Team 3 Architecture Document

This document contains a discussion on design choices made for milestone 2 in our Settlers of Catan project. The original UML from the project and a partially connected UML for milestone 2 are included along with a writeup. We recommend viewing the UML in browser at the following link:

https://github.com/yilmazaj/CSSE375Project/tree/main/SoftwareArtifacts

They are also available for viewing at the bottom of the document in much lower quality.

Main Design decisions for each feature

Feature 1. Added GUI card for dice rolls and player turn

When adding the GUI for player turns and dice rolling, we detached the simple random number generator from the game class and made it into its own Dice class. We could then pass a copy of this to the separate GUI class which allowed us to separate the classes a bit more as they didn't have to share the dice roll information directly. This new GUI class replacing the GUI built into the game class also drastically decreased coupling and gave us a greater cohesion between the two parts. In the future it should be fairly simple to swap out the GUI with another and allows us to expand upon the GUI easily.

Feature 2. Added GUI for displaying player stats

When adding the GUI for displaying player statistics, we first had to do away with the console logs displaying rolls, resource amounts, and other such related information. To create a new GUI with all of the player details, we had to create an instance of the PlayersStatsGUI Class (playerStats) in the Game class, and pass it the array of Players upon construction. From there, the PlayersStatsGUI creates individual PlayerStatsGUI Class instances for displaying each player's statistics, then compiles them together in a JFrame. From this point, the Game class simply needs to make a one line call to playerStats's method: updatePlayerStats() whenever resource amounts are changed, which iterates through each individual PlayerStatsGUI instance and updates resource amounts. These new GUI classes replace the console logs built into the game, and also keeps GUI responsibilities out of the game class. The single method call to update all GUI's and the flexible iteration creating individual PlayerStatsGUI classes means that making UI changes in the future should require no modification to Game, nor PlayersStatsGUI, and should only require changes to be made at the lowest level (PlayerStatsGUI).

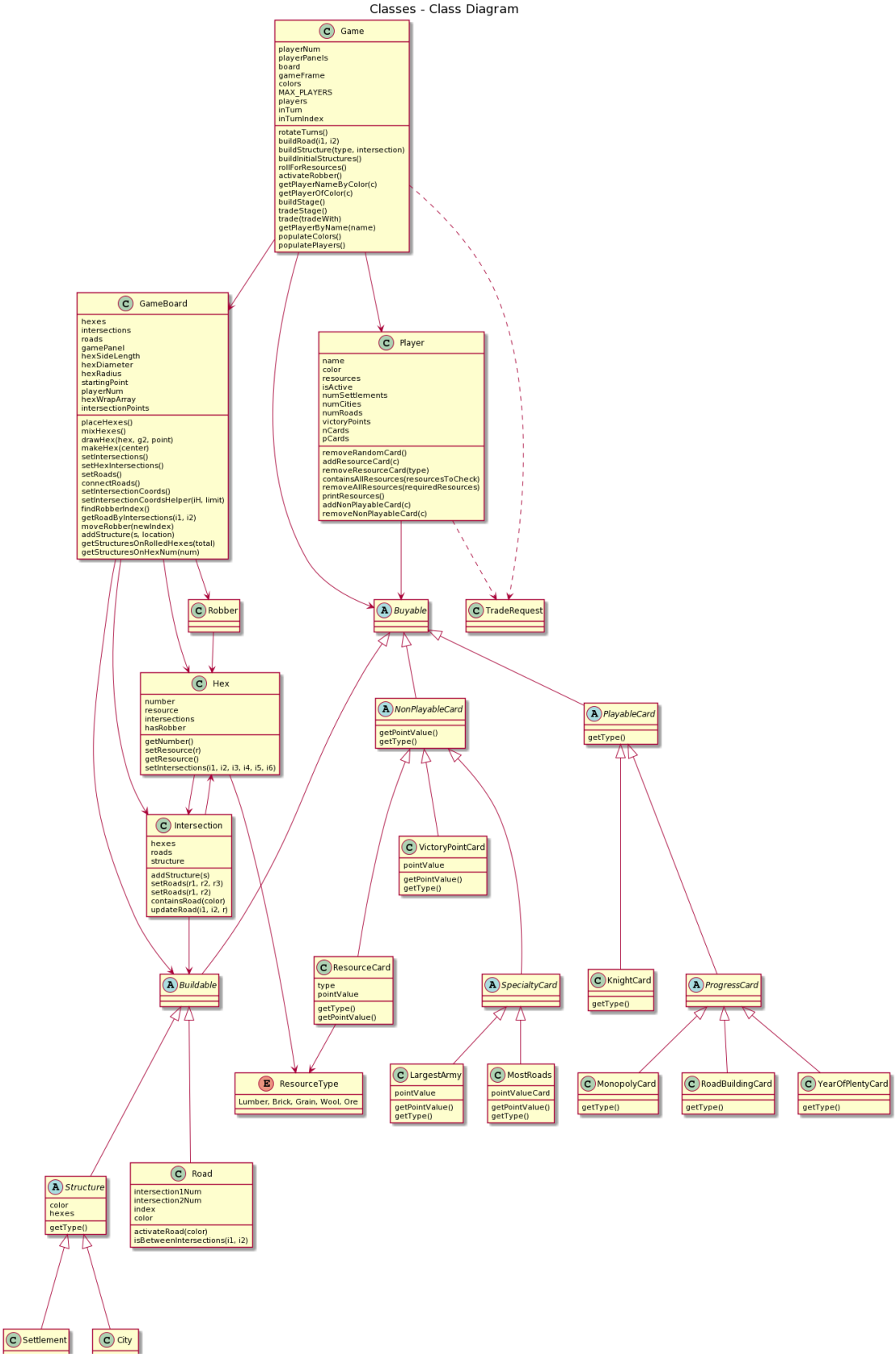
Feature 3. Added and improved icons for buildings and resources

For adding resource icons to the board we decided to use subclasses of an abstract Hex class to separate the different icons. We decided to use the string type of the hex as the icon's image name which allowed us to generalize the icon drawing to the superclass. We decided to draw the icons in the drawHex method because the resource hex icons only need to be drawn once. For the structure icons, we took a similar approach to resources, though we ultimately ended up collapsing the subclasses to the Structure class. The icon drawing is similar however, with the icon's image file having the structure's string type. We decided to call the structure icon drawing method in the paint component class, which would ensure that the structure icons are redrawn and updated as players build structures.

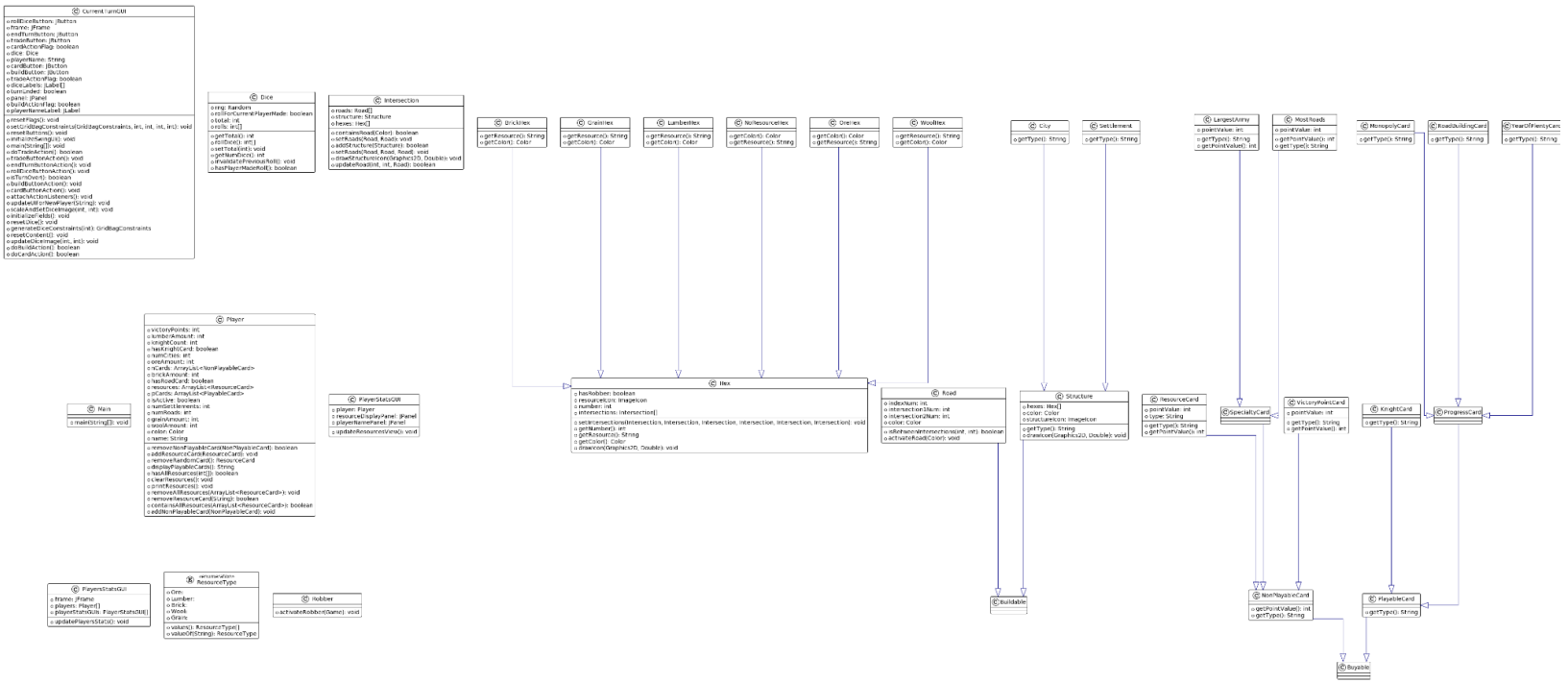
Feature 4. Added implementation for robber

The majority of the design decisions were already made in terms of Robber. There was already a class placeholder for the Robber. The main decision that I made was to move the activateRobber functionality from Game to Robber because it was an instance of feature envy. The activateRobber class already called the required methods, but the methods were not working correctly and certain methods were basically stubs. I filled these out with relevant code to get Robber to function correctly and implemented my own tests to verify. I also verified manually by running the game. That being said, implementing Robber did not involve many major design decisions.

Old UML


Classes - Class Diagram
Game
playerNum
playerPanels
board
gameFrame
colors
MAX_PLAYERS
players
inTurn
inTurnIndex
rotateTurns()
buildRoad(i1, i2)
buildStructure(type, intersection)
buildInitialStructures()
rollForResources()
activateRobber()
getPlayerNameByColor(c)
getPlayerOfColor(c)
buildStage()
tradeStage()
trade(tradeWith)
getPlayerByName(name)
populateColors()
populatePlayers()
GameBoard
hexes
intersections
roads
gamePanel
hexSideLength
hexDiameter
hexRadius
startingPoint
playerNum
hexWrapArray
intersectionPoints
placeHexes()
mixHexes()
drawHex(hex, g2, point)
makeHex(center)
setIntersections()
setHexIntersections()
setRoads()
connectRoads()
setIntersectionCoords()
setIntersectionCoordsHelper(iH, limit)
findRobberIndex()
getRoadByIntersections(i1, i2)
moveRobber(newIndex)
addStructure(s, location)
getStructuresOnRolledHexes(total)
getStructuresOnHexNum(num)
Player
name
color
resources
isActive
numSettlements
numCities
numRoads
victoryPoints
nCards
pCards
removeRandomCard()
addResourceCard(c)
removeResourceCard(type)
containsAllResources(resourcesToCheck)
removeAllResources(requiredResources)
printResources()
addNonPlayableCard(c)
removeNonPlayableCard(c)
Robber
Buyable
TradeRequest
Hex
number
resource
intersections
hasRobber
getNumber()
setResource(r)
getResource()
setIntersections(i1, i2, i3, i4, i5, i6)
NonPlayableCard
getPointValue()
getType()
PlayableCard
getType()
Intersection
hexes
roads
structure
addStructure(s)
setRoads(r1, r2, r3)
setRoads(r1, r2)
containsRoad(color)
updateRoad(i1, i2, r)
VictoryPointCard
pointValue
getPointValue()
getType()
Buildable
ResourceCard
type
pointValue
getType()
getPointValue()
SpecialtyCard
KnightCard
getType()
ProgressCard
ResourceType
Lumber, Brick, Grain, Wool, Ore
LargestArmy
pointValue
getPointValue()
getType()
MostRoads
pointValueCard
getPointValue()
getType()
MonopolyCard
getType()
RoadBuildingCard
getType()
YearOfPlentyCard
getType()
Structure
color
hexes
getType()
Road
intersection1Num
intersection2Num
index
color
activateRoad(color)
isBetweenIntersections(i1, i2)
Settlement
City

New UML


CurrentTurnGUI
Dice
Intersection
BrickHex
GrainHex
LumberHex
NoResourceHex
OreHex
WoolHex
City
Settlement
LargestArmy
MostRoads
MonopolyCard
RoadBuildingCard
YearOfPlentyCard
Player
Main
PlayerStatsGUI
Hex
Road
Structure
ResourceCard
SpecialityCard
VictoryPointsCard
KnightCard
ProgressCard
PlayersStatsGui
ResourceType
Robber
Buildable
NonPlayableCard
PlayableCard
Buyable