**Nikhil Vytla, Jason Gu, Gautam Kapoor**

Description of the Game >

Technology/Libraries Implemented >

Different classes & Data structures Used >

Class diagrams >

Game control integration >

**Description of the Game**

Connect4 is a basic two-player game that consists of each player trying to be the first to connect four of the same colored coins/tokens in consecutive order either vertically, horizontally, or diagonally. However, this is not as simple as it seems! While you attempt to connect four of your pieces, you must also block and/or be aware of your opponent’s moves.

The player is only allowed to place their pieces from the top, and the pieces then fill in the grid from the bottom-up (or in this case drop down due to simulated gravity). The grid itself, against the beginning Computer AI, will consist of a 6 row by 7 column grid, allowing for more than 4.5 trillion positions for all combinations of game boards, filled from 0 to 42 pieces.

Several variations do exist based on the game’s rules, including modes like “Pop Out”, “Pop 10”, and “5-in-a-Row”. We will primarily focus on the most basic version of Connect4, but watch out, there may be surprises to come!

Our game will implement networking in order to add multiplayer functions, and we will also include a Computer-based AI (with potential levels of difficulty).

**Technology/Libraries Implemented**

*Gridworld\**

The game GUI and design is based on Gridworld. We can represent our Connect 4 game board with a grid, and this makes GUI handling and implementation much easier. We will take the concept of using Actors in Gridworld and apply it to the player tokens.

This will be easy since the player tokens don’t even have methods of their own; they’re merely a placeholder in Connect 4. Another feature we will use in Gridworld is to animate the falling token, from the top of the board where the player dropped it to the closest free space at the bottom.

\*On the off case that Gridworld does not work, we’ll dive into creating our own grid using Java’s swing and swingx libraries.

*Networking*

We plan to create a networked Connect 4 where up to 4 players can play on a single game lobby. The board size will also increase as number of players increase.

Our networked game will feature a TCP design, as Connect 4 is a turn based game and TCP is more compatible for this type of game. We also plan to use networking to create a chat room for players.

**Different classes & Data structures Used**

*Connect 4 Game*

The Connect 4 game class simply creates instances of all the objects and runs the game.

*Game Logic*

Game Logic defines the rules the game and the game mechanics. The tokens and their behavior, for example, will be regulated by this class. The tokens can only be dropped into one column at a time and when dropped they must fall to the lowest available spot in the column.

*GUI Interface*

The GUI interface will make extensive use of Gridworld to map out the dimensions of the Connect4 world. Here, we will also program the token-dropping animations as well as the colors and look of the tokens and other behaviors (ex. The BOOM! Function from winning the game).

*Players Interactive*

The Players Interactive class creates an instance of the Connect 4 game. It includes methods for the player such as starting a new game, quitting, and win/lose/draw mechanics.

**Class Diagram**



**Game Control Integration:**

Connect 4’s integration is simplistic as we must track the computer mouse’s location on the game window, recognize when the mouse is clicked and over which column, and appropriately drop a token to the desired location.

We will implement a series of buttons onto the gui display that will allow users when clicked to reset the game and to add more players. When more players are added, the grid will automatically reset, increase its dimensions, and add a one more colored token to be played to accommodate for the new player.