Marcia Gallant and Loryn Losier

October 30 2020

Justifications:

We have a card class because we need to know what card the player picks and the view class needs to know which cards it should display.

We have a crate, robot jewel and stonewall class that extends tile because it uses the majority of Tile's methods. This makes our job easier since if we only have Tile class then when we need to add a stone wall, we have to manually change that this tile can't be moved to and what tile type it is. Now, we can just create an object of that type and those things are done for us. We can create a Tile[][] array and store crates, stonewalls, tiles and robot jewels where needed. We set the tile type to either Normal, Crate, Stone Wall or Robot Jewel depending on what type of tile it is.

We have a `findAndSetMovableDirectionsForCrates(tileList[futurePos[0]][futurePos[1]]);` **method in the GameBoard class which determines and sets which directions a player can move onto the tile with tileType = Crate from. This is because a crate may be movable in 1 direction because it has an empty tile behind it from that direction and it may not be movable from another direction if a crate, stonewall, turtle or the edge of the gameboard is behind it. That's why we must check if it can be moved onto or not before deciding if the turtle can move there.**

In Stone Wall, we set movable directions all to false since you can never move onto a tile with a stone crate.

We decided to not make robot turtle extend tile class because then every time a turtle moved we need to create new tiles to replace where the turtle used to be. Also, in the future if we need to extend our program and add the laser card which will create puddles, it will be confusing since they we may need a tile to be a puddle or turtle simultaneously. We will not run into this problem since Robot Turtle are just stored in a Robot Turtle variable in the Tile class and there is also a Boolean variable occupied in the Tile class which allows us to quickly see which tiles have turtles and which ones don't. So, we just change whether a Tile is occupied or not and set the Tile robot turtle to the one that just moved. We also make occupied false on the old tile that the turtle used to occupy. Each Tile has the ability to contain a robot turtle, so, if the Tile is occupied we can easily access which Robot Turtle is on that tile.

In Robot Jewel we had a getJewelColor method which would have been used for the view class when drawing the jewel since it needs to know which colour. We couldn't call this method, so, we put a getColour method in the Tile class.

We added a GameBoardConverter class which converts the objects of classes in the model into Strings or Integers. The GameBoardController class now passes GameBoardDisplay an object of GameBoardConverter after updating what the model currently is in the GameBoardConverter instead of a Tile[][] object. This allows the GameBoardDisplay to get the model information without creating any dependencies on the display to the model classes. Instead, GameBoardConverter has those dependencies and the display only has dependencies on the controller classes now. This is in case we

make any updates to the Tile[][] we only have to fix how it's converted and do not have to go searching and adding stuff in our display.

Robot turtle update turtle position method is only for the case where a player plays the move forward card. Since the turtle position should never increase by more than 1 at a time. This means that Turtles must be put in their correct positions when first created.

We decided to get rid of the Turtle Master class since it wasn't really doing much for our program.

Our GameBoardModel keeps track of many variables which keep track of the initial robot jewel positions, colour and same for robot turtles. This is important because without it we couldn't set up the game. It also keeps track of GAME_BOARD_DIMENSION which is needed because it tells gameboard how big our Tile[][] should be. Without it we wouldn't know how many tiles need to be on the board and we wouldn't be able ton construct a 2d array keeping track of the board state. It also keeps track of number of players, which players have already won, whose turn it is and the current state of the game which is either in progress (playing) or finished. It also figures out who goes next. It's important to know how many players so we can figure out whose turn is next. This is because we do not want our game thinking it's player's 3 turn when there's only 2 players. Keeping track of whose turn it is important because if player 3 has already won, we don't want player 3 to still be getting turns. We want player 3 robot to just stay on the jewel tile and not move anymore. So, knowing whose already won allows us to skip their turn and go to the next player's turn. Knowing if the game is in progress helps us to know if we should continue figuring out whose turn it is and letting them pick a card to move or end the game.

GameBoardModel also has a moveTurtle method which is really important because it needs to find out if the turtle is even able to move as the card given as a parameter says to. If the turtle is able to move, gameboardmodel must update the x,y positions in the turtle itself and must change occupied var and assign the robot turtle to the new tile in the gameboard. ELSE BUG CARD NEEDS TO GO.

It creates the starter robot turtles and jewels itself based on how many players there are. It has methods which allow the controller to add crates and stonewalls. It moves the given turtle based on the card that's passed to it. It creates a basic card deck for the basic game consisting of step forward, turn right, turn left and bug.

Our GameBoard class is responsible for the Tile[][] 2d array. It creates an empty gameboard consisting of blank tiles for the whole board.

It also adds Crate, Stone Wall, Robot Jewel to the 2d array and sets occupied and stores the robot turtle in the tile the turtle is currently on. It also has a method which is used for updating the turtle's position on the board.

We had to add getColour method to the tile class because we need to get the colour of the jewel in the gameboard view class but it wasn't letting us use the getJewelColour method because the view gets passed a tile[][] and it doesn't let you use jewel class methods on objects in a tile[][] even if they are robot jewel objects because it can't be 100% sure it is one.

We have Player enumeration class which is important because it allows the GameBoardModel to tell GameBoardController whose turn it is. That way the GameBoardController can tell the display to notify the users of whose turn it is. It's also important because the controller must pass the correct robot turtle in with the correct card and the controller only knows what robot turtle to pass into the moveTurtle method if they know whose turn it is.

We have the State enumeration class which allows the model to tell the controller when the game is done and all players finished. It's important because it tells the controller when the game ended, so, the controller can end the display accordingly.

The gameboardcontroller class is important because it receives all user input, such as which tiles they pressed to put crates or stonewalls. It passes this input to the GameBoardModel which puts crates or stonewalls in the correct locations. The GameBoardController also receives card input of which card the user choose which is important because without that information, the model wouldn't know how to move the robot turtle and the game wouldn't progress or work. It allows the user a chance to play the bug card or confirm their card choice. It tells the display to tell the users whose turn it is. It tells the display how to draw the current state of the game by passing a GameBoardConverter which as mentioned previously, converts Tiles and Robots into Strings and Integers since the display doesn't know about Tiles and Robot Turtle objects. It also tells the display when the game ended.

The gameboardcontroller also finds out how many players are in the game which is very important because a GameBoardModel can only be constructed with this information since it affects how many turtles and jewels there are in the game. It also affects whose turn it is since if there's only 3 players, it will sip over player 4 turn since player 4 doesn't exist. It finds out how many players through the numPlayersScreen class.

The GameBoardDisplay is important because it draws the current state to the screen. This allows the user to see the board and know what's going on in the game. It also allows users to see which card they're picking or which tile they're picking to put a crate or stone wall in. It also lets the user see where the robot jewels are which is important because the players want to know what to do to win which is get their turtle on a robot jewel tile which they can only do if they see where the robot jewel is. The GameBoardDisplay also tells the user whose turn it is and this is important because the users may get confused and forget whose turn it is and the wrong player may choose the card for a different player which would be unfair. It also gives popups explaining which tiles you cannot put crates or stonewalls, how many crates and stonewalls you need and a popup when everyone's won the game.

The numPlayersScreen class is important because we need to show the users the options of how many players can play this game at a time. We made this a separate class from the display because we need to receive this information before we can construct a GameBoardModel and without a model we cannot draw the current game.

Overall, we have deviated greatly from our original milestone 2 design: eliminated TurtleMaster class, added two enumeration classes Player and State, added a GameBoardConverter and added a NumPlayerScreen.