Scene Construction and Projection

1 World and Local Coordinate Systems

Local Coordinates - Each object is constructed on a dedicated coordinate system

World coordinates - Apply a single coordinate system to all objects globally.

The purpose of this is to reduce complication for 3D scene construction

2 View Transform

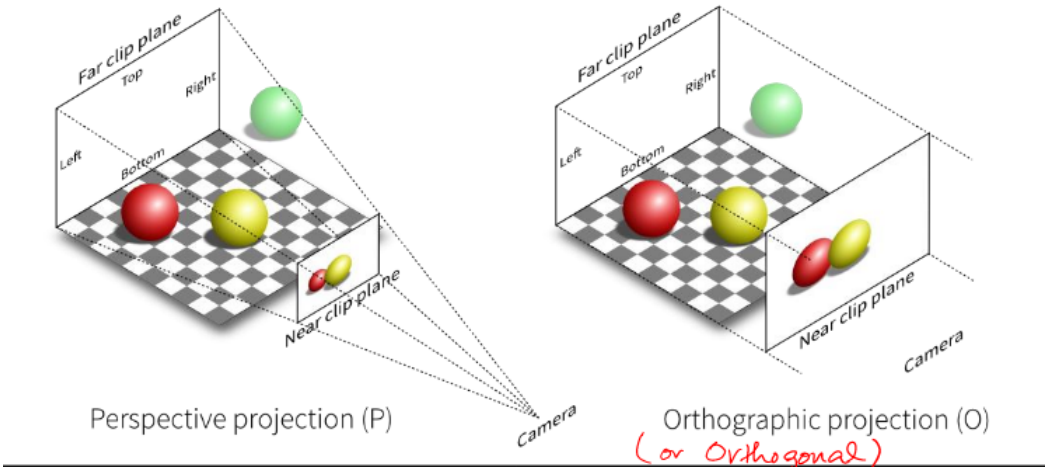
Shift the origin of the world coordinate system to the view origin. The view origin is where our eye (or virtual camera) is located with respect to the world origin

The purpose of this is to allow a user (application) to specify how 2D rendered images of a 3D scene will be generated

3 Projection Transform

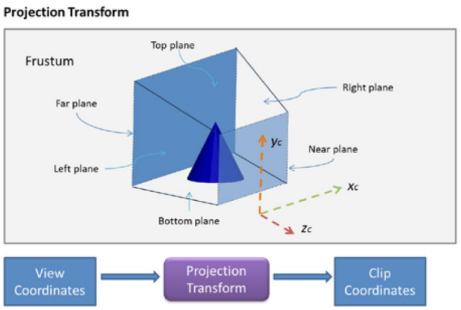
Visible Region - Define which part of a 3D scene will be currently visible

Object appearance - Modify or preserve object shape properties



4 Define a view frustum

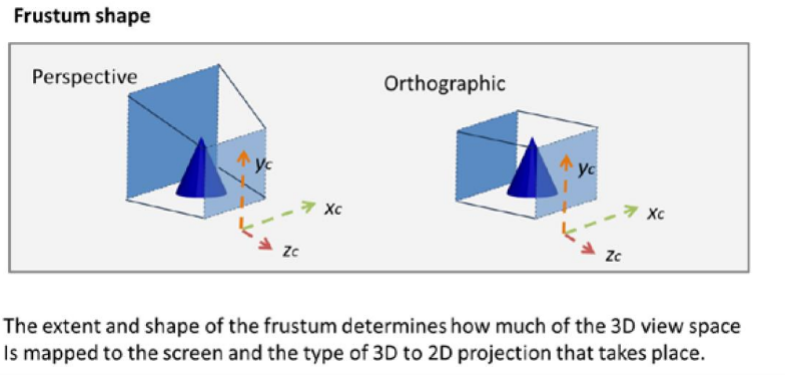
Projection transform is done based on a view frustum and it is defined by six planes (near, far, top, bottom, right and left)



The frustum determines which objects or portion of objects will be *clipped out* and discarded.

5 Types of View Frustum

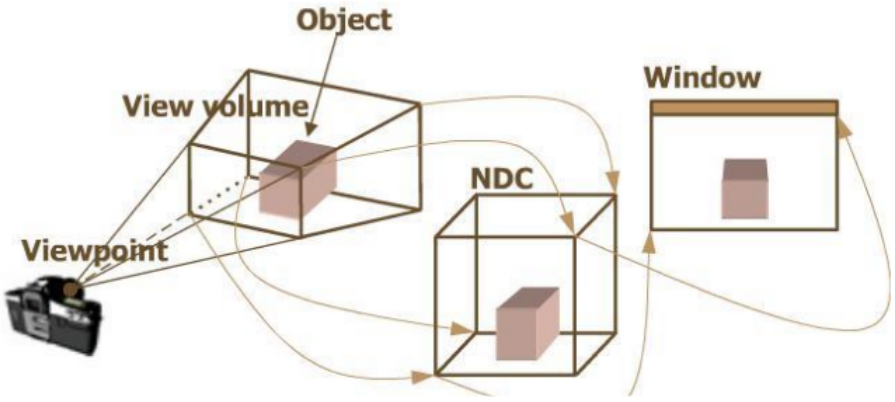
- The shape and extent of the frustum determines the type of view projection from the 3D scene space to the 2D screen
- If the far and near planes have the same dimensions, then the frustum will determine an orthographic projection. Otherwise, it will be a perspective projection



The extent and shape of the frustum determines how much of the 3D view space Is mapped to the screen and the type of 3D to 2D projection that takes place.

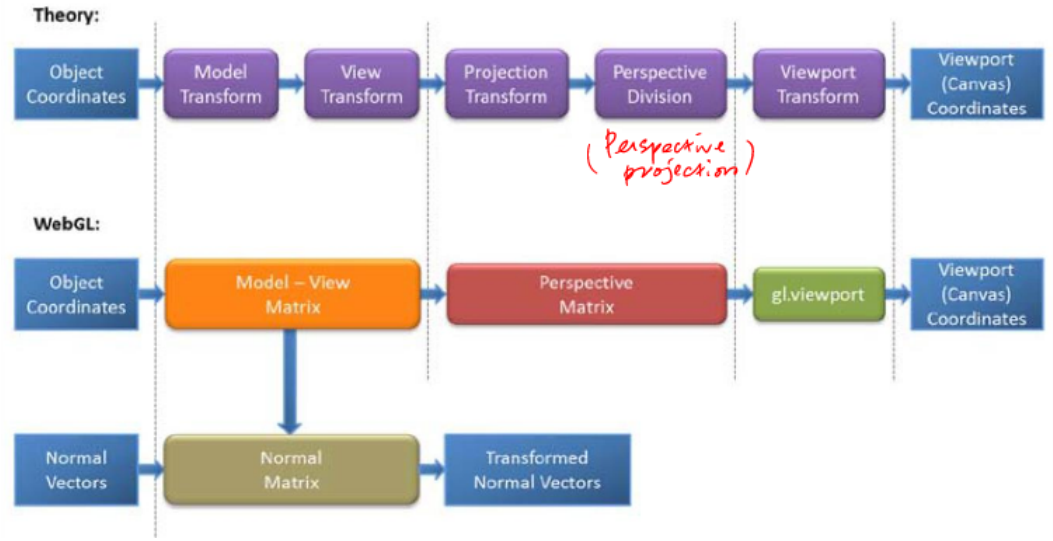
6 Viewport Transform

Map projected view to the available space in the computer screen, i.e. viewport, typically referring to the canvas



NDC - Normalized Device Coordinates. Its x and y coordinates represent the location of your vertices on a normalised 2D screen space

7 Model-View-Projection Transformation



8 WEBGL - setLookAt()

This takes the parameters: Eye Position, Look-at Position, Camera Orientation

- Eye Position - $(x_{eye}, y_{eye}, z_{eye})$
- Look-at Position - (x_{at}, y_{at}, z_{at})
- Camera Position - (dir_x, dir_y, dir_z)

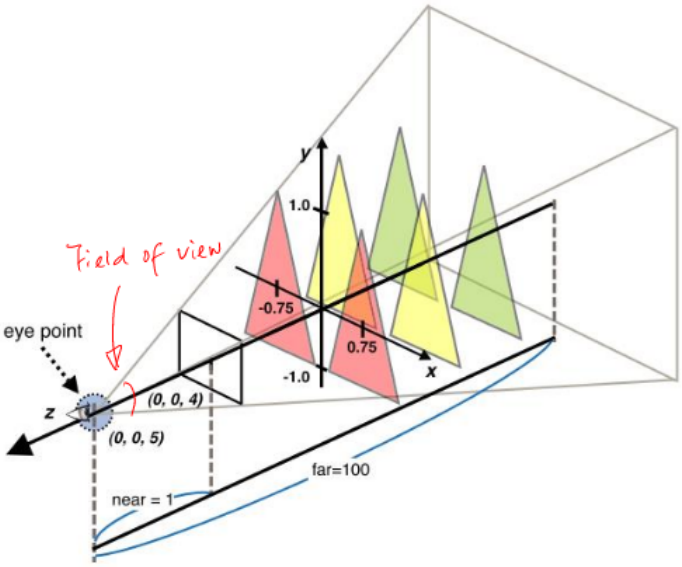
Usage:

```
// Set the matrix to be used to set the camera view
var viewMatrix = new Matrix4();
viewMatrix.setLookAt(0.20,0.25,0.25,0,0,0,0,1,0)

// Set the view matrix
gl.uniformMatrix4fv(u_ViewMatrix, false, viewMatrix.elements);

// Draw the rectangle
gl.drawArrays(gl.TRIANGLES, 0, n);
```

9 WebGL - setPerspactive



Parameters:

1. Field of view (in terms of angle)
2. Aspect ratio
3. Near plane
4. Far plane

```
// Calculate the view matrix and the projection matrix
modelMatrix.setTranslate(0.75,0,0); // Translate 0.75 units along the positive x axis
viewMatrix.setLookAt(0,0,5,0,0,-100,0,1,0);
projMatrix.setPerspective(30, canvas.width/canvas.height,1,100);
// Pass the model, view and projection matrix to the uniform variable respectively
gl.uniformMatrix4fv(u_ModelMatrix, false, modelMatrix.elements);
gl.uniformMatrix4fv(u_ViewMatrix, false, viewMatrix.elements);
gl.uniformMatrix4fv(u_ProjMatrix, false, projMatrix.elements);
// Draw the triangles
gl.drawArrays(gl.TRIANGLES,0,n)
```