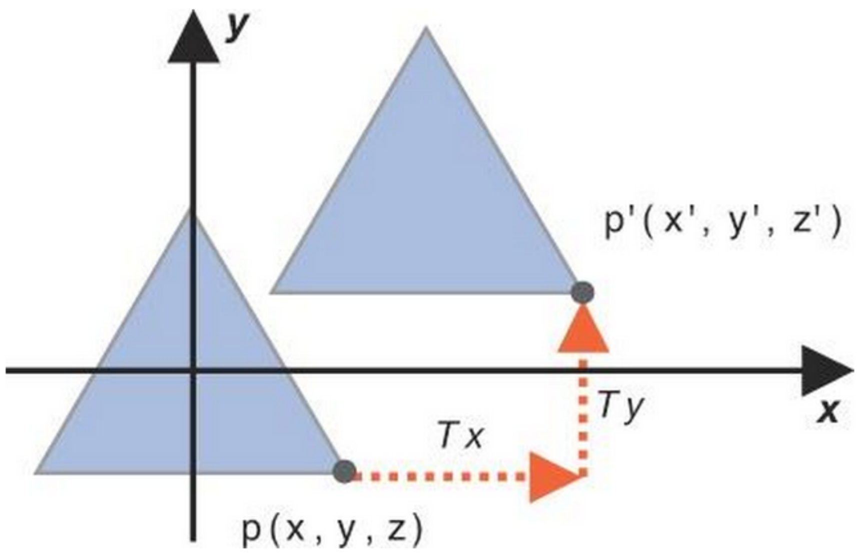
CMSC 161 UV-1L
Interactive Computer Graphics
Meeting 04 - Affine Transformations

Performing Affine Transformations
File: *04-transformations01.html*
04-transformations02.html
04-transformations03.html

Using the formulas from the lecture, affine transfomations can be implemented in WebGL using the vertex shaders to compute of these transformations.

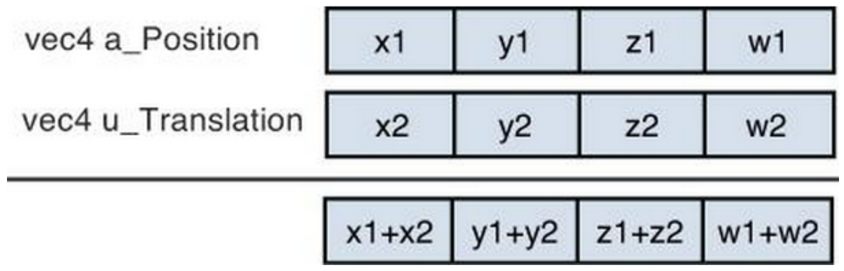


Translation of a whole object

Those affine transformations can be implemented in a WebGL program by just performing arithmetics with each vertex coordinates. These operations are performed as per-vertex operations which should be implemented in the vertex shader.

Translation can be easily implemented by addition a uniform variable to the position variables in the vertex shader

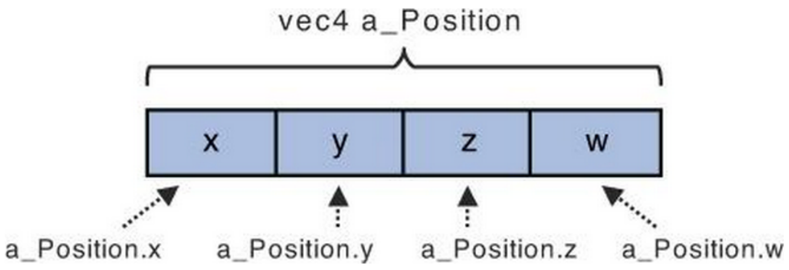
```
gl_Position = aPosition + uTranslation;
```



vec4 a_Position	x1	y1	z1	w1
vec4 u_Translation	x2	y2	z2	w2
	x1+x2	y1+y2	z1+z2	w1+w2

Rotation can not be implemented easily like translation. Each component of the new point position must be assigned to some equation.

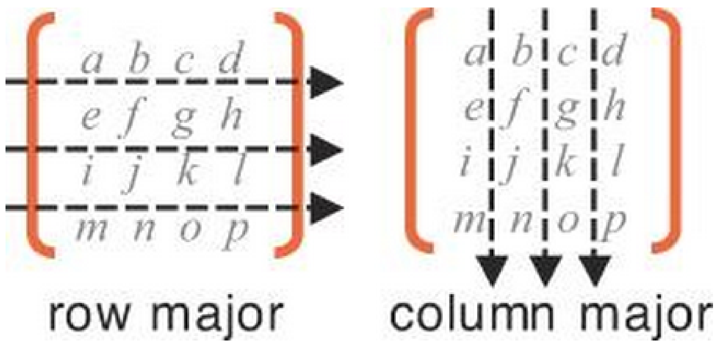
```
gl_Position.x = aPosition.x * cosAngle - aPosition.y * sinAngle;
gl_Position.y = aPosition.x * sinAngle + aPosition.y * cosAngle;
gl_Position.z = aPosition.z;
gl_Position.w = aPosition.w;
```



Performing Affine Transformations using Transformation Matrices
File: 04-transformations04.html

It is known that linear transformations can be expressed as a matrix. Since GLSL has innate matrix operations and optimized matrix operations, affine transformations can be implemented using a transformation matrix

Matrices are still represented as an array in javascript, the order of storage should be in **column major order**.



```
    var transformationMatrix = [a,e,i,m,b,f,j,n,c,g,k,o,d,h,l,p];
                                       or
     var transformationMatrix =[a,e,i,m,
                                b,f,j,n,
                                c,g,k,o,
                                d,h,l,p];
```

Performing Affine Transformations using Transformation Matrices (glMatrix Library)
File: 04-transformations05.html

Creating a transformation matrix in WebGL using glMatrix Library is easier.

```
var transformationMatrix = mat4.create();
mat4.rotateZ(transformationMatrix,transformationMatrix,toRadians);
```

GL Matrix documentation: http://glmatrix.net/docs/2.2.0/

Complex Transformations in WebGL
File: 04-transformations06.html
04-transformations07.html

Multiple linear transformations can be condensed as a single matrix. Using glMatrix we will perform the complex transformation of **rotation with respect to another point.**

Step 1: Compute t as distance of X from the origin

```
var T = vec4.create();
vec4.subtract(T,pointOfRotation,origin);
```

Step 2: Translate primitive by -t. (T^{-1})

```
var translateNegativeTMatrix = mat4.create();
var negatedT = vec4.create();
vec4.negate(negatedT,T)
mat4.translate(translateNegativeTMatrix,translateNegativeTMatrix,negatedT);
```

Step 3: Rotate result by desired amount. (R)

```
var rotationMatrix = mat4.create();
var angleOfRotation = 45.0;
var toRadians = glMatrix.toRadian(angleOfRotation);
mat4.rotateZ(rotationMatrix,rotationMatrix,toRadians);
```

Step 4: Translate result by t. (T)

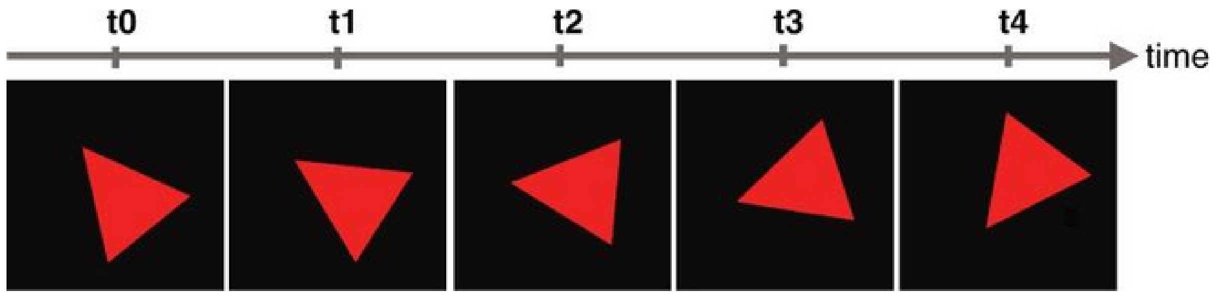
```
var translatePositiveTMatrix = mat4.create();
mat4.translate(translatePositiveTMatrix,translatePositiveTMatrix,T);
console.log(translatePositiveTMatrix);
```

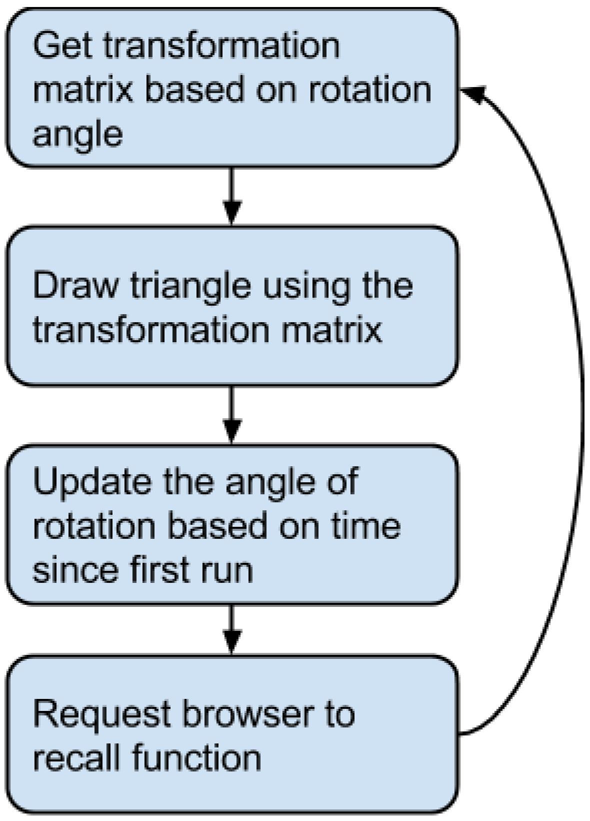
$$P' = TRT^{-1}P$$

```
var transformationMatrix = mat4.create();
mat4.multiply(transformationMatrix,rotationMatrix,translateNegativeTMatrix);
mat4.multiply(transformationMatrix,translatePositiveTMatrix,transformationMatrix);
```

Simple Animation
File: 04-animation01.html

The basics of animation is simply based on redrawing an object after every transformation.




Get transformation matrix based on rotation angle
Draw triangle using the transformation matrix
Update the angle of rotation based on time since first run
Request browser to recall function

CMSC 161 UV-1L
Interactive Computer Graphics
Meeting 04 - Affine Transformations

Exercise 4: **Move that pokemon!!!!!**
File: 04-exersample.html

Create **buttons** near your pokemon that allow your pokemon to do the following:

1. Spin at the center
2. Move to the left continuously
3. Move to the right continuously
4. Move up continuously
5. Move down continuously
6. Grow!! (Scale up)
7. Shrink!! (Scale down)
8. Flip upside down (FREE!)