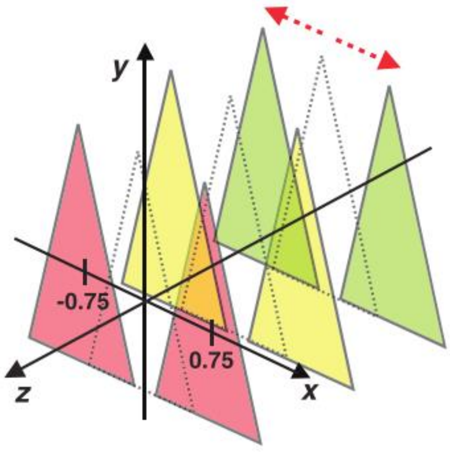
Durham University

Department of Computer Science

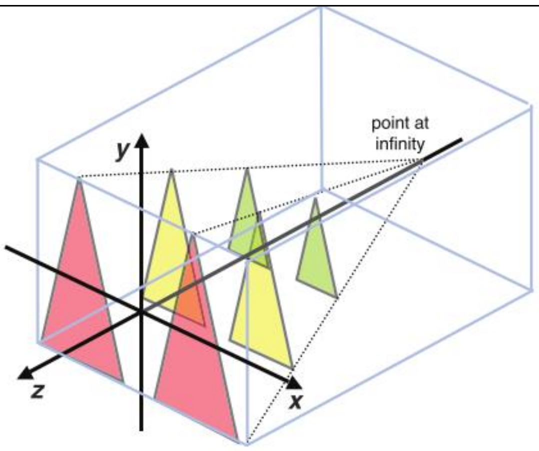
SM Computer Graphics Practical 2

Aim: After the scene composition lecture, it is time to work out how a 3D scene is programmed and rendered using WebGL, understanding how view transform and projection transform work together to allow 3D objects to be displayed in a 3D way.

Reading: lecture 3 – Scene Composition and projection.



Objects in world coordinates



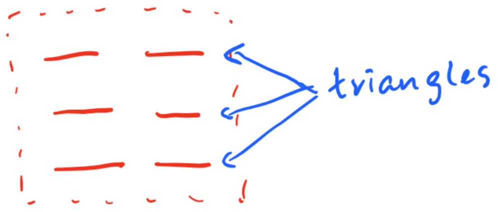
Objects after perspective transformation

Task 1: Check out PerspectiveView.js.

- Understand how the six triangles are constructed by identifying the corresponding vertex data and color data from each of the six triangles.
- Identify the vertex shader attributes to store the vertex data and the color data.
- Explain why the surface of each triangle can be filled up with a gradient of changing color. Identify what is the source of input data of the v_Color attribute in the fragment shader.

Task 2: Study Virtual camera (Lecture note p.13).

- Understand the setLookAt() instruction.
- For example, viewMatrix.setLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0) construct a virtual camera by locating it 5 units away from the origin of the world coordinate system and facing towards the 3D scene with an up-right orientation.
- The diagram below illustrates a top-view of the six triangles in PerspectiveView.js with the two triangles at the bottom representing the front triangles. The dotted lines illustrate the current visible region (view frustum) defined in the program.



- Indicate in the diagram where is the origin of the world coordinate system.
- Identify, in the diagram, the position of the virtual camera as defined by the above setLookAt()

instruction, and what the camera's facing direction is.
- Modify the setLookAt() instruction to viewMatrix.setLookAt(5, 0, 5, 0, 0, 0, 0, 1, 0). Identify, in the diagram, the updated position of the virtual camera.
- Modify the setLookAt() instruction, such that the virtual camera is looking at the six triangles from somewhere round the top-right corner of the near plane of the view frustum.
- Modify the setLookAt() instruction, such that you can zoom-in and zoom-out the six triangles.

Task 3: Study perspective projection (Lecture note pp.15-16).
- Modify the projMatrix.setPerspective() instruction, such that the pair of front triangles disappearing from the rendered output.
- Modify the projMatrix.setPerspective() instruction, such that the pair of back triangles disappearing from the rendered output.
- Modify the aspect ratio in the projMatrix.setPerspective() instruction, making the triangles look thinner (or fatter).
- Increase (or decrease) the field of view angle in the projMatrix.setPerspective() instruction by 20, explaining the change in the rendered result.

Task 4: PerspectiveView_mvp.js.
- Work out the differences between PerspectiveView_mvp.js and PerspectiveView.js, in terms of vertex shader, the use of transform operations, and the vertex data defining the six triangles.
- Explain why these two WebGL programs generate the same output.
- Repeat the "Modify setLookAt()" sub-tasks in task 2 by properly modifying the two modelMatrix.setTranslate() instructions this time instead of modifying the setLookAt() instructions, to generate the same expected results as those in task 2.
- Conclude the similarities and differences between the use of viewMatrix.setLookAt() and modelMatrix.setTranslate().