Introduction to Computer Graphics (CS360A)

Instructor: Soumya Dutta

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: soumyad@cse.iitk.ac.in

Acknowledgements

- A subset of the slides that I will present throughout the course are adapted/inspired by excellent courses on Computer Graphics offered by Prof. Han-Wei Shen, Prof. Wojciech Matusik, Prof. Frédo Durand, Prof. Abe Davis, and Prof. Cem Yuksel

Matrix Mathematics Library: glMatrix.js

- glMatrix-v0.9.5.min.js
- A minimal JavaScript library for helping you with
  - Matrix and vector manipulation
    - Initialization, multiplication, addition, subtraction, dot product, cross product etc.
  - Transformation
  - Projection and viewing

Matrix Operations with glMatrix

- `var mMatrix = mat3.create(); // create a 3x3 matrix`
- `var mMatrix = mat4.create(); // create a 4x4 matrix`
- `mat4.identity(mMatrix); // set the mMatrix to identity matrix`

Matrix Operations with glMatrix

- `mat4.identity(mMatrix);`
- `mMatrix = mat4.translate(mMatrix, [x, y, 0]);`
  - mMatrix is now a translation matrix with translation amount [x,y,z]

- `mat4.identity(mMatrix);`
- `mMatrix = mat4.rotate(mMatrix,angle,[0,0,1]);`
  - mMatrix is now a rotation matrix with respect to [0,0] as the rotation reference and z-axis as the axis of rotation

- `mat4.identity(mMatrix);`
- `mMatrix = mat4.scale(mMatrix,[p,q,0]);`
  - mMatrix is now a scaling matrix with [0,0] as a pivot point and the scaling amount is [p,q,0]

Matrix Operations with glMatrix

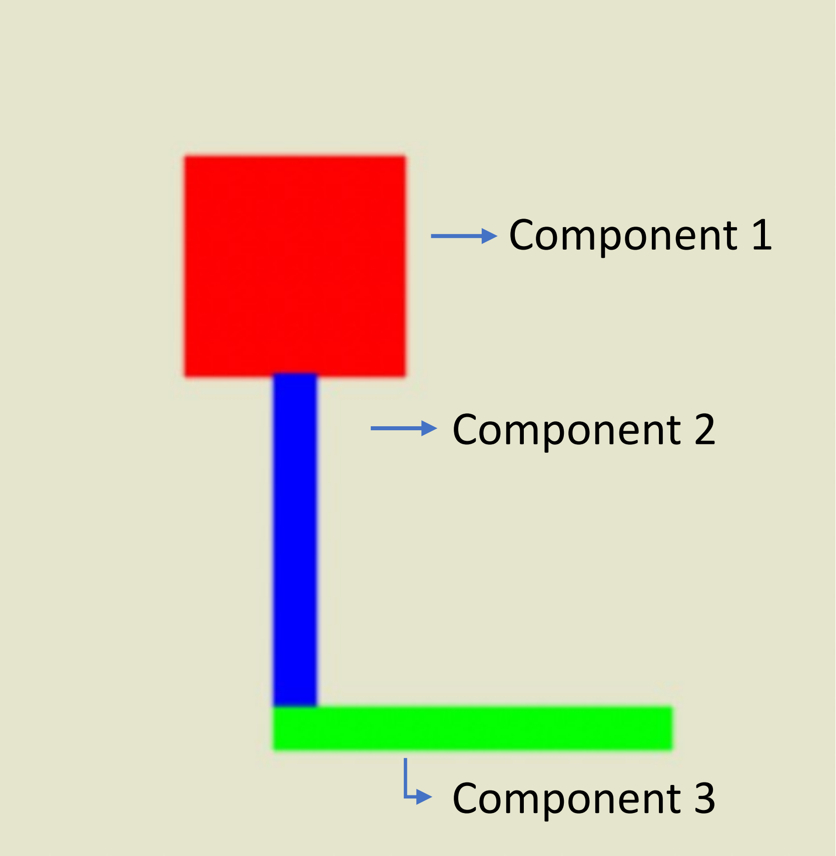
Order of transformation

- mat4.identity(mMatrix);
- mMatrix = mat4.translate(mMatrix, [0.1, 0, 0]);
- mMatrix = mat4.rotate(mMatrix, degToRad(45), [0, 0, 1]);
- mMatrix = mat4.scale(mMatrix, [0.07, 0.25, 1.0]);

- mMatrix = translate[0.1, 0, 0]∗rotate[45]∗scale[0.07, 0.25, 1.0]
  - Scaling will be applied first, then rotation, and then translation
  - This is the order you should follow in your code when applying transformations to an object in its local/object space
- newPosition = mMatrix∗oldPosition

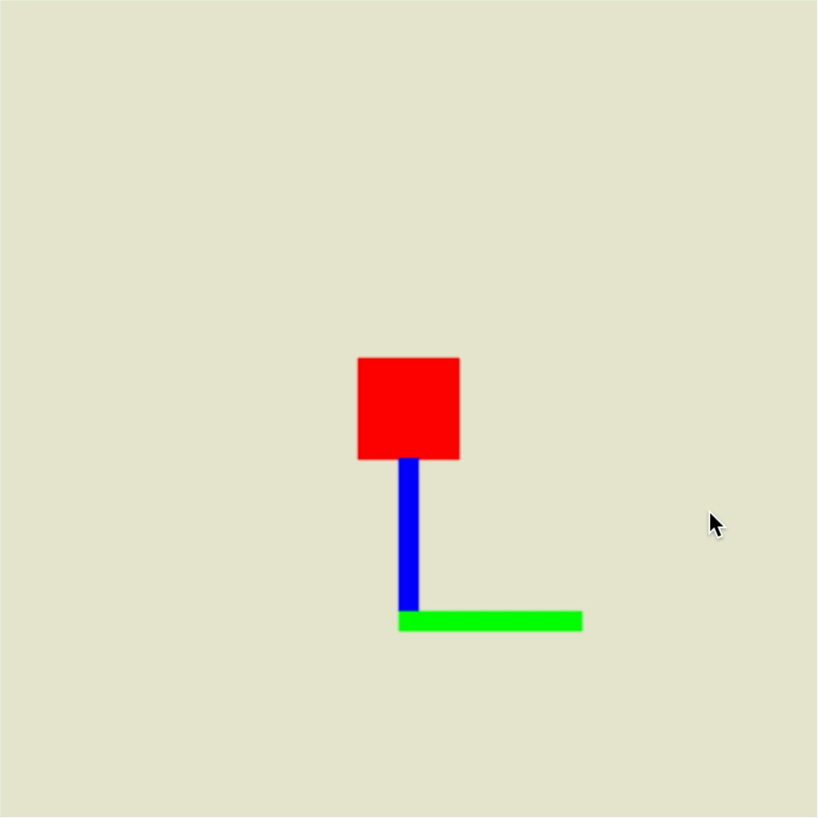
Hierarchical Transformation

- Your objects can have several components
  - Each component will have its own degrees of freedom
  - Form a hierarchy among the components to apply transformations locally and globally



Hierarchical Transformation

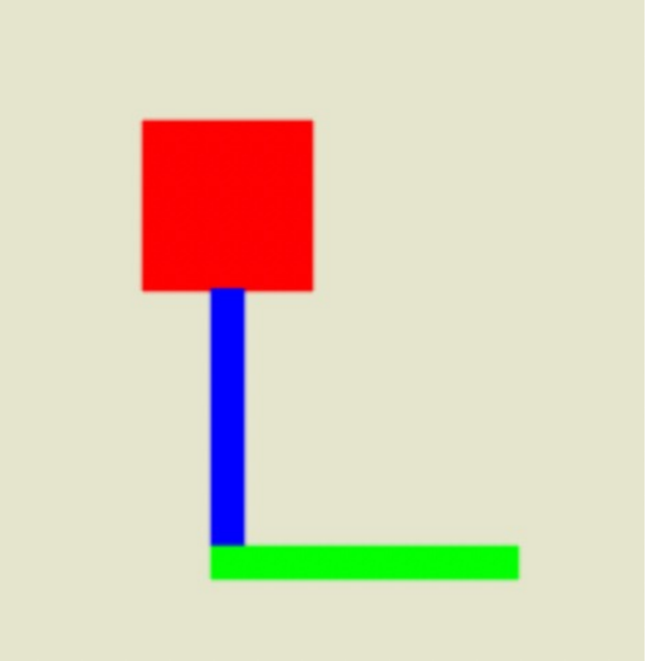
- Your objects can have several components
  - Each component will have its own degrees of freedom
  - Form a hierarchy among the components to apply transformations locally and globally
- How do we achieve this?
  - Carefully maintaining the states of our transformation matrix
  - Specifying transformations for the child components relative to the parent components while constructing an object in a scene

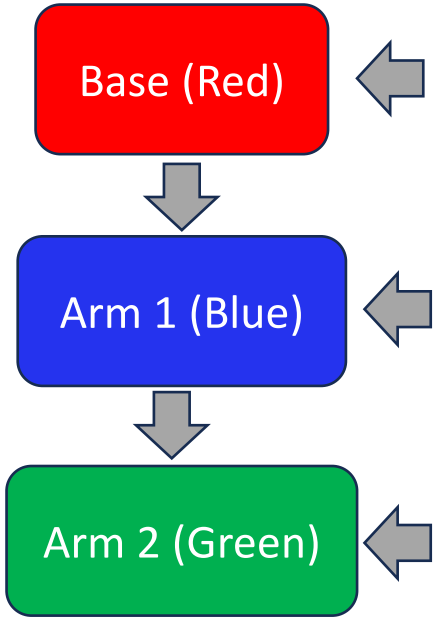


For animation, you can run the example code

Hierarchical Transformation

- We can describe the dependency of components in an object using a <u>tree structure</u>





Transformations here will affect everything below

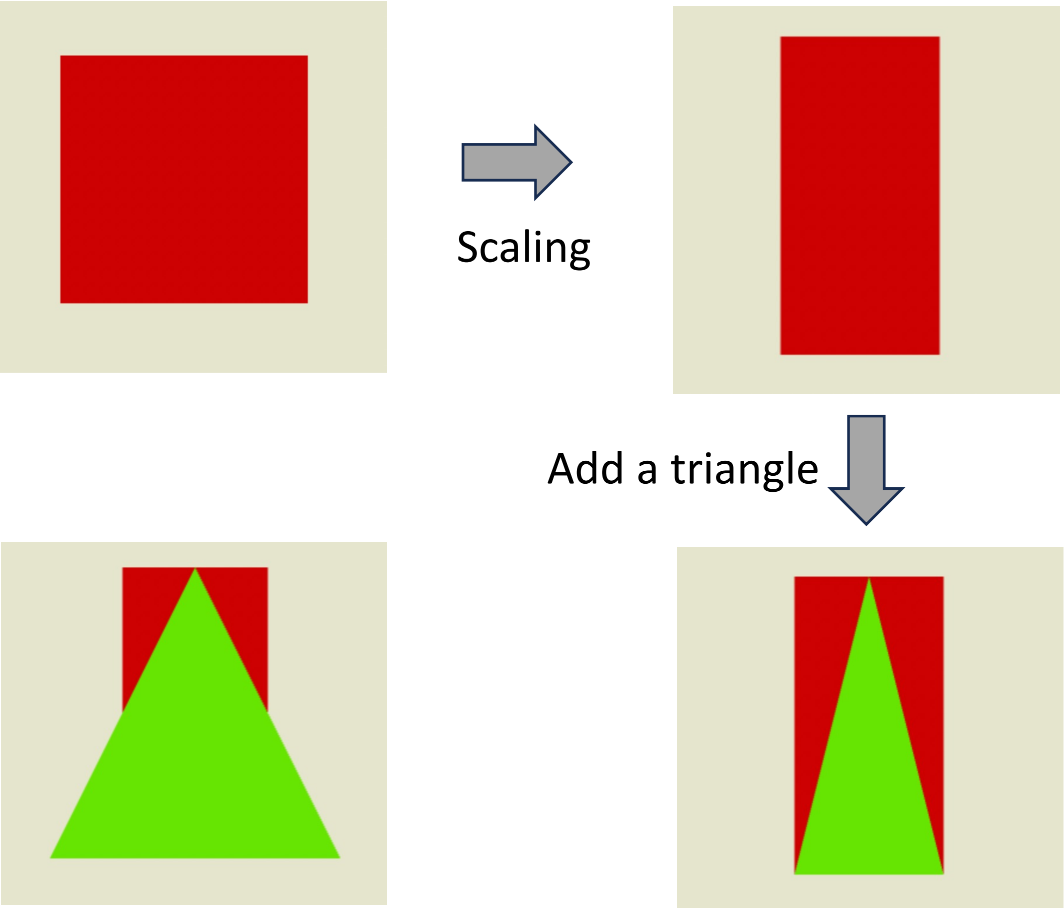
Transformations here will affect blue and green component

Transformations here will affect only green component

The position and orientation of a component can be affected by its parent, grand-parent, grand-grand-parent nodes..

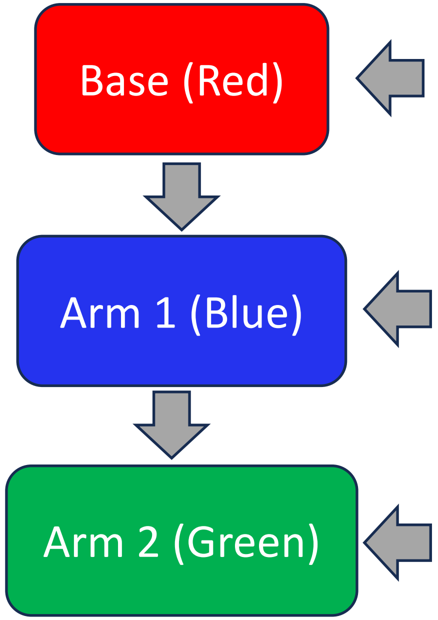
Hierarchical Transformation

- There is a problem here!!??



we expected this

Is this what we expected?



Transformations here will affect everything below

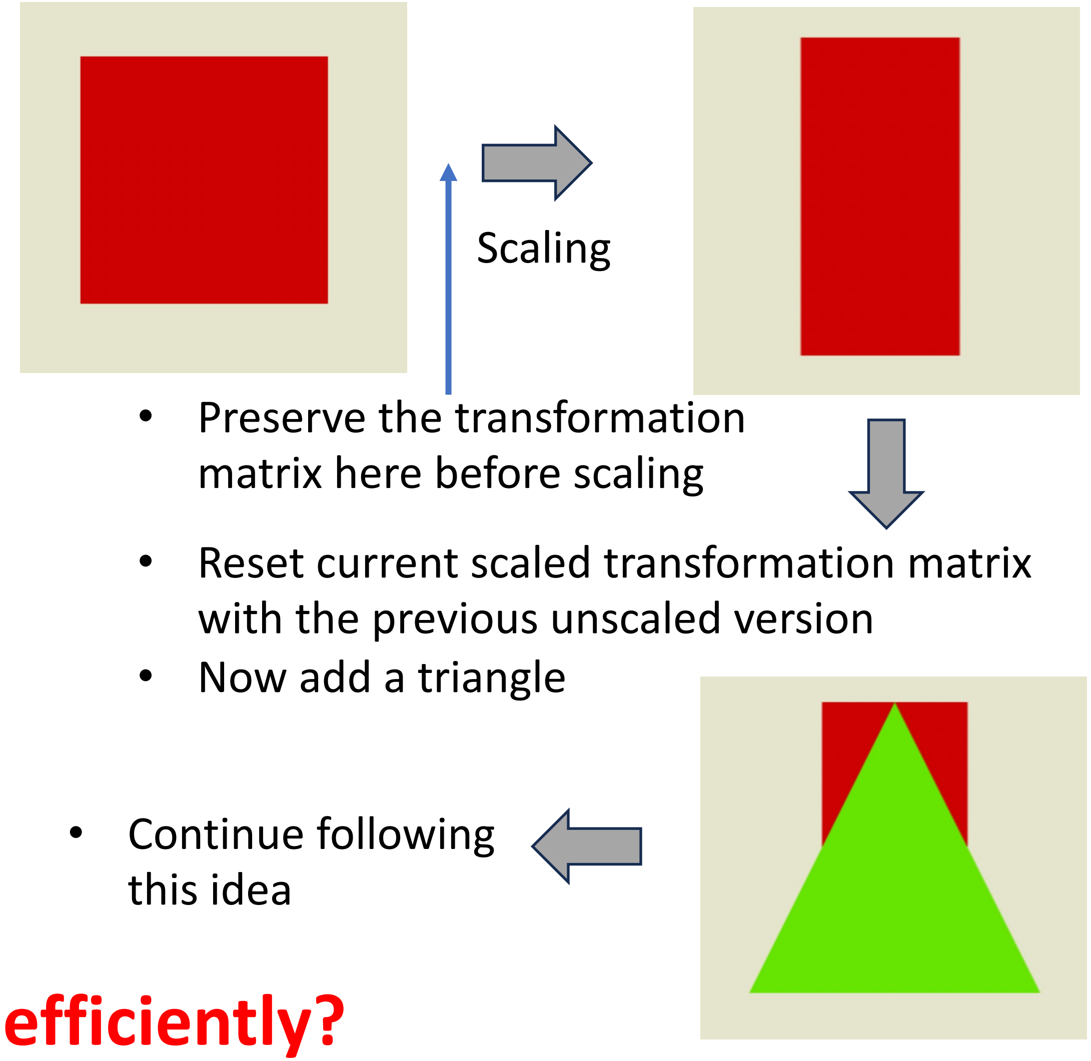
Transformations here will affect blue and green component

Transformations here will affect only green component

Why is this happening?

Hierarchical Transformation

- **How can we solve this issue?**
- Need to somehow preserve the transformation matrix before applying the scaling
- When drawing the triangle, reset the current transformation matrix with the previous unscaled transformation matrix
- Now draw the triangle



- Preserve the transformation matrix here before scaling
- Reset current scaled transformation matrix with the previous unscaled version
- Now add a triangle

- Continue following this idea

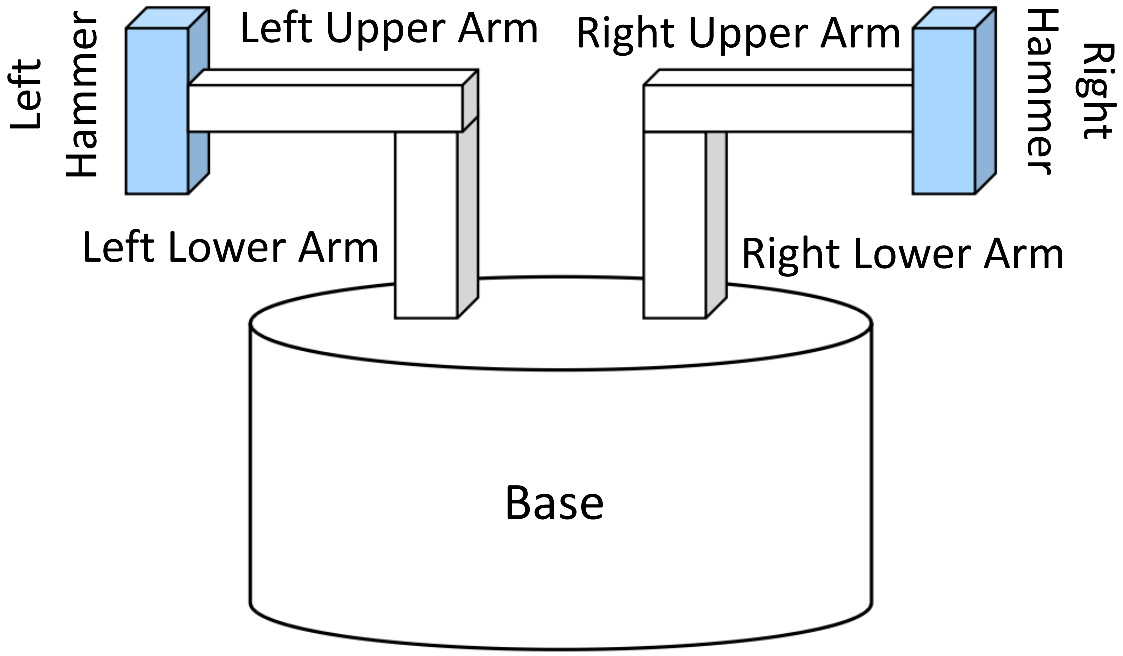
Correct output

How do we implement it efficiently?

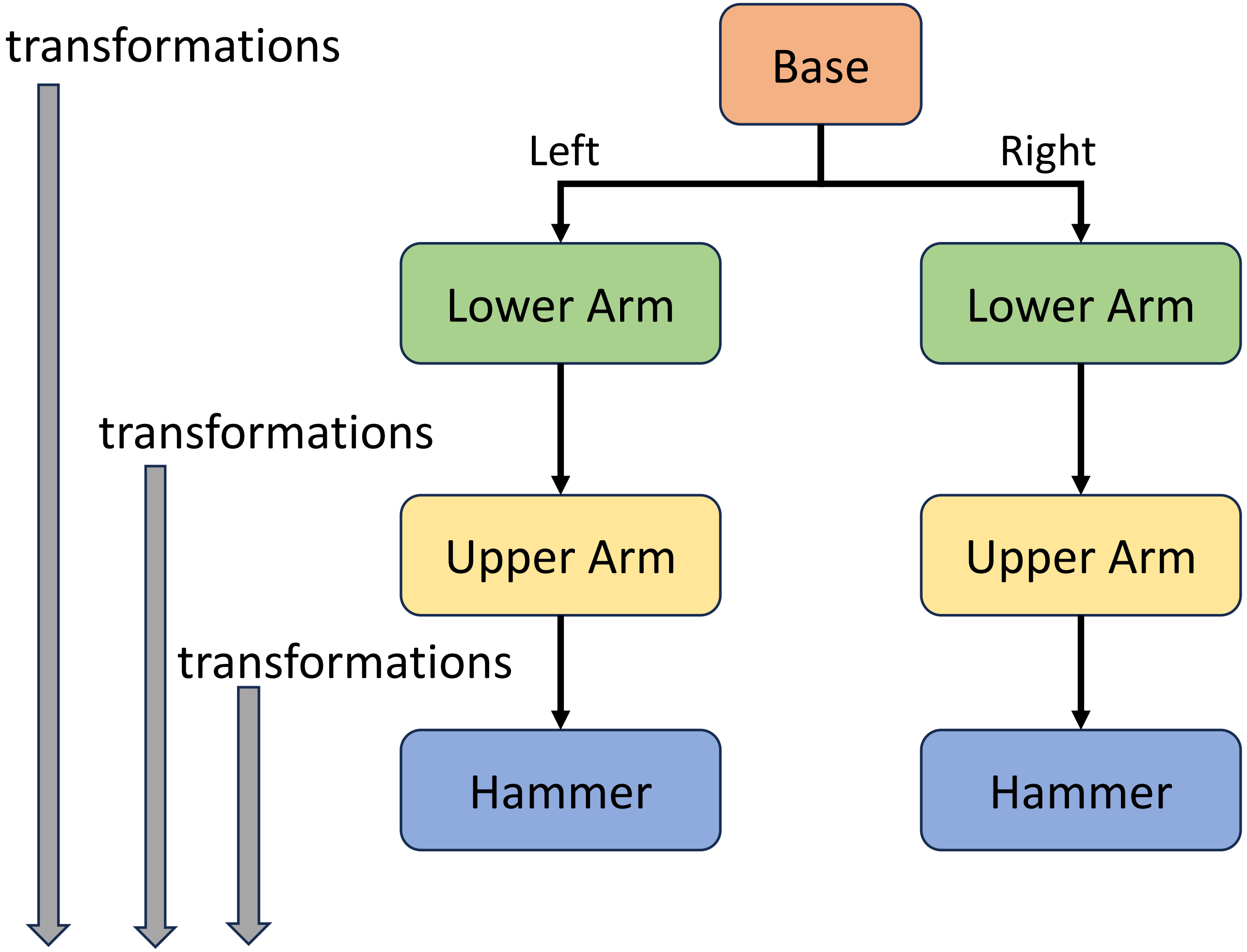
Hierarchical Transformation

- We will use a very popular simple data structure to solve this problem and maintain states of transformation matrix
- What is that data structure? You all know it!
- Use a matrix **stack** to keep track of various state of transformation matrices while building the scene
- Why Stack?
  - **Last-In-First-Out (LIFO) property**
  - Push and pop operation to save and restore matrix states to its immediate previously saved state

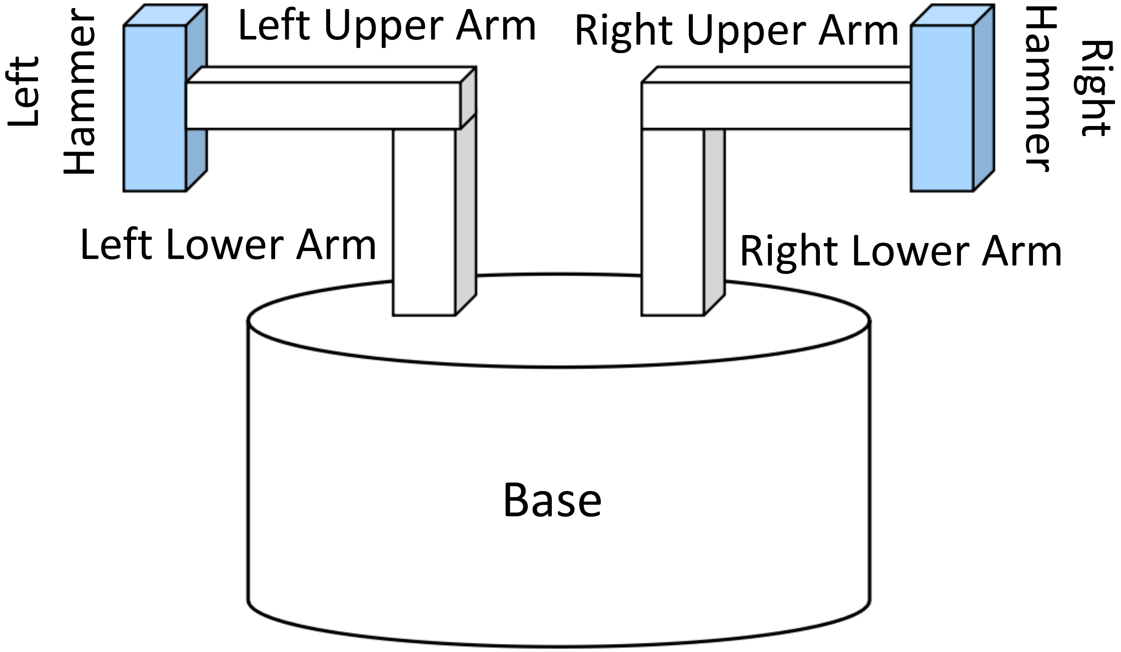
Hierarchical Transformation



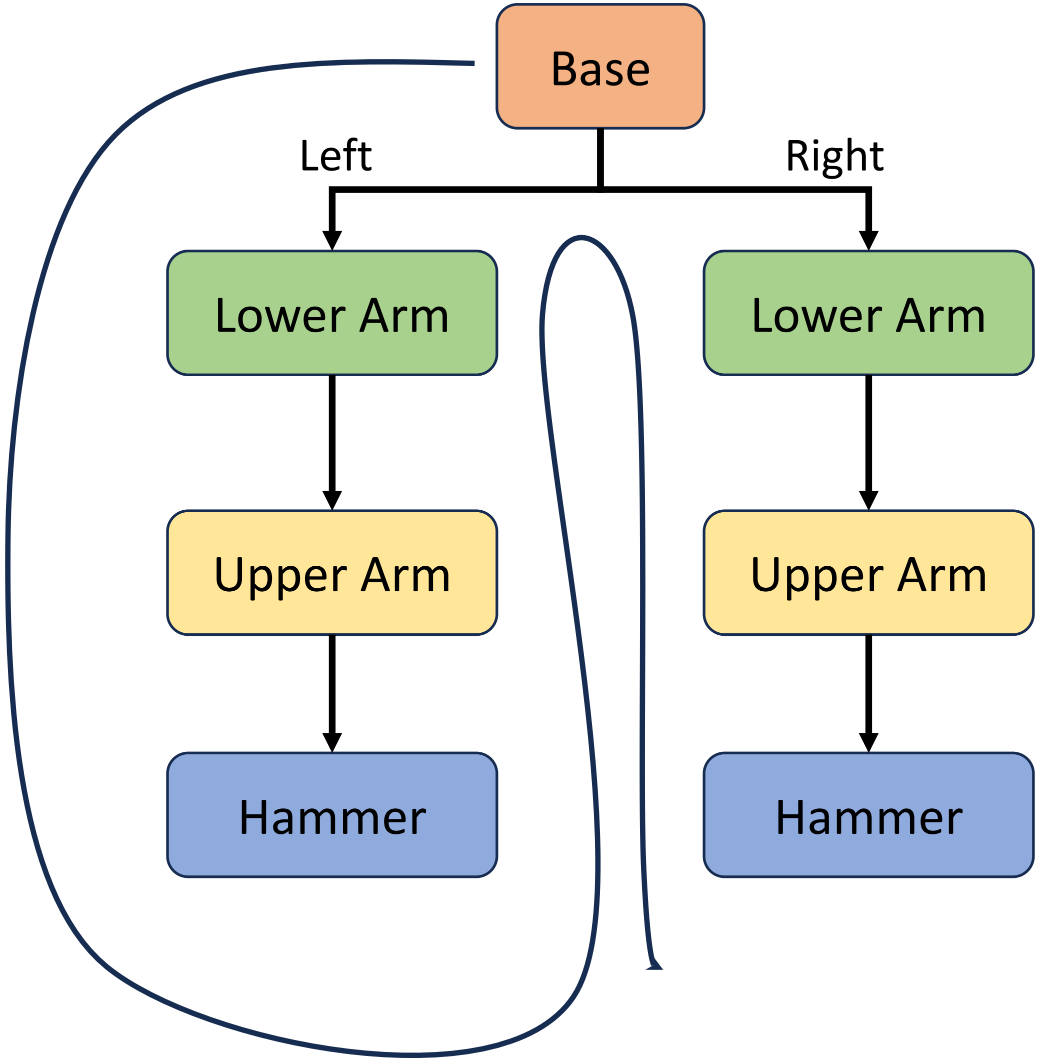
Robot Arm



Hierarchical Transformation



Depth first order



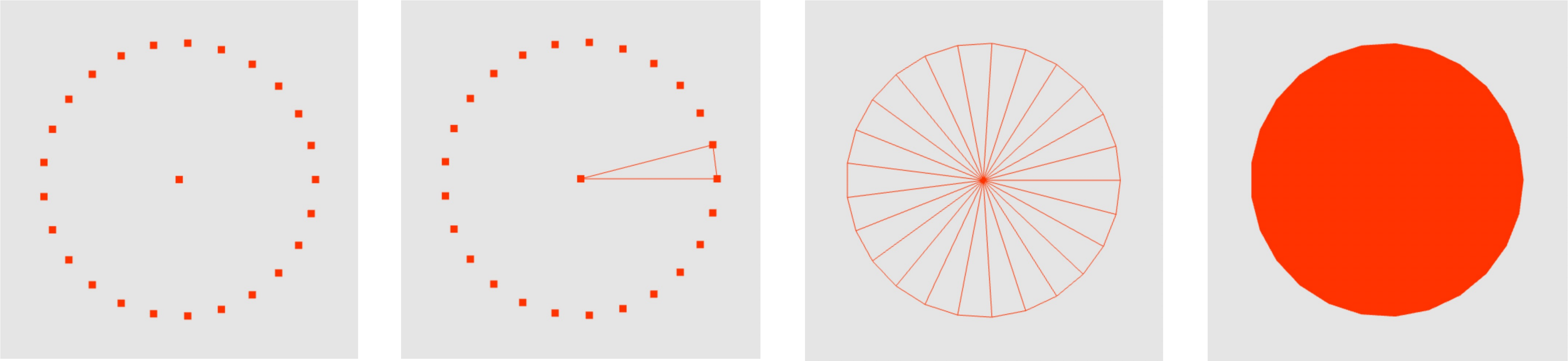
Code Demonstration

- examplePrimitiveRender.js
- examplePrimitiveRender.html

- HierarchicalTransformation.js
- HierarchicalTransformation.html

Circle Drawing

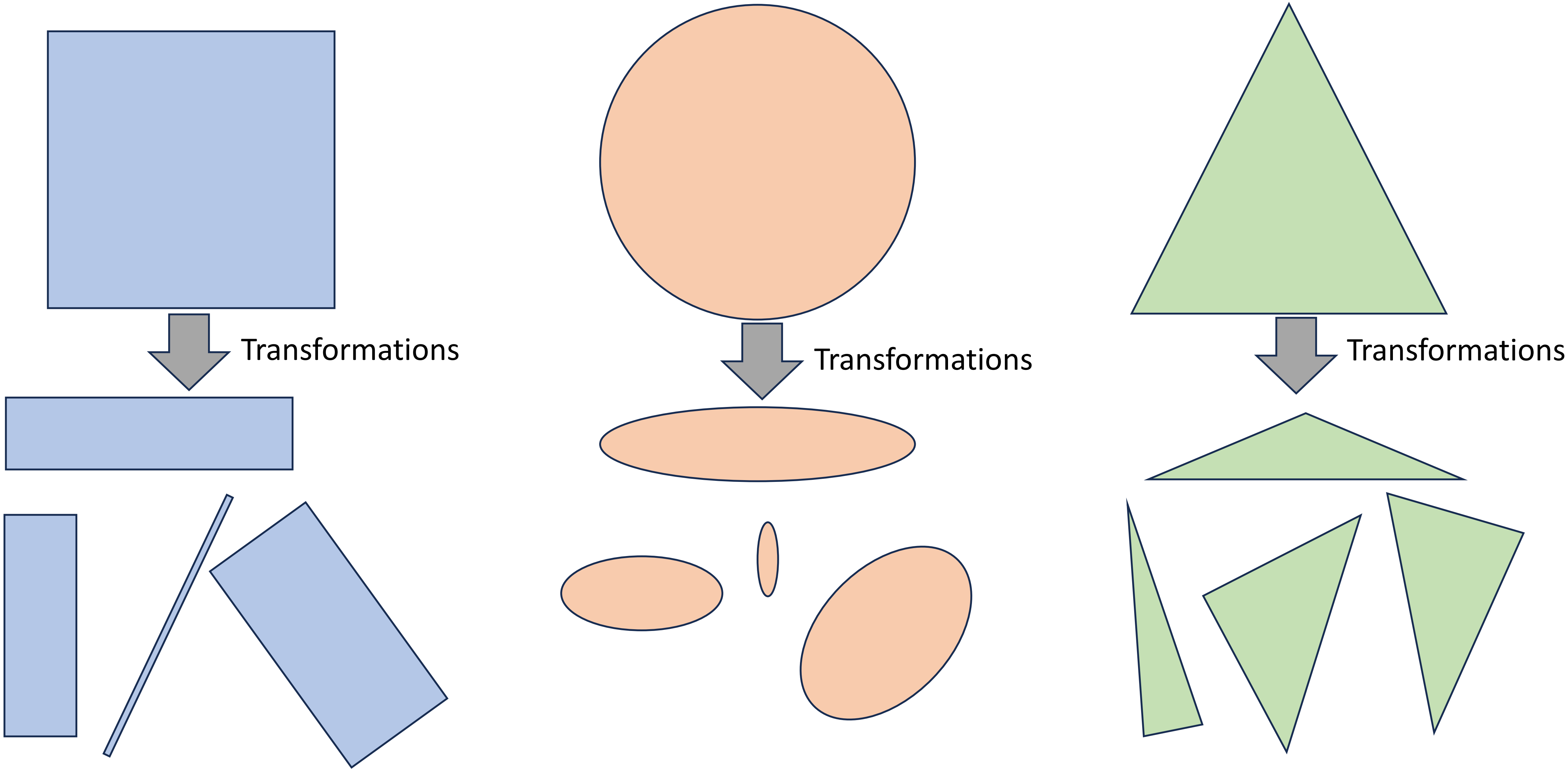
- By now, you know how to render a 2D triangle and a 2D square
- You also know how to transform these and form various shapes
- How about drawing a circle?
  - Generate points on a unit circle
  - Specify their connectivity by forming triangles



Circle Drawing

- Determine how many points to generate
  - Controls the smoothness of the circle surface
- Circle center at [0,0]
- How to get x,y locations for each point on circle?
  - `x = Math.cos((i * 2 * Math.PI) / resolution);`
  - `y = Math.sin((i * 2 * Math.PI) / resolution);`
- How to form the triangles from point indices?
  - Part of your first assignment!
- Add initCircleBuffer() and drawCircle() functions to your code just like square and triangle drawing functions

Basic Shapes and Transformations on Them



Animation in WebGL

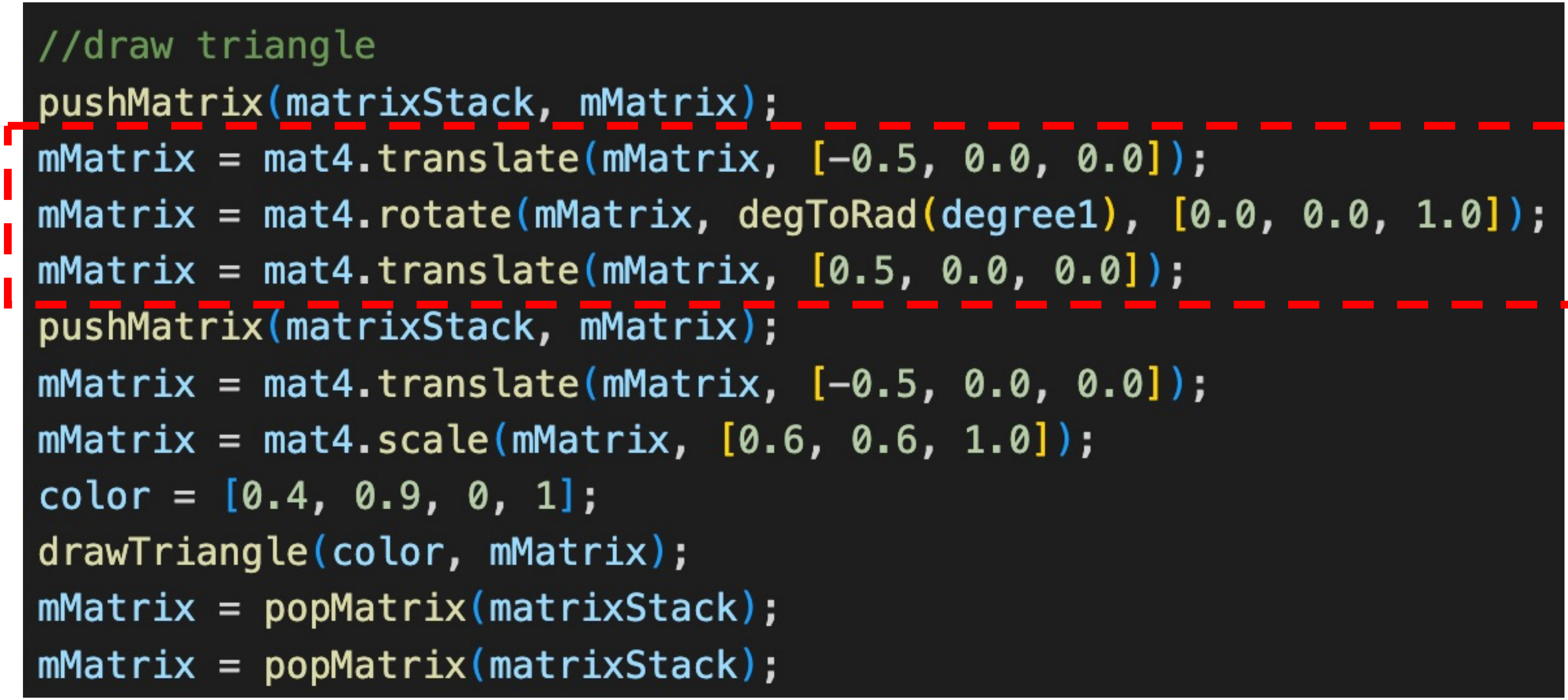
- Simple animation in WebGL is very easy to implement
- Use **window.requestAnimationFrame(callback)** method
  - Tell the browser that you want to perform animation
  - Specify a callback function that will be called before next repaint by the browser
  - The method takes a callback function as an argument
  - **Typically, the callback routine calls itself to animate another frame at the next repaint**
  - The number of callbacks is usually 60 times per second

Example Code for Animation

```
function drawScene() {
  ......
  var animate = function () {

   degree0 += 0.3; // amount of rotation

   // apply rotation to square via transformation matrix
   drawSquare();

   animation = window.requestAnimationFrame(animate); // using
   animate function as the callback to call it again

   };

  animate(); // calling to initiate drawing
}
```

Example Code for Animation

Code to draw a tringle centered at a different point than origin and then rotate it around its new center

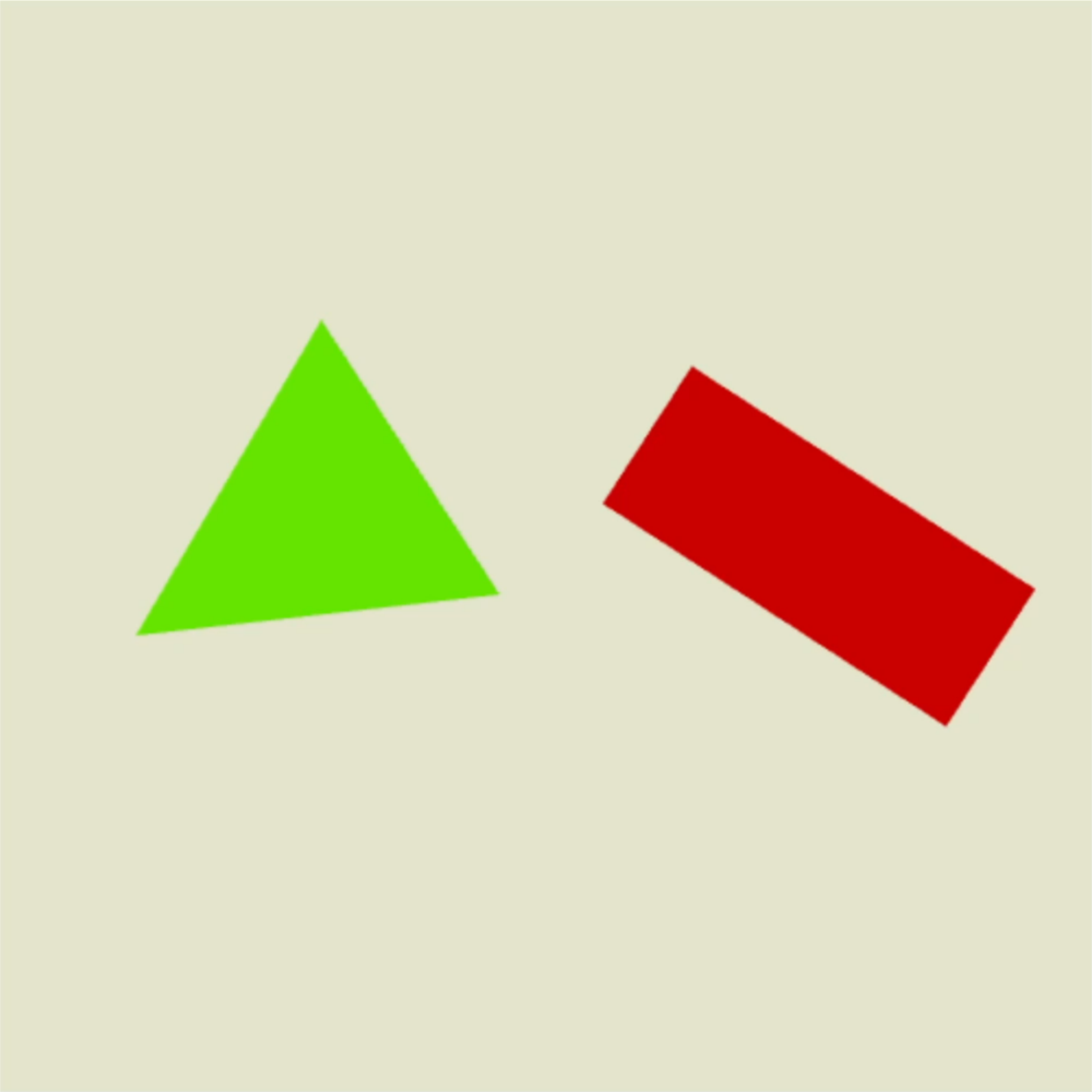


```
//draw triangle
pushMatrix(matrixStack, mMatrix);
mMatrix = mat4.translate(mMatrix, [-0.5, 0.0, 0.0]);
mMatrix = mat4.rotate(mMatrix, degToRad(degree1), [0.0, 0.0, 1.0]);
mMatrix = mat4.translate(mMatrix, [0.5, 0.0, 0.0]);
pushMatrix(matrixStack, mMatrix);
mMatrix = mat4.translate(mMatrix, [-0.5, 0.0, 0.0]);
mMatrix = mat4.scale(mMatrix, [0.6, 0.6, 1.0]);
color = [0.4, 0.9, 0, 1];
drawTriangle(color, mMatrix);
mMatrix = popMatrix(matrixStack);
mMatrix = popMatrix(matrixStack);
```

Translate back to origin, then rotate, and then translate back

First draw the triangle

Example Code for Animation

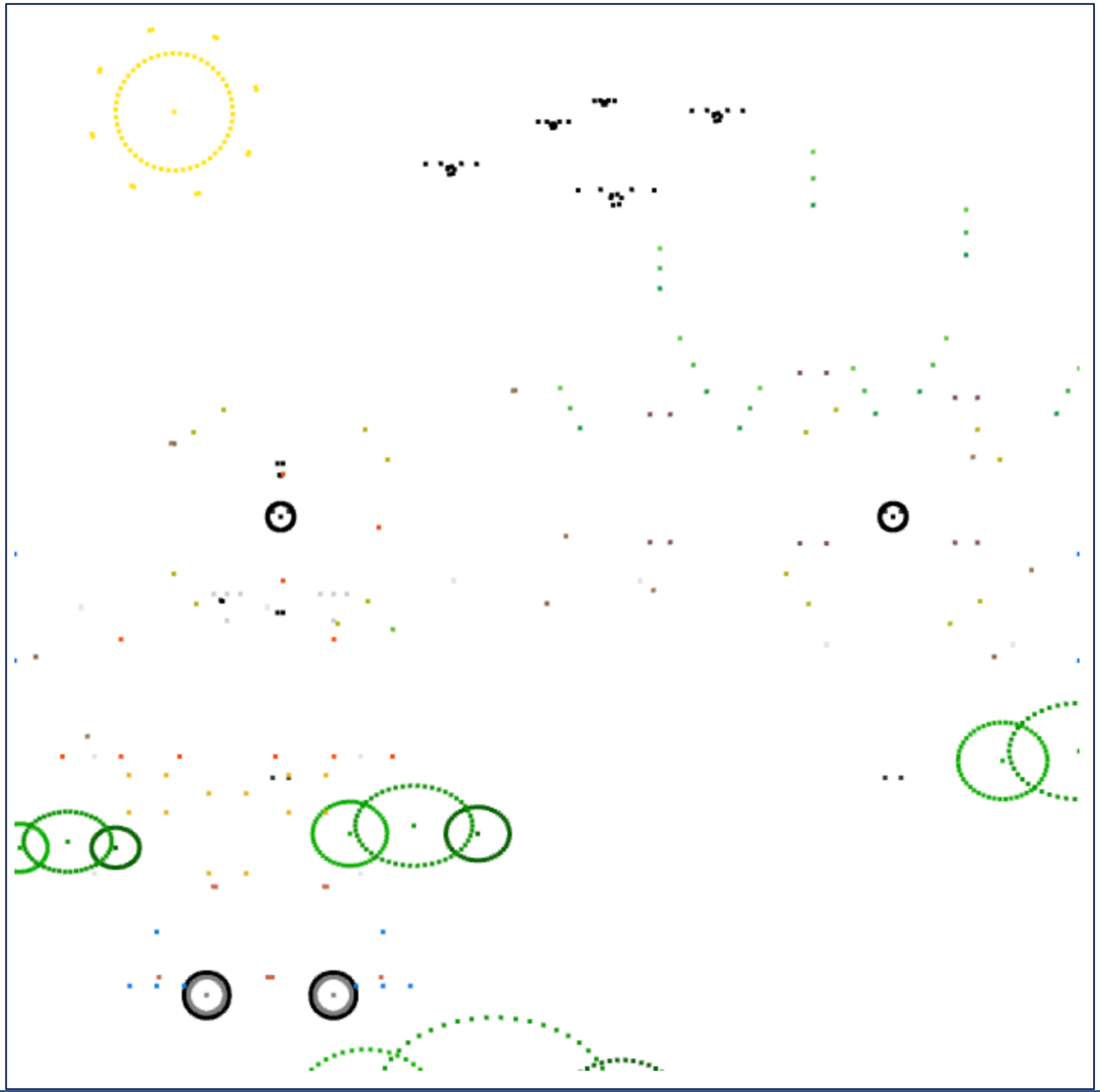


For animation, you can run the example code

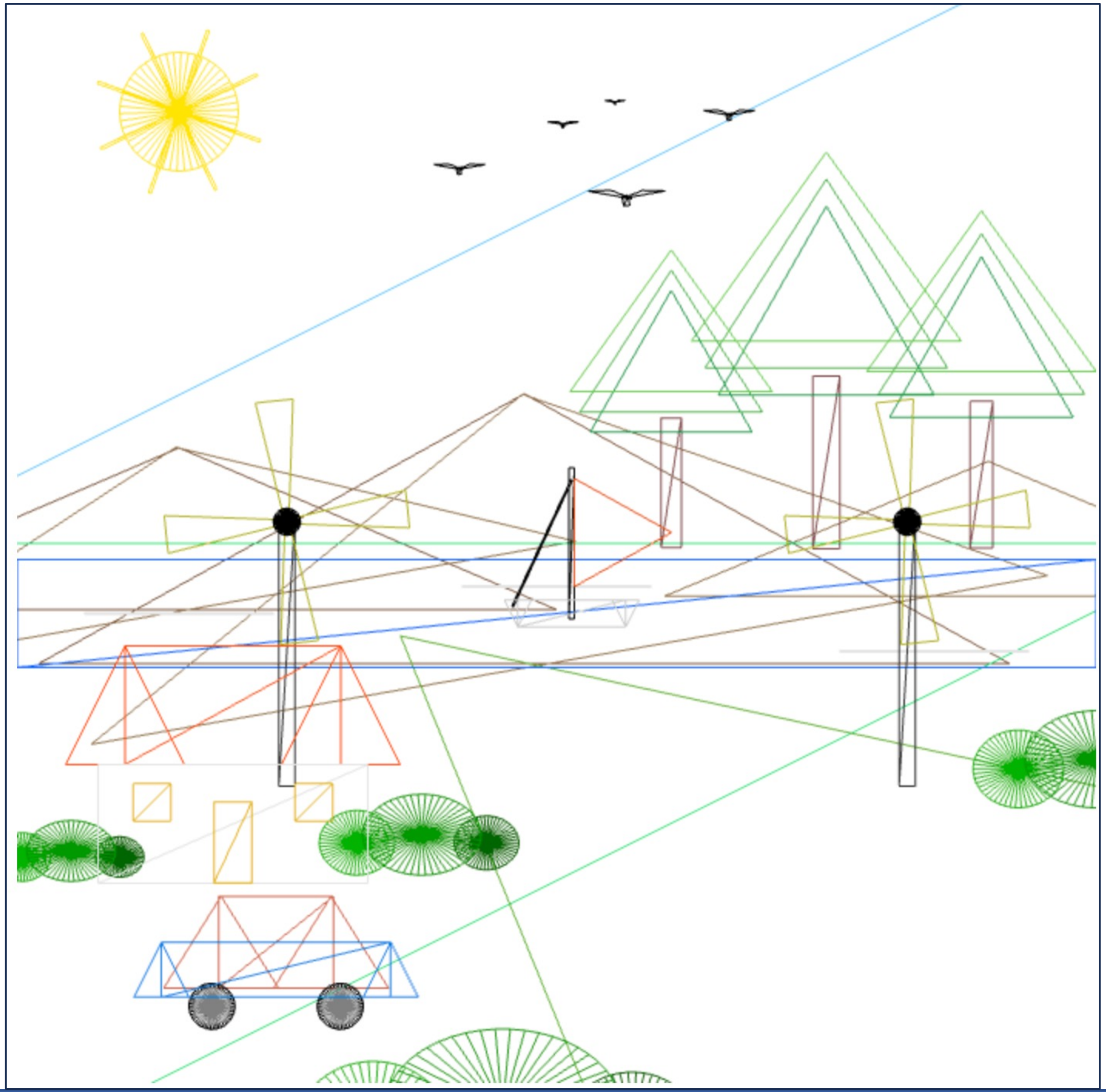
Assignment 1

- Creating a 2D Scene using three basic shapes (circle, square, triangle) and by applying transformations on them
- **Assignment is an individual task**
- **Due date: August 26th 11:59pm at HelloIITK**
- Discussion among your peers are allowed at conceptual level
- **DO NOT COPY or SHARE CODES from anywhere / with anyone!**
  - We will do intelligent software-based plagiarism check
  - If you get caught, you will get 0
  - Will be reported to institute plagiarism committee **without any exceptions**

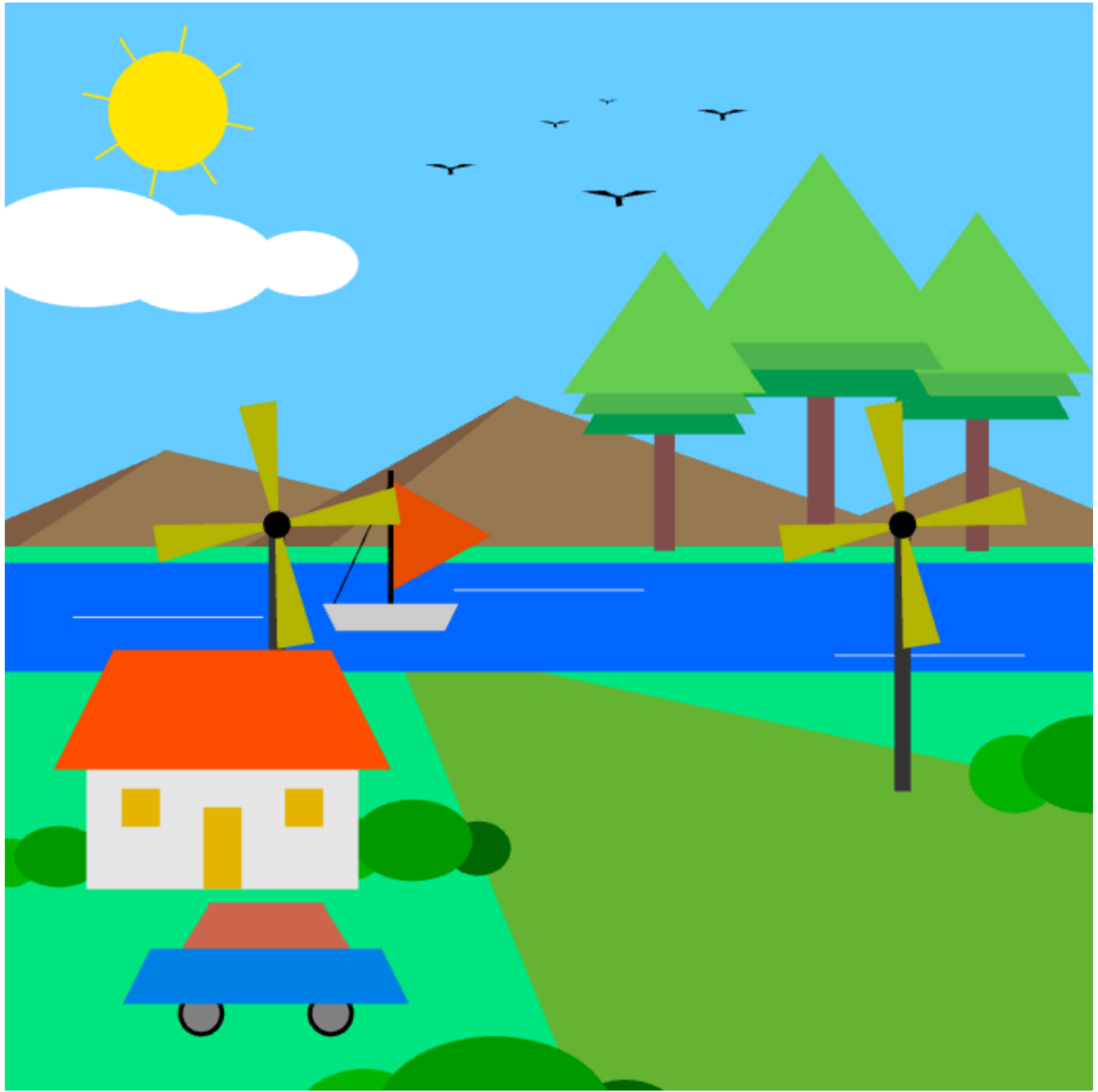
Assignment 1: Point Mode



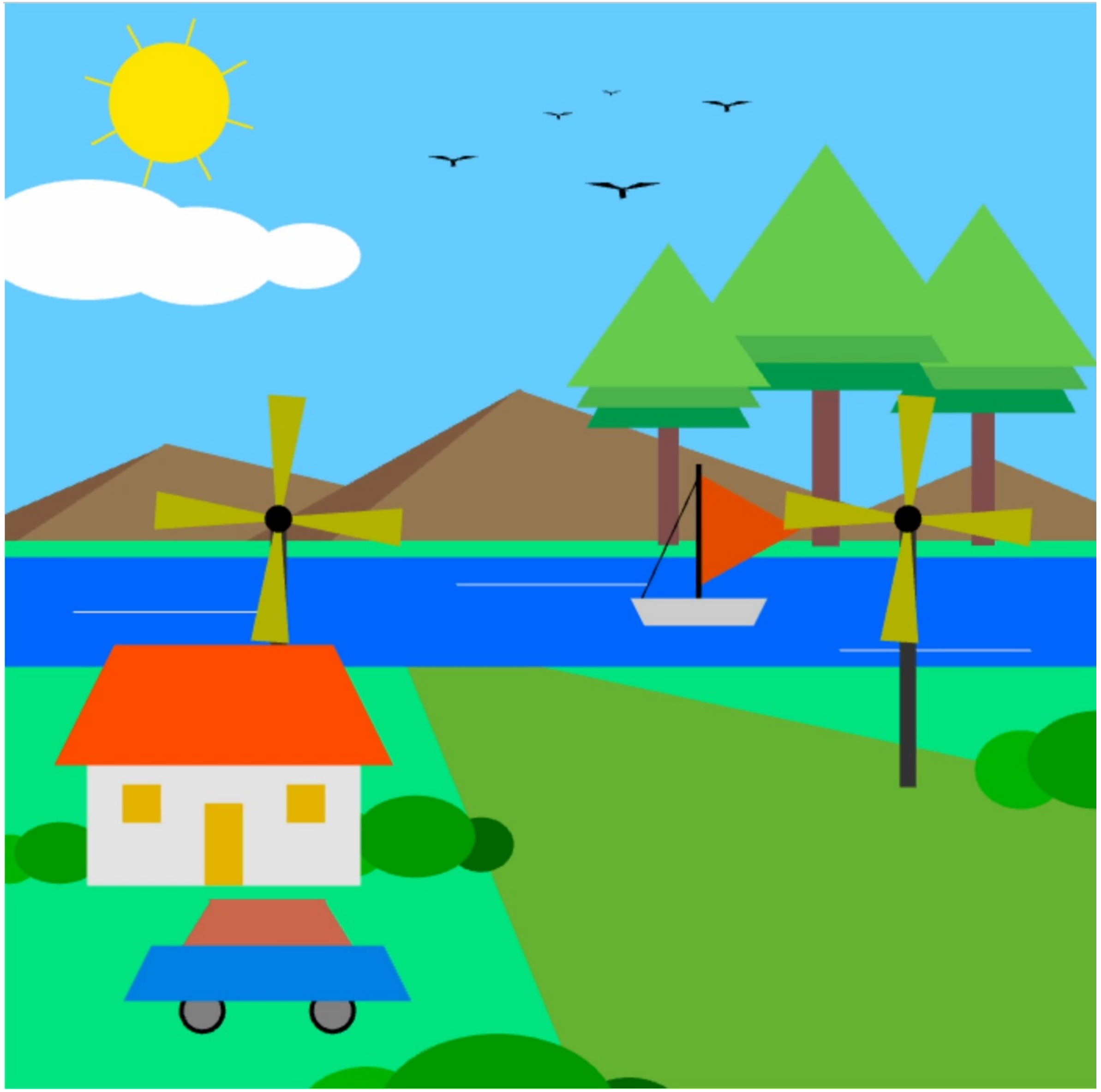
Assignment 1: Wireframe Mode



Assignment 1: Solid Object Mode



Assignment 1: With Animation



A video of the animated version is provided with assignment 1