Meagin Arrocha

Prelim Project 4

I used protected methods in Monster so that the ones the Inky, Dinky, Stinky, and Clyde might share. They can call if from Monster. For example, Clyde uses Inky’s movement for when he is vulnerable, so if I do what I need to do in Monster, they can both use that same method instead of writing the same thing in both classes. The same goes for Stinky when he uses Clyde’s normal state movements for his normal state as well. I also made an int counter in Monster under protected, because each Monster will have a counter for vulnerability. I also made Booleans isNorm, isVuln, and isGoHome so they may return true or false of the state the Monster is in. Each of the Monsters can be in one of these three states and I think these will help me when I start making my methods to easily switch from one to the other or to easily check certain cases. I made a player pointer in Monster also because it needs each Monster needs to be able to get what level player is on so each monster can know how long to be vulnerable for.

I made a virtual bool isOnPlayer() so that every Monster can check if they are on Player for every tick. I also made move(), getDisplayChar, setNormal() and setVulnerable() virtual because each Monster is going to need to do these things but they will each do them differently. I explained what each does next to them in the header file. I left resetDirection() and returnToHome() non-virtual because each Monster will reset to the same place and each Monster will returnToHome the same way (the optimal path back to home).

For determining if GacMan has ever eaten a Monster or vice versa, I think I will do something along the lines of:

-checking every tick

Bool isOnPlayer(){

If(the monster’s getX() == Player -> getX() && the monster’s getY() == Player -> getY())

Return true;

Return false;

}

//then if it returns true check for monster’s state

//I will probably do something along the lines of this in Game only if isOnPlayer returns t true

If(the monster’s state == vulnerable){

//update player score

Player -> inScore(); //where I’ll have a case for if he ate a monster

//send monster home

returnToHome();

}

If(the monster’s state == normal){

//decrement player lives

Player -> decrementLives();

//check lives – because game could be over

Player -> getNumLives(); //if returns 0 then game is over

//if not then reset all actors

Maze -> resetDisplay();

//have them press enter to start again

Console -> printString(0, 24, “You have “);

Console -> printInt(10,24, player -> getNumLives());

Console ->printString(13,24,”left. Press Enter to Continue”);

Console ->waitForEnter();

}

Which brings me to Maze; I a bool getNextCoordinate(int curX, int curY, int &newX, int &newY) so that way I can get the Monster home on an optimal path (explanation in header). I also made a char DFSarray[21][23] so that I can put the depth for search chars in here, the numbers and letters, so that I may store them and easily compare them to get the Monster home.

I did not add anything to Player, but I do have a setDead() that I can more easily implement now. Also, isDead won’t always be false now that there are things to kill me. I’m going to attempt to do all the comparisons (as far as if Monster is on Player) inside of Monster (that’s why I put the Player pointer inside of Monster).

I did not add anything to Game but I did notice I had a getMazeContent function that I haven’t used and I think I’m going to use that later to get the Monster’s state; so when, or if, the Monster and Player meet on the map I can determine what function to call next depending on what state that Monster was in.

Inky does not have any extra methods or data at this moment because all of his methods are being passed down from either Actor or Monster.

So far, I’m thinking for Dinky I am going to allow him to see the whole map. This way he may make his decisions based on where GacMan is (for his normal state). For his vulnerable state, if it is allowed, I’m thinking about making him invisible. So that way the player doesn’t know where he is and when his vulnerability wears off he might appear next to the player ☺ (unless that is considered teleporting).