CMPT 381 Assignment 4: 2D Graphics Transformations

Due: Friday, March 16, 11:55pm

Overview

In this assignment, you will use Android and touch interaction to build a system that supports interactive 2D transformations on graphical objects. You will also build on your experience with Model-View-Controller, and will implement a more complex Controller that can deal with selection handles as well as objects.

Part 1: Non-interactive transformations

Build a simple app in Android that programmatically creates three triangles and displays them on the screen. The triangles store values for translation, scaling, and rotation – for part 1, these are set programmatically, not interactively. Each triangle is displayed at the correct location, size, and rotation given its transformation values.

Interface requirements:

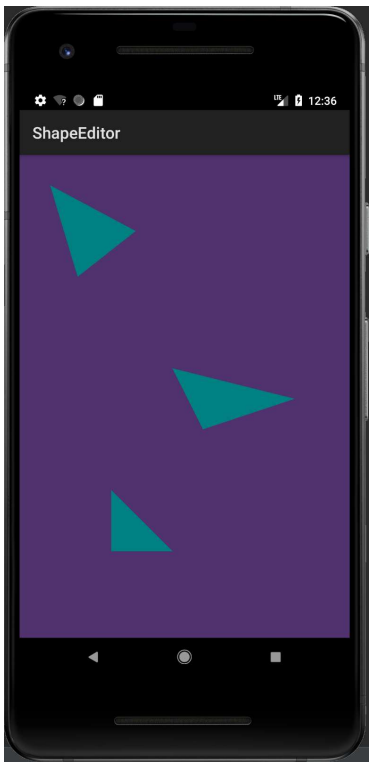
- A custom drawing panel that fills the main part of the screen (no scrolling required); this is where the triangles appear
- Three teal triangles are displayed at the correct translation, scaling, and rotation for their assigned values
- The triangles can have any (triangular) shape, but must appear entirely on the screen

Interaction requirements:

- When the user touches on one of the triangles, it changes colour to pink, and its transformation values are displayed at the top left of the canvas, as shown in the figures below.
- When the user clicks on the background, the selection is cleared and the triangle returns to its original colour.

Software requirements:

- Implement the system using Model-View-Controller, with correct separation between these components (however, it is fine for Triangle objects to have a draw() method that is called from the view)
- Create separate classes for the Model, the View, the Controller, and the InteractionModel
- Implement publish-subscribe communication to notify Views of changes to the Model and iModel
- Implement your controller as a state machine
- The triangles are created in the constructor of the TriangleModel class; a triangle's transformation values are also set here
- You should create classes MainActivity, TriangleModel, Triangle, TriangleView, TriangleViewController, InteractionModel, and interface TriangleModelListener
- Triangles should have the following methods to set the transformations:
    - setTranslate(float tX, float tY)
    - setScale(float sX, float sY)
    - setAngle(float radians)



App showing three triangles

Values for selected triangle

- Triangles should be created and stored using coordinates that centre the object at the origin (i.e., at coordinate 0,0)
- The centre of a triangle can be considered the centre of the triangle's bounding box
- Triangles should also store display coordinates that are adjusted based on the transformations
- When calculating display coordinates, apply the rotation transformation first, then scaling, then translation.
- The system should work correctly on an emulated Pixel 2 device, at Android API level 27

How to draw a triangle in Android:

- There is no Canvas.drawTriangle() method in Android, so you will need to create a Path object
- Create a new Path in each Triangle, and whenever you need to draw the triangle, recreate the path using the triangle's display coordinates. Start by calling the path's rewind() command to reset the path, then moveTo() the first display coordinate, and then lineTo() the remaining coordinates of the triangle.

Resources for part 1:

- Tutorial for custom Views in Android: https://developer.android.com/training/custom-views/index.html
- Equations for the three transformations are shown in the course slides (note that rotation is in radians, not degrees)
- API for Android Path class: https://developer.android.com/reference/android/graphics/Path.html

Part 2: Selection handles and interactive transformation

In the second part of the assignment you will extend your system from part 1 to allow interactive control over translation, scaling, and rotation. When a triangle is selected, it now shows two handles: one at the top-left of the triangle for scaling, and one at the bottom-right to allow rotation.

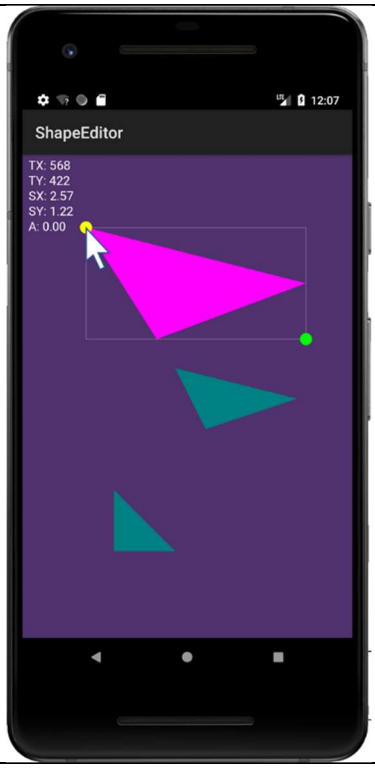
scaling at the center of the bounding box, a line from center to left corner, square x, y scaling
bounding box a fixed property of triangle, store 4 points of the BB, display cornor??
x' = cos(a)*x - sin(a)*y
y' = sin(a)*x + cos(a)*y

Additional interface requirements:

- Touching a triangle now additionally shows the bounding box of the triangle, and the two handles

Additional interaction requirements:

- Dragging inside the triangle changes the translation values for that triangle
- Dragging the scaling handle scales the triangle appropriately for the amount of dragging in the X and Y dimensions
- Dragging the rotation handle rotates the triangle to the relative angle of the touch point to the centre of the triangle (i.e., such that dragging the handle in a circle around the triangle will result in one full rotation of the object). Note that the bounding box rotates along with the triangle.
    - The distance from the touch point to the centre point is not used
- The display of transform values in the upper left of the canvas dynamically changes as the selected triangle is manipulated.

		
Top triangle selected and dragged (cursor shows touch location)	Yellow handle dragged to scale triangle	Green handle dragged to rotate (touch point need not stay on the handle)

inside Triangle: 2 points, 4 lines, 4 bb points(can be done in a vertex class)

Additional software requirements:

- As in part 1, triangles should store "true coordinates" that are centred around the origin, and then calculate "display coordinates" based on the transforms
- You will also need to store true coordinates and display coordinates for the bounding box
- Your controller will now need to check whether the user has touched on a handle, and will need to manage the new states of translating, scaling, and rotating.
- As in part 1, it is fine to incorporate some display elements in the model (in particular, the Triangle class) including the display coordinates and methods to display the triangle, bounding box, and handles.

Hit detection on triangles:

- There are several ways to determine whether a point is inside a triangle, and you can use any method.
- A simple approach is to compare the area of the original triangle with the sum of the areas of the three triangles that exist between the touch point and each pair of the original triangle's points:
    - Given point p and triangle with points a1, a2, a3:
    - A = area of triangle a1, a2, a3
    - B = area of triangle p, a2, a3
    - C = area of triangle a1, p, a3
    - D = area of triangle a1, a2, p
    - If A is equal to (B+C+D) then p is inside triangle a1,a2,a3
    - (in practice, we use: if A – (B+C+D) < 1 to avoid precision errors)

This assignment is to be completed individually; each student will hand in an assignment.

What to hand in

- A zip file of your Android Studio project folder for **either** part 1 (if that is as much as you have completed) **or** part 2. If you have completed all of part 2, you do not have to hand in a project file for part 1. If you have completed some of part 2, but the system does not run correctly, you may hand in two zip files (clearly label these as part 1 and part 2).
- A readme.txt file that indicates exactly what the marker needs to do to run your code. (Systems for 381 should never require the marker to install external libraries).

Where to hand in

Hand in your two files (one zip and one readme.txt) to the link on the course Moodle.

Evaluation

Marks will be given for producing a system that meets the requirements above, and compiles and runs without errors. Note that no late assignments will be allowed, and no extensions will be given, without medical reasons.