CPSC 314
Computer Graphics

Dinesh K. Pai

Geometry 2: Transformations and Homogeneous Coordinates

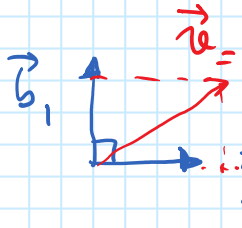
Preliminaries

- Announcements
  - A1. Any questions so far?
    - Note that the assignments are meant to be done in parallel with the classes. If you've done up to 1(b) and read the code so far, you're in good shape.
    - 1 (c) specifically asks you to compute the color in the vertex shader ("Gouraud" shading). Don't use/borrow code to do this in the fragment shader.
- Today: Theory (review Chapter 2 of text)
  - Review and notation
  - Orthonormal Basis.
  - Linear transformations and matrices (mat3 or mat3x3)
  - Homogeneous coordinates

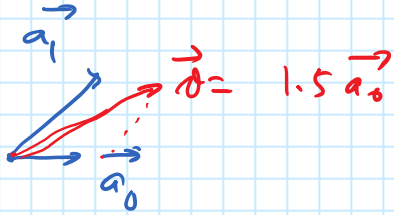
§ Notation

	Ours	Textbook
Points	$\tilde{p}$	$\tilde{p}$
vector	$\vec{v}$	$\vec{v}$
column matrix	$\underline{v}$	$\mathbf{v}$ ← bold font
row matrix	$\underline{v}^T$	$\mathbf{v}^T$
basis	$\underline{\vec{b}}$	$\vec{\mathbf{b}}^T$

§ Orthonormal basis



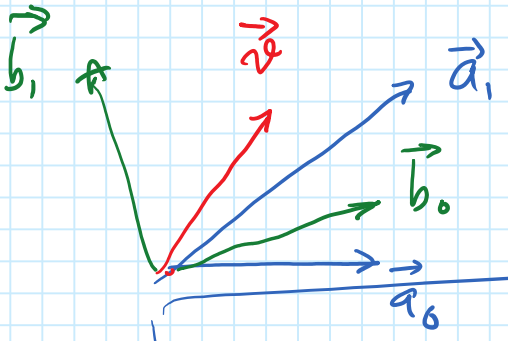
Basis vectors are of unit length, and perpendicular to each other



Computing the component along any basis is easy.

$$\vec{v} \cdot \vec{b}_0 = 1.1 \underbrace{\vec{b}_0 \cdot \vec{b}_1}_{1} + 0.9 \underbrace{\vec{b}_0 \cdot \vec{b}_1}_{0}$$

§ Change of basis



Two bases $\underline{\vec{a}}$ and $\underline{\vec{b}}$

Given $\underline{\vec{v}} = \underline{\vec{b}}\, \underline{v_b} = \underline{\vec{a}}\, \underline{v_a}$ (1)

given ↑ want ↑

Express $\underline{\vec{b}} = (\vec{b}_0 \;\; \vec{b}_1)$

$\underline{\vec{a}} \begin{pmatrix} l_{00} \\ l_{10} \end{pmatrix} \quad \underline{\vec{a}} \begin{pmatrix} l_{01} \\ l_{11} \end{pmatrix}$

$= \underline{\vec{a}} \begin{pmatrix} l_{00} & l_{01} \\ l_{10} & l_{11} \end{pmatrix}$

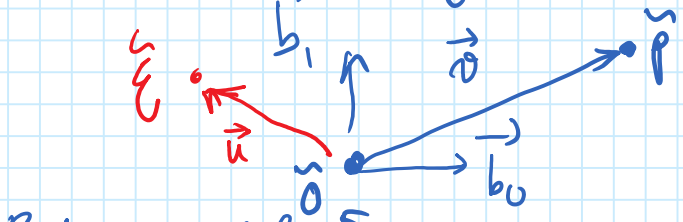
$$\boxed{\underline{\vec{b}} = \underline{\vec{a}}\, \underline{\underline{L}}} \qquad (2)$$

Plugging ② into ①

$$\vec{a}\,\underline{L}\,\bar{v}_b = \vec{a}\,\bar{v}_a$$

$$\square\,1 \qquad 1$$

§ Representing points (and vectors)



Pick a special point "origin"

Extend "+" to allow addition of a point and a vector

$$\check{p} = \vec{v} + \check{o}$$

$$= \underline{\vec{b}_0 v_0 + \vec{b}_1 v_1} + \check{o}$$

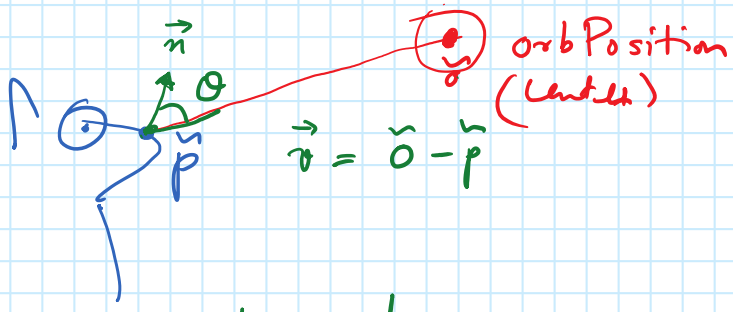
$$= \underbrace{\begin{pmatrix} \vec{b}_0 & \vec{b}_1 & \check{o} \end{pmatrix}}_{\text{A coordinate frame}} \begin{pmatrix} v_0 \\ v_1 \\ 1 \end{pmatrix}$$

homogeneous coordinates of a point

September 20, 2021 2:58 PM

§ A1

1b)



vertex color

$vColor = \max(\cos\theta, 0)$

§ Recap

$$\tilde{p} = \vec{v} + \tilde{o}$$

$$= \vec{b}_0 v_0 + \vec{b}_1 v_1 + \tilde{o}$$

$$= \begin{pmatrix} \vec{b}_0 & \vec{b}_1 & \tilde{o} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ 1 \end{pmatrix}$$

homogeneous coordinate of a point

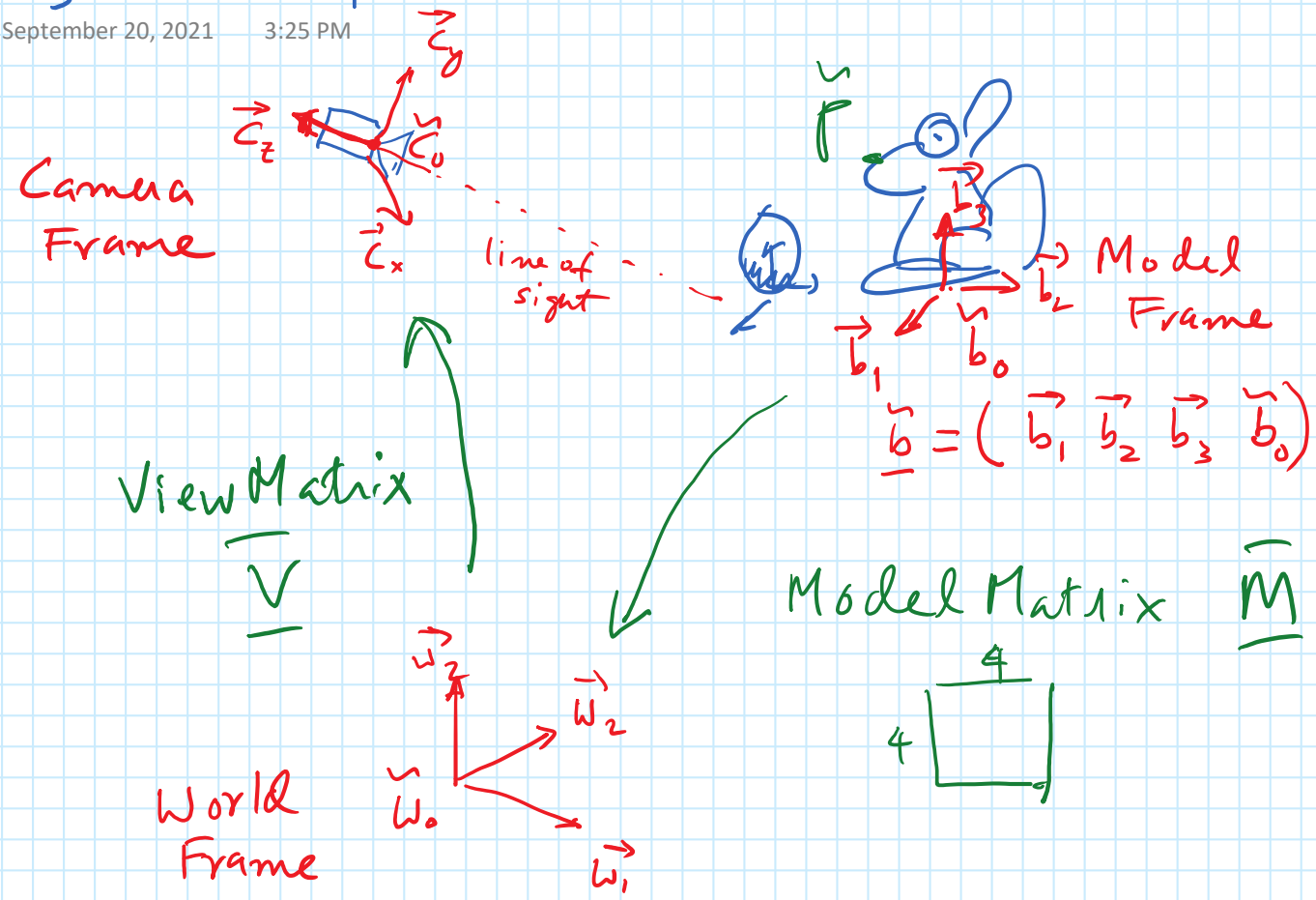
A coordinate frame

Notation:

Frame $\underset{\sim}{b} = \left(\vec{b}_0 \;\; \vec{b}_1 \;\; \tilde{o}\right)$

§ Three important frames

September 20, 2021 3:25 PM



model view matrix

$$\bar{P}_c = (\ )\,\underline{V}\,\underline{M}\,\bar{P}$$

gl_position — Projection (later) — position

Three.js support

- Recall: Built-in uniforms and attributes
  https://threejs.org/docs/#api/en/renderers/webgl/WebGLProgram

Vertex shader (unconditional):

```
// = object.matrixWorld
uniform mat4 modelMatrix;

// = camera.matrixWorldInverse * object.matrixWorld
uniform mat4 modelViewMatrix;

// = camera.projectionMatrix
uniform mat4 projectionMatrix;

// = camera.matrixWorldInverse
uniform mat4 viewMatrix;

// = inverse transpose of modelViewMatrix
uniform mat3 normalMatrix;

// = camera position in world space
uniform vec3 cameraPosition;
```

```
// default vertex attributes provided by Geometry and BufferGeo
attribute vec3 position;
attribute vec3 normal;
attribute vec2 uv;
```

Fragment shader:

```
uniform mat4 viewMatrix;
uniform vec3 cameraPosition;
```

For next class

- Try the transforms app by Eric Haines that I showed in class, it's available here:
  https://www.realtimerendering.com/udacity/transforms.html
- Review Chapter 5 of textbook

For next class

- Review Chapter 4 and 5 of textbook.