Problem Solving Task A

IGB283 – Assessment 01

Due: Firday, 15th September 2017, 11:50pm

Weight: 30%

**Group Memebers:**

Ethan Farley

Jared Carey

# Script/Class Structure

## Scripts

GameMaster = controls the rules of the program

MatrixObject = Creates mesh

IGB283Transformation = includes the necessary transformations functions i.e. translation, rotate, scale

MouseDrag = handles clickable events, which handles the mouse position variable which “GameMaster” reads from to set the y position of the moveables objects when clicking on their accompanying objects (the small objects)

Matrix3x3 = Used to create matrices. Functions from this are primarily used in “IGB283Transform” class, and one function in the “MatrixObject” class called “SetWorldTransforms()”

# Task 1

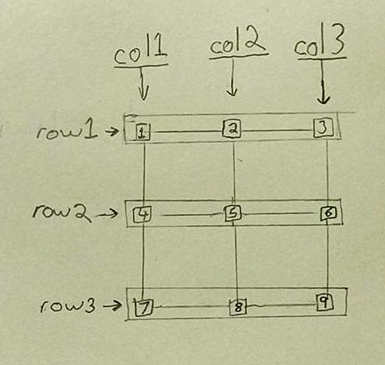
## Saving Vertices to an array

### Summary

For creating vertices for my object, I establish some variables that would be used in the creation of certain shapes, but for ease of use I decided to only go with even triangle/quad counts to avoid complications to form a square or rectangle depending on the sizes set for the object.

### General flow of creation of verts

Since the quad columns vs quad rows are exactly the same length/width (essentially squared), the verts are created by rows of verts. So for example if we have 8 triangles we would have 3 verts in a vertices row. Since the quads are evenly squared by x and y directions, we would have 3 columns of verts and 3 rows of verts. So in saying that my logic goes from column 01, row 01 and increments by column and then row, after all verts in a row have been made. See below example:



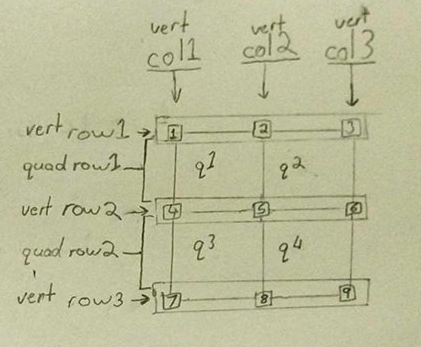
So essentially, verts get created from 1 to 9, on a 4 quad mesh or 8 triangle mesh. Verts get stored into the actual mesh itself, while also being stored into another array for later use in transformations.

### Code that creates the verticestask01 - Example01

Figure 1 Located in the MatrixObject script

## Build triangles from vertices to show mesh

For creating triangles from the vertices, I essentially create each triangles in each quad first. Starting from index 0 of the vertices array. See example below



q = quad

The logic will run through creating each triangles by going through their each quad. So each quad is looped through once to create the triangles for a single quads. So for example, to create q1, the index order will be 1-2-5, then in the same loop to create the second triangle it is 5-4-1 indexes.

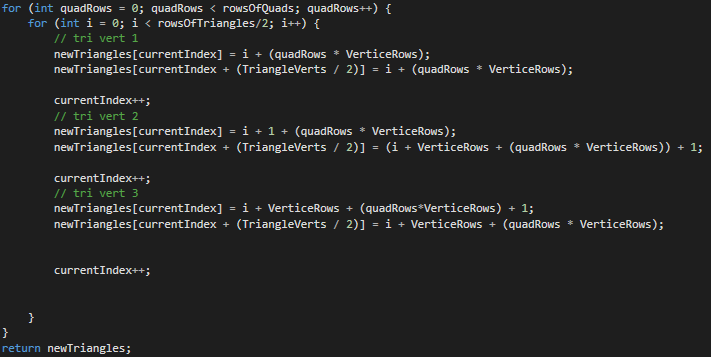
Code;

Figure 2 Located in the MatrixObject script

## Transformation Class

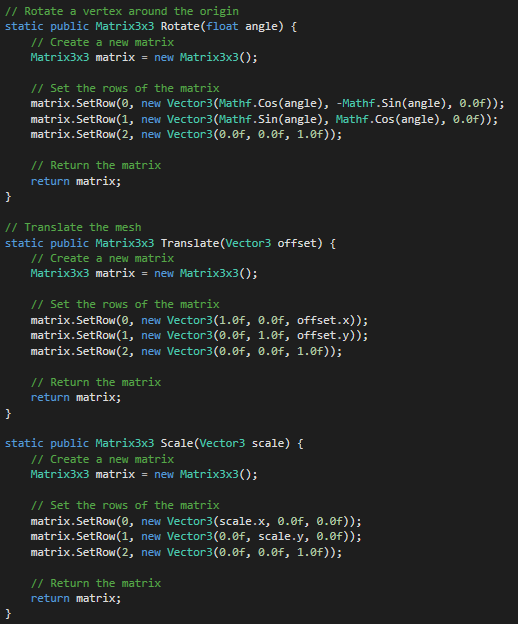


Figure 3 Located in the IGB283Transforms class

## Translation and Rotation of object using self-made transformation functions

### Code:

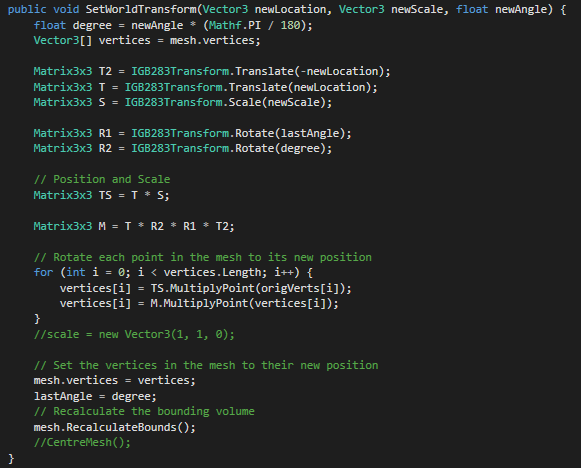


Figure 4 Located in the MatrixObject sript

This function allows me to set the transformations of a meshes vertices by calling the function the IGB283Transformations class. This allows me to easily position, rotate and scale every tick without any issues, except for performance hit. Every tick because the function is called within the update of the object so that’s it transforms are always being set.

task01 - Constant Rotate

Figure 5 Located in the MatrixObject script

Above function when called constantly updates the angle variable.

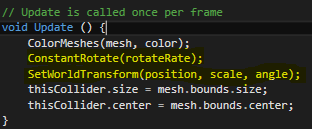


Figure 6 Located in the MatrixObject script

Then above, the angle variable is used in the SetWorldTransform() to update the actually movement of vertices in the mesh.

# Task 2

## Duplicate object with different speed rates on Angle and Horizontal Speed

The following I created in a separate class called “GameMaster”, which handles the randomisations of angles and horizontal speed, sets y movement and duplication of objects.

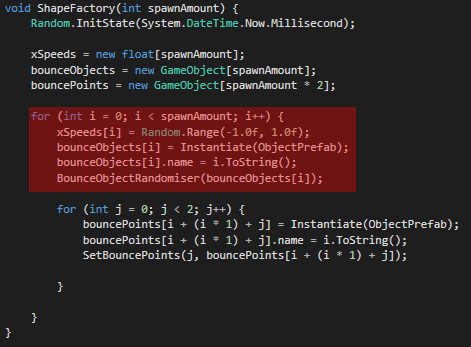


Figure 7 Located in the GameMaster script

Highlighted code in red duplicates the moving object

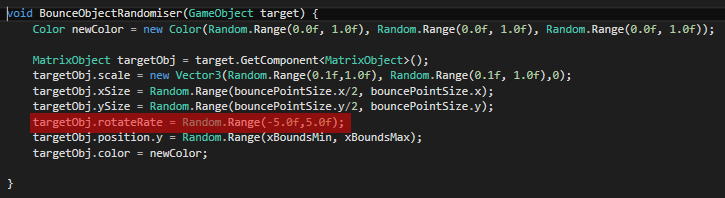


Figure 8 Located in the GameMaster Script

Highlighted part above randomizes the variable associated with the angle speed

# Task 3

## Input

For changing the horizontal speed on the moveable objects, I store the speed rates of the object and then multiple by a global speed multiplier for the x axis.

The multiplier is changed by input from either pressing “+” (“num+”) to increase speed, or pressing “-“(“num -”) to decrease speed. There is maximum to how much the multiplier can do down or up to.

### Code for Speed increase/decrease on key press

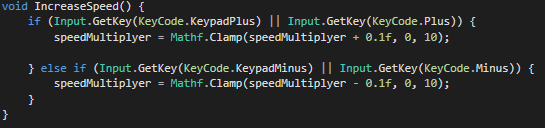


Figure 9 Located in the GameMaster script

## Colour

For this, colour changes based on the colours of the bounce points. Using a colour linear interpolation it takes both of the colours of the bounce points and then using another value to scale between the two colours depending on the location of where the object between the two objects. See below image on the progression of the moveable object between two points.

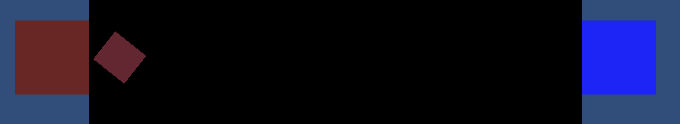


Figure 10 Normalized Value = 1



Figure 11 Normalized Value = 0.5

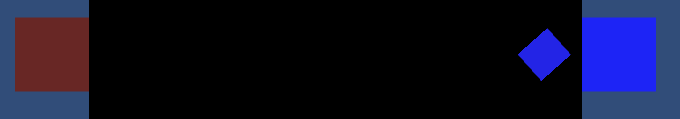


Figure 12 Normalized Value = 0.0

### Code for colour change

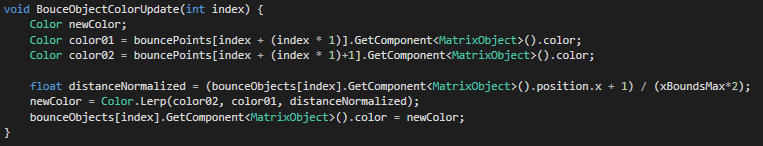


Figure 13 Located in the "GameMaster" Script

# Task 4

## Two objects for each moveable object

My GameMaster class controls the creation of the moveable objects as well as the 2 two objects that represent each of the bounce points.

### Code

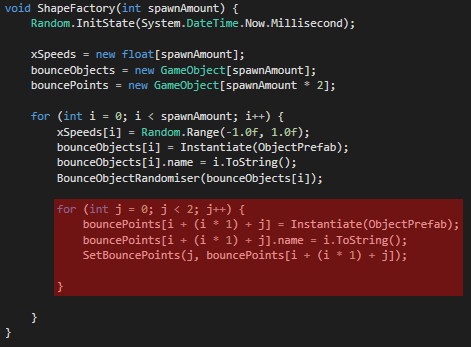


Figure 14 Located in the GameMaster class

## Vertical Drag per row of objects

For this I decided to make a script, called MouseDrag that updates a public variable with the y position of the mouse in world space. Then in the GameMaster script, I reference the MouseDrag and get the y position to update and apply it to the movement of the object. Once this is done I then set y position of the bounce points to the y position of their corresponding bounceObject.

### Code

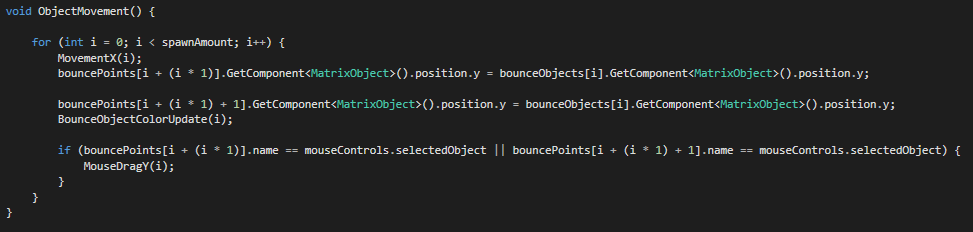


Figure 15 Located in the GameMaster script

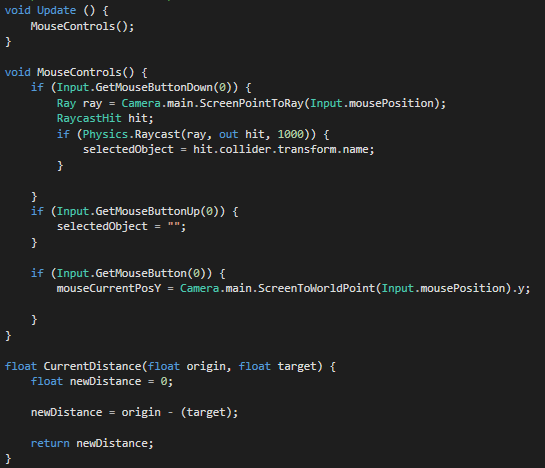


Figure 16 Located in the MouseDrag script

# Task 5

## Randomise Colour button

### Reason

To let the user see a different set of colours. Since the side objects determine how the moveable object colour changes, it is also a good indicator the colours can be changed at runtime.

The users can press a UI button to select new colours at random for the bounce points, which in turn affect the moveable objects colour, depending on the their location.

### Code

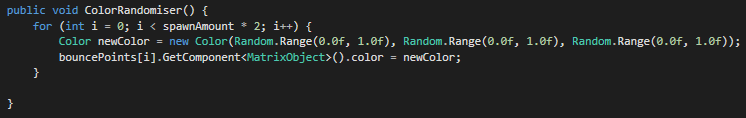


Figure 17 Found in GameMaster script

### UI

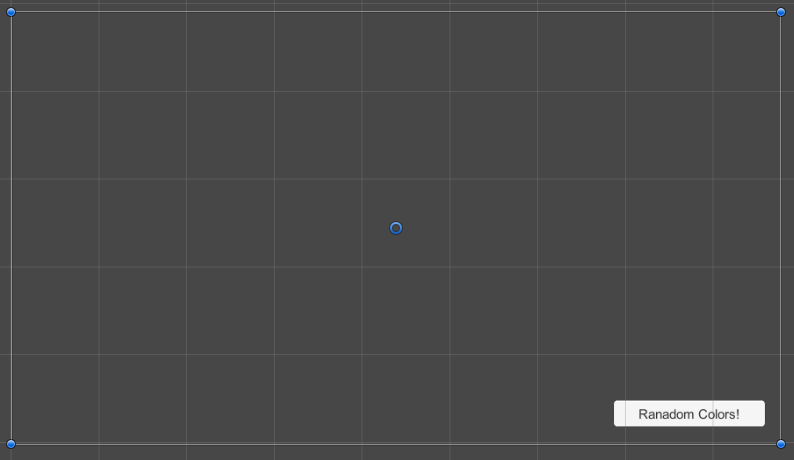


Figure 18 In the "Play Level" scene

## Show colour of bounce point

### Reason

Allows the user to see what the colour is at each bounce point. Colours are simple displayed like they would be with the moveable objects, but the only different between the bounce object and bounce points is that the bounce points are only updated when a new random colour has been selected, whereas the bounce objects colours are constantly updating depending of their location to the bounce points.