Project 6

Suresh Nayak
suresh.nayak@colorado.edu

Rajagopal Anandan
rajagopal.anandan@colorado.edu

GitHub Repo - https://github.com/sureshnnayak/Ludo-Game

Project Summary (Taken from proposal for context)

Ludo is an Indian-origin strategy game that 2-4 people can play. Each player in the game races toward the end position by moving any of their tokens out of 4 available tokens. The game would generally start with players choosing a color for their token and placing them in the home. A player should toss 6 in order to get the token to the starting position. Once the token is out, the player can move the token by boxes equivalent to the number obtained on the dice in each upcoming turn.
During the game, a player can send other players' tokens back home if they move their tokens to the same box the other player's taken was kept. There will be a few save point boxes where all players are allowed to place their tokens without sending other players' tokens back home. The game will end when all the players, except for one, move all four tokens to the end position.

Status summary:

Work Done:
1.We are following the MVC model
The Board is composed of 15x15 matrix where each cell is drawn using these parameters:

a. Color
b. Is safe
c. Connected cell

Based on the index value of the matrix, the board cell can be accessed, and tokens can be placed using the cell id.
The Game engine holds the model logic. We can manipulate the tokens based on the dice value with compliance with the constraints. places in the gameEngin.

Changes or issues encountered

1. Swift and ios application development was new to both of us. So it took us significant time to understand the general app development.
2. Some of the data structures in swifts were slightly different from what we knew before. For example, optional is a datatype that can have nil values. If we are creating a dictionary (analogous to default dict in python). The value returned upon quarrying a dictionary value is of type Optional. To use this data as an array or int, we have to convert it using “!”.

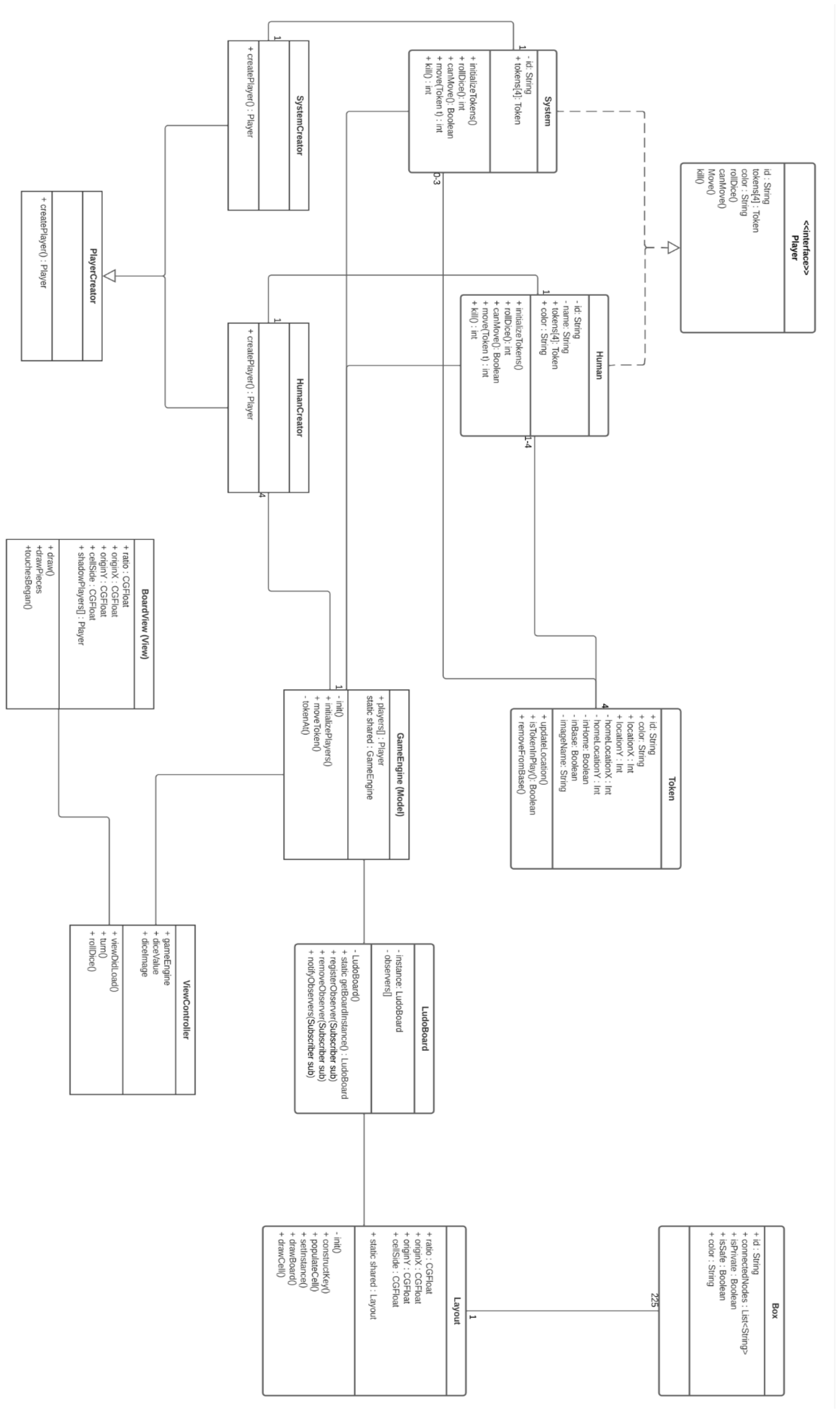
Patterns Used

1. Factory pattern - We have used factory pattern to create players
2. Singleton - We have used Singleton to have a single instance of the board (Layout class) and the Model(GameEngine class).

Plan for the next iteration

1. We want to make the UI elegant and smoother.
2. As of now, we have included two players and their tokens. We need to create two more optional objects to have more players in the game.
3. In the MVC Architecture, we are yet to establish the complete communication between the view and model units.
4. Game rules needs to be implemented
5. In the model component, the player will use the dice roll values to move the token. We want to make the tokens interactive and allow the user to select which token they want to move.
6. The view controller is updating the view based on the changes made in the board view in the next phase, we want to implement this as an observer pattern.
7. The connected cells and is_safe values need to be populated.

CLASS DIAGRAM



SystemCreator
+ createPlayer() : Player
PlayerCreator
+ createPlayer() : Player
HumanCreator
+ createPlayer() : Player
System
- id: String
+ tokens[4]: Token
+ initializeTokens()
+ rollDice(): int
+ canMove(): Boolean
+ move(Token t) : int
+ kill() : int
<<interface>> Player
id : String
tokens[4] : Token
color : String
rollDice()
canMove()
Move()
kill()
Human
- id: String
- name: String
+ tokens[4]: Token
+ color : String
+ initializeTokens()
+ rollDice(): int
+ canMove(): Boolean
+ move(Token t) : int
+ kill() : int
BoardView (View)
+ ratio : CGFloat
+ originX : CGFloat
+ originY : CGFloat
+ cellSide : CGFloat
+ shadowPlayers[] : Player
+ draw()
+drawPieces
+touchesBegan()
GameEngine (Model)
+ players[] : Player
static shared : GameEngine
- init()
+ initializePlayers()
+ moveToken()
- tokenAt()
Token
+ id: String
+ color: String
+ locationX : Int
+ locationY : Int
- homeLocationX : Int
- homeLocationY : Int
- inHome: Boolean
- inBase: Boolean
- imageName: String
+ updateLocation()
+ isTokenInPlay(): Boolean
+ removeFromBase()
ViewController
+ gameEngine
+ diceValue
+ diceImage
+ viewDidLoad()
+ turn()
+ rollDice()
LudoBoard
- instance: LudoBoard
- observers[]
- LudoBoard()
+ static getBoardInstance() : LudoBoard
+ registerObserver(Subscriber sub)
+ removeObserver(Subscriber sub)
+ notifyObservers(Subscriber sub)
Layout
+ ratio : CGFloat
+ originX : CGFloat
+ originY : CGFloat
+ cellSide : CGFloat
+ static shared : Layout
- init()
+ constructKey()
+ populateCell()
+ setInstance()
+ drawBoard()
+ drawCell()
Box
+ id : String
+ connectedNodes : List<String>
+ isPrivate : Boolean
+ isSafe : Boolean
+ color : String
1
1
0-3
1
1
1-4
4
1
4
1
225