-Model stores X’s and O’s

-3 x 3 grid tic tac toe

-View = Graphics window, UI (gets used)

-Model = keeps track, and tells view

-Controller = Interacts w/ both ^^^

-Class diagram = Class names, methods

***User Interface (UI):***

Window 1200 x 800

3x3 Grid displayed

Colors (easy to see background, grid lines, X/O’s)

Create unique header “Tic Tac Toe”

Display who’s turn it is (i.e. Player 1 (X) or Player 2 (O))

Display when someone has clicked in the grid

Error messages for bad decisions (clicking in an already used square)

Who’s turn it is will be displayed

Who won the game

Who lost the game

Draw if no one wins

View ---->model------>controller

***Game Play Design:***

|  |  |
| --- | --- |
| ***USER*** | ***PROGRAM*** |
| Start Program | Present empty grid  \*\*Present who’s turn it is |
| Player clicks on a grid square | Game draws X  \*\*Game switches to ---> Player #2 (O’s) |
| Player 2 clicks on a grid square | Game draws O  \*\*Switches back to ------> Player #1 (X’s) |
| Clicks outside of grid | Returns nothing |
| Clicks on an already filled square | Returns Message: “Click another square that isn’t filled” |
| Clicks in grid square | Checks if player is winner |
| Clicks in grid square | Checks if players Draw |
| Player 1 wins | Displays winner message  \*\*displays loser message to P2 |
| Player 2 wins | Displays winner message  \*\*displays loser message to P1 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***Class Design***

**CONTROLLER**

*Instance Variables*

* Self.v
* Self.m
* Self.v.message
* Self.v.click
* self.m.isValid
* self.m.ControlX
* self.m.isWinner
* self.m.ControlO
* Self.v.done
* self.m.resetGrid
* self.playgame

*Methods*

* \_\_init\_\_
* playAgame
* Play
* ControllerTest

**MODEL**

*Instance Variables*

* Self.v
* Self.grid
* self.v.drawX
* Self.grid.pop
* Self.grid.insert
* Self.v.message
* self.v.drawO

*Methods*

* \_\_init\_\_
* isWinner
* isValid
* ControlX
* ControlO
* resetGrid
* ModelTest

**VIEW**

*Instance Variables*

* Self.win
* Self.items
* self.win.setCoords
* Self.one
* Self.two
* Self.three
* Self.win.getmouse
* Self.items.append
* Self.items.undraw
* self.win.setBackground

*Methods*

* Click
* \_\_init\_\_
* Message
* drawX
* drawO
* Done
* Reset
* viewtest