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

Point Salad

A python implementation of the board game point salad where players take turns drawing point cards or veggie cards and at the end of the game use them in combination to score points.

* **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.

Competitors include the uno game across different devices and the Hanafuda and Uno implementations in the 51 Worldwide Classes game on nintendo switch, all similar board game implementations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Levels of AI Players** | **How to interact** | **Platform** | **Online Play** | **Graphic Interface** |
| Competitor 1 | 1 (No choice) | Joycon and touchscreen | A range of devices | Yes | Good |
| Competitor 2 | 4 | Joycon | Console only | Yes (require nintendo account) | Very good |
| Competitor 3 | 4 | Joycon | Console only | Yes (require nintendo account) | Very good |

The main competitiveness comes from implementation of AI players at different levels, which will use algorithms to find out best moves in each turn. My game is also the very first electronic implementation of the point salad game, which is a gap in my competitors.

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

The cards, draw plies, veggie market, and the players will be written as classes, allowing for a cleaner code.

# object deck

# attribute cards

#

# object pile

# attribute cards

# attribute pileNum

#

# object vegMkt

# attribute cards

# attribute vegNum

#

# object Player

# attribute hand

# attribute playerNum

#

# object BotPlayer

# inherits from Player

# method autoPilot

#

# object Card

# attribute cardNum

# attribute front

# attribute back

* **Algorithmic Plan** [2.5 pts]: A detailed algorithmic plan for how you will approach the trickiest part of the project. This plan should follow the format in our "large project design" notes

First determine if the game is in bot mode.

Then, if in bot mode, call the autoPilot method in BotPlayer class.

In easy mode, the autoPilot randomly choose cards.

In normal mode, the autoPilot choose veggies according to needs by point cards in its hand.

In hard mode, the autoPilot use minimax to figure out the best move.

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

3 levels of AI players by TP2. (Done and added dynamic difficulty mode)

Optimization of AI players, GUI, a third player, socket are all options if time allows.

* **Version Control Plan** [2.5 pts]: A short description and image demonstrating how you are using version control to back up your code. **You must back up your code somehow!!!**

Backup using OneDrive and GitHub

A screenshot of a computer screen

Description automatically generated

* **TP2 Update**
* No fundamental design changes have been made. A hard bot mode with minimax has been implemented, and the only thing added that was not in the original plan is a dynamic difficulty mode which adjusts the difficulty of the bot depending on the evaluation of the current game state. Looking ahead, a more polished UI, adjustments to the AI algorithm, adjustment to the code structure, multiplayer using socket are all options for post MVP.