

TP2 – Geometric Transformations

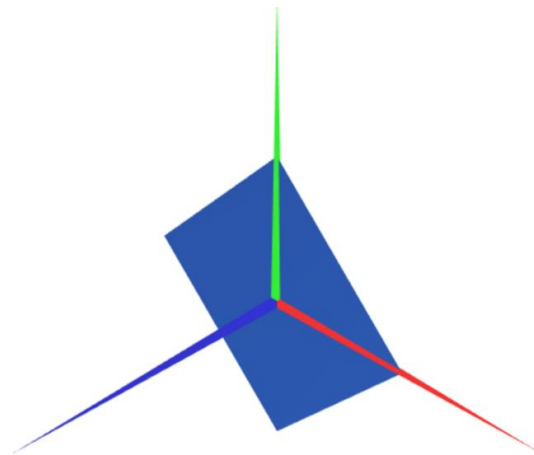
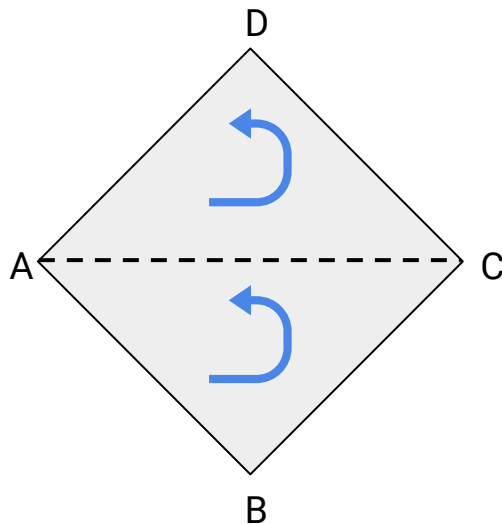
Concepts and Practice

Previous lesson

Last week we learnt how to create geometries using **vertices** and **indices**

```
this.vertices = {  
  xA,yA,zA,  
  xB,yB,zB,  
  xC,yC,zC,  
  xD,yD,zD  
}
```

```
this.indices = {  
  0, 1, 2,  
  0, 2, 3  
}
```



Transformation matrices in OpenGL/WebGL

Geometric transformations are used to **move**, **scale** and **rotate** the scene objects

The geometric transformations may be represented using **vectors** or **matrices**

For the 3D space, we use **square matrices** with 4x4 dimensions

WebGL matrices are **written differently** from their mathematical equivalent

Math translation matrix

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transposed
→
(columns ↔ rows)

OpenGL/WebGL equivalent

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

Transformation matrices in WebCGF

Geometric transformations are applied to the scene – **transformation matrix**

They **affect all elements** drawn after its application

We can define and apply these transformations using **vectors** or **matrices**

In the **WebCGF** library:

Functions for vectors

`CGFscene.translate(x,y,z)`

`CGFscene.scale(x,y,z)`

`CGFscene.rotate(angle,x,y,z)`

Functions for matrices

`CGFscene.multMatrix(matrix[4x4])`

Combining/multiplying transformations

Several transformations may be combined as follows, in the form of matrices:

Example: Apply a rotation in X axis, then a translation (as seen in theory class)

$$\begin{bmatrix} x_f \\ y_f \\ z_f \\ 1 \end{bmatrix} = T(T_x, T_y, T_z) \cdot R_x(\alpha) \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

The matrices are applied **“from right to left”**, in reverse order of how it is described textually

Combining transformations in WebCGF

Using this example, how would we apply the matrices in our WebCGF code?

In `display()` function of **MyScene**:

```
display(){  
    ...  
    matrixRotate = [...]  
    matrixTranslate = [...]  
  
    this.multMatrix(?)  
    this.multMatrix(?)  
    this.object.display();  
}
```

} Creating the matrices

} Applying matrices

— Call for object draw, after transformations

Combining transformations in WebCGF

Transformations are written in **reverse order** of how it is “described textually”

```
display(){  
  ...  
  matrixRotate = [...]  
  matrixTranslate = [...]  
  
  this.multMatrix(matrixTranslate);  
  this.multMatrix(matrixRotate);  
  this.object.display();  
}
```

Or using **vector** functions:

```
display(){  
  ...  
  
  this.translate(...);  
  this.rotate(...);  
  this.object.display();  
}
```

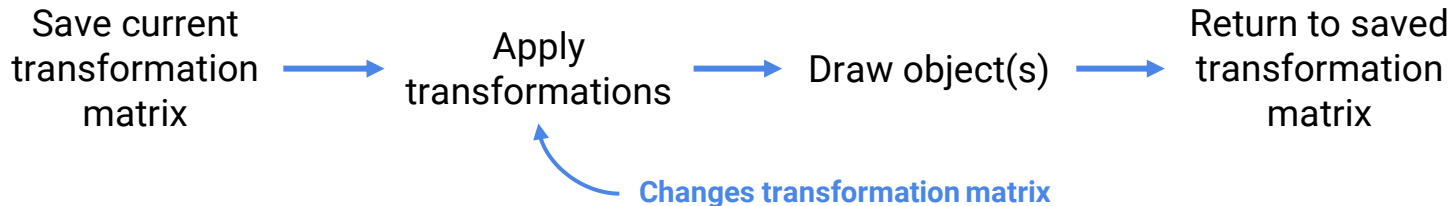
Transformations for multiple objects in WebCGF

Considering that transformations affect all objects drawn after its application:

*How to apply **different transformations** to different objects?*

The scene (**CGFscene**) contains a **matrix stack**, which can be used to store and recover previous versions of the transformation matrix

The **process** of applying the transformations to **specific objects** is as follows:



Transformations for multiple objects in WebCGF

The **CGFscene** class provides functions for manipulating the matrix stack

```
display(){  
    this.LoadIdentity();  
    ...  
    this.pushMatrix();  
    this.multMatrix(...);  
    ...  
    //Draw the objects  
    this.popMatrix();  
    ...  
    //Repeat the process  
}
```

— Sets the scene's transformation matrix to the identity matrix (already in provided code)

— Saves current transformation matrix to the stack

— Sets the scene's transformation matrix to the previously saved matrix

These functions can be **nested** to create complex objects composed of **different geometries**

Note: loadIdentity() is already present in the display() function in the provided source code. You don't need to use it again.