ASSIGNMENT 1

Name: Tay Shao An
UID: u7553225
Laboratory time: Wednesday 5:00pm
Tutor: Pramo Samarasinghe

INTRODUCTION

Assignment1 is a simple art program which is able to create pictures using 6 different types of shapes through mouse clicks and key presses.

PROGRAM DESIGN

MODEL.HS - Defines all relevant data types which are used throughout the program such as:

> Shape

> Tool

> ColourShape - A data type which holds a colourName and Shape

> Model - A data type which describes the state of the program by keeping track of all shapes generated, the current tool and current colour selected. This data type is called through handleEvent in Controller.js when a new user input is taken.

VIEW.HS - Takes a Model, provided by Controller.js, and converts it into a codeworld Picture to be displayed on the screen.

> modelToPicture - The “main” function which calls all other functions in order to convert a Model to a Picture

> toolToLabel - Takes a Tool as an input, and returns its description. This description is displayed on to the canvas through modelToPicture

> colourShapesToPicture - Takes a list of ColourShapes and converts them to Pictures through calling the function colourShapeToPicture on each ColourShape

> colourShapeToPicture - Takes a ColourShape, and calls the codeworld function “coloured” to colour a Picture.

> colourNameToColour - Takes a ColourName and returns a Colour

> shapeToPicture - Takes a Shape and returns a Picture

CONTROLLER.HS- Handles all user inputs and generates a new Model.

> handleEvent - handles all user interactions with the programs including key presses and mouse clicks. Generates a new Model based on each event.

> nextColour - Takes a ColourName and returns the next ColourName in order.

> nextTool - Takes a Tool and returns the next Tool in order.

ASSUMPTIONS

> To ensure that the eccentricity is set as close as possible to the user’s desired value, it is set by default at 0.500
> Coordinate Plane was removed so as to give the user a bigger canvas.
> To ensure that a similar ellipse can be drawn repeatedly, the eccentricity is kept the same after each ellipse is drawn
> Eccentricity’s lower bound is set at 0.
> Colours can be changed in between a pointerPress and a pointerRelease

TESTING

> Simple functions like colourNameToColour & toolToLabel can be tested by running cabal v2-repl comp-1100 assignment1, providing an input and checking the output.
> To ensure modelToPicture works, I had to break it down into its various functions being called within it.

1. For the shapeToPicture function, I tested its functionality by comparing my image generated with the actual sample provided.
2. ColourShapeToPicture can be tested by inserting an input and checking the output

> Tools are tested by drawing the shapes in various directions and seeing whether the shape generated is the one imagined by the user.
> Key Presses, specifically the “+”, “-” and spacebar, are tested by pressing all relevant keys and watching for the change in model by pressing d on the keyboard.
> Edge cases were tested such as:

1. Trying to change eccentricity when current Tool is not ellipse
2. Pressing spacebar when current Tool is not polygon
3. Changing Tools while pointer has not been released
4. Changing Colours while pointer has not been released
5. Drawing Shapes continuously
6. Pressing backspace when no other shapes are left
7. Increasing eccentricity beyond the lower or upper bounds

REFLECTION

> Encountered problems:

1. Creating an Ellipse and Triangle from the shapeToPicture function was especially difficult. To solve this, I used a helper function in order to group together long blocks of repeated code so as to visualize the problem easier.
2. At the start, I used generic and bad names for variables and I became confused as to what I was referring to. This made me realize the importance of naming conventions

> If I were to redesign the program from scratch, I would:

1. Designate a specific part of the canvas for a toolbar so that the text which informs the user the current tool selected and current colour will not be covered up by a black Picture.

2. Add a variable to store the last inserted shape to allow for redoing an undo.
3. Package functions together. Eg. removing the function nextTool and nextColour and add each of their codes directly into handleEvent.
4. Remove the ColourName datatype, change each function to handle the Colour datatype instead, and directly package the Colour into the ColourShape datatype.