CSC 240 Computer Graphics Video 12: Animation & 3D Transformations in WebGL

Nick Howe
Smith College

Animation

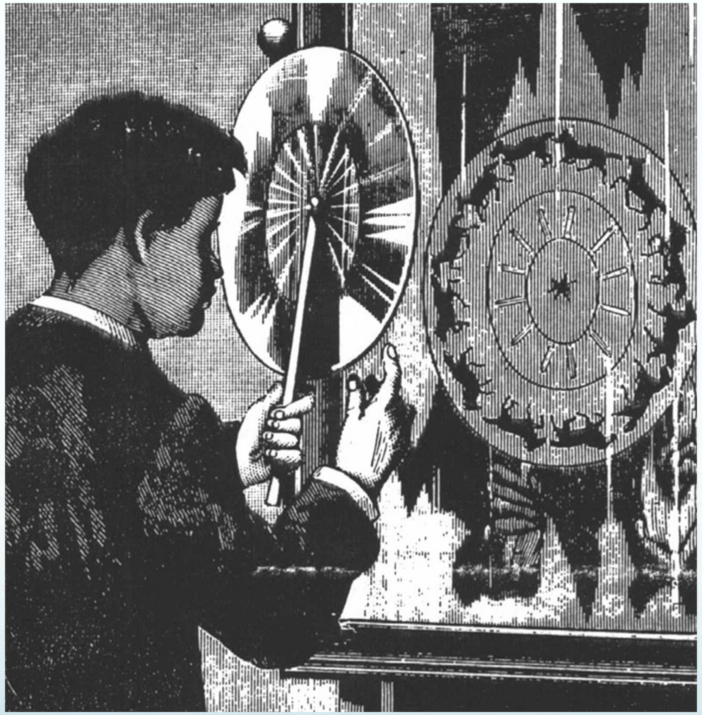
Persistence of vision is a property of the human eye

- Human visual system process at 10-12 frames/second
- Glimpsed image retained for 0.15 seconds
- Illusion of continuity if replaced by another in that time

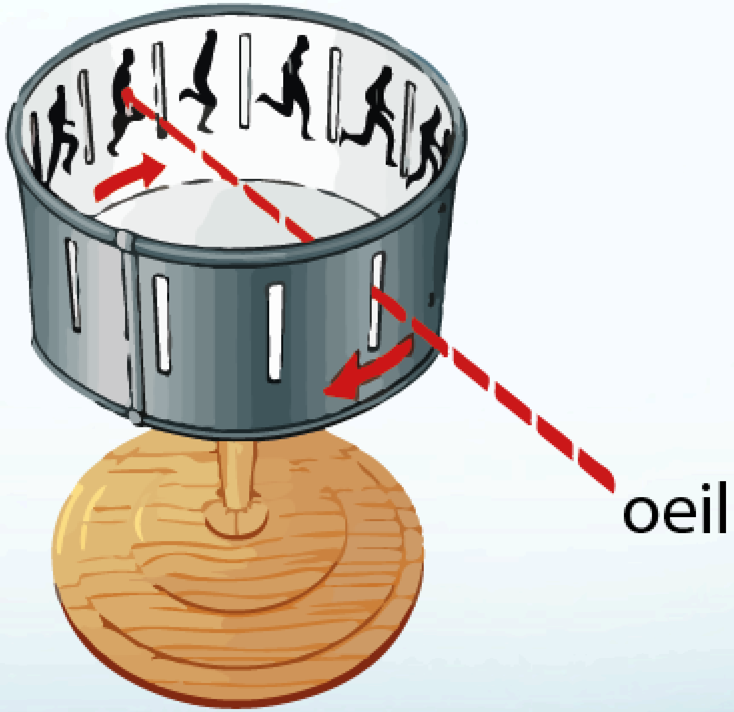
https://www.aao.org/eye-health/anatomy/parts-of-eye

Animation

Early animation devices



phenakistoscope



zoetrope

http://www.lyceelecorbusier.eu/insituhacklab/?p=1670
http://thirdmonk.net/high-culture/phenakistoscope-psychedelic-birth-animation-gallery.html

Animation

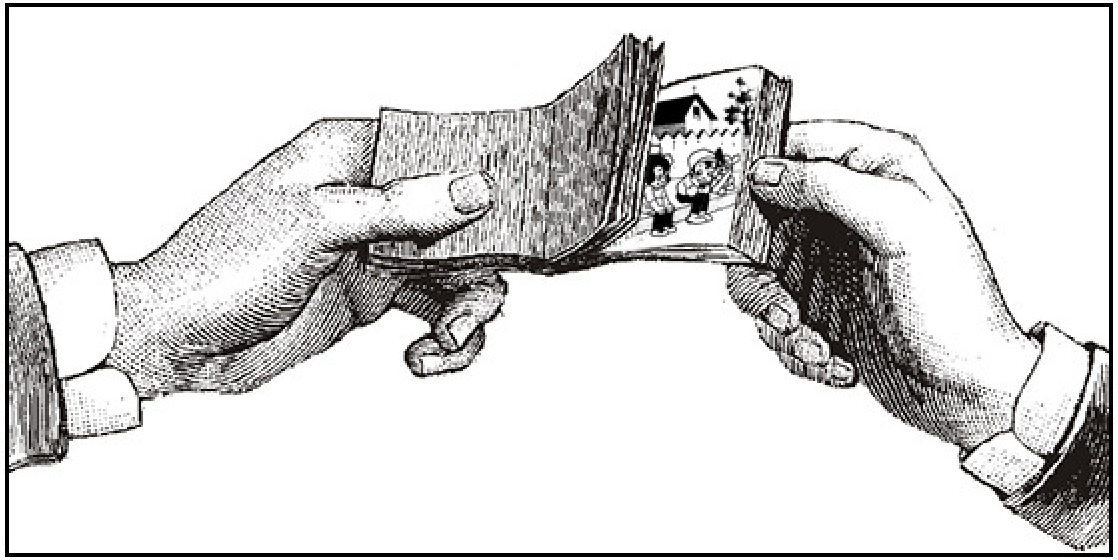
Film projection and raster displays

https://en.wikipedia.org/wiki/Cathode_ray_tube
https://www.amazon.com/Projector-Handcraft-Photograph-Bookstore-Decoration/dp/B06XCKCP7T

Animation

Web browsers typically refresh animations at 60 fps

- No refresh if frame is hidden
- Older or heavily used systems may be slower
- All animations automatically double buffered



http://infinitedictionary.com/blog/2015/06/03/design-exercise-flip-book-animation/

Animation

What's the best way to implement WebGL animation?

- We could use `setInterval` or `setTimeout`
- Better: `requestAnimationFrame(callback)`
- The argument is a **callback function**, to be invoked when the next frame is ready for drawing
- In lab6, it was called render():

These are not the same!

```
// Render the scene. This is called for each frame of the animation.
function render() {
	renderer.render(scene, camera);
}
```

Animation

Our rendering callback should accomplish three things:

1. Update the scene parameters
2. Call the WebGL renderer
3. Set a new callback for the frame after

Most of our work will happen here!

```
// Render the scene.
// This is called for each frame of the animation.
function render() {
    cube.position.z -= 0.1;
    renderer.render(scene, camera);
    requestAnimationFrame(render);
}
```

Animation

Callback functions can take a timing argument

- Use to compute elapsed time between frames

```
var then = 0;

// Render the scene.
// This is called for each frame of the animation.
function render(now) {
    var elapsed = now-then;
    console.log(elapsed);
    then = now;

    cube.position.z -= elapsed/1000;
    renderer.render(scene, camera);
    requestAnimationFrame(render);
}
```

Update position based on time elapsed for smoother motion

Questions

PAUSE NOW & ANSWER

1. Why do we perceive continuous motion when viewing a rapid sequence of still images?

   *A property of the eye called **persistence of vision**.*

2. When using `requestAnimationFrame(callback)`, what does the `callback` function need to do besides rendering the scene and calling itself again?

   It needs to update the scene.

3. How can we ensure smooth animation even with interruptions?

   Measure the time between frames and update accordingly.

Transformations in 3D

Transformations in 3D

Most transformation matrices generalize from 2D to 3D

2D:

Identity

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

3D:

Identity

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformations in WebGL

Translation: WebGL objects have a `position` property

- The `position` property has `x`, `y`, and `z` components
- Set them individually or as a group using `set()`
- Can also use `translateX()`, `translateY()`, etc.

Scale: WebGL objects have a `scale` property

- The `scale` property has `x`, `y`, and `z` components
- Set them individually or as a group using `set()`
- NO `scaleX()`, `scaleY()`, etc. provided!

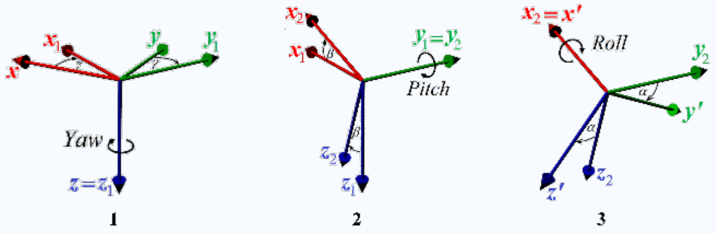
General: WebGL objects have a `matrix` property

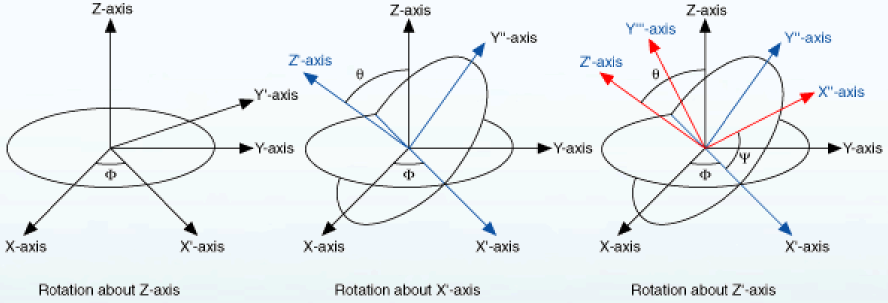
- Modify using methods above, or via `applyMatrix4()`

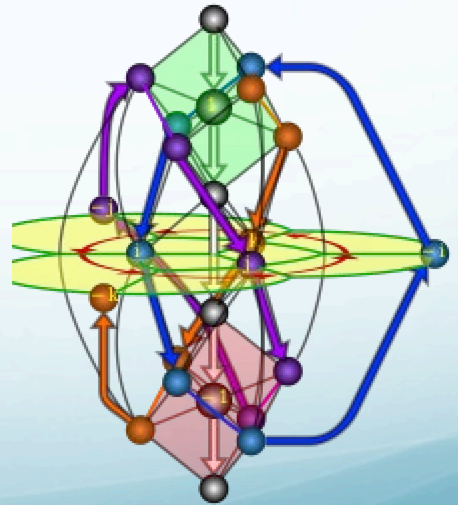
Rotations in 3D

Rotation is more complex. How to specify it?

- Azimuth + elevation
- Yaw + pitch + roll
- Euler angles
- Quaternions
- Lookat & up







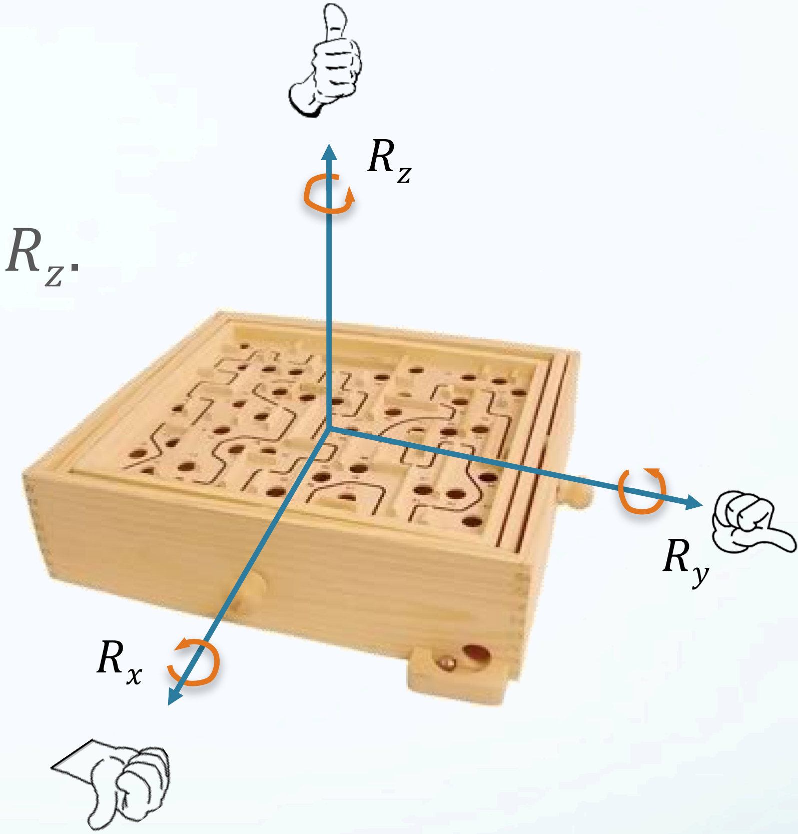
https://www.mdpi.com/1424-8220/15/3/7016
https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation
http://zone.ni.com/reference/en-XX/help/371361P-01/gmath/3d_cartesian_coordinate_rotation_euler/
https://www.turbosquid.com/3d-models/cartoon-observatory-toon-3d-model/1020233

Rotations in 3D

Three.js builds rotations out of axial components: R_x, R_y, R_z.

$$R = R_x \cdot R_y \cdot R_z$$

- WebGL objects have a `rotation` property
  - The `rotation` property has `x`, `y`, and `z` components
  - Represent rotation in radians around corresponding axis
  - Applies R_z, then R_y, then R_x
  - Can also use `rotateX()`, `rotateY()`, etc.

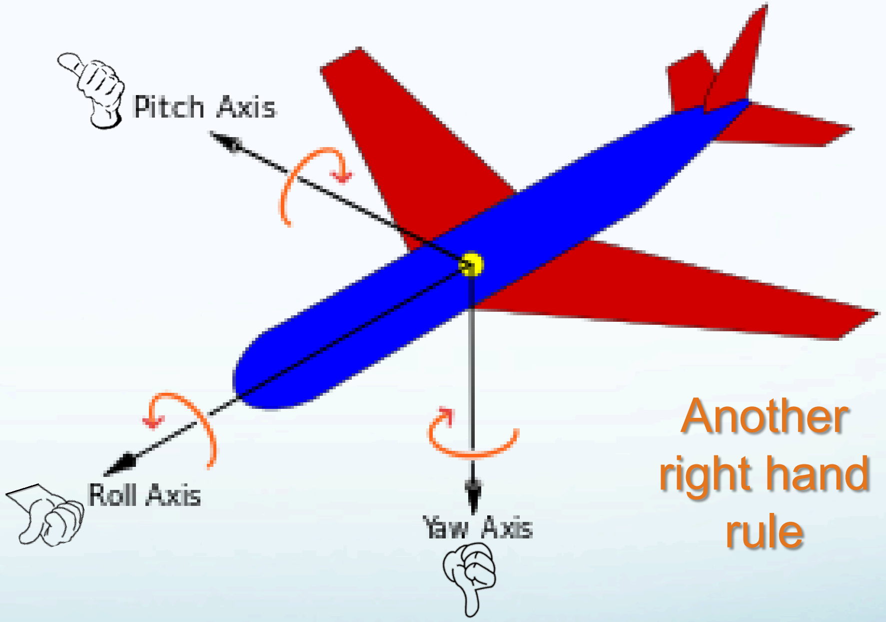


$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations in 3D

Axial component rotations also sometimes called pitch, yaw, and roll.

- Terminology from airplanes/boats
- R_x is roll
- R_y is pitch
- R_z is yaw



Another right hand rule

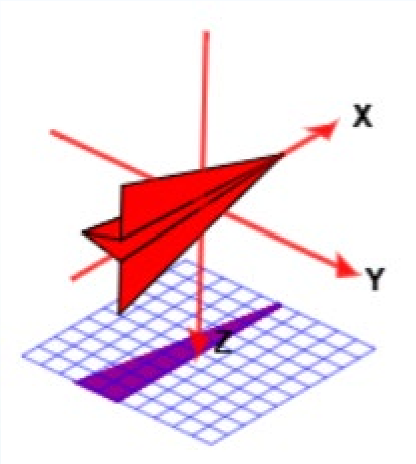
R_x, R_y, R_z also known as **Euler Angles**

https://en.wikipedia.org/wiki/Aircraft_principal_axes

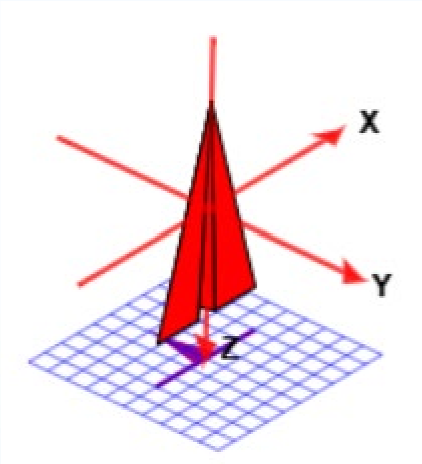
Rotations in 3D

Interactions between axial rotations can be tricky/confusing.

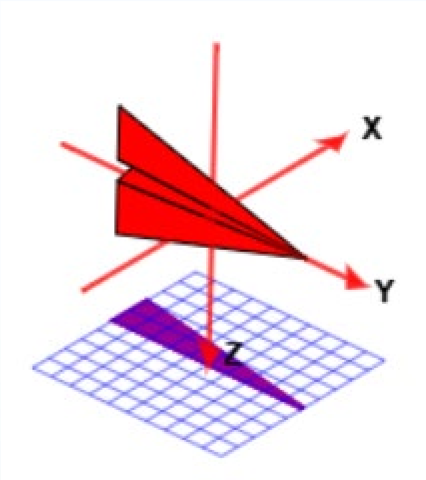
- R_x applied after R_y and R_z, which affect the end result



$R_x = 90°$ alone

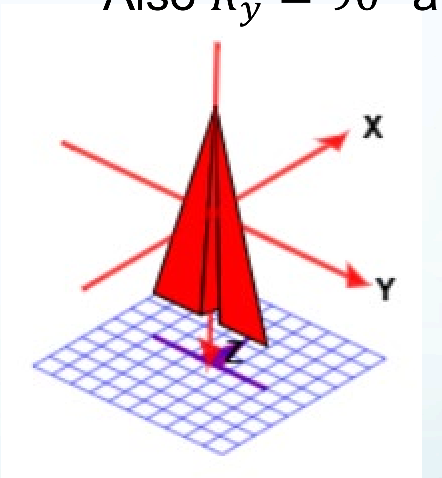


$R_x = 90°$
with $R_y = 90°$



$R_x = 90°$
with $R_z = 90°$

Also $R_y = 90°$ alone!



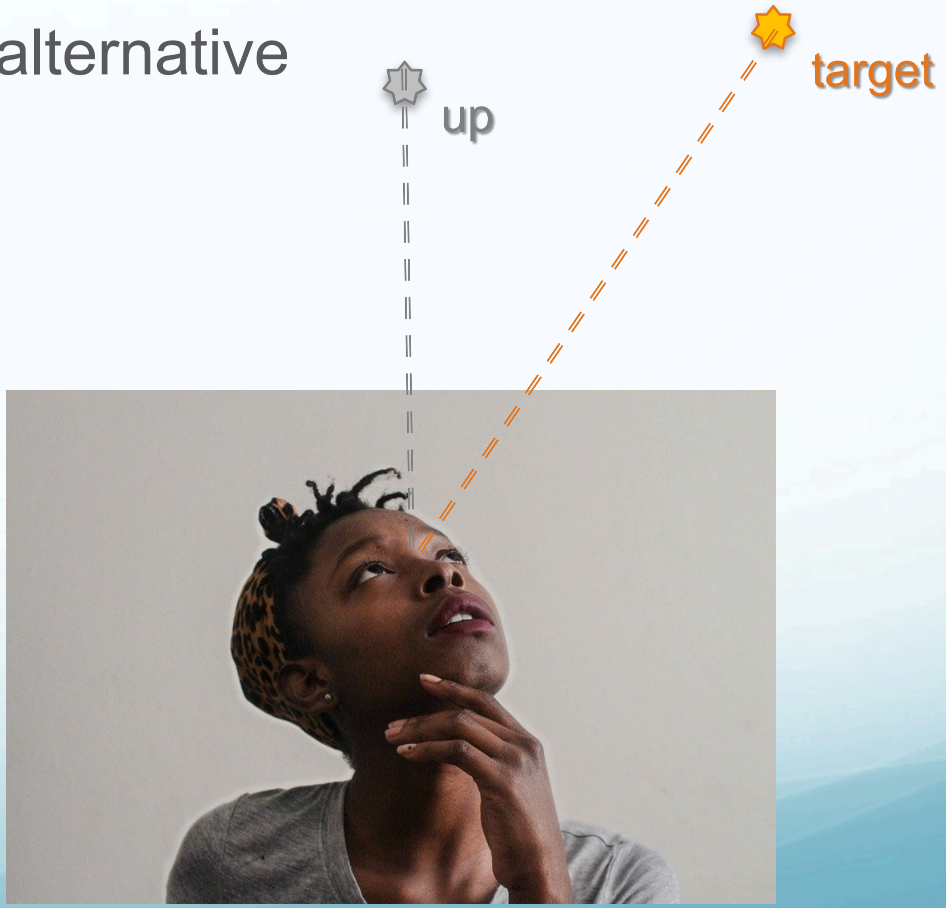
$R_x = 90°$
with $R_y = 90°$
and $R_z = 90°$

http://danceswithcode.net/engineeringnotes/rotations_in_3d/demo3D/rotations_in_3d_tool.html

LookAt

Other methods of specifying rotation can sometimes be simpler.

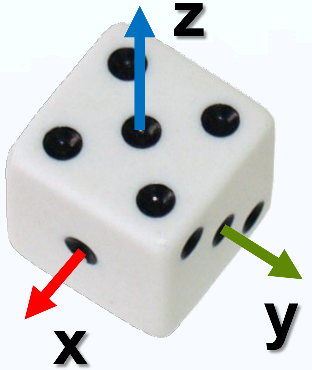
- Three.js provides a `lookAt` method as an alternative
- Specify target point; rotates object to align
- Object uses property `up` to get twist around the line of sight



Questions

PAUSE NOW & ANSWER

1. What number would be on top after each rotation, always starting from the position shown?
   a. rotation.x = Math.PI/2; *3*
   b. rotation.y = Math.PI/2; *6*
   c. rotation.z = Math.PI/2; *5*



(Note: face opposite 5 is 2; opposite 3 is 4; opposite 1 is 6.)

2. What transformations are performed by the matrices below?

a. $\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Translation

b. $\begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Rotation around Y

c. $\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Scale & Translation

d. $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & .71 & -.71 & 0 \\ 0 & .71 & .71 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ Rotation around X

3. What WebGL code would generate such a transformation for object Q?

```
Q.position.set(3,2,1);
Q.rotation.y = Math.PI/2;
Q.scale.y = 5; Q.position.x = 5 = 5;
Q.rotation.x = Math.PI/4;
```

Review

After watching this video, you should be able to…

- Write a callback function to perform animation in Three.js
- Express 3D transformations mathematically as 4D homogeneous matrices
- Express 3D rotation as a composition of R_x, R_y, and R_z component rotations
- Apply translation, scaling and rotation to 3D objects in Three.js
- Use `lookAt` as an alternative to component rotations