Scott Forsberg
CSC 143
2019-10-19

Homework 2 Report

Planning and Design

Six Unit Pieces

The original piece class was divided into an interface Piece, a superclass AbstractPiece, and several individual piece subclasses SquareShape, BarShape, SShape, JShape, LShape, and ZShape.

Each shape class extends the AbstractPiece class, and the AbstractPiece class implements the Piece interface. The bulk of the code from the old Piece class was transferred to the AbstractPiece class. An empty rotate method is present in the Piece interface, and implemented in AbstractPiece class.

Each piece class has a series of x and y offsets, with square 1 representing the origin where x,y = 0,0. Each other square is represented as an x,y offset from the origin, for example square 0 of the LShape would be at x,y=0,1. These offsets are sent to the constructor of the superclass along with the specific color of the given piece, the superclass being AbstractPiece.

The constructor of AbstractPiece then creates new square objects using the x,y coordinates as offsets which are combined with the initial row and column values.

Piece Rotation Math

The first step was capturing the col,row of square 1, and using these values as offsets. Each subsequent square then also had it's col,row information gathered, and then the offsets were subtracted from these values. The resulting numbers are then the x,y values of a Cartesian coordinate plane.

For each square y value (row) was then set to (-1)* as the rotated x value (col). Then, the x value (col) was set to equal the y value (row). This results in a -((pi)/2) rotation for all coordinates “about” the origin piece 1. Finally the x,y values for each square are subtracted from the offsets to get the resulting row and column positions for each square.

Piece Rotation Programming

The keyPressed method of the EventController class had a case statement matching clause added, which invokes the handleRotate method also in the EventController class. This class then invokes the rotatePiece method of the Game class, which sends a message to the rotate method of the piece object. The rotate method located in the AbstractPiece class then invokes the getRotatedCoords method which returns a multilevel array representing each square and the corresponding row,col values.

The canRotate class of the AbstractPiece is then used with an input of the rotated array, and uses the canRotate method of the Square class to individually check each square. If all of them can rotate a Boolean true is returned to the rotate class of the AbstractPiece class, and then runs the rotate class against each square. This class only needs to set the row and col values to the input, as these identifiers are declared in the Square class.

Unit Testing

The testRotation unit test first creates a grid object, and then populates it with pieces. After each test, the rotated variable is checked. The rotated boolean was created in the AbstractPiece class and set to true when the successful rotation code is executed within the class.

First an LShape piece is created with no nearby obstrucitons, and the rotate method is called followed by the assertTrue unit test. Next a shape is placed next to the wall, rotated, and checked to see that it did not rotate. Various shapes are checked, and the square is checked to assure it did not rotate. The code appears to be bug free and operating as intended.

Project Evaluation

This project involved common and useful principles of object oriented programming. Subclasses extending superclasses, and implementing interfaces, as well as the separation between model view and controller were employed to created a functional program. In particular I observed the benefits of reusing common code within a superclass. Also, the project involves separation of methods between classes via polymorphism. It uses methods with the same name in different classes, which allows for a categorization of similar nationalities.

This project was certainly worth the effort, as it represents a framework for user interactive applications. Nearly any kind of software may be created in this way, responding to key presses and displaying pertinent information.