CS380: Introduction to Computer Graphics

Hello World 3D

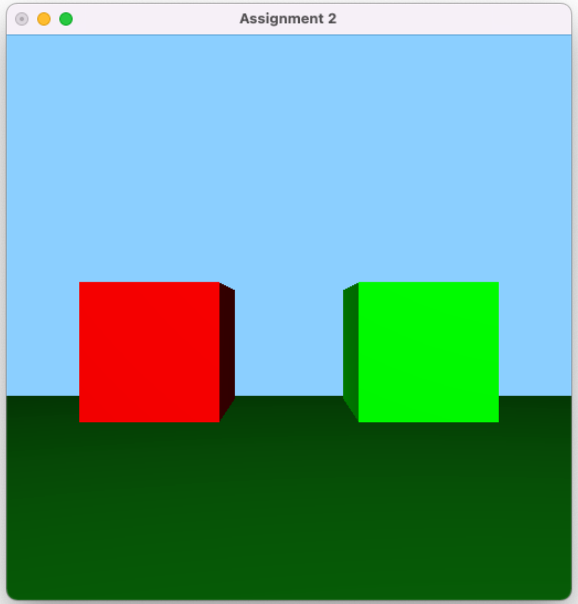
Lab Session 2
Juil Koo

Spring 2023
KAIST

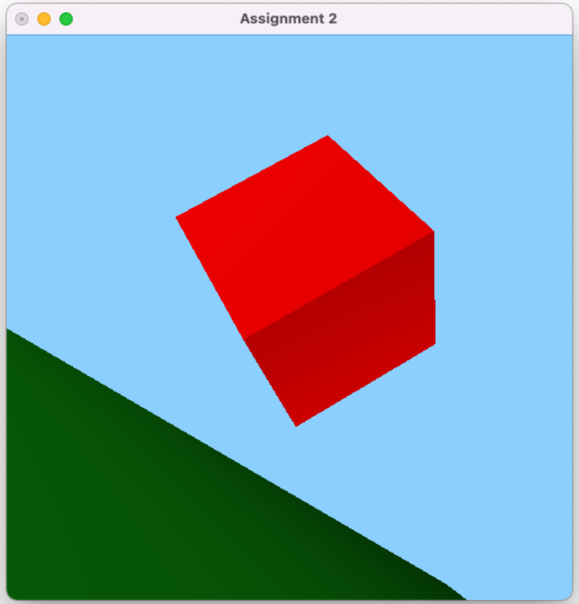
Tasks

- Complete **linFact** and **transFact** functions in **matrix.h**.

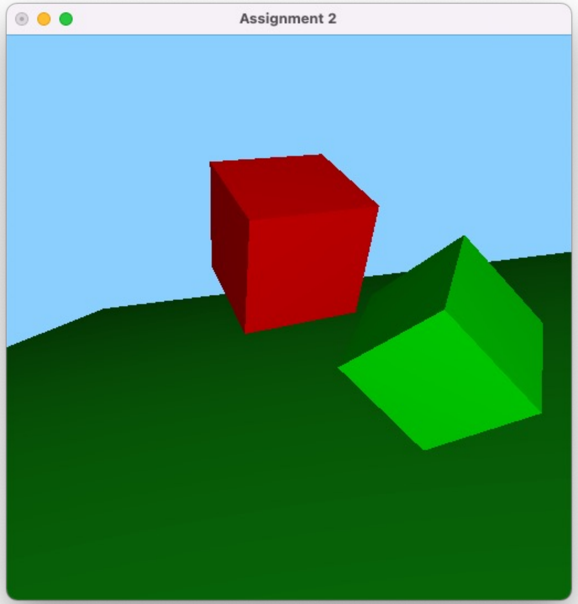
- Drawing cubes.



- Viewpoint change.



- Object manipulation w.r.t. a current viewpoint.



Recap: Affine Transformation Matrix

Factorization of an affine transformation matrix.

Full affine matrix A — Translation T — Linear L

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a & b & c & 0 \\ e & f & g & 0 \\ i & j & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A = TL$$

Recap: Notations

$$\tilde{p} = \vec{\mathbf{f}}^t \mathbf{c} = (\vec{\mathbf{f}}^t A)(A^{-1}\mathbf{c})$$

$$= \vec{\boldsymbol{a}}^t \mathbf{d}, \text{ where } \vec{\boldsymbol{a}}^t = \vec{\mathbf{f}}^t A \text{ and } \mathbf{d} = A^{-1}\mathbf{c}.$$

We can express a point $\tilde{p}$ with a new frame $\vec{\boldsymbol{a}}^t$ and new coordinates $\mathbf{d}$.

Recap: Respect

- Transform a point.

  $\tilde{p} = \vec{\mathbf{f}}^t \mathbf{c} \Rightarrow \tilde{q} = \vec{\mathbf{f}}^t S c : \tilde{p}$ is transformed by S with respect to $\vec{\mathbf{f}}^t$.

  $\tilde{p} = \vec{\mathbf{a}}^t A^{-1} \mathbf{c} \Rightarrow \tilde{q} = \vec{\mathbf{a}}^t S A^{-1} c : \tilde{p}$ is transformed by S with respect to $\vec{\mathbf{a}}^t$.

- Transform a frame.

$\vec{\mathbf{f}}^t \Rightarrow \vec{\mathbf{f}}^t S : \vec{\mathbf{f}}^t$ is transformed by S with respect to $\vec{\mathbf{f}}^t$.

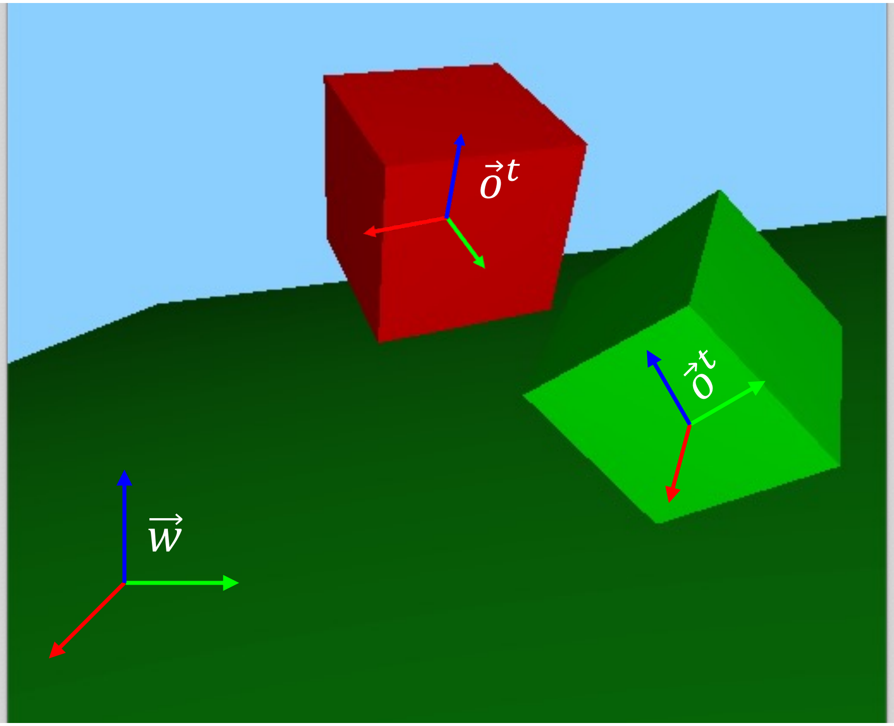
$\vec{\mathbf{f}}^t = \vec{\mathbf{a}}^t A^{-1} \Rightarrow \vec{\mathbf{a}}^t S A^{-1} : \vec{\mathbf{f}}^t$ is transformed by S with respect to $\vec{\mathbf{a}}^t$.

Frames

- World frame $\vec{\boldsymbol{w}}^t$
  - An absolute frame in the 3D space.
  - All other frames are represented based on this frame.

- Object frame $\vec{\boldsymbol{o}}^t$
  - All objects have their own frames.
  - $\vec{\boldsymbol{o}}^t = \vec{\boldsymbol{w}}^t O$

- Eye frame $\vec{e}^t$
  - $\vec{\boldsymbol{e}}^t = \vec{\boldsymbol{w}}^t E$



Eye Coordinate

We explicitly store the matrix E. $(\vec{\boldsymbol{e}}^t = \vec{\boldsymbol{w}}^t E)$

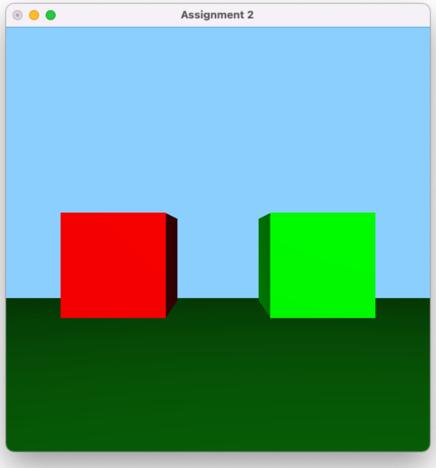
$$\tilde{p} = \vec{\mathbf{o}}^t \mathbf{c} = \vec{\boldsymbol{w}}^t O\mathbf{c} = \vec{\boldsymbol{e}}^t E^{-1} O\mathbf{c}$$

- Object coordinates: $\mathbf{c}$
- World coordinates: $O\mathbf{c}$
- Eye coordinates: $E^{-1}O\mathbf{c}$ → MVM = inv(eyeRbt) * objRbt; (Model View Matrix)

E^{-1} above inv(eyeRbt); O above objRbt

Task 1: Drawing Cubes

- The base code draws only a red cube.
- You need to add another cube with a different color.
- Two cubes should be displayed in the first screen of the execution.



Task 2: Complete linFact and transFact

In matrix4.h, complete linFact and transFact functions.

```
inline Matrix4 transFact(const Matrix4& m) {
  // TODO
}

inline Matrix4 linFact(const Matrix4& m) {
  // TODO
}
```

Task 3: Viewpoint Change

- The base code contains a matrix representing the eye frame.

```
static void drawStuff() {
  // short hand for current shader state
  const ShaderState& curSS = *g_shaderStates[g_activeShader];

  // build & send proj. matrix to vshader
  const Matrix4 projmat = makeProjectionMatrix();
  sendProjectionMatrix(curSS, projmat);

  // use the skyRbt as the eyeRbt
  const Matrix4 eyeRbt = g_skyRbt;
```

- Make the eye frame matrix change when the “v” key is pressed.

- The eye frame should cycle between the sky-camera, Cube 1 and Cube 2.

Task 4: Object Manipulation

- The base code is only able to manipulate the red cube.

- Make the object being modified change when the "o" key is pressed.

- $\vec{\mathbf{a}}^t$, the frame w.r.t the object being manipulated, should be transformed according to a current view and an object being modified.

- The signs of the rotations/translations should be inverted according to a combination of (1) a current view, (2) an object being modified and (3) a current $\vec{\mathbf{a}}^t$.

Recap: Desired Auxiliary Frame $\vec{\mathbf{a}}^t$

- Object's affine transformation: $O = O_T O_R$
- Eye's affine transformation: $E = E_T E_R$

$$\vec{\mathbf{a}}^t = \vec{\mathbf{w}}^t O_T E_R$$

O_T :Transform the object at its origin.
E_R: Rotation axis is the y axis of the eye.

Task 4: Object Manipulation

In the case below, $\vec{\mathbf{a}}^t$ should be switched between *world-sky* frame and *sky-sky* frame by pressing "m".

	Eye frame	Manipulated object	
selected	Sky-camera	Sky-camera	selected
	Cube 1	Cube 1	
	Cube 2	Cube 2	

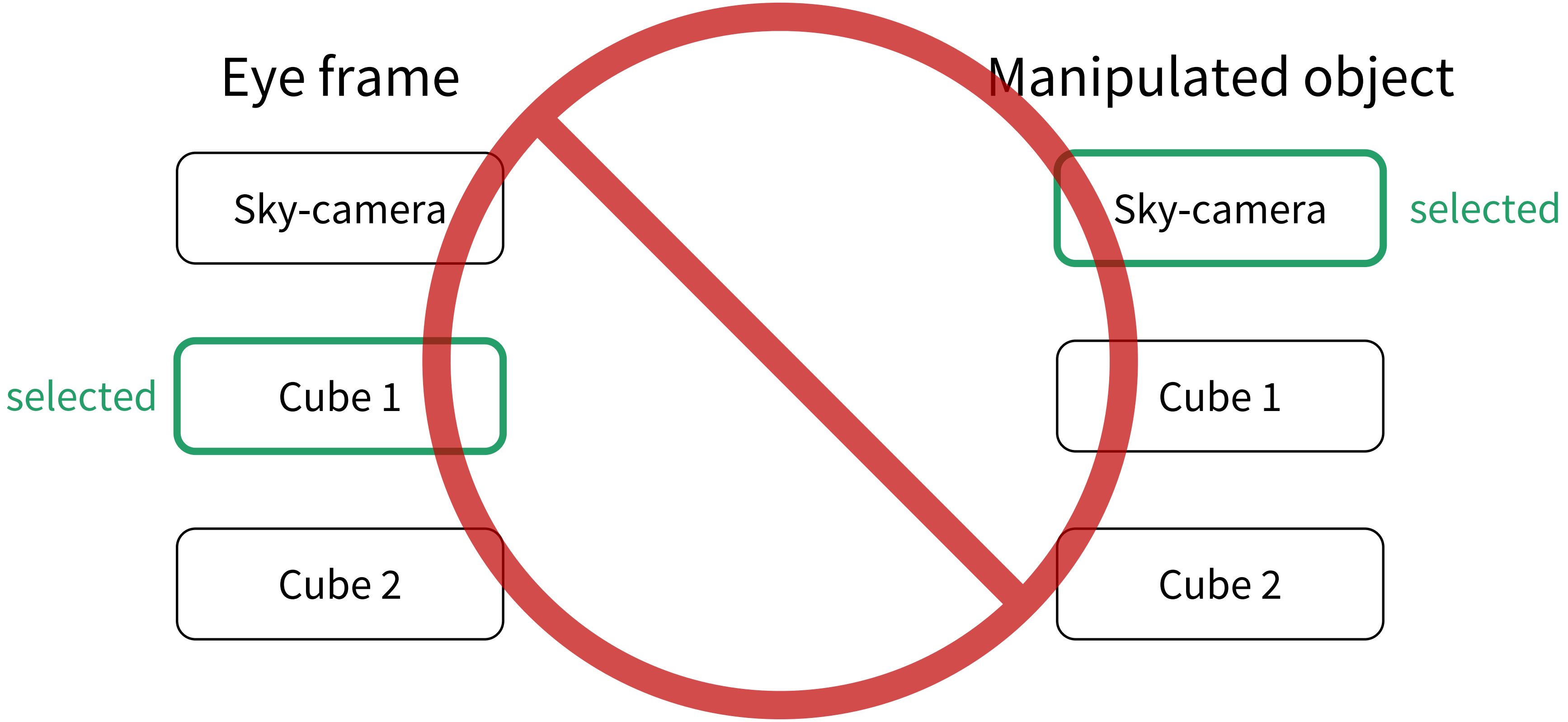
Task 4: Object Manipulation

In the case below, $\vec{\mathbf{a}}^t$ should be switched between *world-sky* frame and *sky-sky* frame by pressing "m".

- *World-sky* frame: center at world's origin and axes aligned with the sky camera.
- *Sky-sky* frame: the sky camera's frame itself.

Task 4: Object Manipulation

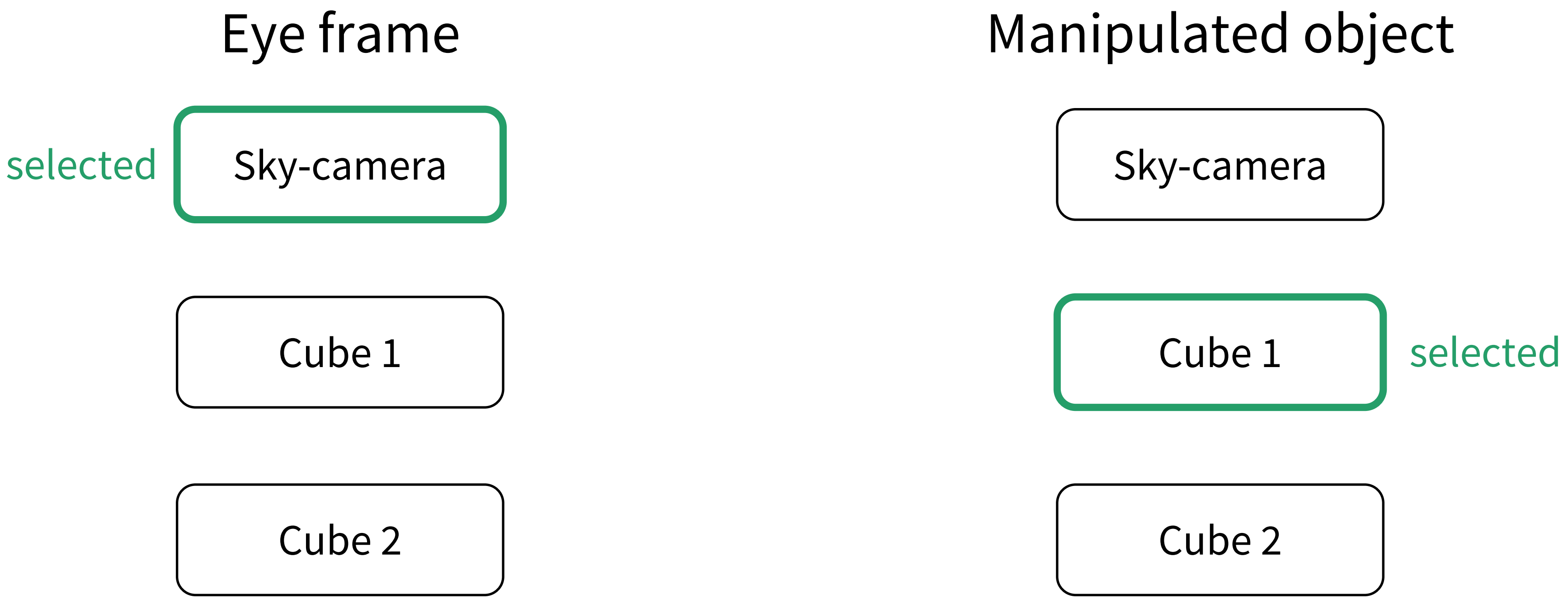
It is NOT allowed to manipulate the sky-camera when a current eye is one of the cubes.



Task 4: Object Manipulation

The directions of the rotation/translation depend on the following three cases:

1. When manipulating one of the cubes and the eye is different from the cube,



Task 4: Object Manipulation

The directions of the rotation/translation depend on the following three cases:

1. When manipulating one of the cubes and the eye is different from the cube,
   - (a) Pressing the left mouse button and moving the mouse to the right : a positive y-rotation.
   - (b) Pressing the left mouse button and moving the mouse upwards: a negative x-rotation.
   - (c) Pressing the right mouse button and moving the mouse to the right: a positive x-translation.
   - (d) Pressing the right mouse button and moving the mouse upwards: a positive y-translation.

   (These are identical to the default directions in the base code.)

Task 4: Object Manipulation

The directions of the rotation/translation depend on the following three cases:

2. If both the object being manipulated and the eye are the sky camera and $\vec{\mathbf{a}}$ is the world-sky frame,

Eye frame		Manipulated object
selected: **Sky-camera**	$\vec{a}$: *world-sky* frame	**Sky-camera** selected
Cube 1		Cube 1
Cube 2		Cube 2

Task 4: Object Manipulation

The directions of the rotation/translation depend on the following three cases:

2. If both the object being manipulated and the eye are the sky camera and $\vec{\mathbf{a}}$ is the world-sky frame, invert the sign of both the rotations and the translations.

Task 4: Object Manipulation

The directions of the rotation/translation depend on the following three cases:

3. Else, invert the sign of only the rotations.

Task 4: Object Manipulation

To sum up,

1. When manipulating one of the cubes and the eye is different from the cube,
   (a) Pressing the left mouse button and moving the mouse to the right : a positive y-rotation.
   (b) Pressing the left mouse button and moving the mouse upwards: a negative x-rotation.
   (c) Pressing the right mouse button and moving the mouse to the right: a positive x-translation.
   (d) Pressing the right mouse button and moving the mouse upwards: a positive y-translation.
   (These are identical to the default directions in the base code.)

2. If both the object being manipulated and the eye are the sky camera and $\vec{a}$ is the world-sky frame, invert the sign of both the rotations and the translations.

3. Else, invert the sign of only the rotations.

How to Submit

- Record a video with your mouse cursor that shows the following actions:
    1. Changing the viewpoint between the sky-camera, Cube 1 and Cube2.
    2. Rotating/translating Cube 1 when the eye is the sky-camera.
    3. Rotating/translating Cube 1 when the eye is the Cube2.
    4. Rotating/translating the sky-camera when $\vec{\mathbf{a}}$ is *world-sky* frame or *sky-sky* frame.

- Compress the files including both the video and your code, and submit a zip file on GradeScope.

- Due date: Mar. 29 (Wed) 23:59 KST.