**Project Report:**

1. **Title Page**

* Project Title: FX Yut
* Authors: Ian, Jaiman
* Date Started: 2018/06/01
* Course: ICS3U1
* Last Updated: 2018/06/015
* Date Delivered: 2018/06/15

1. **Customer Requirements**

* Classic non-video game
* An FX twist

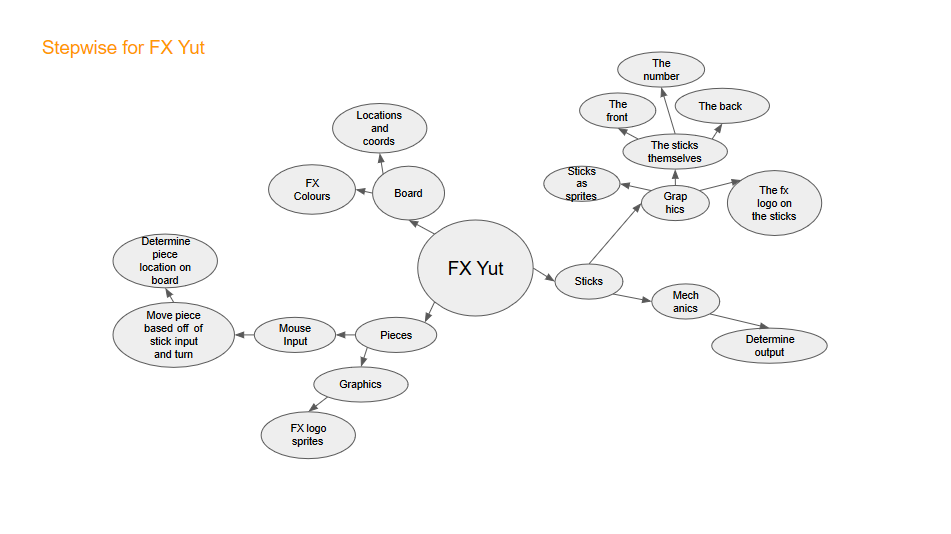
1. **Project Timeline**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Requirements** | **Design** | **Implementation** | **Testing** | **Deployment** |
| Proposed | June 1st | June 5th | June 12th | June 14th | June 15th |
| Completed | June 1st | June 5th | June 14th | June 14th | June 15th |

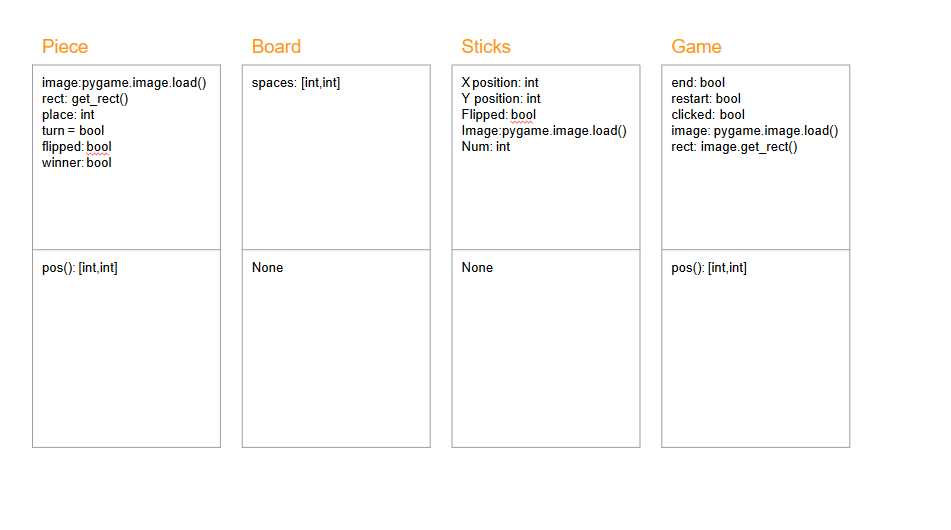
Implementation took a while longer than expected due to some problems with animation and piece movement. This resulted in the implementation being finished June 14th rather than June 12th. We had less time for testing as a result, although throughout implementation, we were constantly checking for errors and fixing them as they arrived.

1. **Design Proposal**

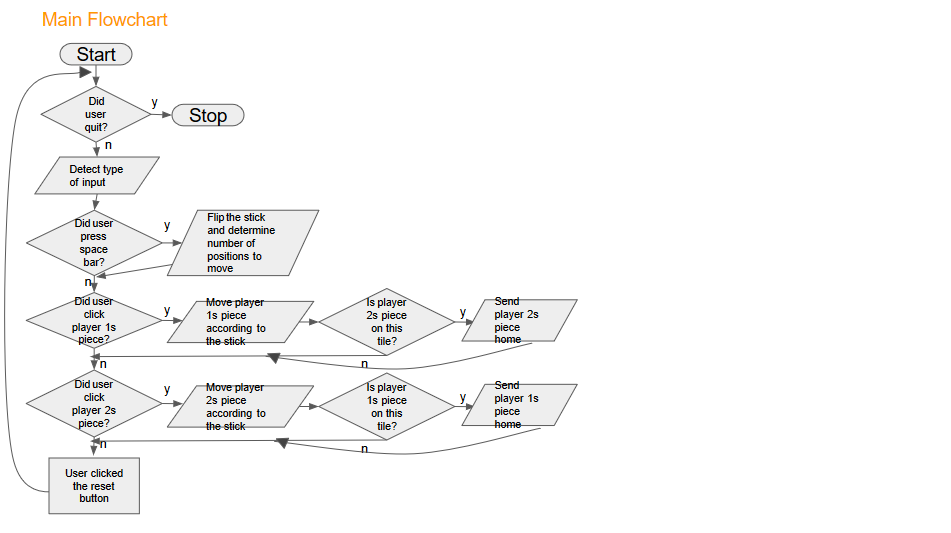
* Board
* Yut stick
* 1 piece per player (2 players)
* Stepwise



* UML

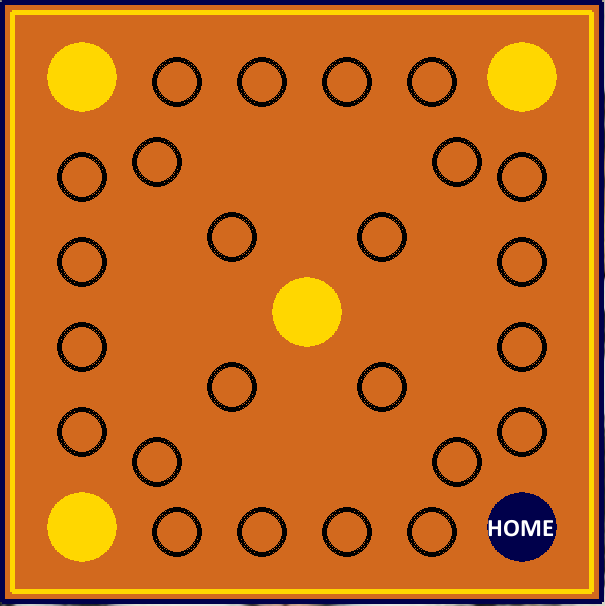


* Flowchart



1. **Implementation Details and Deliverables**

* The product we created was an edited version of the classic Korean board game “Yut.” Our product has the same board format as the original board game but has FX themed colours as requested by the client:



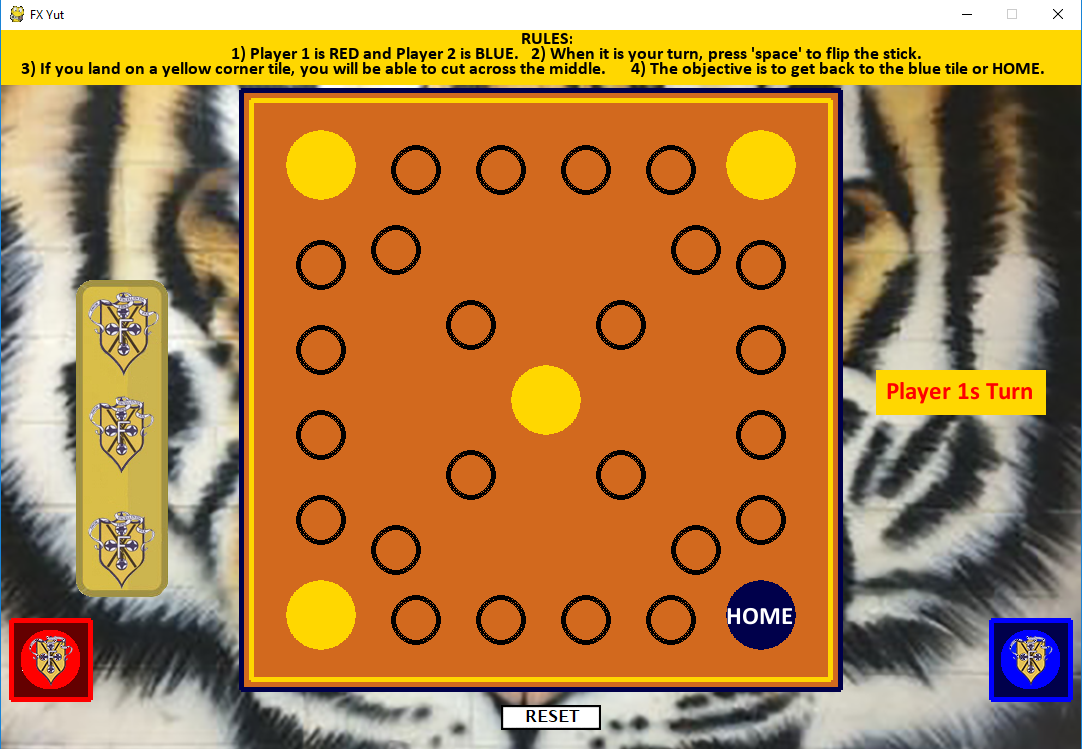
* This was done by using the pygame functions to draw the board and text. The board has a coordinate assigned to each spot which is done through the use of the class Board() which assigns the coordinates and creates a list of those coordinates in order for the pieces to be able to function.
* The project has pieces, one per player. This was done through the creation of a Piece() class with all the necessary functions to change positions based on the board locations and mouse input, and other attributes to determine the player’s turn, win status, etc. The mouse detection detects the mouse input using sprites and sprite collision detection and moves the piece according to certain cases. These cases are in the top right, top left and center spaces which alters the piece’s path. Areas that were capable of reaching the end were also cases that needed to be checked due to errors involving the spaces list. The moving of the pieces is done by checking certain conditions and moving using the random value given by the stick.

(Red is player 1 and Blue is player 2 and the surrounding boxes are part of the UI)

* There is a stick that flips to reveal a random value from 1-5. This value determines how many spaces the pieces move. This is done through the use of the class Stick() which contains the positions of the stick and the image and the random.randint() function. The flipping is done through detecting a keyboard input (space bar) and replacing the image on screen with the one off screen.

(The number that is displayed is determined by the random function.)

* Specific rules that needed to be assigned include; detecting if it is the player’s turn, detecting if one player’s piece has sent the other back to ‘home’, detecting the end of the game and resetting it. Detecting the player was done by assigning a boolean variable for both instances of the Piece() class with player 1 always being first, and changing it after a piece has been moved. Detecting if the player’s piece was sent home was done by testing for sprite collision between the two pieces and sending the piece back to the start depending on which turn it’s on. Detecting the end of the game was done with a boolean variable set to false until any of the pieces reached home. If the user clicked the reset button (which was defined in the class Game()), all the values at the beginning of the program would reset.



* The background is an image of a tiger which was blitted before all the drawings
* The rules of the game are displayed at the top of the screen

1. **Maintenance**

In this project we had many things go well along with a few things not go as planned. The things that went well were implementation of sprites, the turn mechanic, the whole UI and the stick mechanics to name a few. Something that we couldn’t add or didn’t have the time to implement was animations for the sticks and pieces. Some of the things that we could improve upon given the time is code efficiency and a more interactive feel to the game. Things that we were really proud of are; the drawing of the sprites and the overall functionality of the code, considering our limited experience.

1. **Resource Allocations**

|  |  |  |
| --- | --- | --- |
| **Task** | **Responsibility** | **Description** |
| Ian | Pieces/Mouse input | Involved the creation of the piece images, the implementation of how pieces functioned and how it interacted with the board and sticks. |
| Ian | Board | Involved the creation of the board’s visuals, the implementation of locations on the board and how it interacted with the pieces and sticks. |
| Ian | Game end/restart  and UI | Checked for the end of the game and allowed the user to restart the game if it ended. UI in terms of the turn display and restart button. |
| Jaiman | Sticks/Keyboard input | Involved the creation of stick images, stick functions and how it interacted with the pieces. |
| Jaiman | UI | Displayed the general rules of yut and the stick output |

1. **References**

* <http://programarcadegames.com/>
  + Pygame template
* <https://stackoverflow.com/questions/10990137/pygame-mouse-clicking-detection>
  + How to check for mouse click on the piece
* St Francis Xavier SS twitter
  + Profile picture tiger as game background
  + STFXSS logo