Technical Report for Assignment 1 - COMP1100

Aryan Odugoudar

Student Information

- **Name:** Aryan Odugoudar
- **Laboratory Time:** Friday 8:00am - 10:00am
- **Tutors:** Jess Allen and Madeleine Stewart
- **University ID:** u7689173

Contents

1 Introduction

In this technical report, I provide an overview of my implementation for Assignment 1 in COMP1100. This assignment involves creating a program that allows users to draw various shapes on a canvas, change colours, and utilize different tools for drawing. The program is structured using the model-view-controller pattern, with Haskell as the programming language.

2 Analysis of the Program

2.1 Program Features

My program provides the following features:

1. Drawing various shapes, including lines, rectangles, polygons, semicircles, and ellipses.
2. Changing the colour of the shapes being drawn.
3. Switching between different drawing tools.
4. Removing the last added shape (Undo functionality).
5. Completing the drawing of a polygon by pressing the spacebar.
6. Rotating semicircles anti-clockwise or clockwise.
7. Adjusting the stretch of an ellipse in two directions.
8. Displaying the current state of the program (useful for testing) by pressing 'D'.

2.2 Implementation Details

2.2.1 Model.hs

'Model.hs' defines all relevant data types used throughout the program.

Shape

- **Description**: Defines the Shape data type, which represents various geometric shapes such as Line, Rectangle, Polygon, Semicircle, and Ellipse.
- **Purpose**: To encapsulate the properties of different shapes within the program.

Tool

- **Description**: Defines the Tool data type, which represents the currently selected drawing tool (e.g., LineTool, RectangleTool).
- **Purpose**: To keep track of the user's current drawing tool.

ColourShape

- **Description**: Defines the ColourShape data type, which combines a colour (ColourName) and a shape (Shape).
- **Purpose**: To represent a shape with its associated colour for rendering.

2.2.2 View.hs

'View.hs' is responsible for converting the Model into a CodeWorld Picture for display.

modelToPicture

- **Description**: The main function in 'View.hs', it takes a Model as input and converts it into a CodeWorld Picture.
- **Purpose**: To generate the visual representation of the program's current state.

toolToLabel

- **Description**: Takes a Tool as input and returns a Text description of the tool.
- **Purpose**: To provide clear instructions for the selected drawing tool.

colourShapesToPicture

- **Description**: Takes a list of ColourShapes and converts them into Pictures.
- **Purpose**: To render a list of coloured shapes into a single Picture.

colourShapeToPicture

- **Description**: Takes a ColourShape and uses the ‘coloured‘ function to colour a Picture with the specified colour.
- **Purpose**: To render a shape with a specified colour.

colourNameToColour

- **Description**: Takes a ColourName and returns a Colour corresponding to that name.
- **Purpose**: To convert a colour name into a CodeWorld-compatible colour.

shapeToPicture

- **Description**: Takes a Shape and returns a Picture representing that shape.
- **Purpose**: To convert a shape into a graphical representation.

2.2.3 Controller.hs

‘Controller.hs‘ handles user inputs and generates a new Model based on these inputs.

handleEvent

- **Description**: Handles all user interactions with the program, including key presses and mouse clicks. It generates a new Model based on each user event.
- **Purpose**: To update the program’s state in response to user actions.

nextColour

- **Description**: Takes a ColourName and returns the next ColourName in a predefined sequence.
- **Purpose**: To determine the next colour to use when drawing shapes.

nextTool

- **Description**: Takes a Tool and returns the next Tool in a predefined sequence.
- **Purpose**: To handle tool switching and ensure that tools change appropriately based on the program’s state.

3 Rationale and Reflection

3.1 Design Choices

- I chose to implement the ‘toolToLabel‘, ‘nextColour‘, and ‘nextTool‘ functions as simple pattern matching tasks since they don’t involve complex list manipulations or recursive logic.
- For the colour transition, I decided to use a predefined sequence of colours to ensure a smooth transition from one colour to another.
- When implementing ‘nextTool‘, I considered the scenario where the user is in the middle of drawing a shape and ensured that the tool state remains unchanged in such cases.

3.2 Assumptions

- I assumed that the order of colour transition from red to blue was not critical as long as the colours were distinct.

4 Testing

4.1 Overall Testing

I tested the program as a whole by running it and using various keyboard and mouse inputs to ensure that it responds correctly to user actions. I checked if shapes were drawn accurately and if tools and colours changed as expected.

4.2 Function Testing

For function testing, I conducted extensive testing of all program functions, ensuring that they perform as intended and handle edge cases appropriately.

- `colourNameToColour` and `toolToLabel` functions were tested by running `cabal v2-repl comp-1100 assignment1`, providing inputs, and verifying the correctness of the output.
- To test the `shapeToPicture` function, I compared the generated image with actual samples provided. This ensured that shapes were accurately represented.
- `colourShapesToPicture` was tested by inserting inputs and verifying the correctness of the output images.

Additionally, I tested edge cases,

- Pressing the spacebar when the current Tool is not a polygon.
- Changing Tools while the pointer has not been released.
- Changing Colours while the pointer has not been released.
- Drawing shapes continuously to check for any unexpected behavior.
- Pressing the backspace key when no other shapes are left.

5 Conclusion

In conclusion, this technical report provides an overview of my implementation for Assignment 1 in COMP1100. It discusses the program's features, implementation details, design choices, assumptions,and testing procedures. My implementation successfully meets the requirements of the assignment, and the program functions as expected.