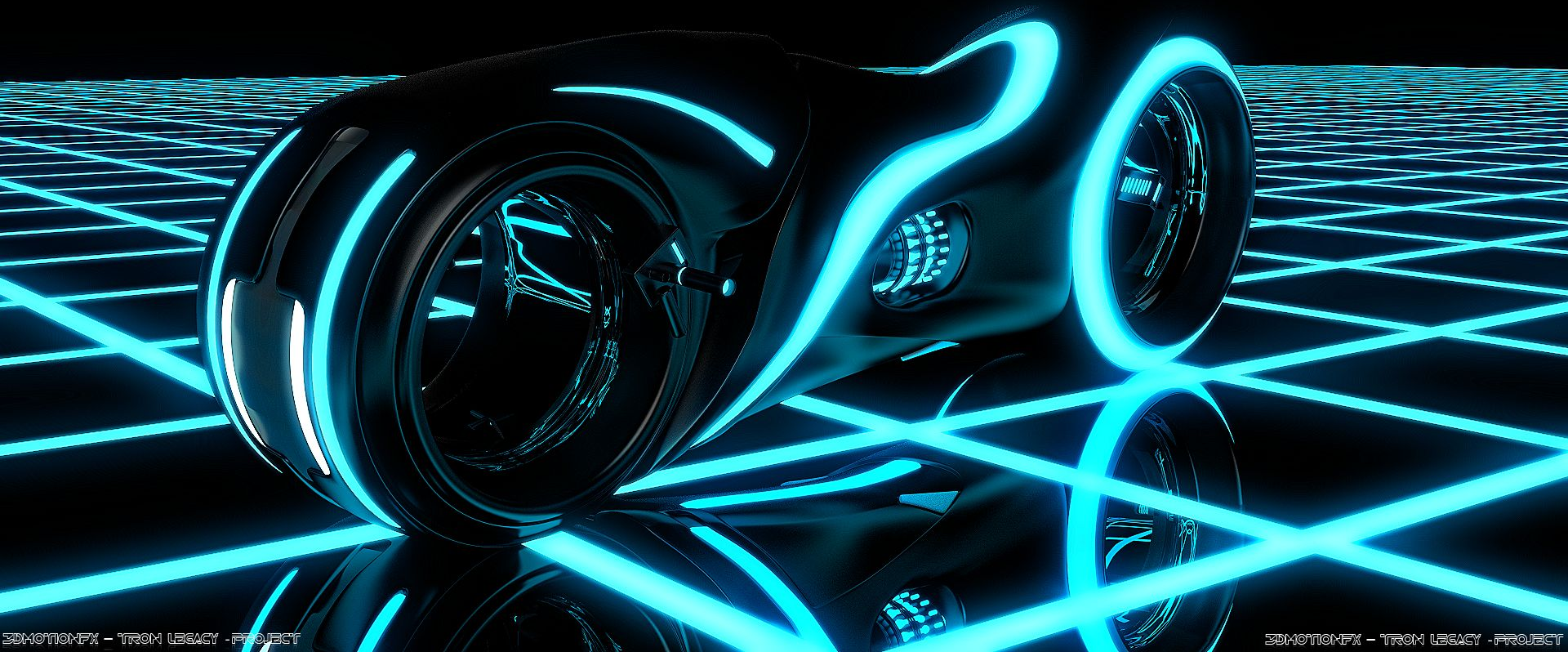
REVAMPING TRON

Naomi Chiu | Jessie Potter | Alex Frye



Agenda

1. Overview of our chosen direction
2. Review of what we have accomplished since the last AR
3. Quick demo of new packaging
4. Updated system architecture
5. Style update
6. Next steps

Key Questions

- What are your thoughts on our current plans moving forward?
- Do you have any suggestions to clarify our code? What needs more commenting?
- Do any of you have experience with minimax?
- Do you think a aggressive bot that seeks to cut off opponents or a defensive bot that seeks to survive the longest would be a more formidable player?

Where are we going?

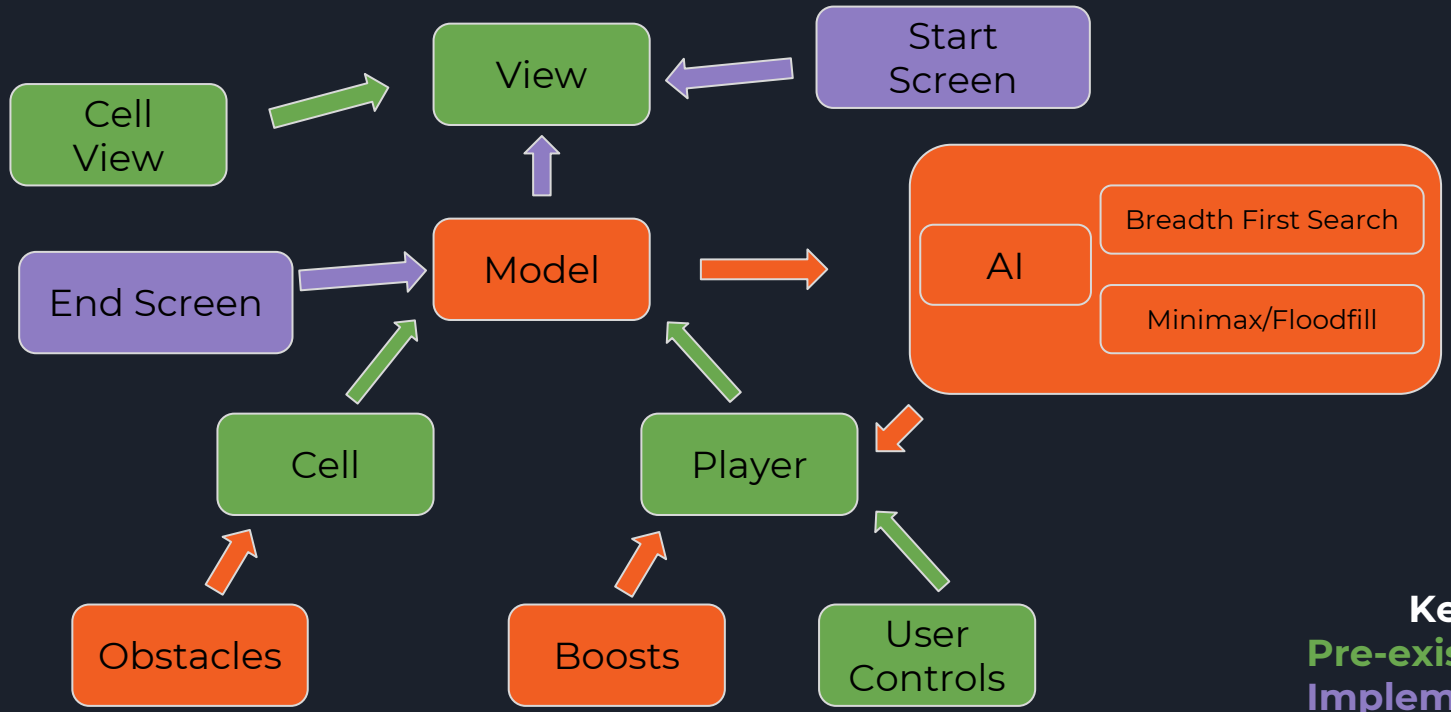
- Focusing on developing an semi-competitive AI
- Hoping to implement with player v. player v. CPU free-for-all
  - Stretch goal: make user interface so number and combinations of players can be customized
- May try to create game maps to increase variety and difficulty

What have we done so far?

- Improved game packaging with a start and end screen now
- Divided up classes into separate files for better readability instead of one ginormous file
- Cleaned up unnecessary code and compiled classes for concision
- Did research on possible AI types (A* and Breadth First) and attempted implementation

DEMO TIME

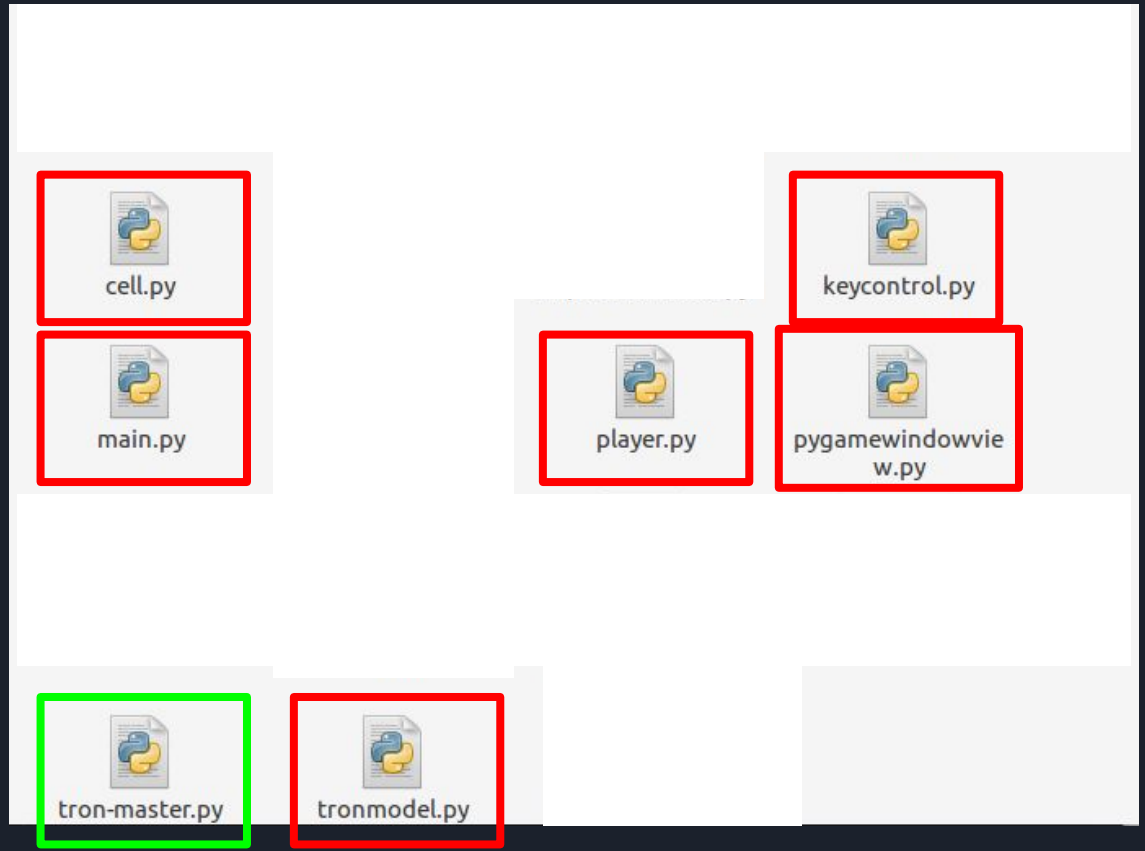
Updated System Architecture Diagram



Key
Pre-existing
Implemented
In Progress

File Breakdown

- tron-master.py was broken into separate files for each class.
- Using file imports to import classes to reference in other classes



Our code for our model class for our MVC

Imports to get needed classes from class files

Docstrings for each method in the class

Docstring about function of file

```
"""
Contains the model class for our MVC structure.
Model updates the state of the game.
"""

import pygame
from pygame.locals import*
import time
import os

from player import Player
from cell import Cell

class TronModel(object):
    """Model object containing the players, the game state, all cells, and the cells that have been hit."""
    def __init__(self,cell_length,width,height):
        pygame.init()
        size = (width,height)
        self.screen = pygame.display.set_mode(size)
        self.width = width
        self.height = height
        self.cell_length = cell_length
        self.cell_lst = []
        self.player_paths = []
        self.player1 = Player(self.screen,10,(self.width/2+100),(self.height/2),"r",(255,140,0))
        self.player2 = Player(self.screen,10,(self.width/2-100),(self.height/2),"l",(0,250,0))
        for i in range(self.height//cell_length):
            for j in range(self.width//cell_length):
                self.cell_lst.append(Cell(self.screen,(i*self.cell_length,j*self.cell_length),cell_length))
        self.game_over = False
        self.end_start = False

    def _draw_players(self):
        """Calls the player objects' draw functions"""
        self.player1.draw()
        self.player2.draw()

    def in_cell(self):
        """Loops through cell_lst to find the cell whose xrange contains player.x
        and whose yrange contains player.y, and sets the player location to be within that cell."""
        for cell in self.cell_lst:
            if self.player1.x in cell.xrange and self.player1.y in cell.yrange:
                self.player1.current_cell = cell
                break
        for cell in self.cell_lst:
            if self.player2.x in cell.xrange and self.player2.y in cell.yrange:
                self.player2.current_cell = cell
                break

    def update(self):
        """Checks for new inputs and updates the game model."""
        self.player1.update()
```

Where do we hope to go?

- After asking some experienced game makers, we found that our best bet for AI for this game would be through Minimax and Floodfill
- We are beginning to build our own AI and are hoping to get it running ASAP
- AI is our primary concern but we still have additional features in gameplay in consideration to implement

Key Questions

- What are your thoughts on our current plans moving forward?
- Do you have any suggestions to clarify our code? What needs more commenting?
- Do any of you have experience with minimax?
- Do you think a aggressive bot that seeks to cut off opponents or a defensive bot that seeks to survive the longest would be a more formidable player?