Project 3 Design Document

Team 33

Eddie Canales
Joseph Hernandez
Lasse Starklit
Mark McCauley
Siman Shrestha

1. Project Purpose

1.1. The Problem

Artificial Intelligence is attempting to develop computer systems which can do things that if humans did them would be considered "intelligent" behavior. The goal of this project is to create a computer game that incorporates the use of an AI searching algorithm and a basic graphical user interface. In this game, the human player and the AI search both have the same centered stack of pancakes set in some scrambled order. The objective of the game is to get the stack in order from smallest pancake to the largest by repeatedly flipping over a top partial stack of a number of pancakes. The human player is trying to get the stack in the correct order in less moves than the computer, or AI search. By completing this project, we hope to learn and code some fundamental AI techniques.

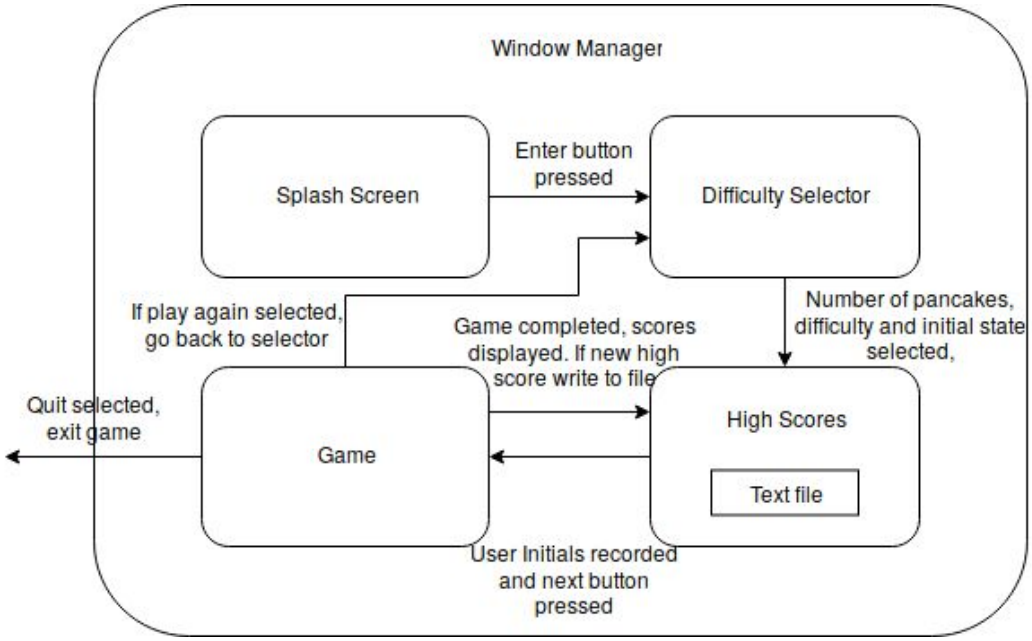
1.2. The Motivation

AI has proven to be extremely effective in helping get work done faster with accurate results. As the technology continues to improve, companies in various industries have been implementing Artificial Intelligence to help with different activities. This project is a perfect opportunity for us to get some first hand experience with AI and learn some valuable techniques while improving our collaborative skills. There is still a vast amount of room for this technology to grow and this project might allow us to see how we can contribute to this growth.

1.3. The Users

Since this is a friendly game, the scope of the users for this project is broad. Of course we need to make the program easy to use for ourselves and those grading our project, but we should also take into account our friends and family who might want to try our game. These users might not come from a strong technical background so it is important to make the game exceptionally user friendly as well as enticing to play.

2. High-Level Design



2.1. Task #1: Splash Screen

- Purpose: This task is done in order to get users interested in the game and want to play. The splash screen should be adequately descriptive and more importantly attention grabbing.
- Justification: Making users want to play your game is important for business. Hypothetically, if we want people to pay to play our game, it needs to be attention grabbing and enticing for a broad audience.
- Role: Since we want our splash screen to be aesthetically pleasing, we are planning to implement an animation of our game being played. The button on our screen needs to be functional and lead us to the next screen.

2.2. Task #2: Pancake Count and Difficulty Level

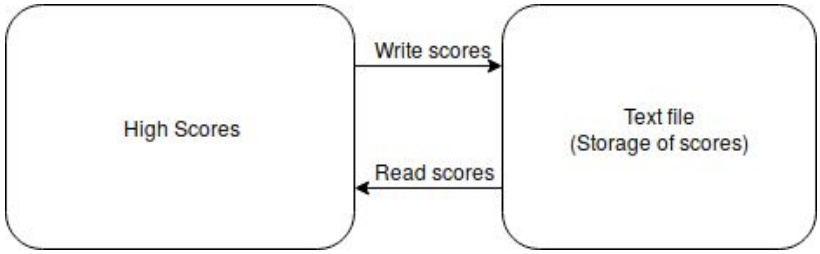
- Purpose: Take input from the user to cater to how they want to play the game. This input is used to set the stage of the game in later tasks.
- Justification: Our goal is to make the game user friendly. If a new player wants to start playing the game, they should have the option to play on easier settings in order to first get used to the game. If a veteran player wants to be challenged, they should also have the option to play on the hardest settings.
- Role: This task appeals to the user allowing them to choose how they want to play. This screen is another important task that needs to be exceptionally user friendly and attention grabbing.

2.3. Task #3: Initial Order State

- Purpose: Allow user to choose the initial state of the pancake stack or use a random order. This is also used to set the stage of the game in later tasks
- Justification: This adds another layer of customization to the game. The initial order of the pancakes factors a lot into how difficult the game will be. By using a random order, we will create variations to the game so that few gameplays will start out the same.
- Role: This is another task that needs to be attention grabbing. It can also be included on the same screen as the previous screen prompting for pancake count and difficulty level.

2.4. Task #4: Top Scores

- Purpose: Instill a competitive spirit in our users to motivate them to continue to play. This will let users know where they stand and give them goals to aim for while playing the game.
- Justification: Almost all popular games today have a high level of competitiveness to them. This gives users a reason to come back and keep playing.



- Role: Because of our broad range of users, it might be smart to have a different. This screen needs to be attention grabbing and match the themes of previous screens.

2.5. Task #5: Visualization

- Purpose: Set the game in motion and set the stage to let the user play the game.
- Justification: This is a crucial step in initializing the game based on the users settings. This is the user's first impression of the actual game so it needs to be smooth and visually appealing.
- Role: Converting the user input into the actual game will be a challenge and the input needs to be stored in a way that this task can have easy access to it. This step also needs to provide the user with a good first impression of the game.

2.6. Task #6: Game Controls

- Purpose: Make the game easy to play for the user. This should also make the game fun to play and not feel clunky or unpleasant.

- Justification: This is the core of our game so it needs to be fun to use as well as smooth when transitioning between moves.
- Role: Game controls will be one of the most challenging tasks and are crucial to really make this an enjoyable game. To appeal to traditional gamers, it would be smart to stick with the "WASD" format to play the game. It is important to put a lot of time into this task to ensure the mechanics of the game are solid.

2.7. Task #7: Minimax Search

- Purpose: Create an effective AI search and write how the AI will flip its pancakes.
- Justification: The minimax search is what will give this program some Artificial Intelligence by the fact that the "AI" will be able to make intelligent decisions on the next move to make. The deeper the minimax tree is, the better decisions the AI will be able to make.
- Role: Minimax search will be another challenging task and is important so our project includes AI search functionality. We need to be sure our minimax search is accurate to create a competitive AI.

2.8. Task #8: Game Over

- Purpose: Check to see if the stacks are sorted. Calculate the users score and display it along with the top 5 scores. If a top score is beat, rewrite the file with the scores. Also prompt the user if they want to play another game.
- Justification: This gives the user a sense of competition and immediate feedback on how they played the game. Making a challenging and competitive game will make more users keep playing.
- Role: This task wraps up the game and calculates the score based off difficulty and number of moves. This screen needs to be able to return to square one and start a new game as well as quit the game entirely.

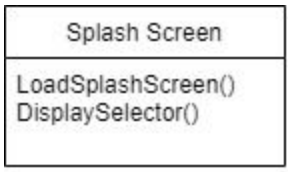
2.9. Extra Task: Splash Screen Demo

- Purpose: Attract more attention to the game and showcase the appeal of the actual gameplay.
- Justification: Having an attractive game all around will allow us to showcase its pleasing visuals up front and entice users to play the game.
- Role: Visualization of our program needs to be consistent in order for this to have the proper effect. The gameplay of our game needs to tie in nicely with our splash screen.

3. Low-Level Design

3.1. Task #1: Splash Screen

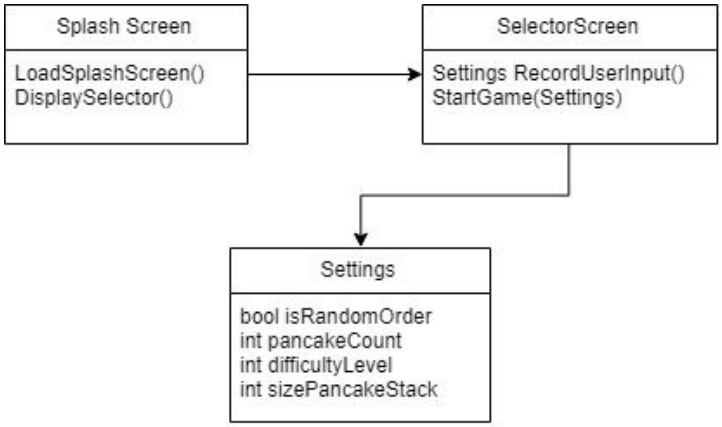
- Usage: The splash screen will be shown a the beginning of each game until the user clicks enter to start the game.
- Model: The settings of the game will be stored in a struct so that it will be easier to reference these settings in the Minimax search, visualization, and game control tasks.



- Interaction: The splash screen will be shown a the beginning of each game so that the player may navigate to play the game or change settings like difficulty, the size of the pancake stack, or initial order of the pancake stack. The user will interact with the splash screen by moving the arrow keys to select the certain settings the user desires.

3.2. Task #2: Pancake Count and Difficulty Level

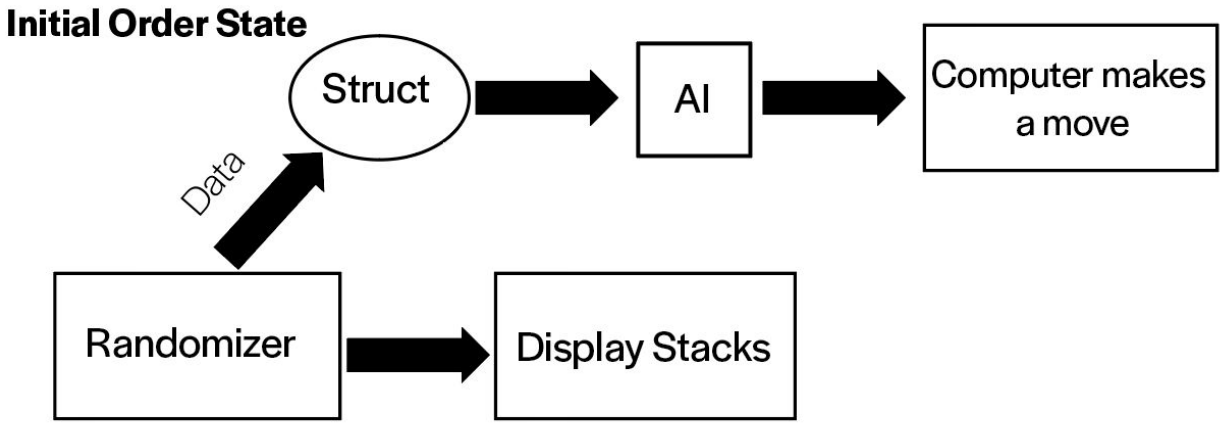
- Usage: Pancake count will be directly correlated to the visualization of the pancake stack on screen as well as the number of leaf nodes that the minimax search tree will have. Difficulty level will directly correlate to the number of searches that the minimax tree looks towards each move.
- Model: Pancake count and difficulty level values will be stored in a settings struct so that the values will be used in the minimax search, visualization, and game control tasks.



- Interaction: The pancake count and difficulty selection will be done by the Selector Screen prompting the user to input the difficulty selection and pancake count from the keyboard. The program will wait for the user to input values and make sure that they are valid before storing them in the settings struct.

3.3. Task #3: Initial Order State

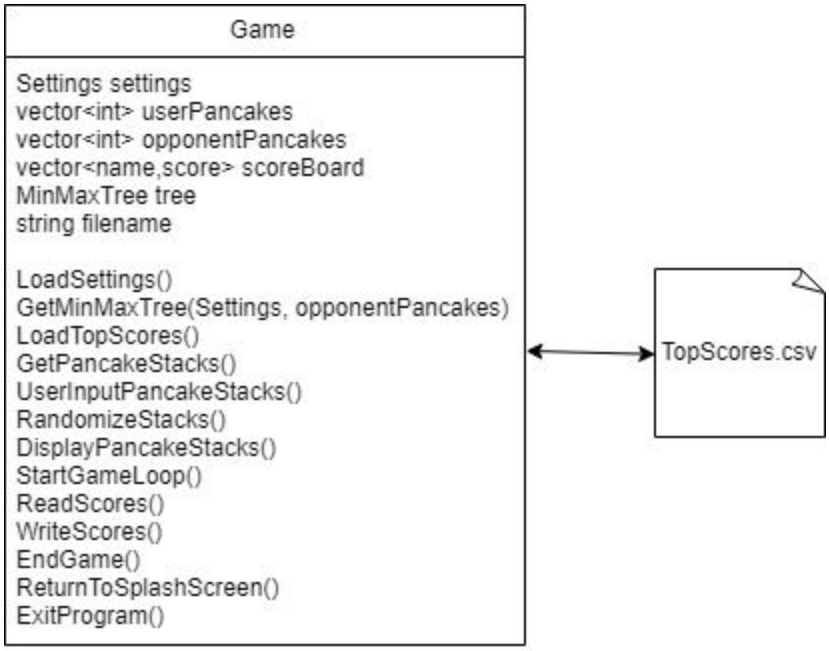
- Usage: Initially, the human player and the AI search both have the same centered stack of pancakes, each one a different size, in some scrambled order.
- Model: A randomizer function will be implemented in order to scramble the order of the initial pancake stack. The order of the initial stack must be stored in a struct in order for the AI to determine what moves to make.
- Interaction:



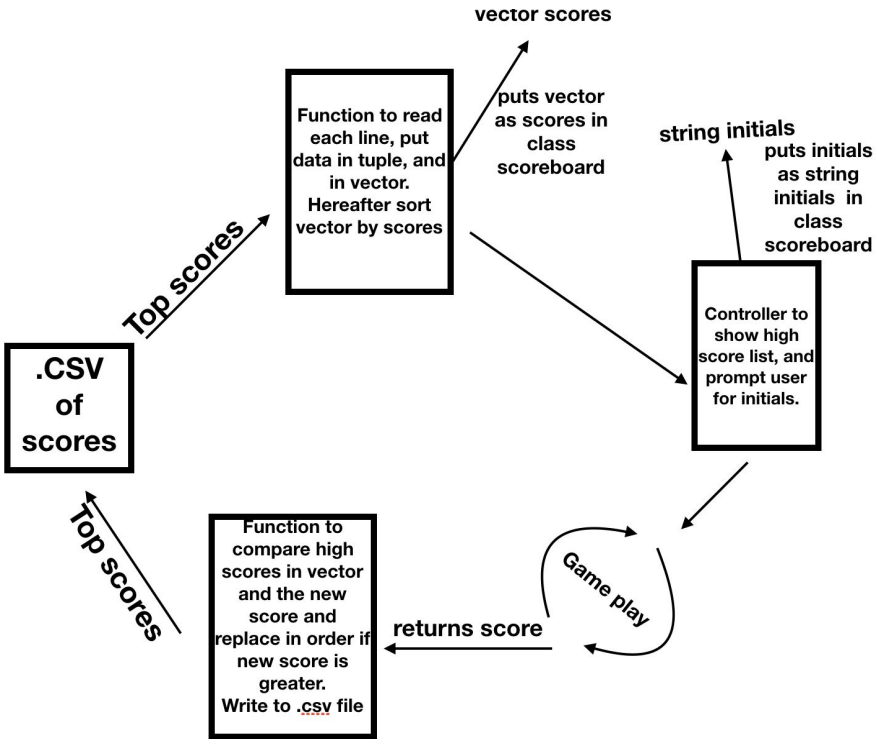
3.4. Task #4: Top Scores

- Usage: Top scores will be saved an updated as the game is played. The top scores will be displayed at the beginning and end of each game played. If the player scores a new top score then they will be able to put their name next to their score and the top scores list will be updated. The scores will be written everytime the game is played and will be reloaded every time the game or program restarts so that scores will not be lost.
- Model: The top scores will be saved in a csv file in the format: initials,score. The top scores file will only support three inputted characters for the initials, while the score itself will be stored as an unsigned int. When the high scores are requested to display a function will be called that reads the csv file with the high scores. Each line

in the file will then be saved as a tuple, where the first element is a string consisting of the initials and the second element is a double consisting of the score. The vector will then get sorted, so the greatest score is the first tuple and the lowest score will be the last. The top scores will then be able to be displayed when it is needed. Once a new game starts the player will be prompted to insert their initials. When the game completes the score is compared to the vector of scores. If the score is greater than one or more scores from the scores vector, it will get inserted in order and pop the lowest score . Next, the top scores text file.

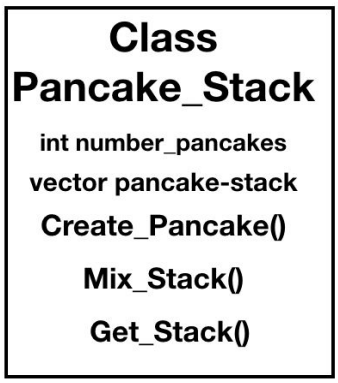


- Interaction: Whenever the game is over the program will stream in the file and store the top scores inside a vector. If the score of the completed game is better than one of the scores in the vector then the score is inserted into the corresponding place. Next the program will write into the Top Scores csv file the new list of top scores.



3.5. Task #5: Visualization

- Usage: The visualization makes it easier for the user to get a grasp of the actual game play and the current position to be able to make the best possible move.
- Model: A pancake of size x, initializing a string of name pancake-x. Creating a for loop iterating x times makes it possible to add characters to the string, so the string pancake-x can be shown by just printing the string. After each pancake is created and added to a vector, the order of strings will get mixed, so the stack is mixed up. After this the stack be used in the game play.
  The following illustrates the class used to create the stack.



- Interaction: Each pancake in the stack can be used by the controller that initiates the game play. The stack can easily be called from the controller and shown on the window, since each string in the stack is made as a pancake.

3.6. Task #6: Game Controls

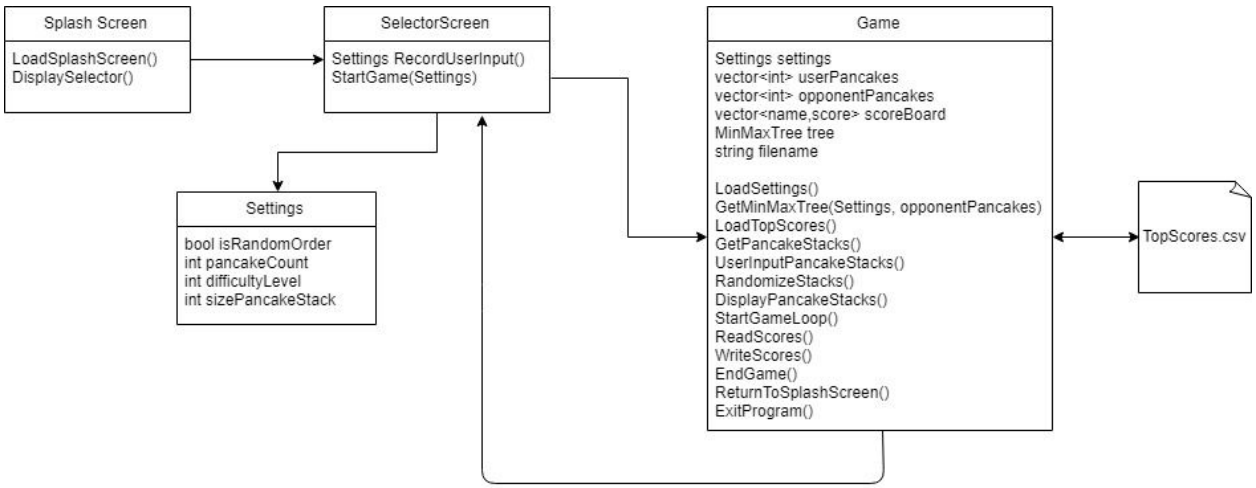
- Usage: The game controls will be the arrow keys on the keyboard. This will also the user to navigate, during their turn, to the position they would like to flip their pancakes.
- Model: When it is the user's turn to flip their pancakes the user will use the arrow keys to select the spot to flip and then hit enter to flip them. After the user selects enter on their keyboard, the pancakes will flash for three seconds and then the flip will be made.
- Interaction: The game is controlled by user input. Whenever it is the user's turn to select a pancake they are prompted to select one to flip. The program waits for the user input to continue running the program.

3.7. Task #7: Minimax Search

- Usage: Minimax is what will make up the AI aspect of the computer player. It is a type of backtracking algorithm that can be used in decision making in order to find an optimal move for a player.
- Model: The minimax algorithm will be implemented using a tree data structure. The algorithm will try all possible moves while storing it in the tree. After the construction of the minimax tree is complete, the algorithm backtracks and makes a decision.
- Interaction: The minimax algorithm will be used to help decide the computer player's next move in our game. Data of the initially randomized pancake positions will be fed into the algorithm so that it can construct the minimax tree, which is needed in order to find the optimal list of moves to win the game.

3.8. Task #8: Game Over

- Usage: Once either one or both stacks are sorted the game will end. The top 5 scores and the current score of the player will be sorted and the top 5 will be written to a file that will be read in the next time the game is played.
- Model: The top 5 scores will be read in from a file and stored into a sorted vector. Another function will check each stack to see if they are sorted or not after each turn and end the game once one or both are sorted at the end of the turn.



- Interaction: After each turn in the game both stacks are checked for correctness. If a stack is completely correct then the game is over. From there the game class will calculate the score and update it in the Top Scores csv file. Next the user will be prompted to either leave the program or play the game again. If the user decides to play again they will be directed back to the splash screen. The program will wait for the user to arrow to their choice and press enter to continue to the desired functionality.

3.9. Extra Task: Minimum Move Count

- Usage: On the screen there will be a string displayed that contains the minimum number of moves needed to sort the stack.
- Model: A function that finds the solution to the current stack will return the minimum # of moves needed to sort the stack and that output will be displayed on the screen.
- Interaction: The minimum move count display will interact with the minimax tree structure to find the shortest possible moves to complete the game. After each user's turn a new minimax tree would need to be made to find the shortest route to completion. The structure will count the depth of the first possible leaf in the structure and display that value on screen while the user is deciding their next move.

4. Benefits, Risks/Issues, and Assumptions

4.1. Benefits

- Maximizing usability of our program will make it easy for a broad range of users to play the game

- Creating an aesthetically pleasing splash screen and graphical user interface will attract users to play our game
- Emphasizing responsive and smooth game mechanics will make the game enjoyable to play
- Implementing a functional AI search will challenge users and give them incentive to come back and play
- A working system to display high scores will also contribute to our desire to create competition within our game

4.2. Risks/Issues

- Having each screen communicate through a window manager should allow variables to be passed within each screen, but there is a possibility all needed variables haven't been accounted for
- Since our team is new to the graphics library we are using, it will be a challenge for us to maximize the usability and look of our program
- Creating the best possible game mechanics will take research and getting used to ncurses.
- Implementing a functional AI search will pose risk to our program since we will have to implement a full minimax tree. This part of our project requires special attention to ensure we get an accurate AI search
- We must design our code to be free of global variables as a safe coding practice

4.3. Assumptions

- Our target users for the program is broad and the code should be user friendly assuming those not experienced in programming will also be playing our game
- We will be using text files to store the high scores for difficulty levels. We will assume writing to a text file is the same as writing to a disk
- Each screen we need to display will be contained in its own class, we will have a class for splash screen, difficulty selector, game, and high scores. These classes will be able to communicate to each other through a window manager.
- To implement AI, we will use a minimax data structure which we will write ourselves
- Using a minimax tree is the most optimal way for us to implement Artificial Intelligence for the user to play against