2. Class Diagram

demo

UserInterfaces

FXGLMenu
fireResume():void
fireExitToMainMenu
fireExit():void

MainMenu
- selectedButton: ObjectProperty<Button>
- SELECTED_CLOR: Color
- NOT_SELECTED_COLOR: Color

GameMenu
- handleSaveClick():void

<<interface>> LoadSave
+ getSaveFilePath(int slotIndex): Path
+ load(int slotIndex): void
+ save(BufferedWriter writer, int slotIndex)
+ stringTypeMapping(String animalTypeString): AnimalType

LoadSaveUI
+ mainMenuContainer: VBox
+ gameMenuContainer: VBox
- loadContainerInitialised: boolean
- saveContainerInitialised: boolean
- application: FieryDragonsApplication
- slotDescriptions: String[]
- slotTexts: Text[]
+ loadNewGame(): void
+ handleLoadSlotClick(int slotIndex): void
+ showSaveMenu(): void
+ showMainMenu(): void
- getSaveFilePath(int slotIndex): Path
- readSaveFileMetadata(): voif
- createNewSaveFileMetadata():void
- handleSaveClick(int slotIndex): void
- hide(VBox container)

utils

Utils
+ generatePairs(int min, int max):ArrayList<int[]>
+ shuffleIntArray(ArrayList<T> array, long seed):void

<<interface>> EntityFactory
+ <methodName> (SpawnData data):Entity

<<interface>> InitModel
+ createModel(SpawnData data):void

EntityFactory

<> SpawnFactory
spawnData: SpawnData: SpawnData
+ spawn():void

ChitCardFactory
- coveredChitCardShapes: Circle[]
- chitCardAdapter: ChitCardAdapter
+ newChitCard(SpawnData data):Entity
+ getRandomCoordinates(): List<int[]>
- createCoveredChitCardFilledShape(): Circle
- createUncoveredChirCardShape():Circle
- spawnLoaded(): void

TextFactory
+ winMsg(SpawnData data):Entity
+ turnUpdateMsg(SpawnData data):Entity
+ movementCountMsg(SpawnData data_:Entity
+ playerPointsMsg(SpawnData data): Entity

CaveFactory
- circle_radius: double
- playerTurnManager: PlayerTurnManager
+ loadCaves(SpawnData data):Entity
+ newCaves(SpawnData data): Entity
- createCaveShapes(double x, double y, int i):Circle

DragonTokenFactory
- DRAGON_TOKEN_IMAGE_PATHS: String[]
- playerTurnManager: PlayerTurnManager
+ newDragonTokens(SpawnData data):Entity
+ loadDragonTokens(SpawnData data):Entity
- createPlayerIDLabel(double x, double y, int id):Text
- createTokenImage(double x, double y, int idx)

ButtonFactory
- exitBtn: Circle
- textDesc :Text
- EXIT: ImagePattern
- EXIT_WHITE: ImagePattern
+ exitBtn(SpawnData data):Entity
+ setListeners():void

VolcanoSegment
- segmentCards: Volcano[]
- counter: int
+ addVolcanoCard(VolcanoCard volcanoCard):void
+ getSegmentCards():VolcanoCard[]

VolcanoRingFactory
- volcanoRing: Volcano[]
- segmentLength: int
- numCards: int
- numSegments: int
- rotationAmount: double
+ getVolcanoCardByID(int id):VolcanoCard
+ newVolcanoRing(SpawnData data):Entity
+ loadVolcanoRing(SpawnData data): Entity
+ createSegments():VolcanoSegment
- createVolcanoCardShape(double x, double y, VolcanoSegment segment, int j, int cardCounter)" Rectangle

Controller

ChitCardAdapter
- viewModelMapping: HashMap<Circle, ChitCard>
+ addMapping(Circle chitCardView, ChitCard chitCard): void
+ getViewControllerMapping():HashMap<Circle, ChitCard>

DragonTokenManager
- closestTokenDiff: int
- dragonTokebs: DragonToken[]
- counter: int
+ addDragonToken(DragonToken dragonToken)
+ findClosestDragonToken(DragonToken dragonToken):DragonToken
+ getClosestDragonTokenDistance():int

PlayerTurnManager
- RAND_ANIMAL_CHOICES: ArrayList<Integer>
- players: Player[]
- currPlayer: Player
- playerTurn: int
- winner: Player
+ getPlayers(): Player[]
+ getCurrPlayer(): Player
+ handleTransition(): void
+ checkWinCondition: boolean
+ getWinner(): Player

TextDisplayManager
- currentTurnTextEntity: Entity
- currentMovementCount: Entity
- gameEnd: boolean
+ getInstance():TextDisplayManager
+ displayTurnUpdateMsg(int currPlayerID): Entity
+ displayWinMsg(int currPlayerID): void
+ displayMovementCountMsg(int currPlayerID, int totalMovement): Entity
+ initialize(int currPlayerID):void
+ handleEndGame(int currPlayerID):void
+ handleTurnTransition(int currPlayerID):void
+ handleMovementCountUpdate(int currPlayerID, int totalMovement)
+ displayPointsMsg(int currPlayerID, int points): void

ChitCardFlipManager
- chitCardFlipManager: ChitCardFlipManager
- coveredChitCardShapes: Circle[]
- uncoveredChitCards: ArrayList<Circle>
- setCoveredChitCardShapes(Circle[] coveredChitCardShapes):void
+ getInstance(): ChitCardFlipManager
+ handleUncover(Circle chitCardChosen, ChitCard chitCardChosen):void
+ resetOnClickListener(HashMap<Circle,ChitCard> mapping):void
+ handleTurnEnd(): void

FieryDragonsApplication
- endGame: boolean
- playerTurnManager: PlayerTurnManager
- chitCardAdapter: ChitCardAdapter
- textDisplayManager: TextDisplayManager
- loadSaveUI: loadSaveUI
- loadSaves: ArrayList<LoadSave>
- slotToLoad: int
initSettings(GameSettings gameSettings):void
initUI(): void
initGame():void
onUpdate():void
+ main(String[] args):void
+ registerLoadSave(LoadSave loadSave)
+ loadGameState(int slotIndex)
- handleCircleClick(Player player, Circle chitCardChosen): void
- playTurn(Player newPlayer):void

Animals

<<enum>> AnimalType
SALAMANDER,
BABY_DRAGON,
BAT,
SPIDER,
DRAGON_PIRATE,
LEPRECHAUN

<> Animal
- baseImgPath: String
- count: int
- animalType: AnimalType
+ getAnimal(AnimalType animalType, int count):Animal
+ fillUIShape(Shape uiShape):void
+ calculateDestinationID(Player player, DragonToken dragonToken)
# getSuffixedImgPath():String

BabyDragon

Bat

Salamander

Spider

DragonPirate
+ calculateDestinationID(Player player, DragonToken dragonToken)
# getSuffixedImgPath(): String

Leprechaun
+ calculateDestinationID(Player player, DragonToken dragonToken)

Model

Player
- dragonToken: DragonToken
- doNothingContinueTurn: boolean
- ID: int
- turnEnded: boolean
- animalType: AnimalType
- points: int
+ validateMove(Circle cardChosen, ChirCardAdapter chitCardAdapter)
+ moveToken(int animalCount,int destinationID, boolean isSwap): void
+ setEqualityBoost(boolean hasEqualityBoost): void
+ useEqualityBoost(): void
+ loadSaveString(): String

VolcanoCard
- ringID: int
- animalType: AnimalType
- occupied: boolean
- xPos: double
- yPos: double
+ setOccupied(boolean occupied):void
+ getOccupiedStatus():boolean
+ getAnimal():AnimalType
+ getRingID():int
+ setAnimal(AnimalType animal): void
- getxPos(): double
- getyPos():double

DragonToken
- totalMovemenrCount: int
- initialVolcanoCardID: int
- tokenImage: Rectangle
- currentVolcanoCard: VolcanoCard
- movedOutOfCave: boolean
- playerLabel: Text
- currPosAngle: double
+ moveToken(int animalCount):void
+ updateTotalMovementCount(int steps):void
+ getTokenImage(): Rectangle
+ getCurrentPositionInRing(): int
+ setCurrentPosition(VolcanoCard newVolcanoCard)
+ getPlayer(): Player
+ loadSaveString(): String

ChitCard
- index: int
- animal: Animal
- coveredShape: Circle
- uncoveredShape: Circle
- x: double
- y: double
- coveredVisible: boolean
- uncoveredVisible: boolean
+ getAnimal(): Animal
+ getX(): double
+ getY(): double
+ isCoveredVisible():boolean
+ isUncoveredVisible(): boolean

Cave
- playerID: int
- xPos: double
- yPos: double
- animalType: AnimalType
+ getyPos():double
+ getxPos():double
+ getPlayerID():int
+ getAnimal(): Animal