Computer Graphics

Minho Kim

Dept. of Computer Science

University of Seoul

Chapter 4: More Transformations and Basic Animation

What to Learn

- Be Introduced to a **matrix transformation library** that hides the mathematical details of matrix operations
- Use the library to quickly and easily **combine multiple transformations**
- Explore **animation** and how the library helps you animate simple shapes

Example #1: `RotatedTriangle_Matrix4`

Example #1: `RotatedTriangle_Matrix4`

- http://rodger.global-linguist.com/webgl/ch04/RotatedTriangle_Matrix4.html
- What to learn
  - How to use the library `cuon-matrix.js` to manipulate WebGL transformation matrices

Transformation Matrix Library: `cuon-matrix.js`

- Can be found at
  - https://github.com/yukoba/WebGLBook/tree/master/lib (Japanese)
  - https://github.com/huningxin/webglbook_examples/blob/master/lib/cuon-matrix.js
- To simplify matrix manipulations
- `Matrix4`, `Vector3`, `Vector4` for matrix and vectors
- Use `M.elements` to access the data (`Float32Array` type)
- Other alternatives
  - glMatrix "Stupidly fast WebGL Matrices" (2010)
  - glm-js (experimental)
  - J3DIMath.js

`cuon-matrix.js`: Linear Algebra

- `A.setIdentity()`: $A \leftarrow I$
- `A.set(B)`: $A \leftarrow B$
- `A.concat(B)` or `A.multiply(B)`: $A \leftarrow AB$
- `A.multiplyVector3[4](v)`: return Av
- `A.transpose()`: $A \leftarrow A^T$
- `A.setInverseOf(B)`: $A \leftarrow B^{-1}$
- `A.invert()`: $A \leftarrow A^{-1}$

`cuon-matrix.js`: Transformations

- `A.setScale(…)` : Sets A to a scaling matrix ($A \leftarrow S$)
  - `A.scale(…)` : Concatenates a scaling matrix to (the right of) A ($A \leftarrow AS$)
- `A.setTranslate(…)` ($A \leftarrow T$)
  - `A.translate(…)` : Concatenates a translate matrix to A ($A \leftarrow AT$)
- `A.setRotate(…)` : Rotation about an arbitrary axis ($A \leftarrow R$)
  - The axis (x,y,z) does not need to be normalized
  - `A.rotate(…)` : Concatenates a rotation matrix to A ($A \leftarrow AR$)
- `A.setLookAt(…)` : Sets a camera transformation (Chapter 7)

`cuon-matrix.js`: Viewing (Chapter 7)

- `A.setOrtho(…)`: Sets A to an orthogonal projection matrix
- `A.setFrustum(…)`: Sets A to a perspective projection matrix
- `A.setPerspective(…)`: Sets A to a perspective projection matrix

`cuon-matrix.js`: Basic Procedure

1. Create a `Matrix4` object.
2. Sets a proper transformation matrix.
3. Pass the matrix as a uniform variable (by passing the `elements` member variable)
4. Matrix-vector multiplication is done in the vertex shader

- Example ([`RotatedTriangle_Matrix4`]())

```
// Create Matrix4 object for a rotation matrix
var xformMatrix = new Matrix4();
// Set the rotation matrix to xformMatrix
xformMatrix.setRotate(ANGLE, 0, 0, 1);
...
// Pass the rotation matrix to the vertex shader
gl.uniformMatrix4fv(u_xformMatrix, false, xformMatrix.elements);
```

Example #2: `RotatedTranslatedTriangle`

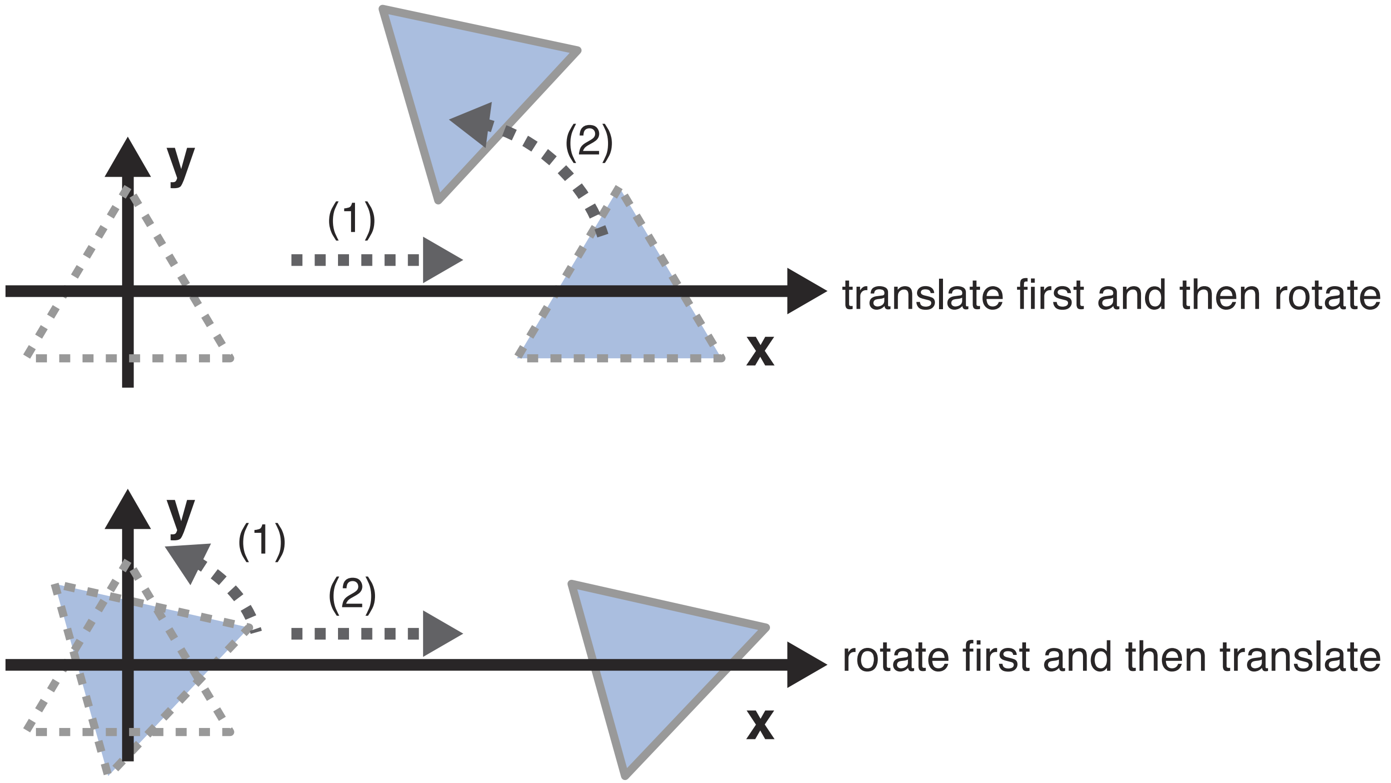
Example #2: `RotatedTranslatedTriangle`

- http://rodger.global-linguist.com/webgl/ch04/RotatedTranslatedTriangle.html
- What to learn
  - How to combine multiple transformations
- Rotation after translation

  ```
  M.setRotate(…);
  M.translate(…);
  ```
- Note the order of function calls!
- Using other libraries
  - https://xregy.github.io/webgl/src/RotatedTranslatedTriangle_glMatrix.html (glMatrix)
  - https://xregy.github.io/webgl/src/RotatedTranslatedTriangle_J3DI.html (J3DIMath)

Lab Activities

- Modify the source code to implement a “translation after rotation”.
- “Translation after rotation” is more common.


y
(1)
(2)
x
translate first and then rotate
y
(1)
(2)
x
rotate first and then translate

Example #3: `RotatingTriangle`

Example #3: `RotatingTriangle`

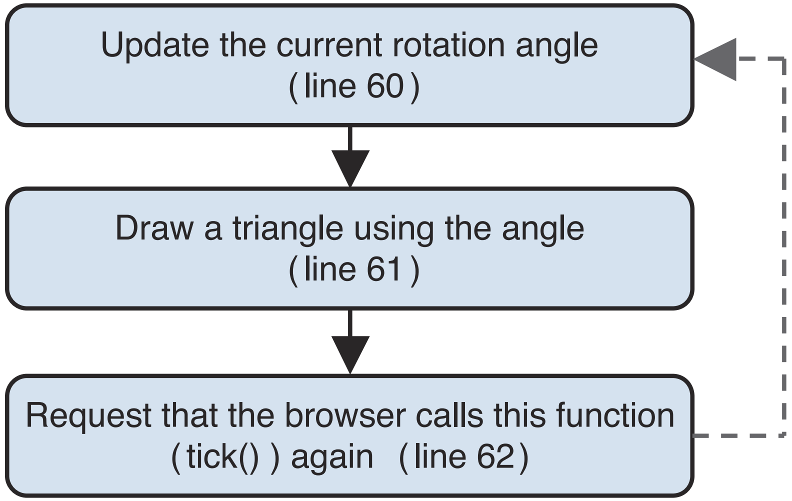
- http://rodger.global-linguist.com/webgl/ch04/RotatingTriangle.html
- Two key mechanisms required
  1. Repeatedly calls a function to draw a triangle at times t0, t1, t2, t3, and so on.
  2. Clear the previous triangle and then draws a new one with the specified angle each time the function is called.
- What to learn
  - How to animate an object using JavaScript animation feature
  - How to implement mechanism #1 using `window.requestAnimationFrame()` function

Example #3: `RotatingTriangle` (cont’d)

- Initial values are set only once in `main()`
- We need a separate `draw()` function to be called repeatedly.
- Procedures in `draw()` function
    1. Update the values required for animation and then send them to the shaders
    2. Clears `<canvas>`
    3. Draws the 3D scene

Callback Function `tick()`

- Called whenever the `<canvas>` needs to be re-drawn
- Calls `draw()` function to draw the 3D scene
- Defined and called initially in `main()`
- Defined as an anonymous function to pass the local variables in `main()` to `draw()` function.
- A `Matrix4` object, `u_ModelMatrix`, is defined in `main()` and passed to `draw()` to prevent re-creation.
- More on `requestAnimationFrame()` later


Update the current rotation angle (line 60)

Draw a triangle using the angle (line 61)

Request that the browser calls this function (tick()) again (line 62)

Request to Be Called Again

- `setInterval()`
  - Traditional way
  - Designed before “browser tabs” and executes regardless of which tab is active → performance problem
- `requestAnimationFrame()`
  - Called only when the tab in which it was defined is active.
  - Currently adopted by most browsers as `window.requestAnimationFrame()`
  - `requestAnimationFrame()` in `webgl-utils.js` handles the differences among browsers (more safe)
- Reading material: http://creativejs.com/resources/requestanimationframe

`requestAnimationFrame()` (cont'd)

- You cannot specify an interval. The callback function will be called when the browser wants the web page containing the element (2nd parameter) to be painted.
- After calling the function, you need to request the callback again because the previous request is automatically removed once it's fulfilled.
- Can be canceled by `cancelAnimationFrame()`.

Using the Rotation Angle (`animate()`)

- For consistent speed, we need to compute the values based on the elapsed time since the last call.
- JavaScript `Date` object
  - `now()`: returns the current time in milliseconds
- Current time needs to be stored in a global variable (`g_last`) for the next call.
- Do not forget to set the initial value for `g_last`!

Lab Activities

- Modify `RotatingTriangle` to
  - `RotatingTranslatedTriangle`
  - `RotatingTriangle_withButtons` (dynamic control of the rotation speed using buttons)