UW Collegiate  
Computer Science 40S  
Graduation Project 2017

As a group create a GUI based Tic-Tac-Toe game. Success will be achieved if the game cannot be beaten by a human opponent.\

One stipulation: The game logic must be accomplished using a tree data structure.

Phase 1: Choose development language and environment, and version control

* + Java
  + Netbeans
  + Github
  + MVC design pattern

Phase 2:

3 division

1. Gui
2. Game logic
   1. Tic-tac-toe rules
   2. Each time a cell is clicked by the player or new state is returned by the model
      1. Gui asks game logic or model asks game logic
         1. Is this a legal move?
         2. Is this a win
         3. Is this a lose
         4. Is this a draw
         5. Is this a continue
      2. Controller sends message to model with new state asking for computer move
3. Machine logic
   1. How the computer decides its next move (must be done using a tree data structure)
   2. When a request is received the next move is computed and sent back to the controller

How will we approach the development of these three things?

* Division of labour model. ~~Redundancy? Can someone work on more than one module?~~
* Each member has one primary responsibility. It is expected that each member will participate in other areas as needed/they want to.

Who’s doing what?

* GUI: Joseph, Orion
* Game Logic: Colin
* Machine Logic: Thomas, Creston

Updates: first few minutes of each class, everyone has committed to arriving to class on time for the next 6 classes

Version control: Thomas, using git and git hub.

* Day 1: tutorial, get everyone onto github and create the project folder local and remote.

UW Collegiate  
Computer Science 40S  
Graduation Project 2017

Day 2: Thursday March 24

1. Thomas finished git tutorial
2. Interface discussion
   1. GUI passes x and y of proposed play (from player) to game logic
   2. game logic passes current state including proposed move to machine logic
   3. machine logic passes x and y of proposed move to game logic
   4. game logic passes new current state to gui
3. Formal declaration of division of labour
   1. GUI: Joseph and Orion
   2. game logic: Colin
   3. machine logic: Creston and Thomas
4. First updates take place on Tuesday March 28, announced and acknowledged.
5. Creston: machine logic
   1. get current state from machine logic
   2. create a tree of all possible moves and counter moves
   3. discard losing branches
   4. generate next move from possible winning branches
   5. pass new move as x and y (integers)
6. Colin: game logic
   1. maintain current state
   2. receive new proposed play move from gui
   3. check new move for;
      1. validity
      2. is it a win
      3. is it a lose
      4. is it a draw
      5. is it a conintue
   4. update state
   5. pass new state to machine logic
   6. receive new proposed play move from machine logic
   7. check new move for;
      1. validity
      2. is it a win
      3. is it a lose
      4. is it a draw
      5. is it a conintue
   8. update state
   9. pass new state back to gui

UW Collegiate  
Computer Science 40S  
Graduation Project 2017

Day 3: Thursday March 28

Updates:

Joseph and Orion: GUI

* Getting the gui to place correct symbol in response to a click
* symbol is immutable for run of game

Thomas:

* git managing, no update

Colin: Game logic

* created game board
* created methods for win and who won, lose, draw needs to be added,
* discussed the possibility of implementing the win test in 3 methods, one for rows, one for columns, and one for dagonals.
* This would create the need for another method to determine who won.

Creston: Machine logic

* nothing done yet