**The College of Staten Island**

**Java Design and Implementation of Gomoku**

Lab5

CSC 330 - Object Oriented Software Design

Professor Richard Weir

To design and implement a 2-D based game that is entertaining to the user’s imagination. The 2-D based game should be flexible enough to easily incorporate a smart computer player.

**Project Priorities**

1. Code must be flexible enough to achieve Player vs Player (P v P) and Player vs Environment (P v E).
2. Code must include an abstract class, an interface class, and polymorphism.
3. Utilize JavaFX packages to design game.
4. Research AI intelligence for P v E.

**System Platform Requirements**

1. Programming Language: Java and JavaFX packages
2. Operating System: Windows 7 or up, macOS High Sierra or Up
3. IDE: Eclipse or NetBeans

**Project Approach : 2-D Based Gomoku Game**

Our team will deliver a 2-D based Gomoku game that will allow the user to select Player vs Player OR Player vs Computer. This strategy game will allow players to use their critical thinking skills as they battle against a computer or their friends, for ultimate entertainment experience. . Gomoku’s simplicity is suitable for all gamers of all ages.

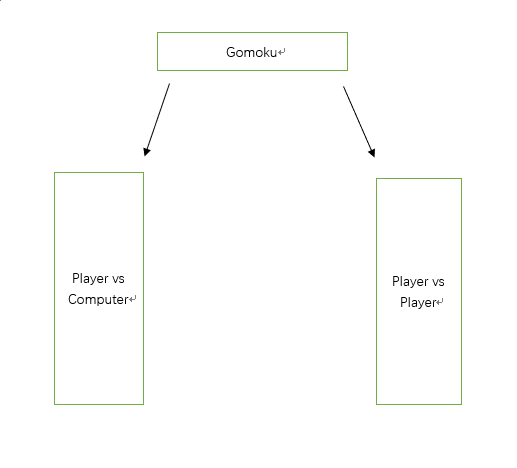
In order to deliver a final product, our group will be following the Agile Software Development approach. This methodology will not restrict us to our original design and ideas, but instead allows us to work together to deliver working code as quickly as possible. The Agile Process is suitable for planning tracking development progress of a Minimum Viable Product. The Agile Process allows for the opportunity to make improvements as we continue to optimize our requirements with proper testing and feedback.

**Planning**

Gomoku is a traditional Japanese board game for two players. The players take turns putting black and white stones on the board. The black stone plays first and each player alternates in placing a stone of their color on an empty intersection. The winner is the first player to get an unbroken row of five stones horizontally, vertically, or diagonally. The game must have a board that is approximately 20 x 20 in size.

1. Gomoku is a game that requires players to interact with the board. A function must be included to handle the actions for On-Click Mouse events.
2. There should be three buttons to serve the Players’ needs: Start Game, End Game, Challenge AI.
3. The game must include labels to display text to that will announce each move made and the winner.
4. Build a function that decides whether the top, bottom, left and right of the color stones pieces meet the five requirements in a row. If the condition is true, the winner will be announced. If the condition is false, the game should remain in progress.
5. The replacement of players will need to be automatic. Every time the board is clicked to place a stone, the player’s turn will alternate. The alternating between players will continue until one player wins.

**Analysis**

The game will consists of the following modes:

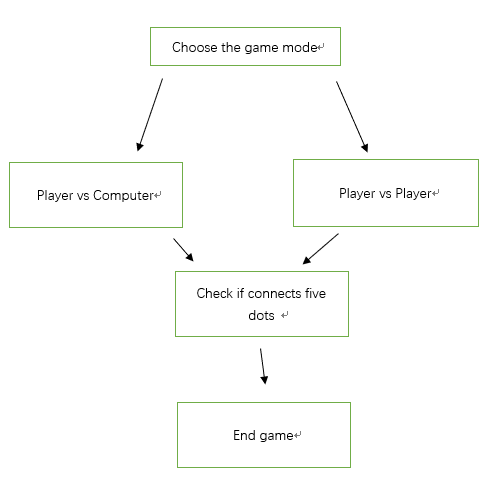
Player vs Computer

This mode will allow the user to play versus a computer player. Upon selecting this mode, the computer player will be the second player by default and will randomly select a place to set a stone. The computer player will play until a winner is prompted. After each stone is set, the board will decide if a player has won after yet. The board will prompt the winner and end the game.

Player vs Player

This mode will allow the user to play versus another player. The board will prompt which player will play first, then the players will alternate turns. After each stone is set, the board will decide if a player has won after yet. The board will prompt the winner and end the game.

**Design**

The Gomoku game will having the following game modes and design:****

|  |  |
| --- | --- |
|  | Initial design of board game. The game will begin with a blank board game and the button options. |
|  | The text will change as each player alternates turns.  As each stone is “placed”, the tile will change color. |
|  | The label will display text corresponding to each player’s move. |
|  | The text will change as each player alternates turns.  As each stone is “placed”, the tile will change color. |
|  | The label will display text corresponding to each player’s move. |
|  | If a player places five stones, the game text will announce the winner and end the game. |

**Implementation and Integration**

1. This game will have three options: "New Game", "New AI" and “End Game”.’
   1. New Game will allow for in Player vs Player.
   2. New AI will allow for in Player vs Environment.
   3. End Game will end the game.
2. The board panel must be over 20 x 20 in size to display well on the user’s screen. The button sizes must include JavaFX packages such as *Javafx.scene.control.Button* and *.Label* to create, track and call buttons. Creating a simple for loop will allow the buttons to remain on the screen until the game is over.
3. Rather than using JavaFX Pane to draw the board, we will use JavaFX packages to create buttons as main panels to track the player’s moment and calculate the computer player’s moves.
4. We will need to create a function with allocated memory to track if the current player has achieved five connected stones on the board. It will need to track each round with the corresponding player. We can set actions on buttons to track the game progress.
5. To achieve P v E, we must create a function that will randomly generate plays for the computer player. Using Math.Random can help achieve the randomness.
6. We must implement a function that can track the plays / turns and check for a winner.

**Take Away and Results**

The Agile Software Development Approach worked best for our team because it allowed us to utilize our strengths in self-organization and provide a structure for us to set tangible goals as we worked incrementally. Working in increments allowed us to predict the next steps and calculate the length of our tasks. By capturing our requirements at a high level with trivial designs, we gained a clear idea of the outcome we would like to achieve. This provide adaptability as we scale to improve our product and does not constrict us to our original designs and implementations. We logged our progress daily to reduce risk and ensure our team knows exactly what is complete and what is not.

Although the Agile Model shares similar methodologies as the Waterfall Model (Design, Planning and Analysis), the Agile approach was much more flexible for our team because working in increments allowed us to catch our error early with repeated testing. Working in increments allowed us to catch the following technical hurdles:

* + - 1. The initial board implementation could not handle the mouse trackpad movement.
      2. The AI could select a random title to place a stone but could not “click” the board on the GUI.

Catching these errors early, saved us future debugging time because it made the project details more visible and our code flexible.