Lab 8 Tangram – Research Document

Ben Millar

Narrative Description

Ben ran the program, and he saw on-screen an assortment of coloured shaped, numbered 1 through 7, comprising FIVE triangles of varying sizes, ONE square, and ONE parallelogram. In the background was the silhouette of a house, which Ben realised he could create using these shapes.

Instructions along the bottom of the screen told Ben he could select a shape with the number keys, pan it up, down, left and right with the arrow keys, and rotate it in place using the + and – keys.

Ben pressed the '1' key on his keyboard, and the triangle marked 1 was highlighted; he then used the arrow keys to move it into position over the silhouette and rotated it clockwise into a position he was happy with using the + key. Ben then pressed the '7' key on his keyboard, highlighting the square marked 7. As with the previous shape, he moved it over the silhouette of the house and placed it in the position he thought it would go.

Not being an 1818 Parisian, Ben quickly became confused, and decided to check the solution by pressing and holding the 'S' key. While Ben held the 'S' key, the silhouette in the background changed into a full colour image, showing where each piece was supposed to go.

Inner Workings

At its foundational level, this game comprises of a relatively simple system: An array of vertices stores a set of vertices (these are essentially a modelled as a 3-dimensional vector, with some additional colour and texture information) each representing the points of a shape. This means that, for our 7 shapes comprising 5 triangles and 2 quads, we need keep track of (5x3) + (2x4), or 23 points in space.

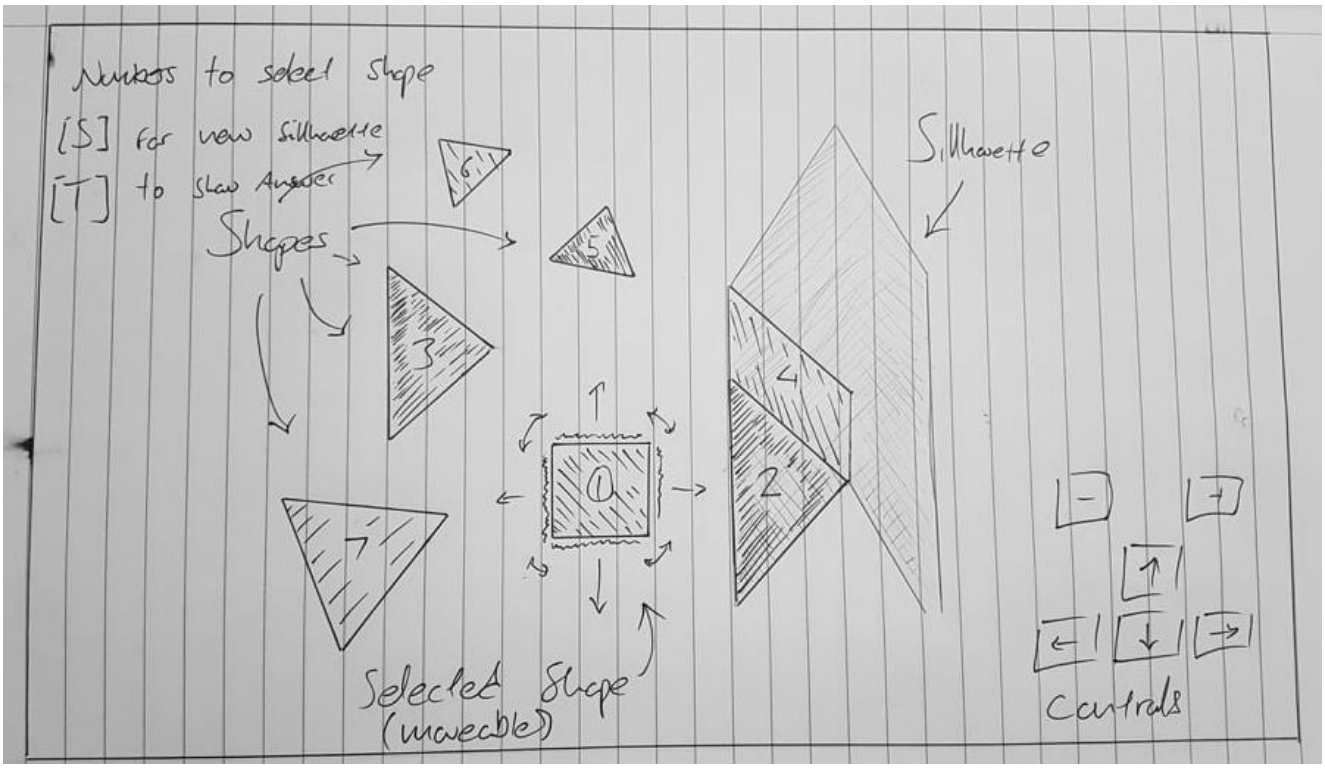
These are then passed into an SFML VertexArray object, which is a construct allowing more things to be done with our vertices. In this case, we'll use it to classify our vertices into **triangle** and **quad** primitives; this forms each group of 3 or 4 vertices respectively into a contiguous shape when displayed on-screen.

In essence, we turn our vertex points into a set of triangles and quads (one square, one parallelogram).

From a gameplay point of view, these points are manipulated using 3D matrices. For example, we can use a translation matrix to translate a point about the XY plane; similarly, we could construct a rotational matrix to rotate a point about its origin. This concept can be extended and applied to all points on a given shape in order to move and rotate this shape on the screen.

This is how the game works, with the user using their keyboard to select and move each of the shapes as they wish.

Sketch


Numbers to select Shape
[S] for new Silhouette
[T] to slow Angular
Shapes
Silhouette
Selected Shape
(moveable)
Controls