CS380: Introduction to Computer Graphics

Linear Transformation

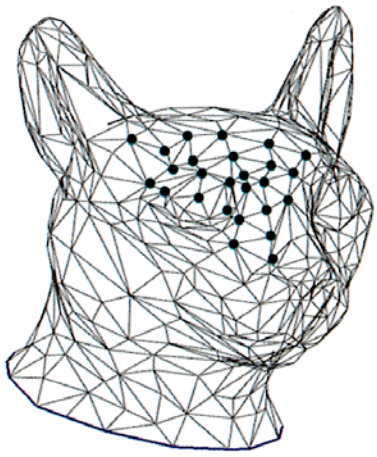
Chapter 2

Min H. Kim

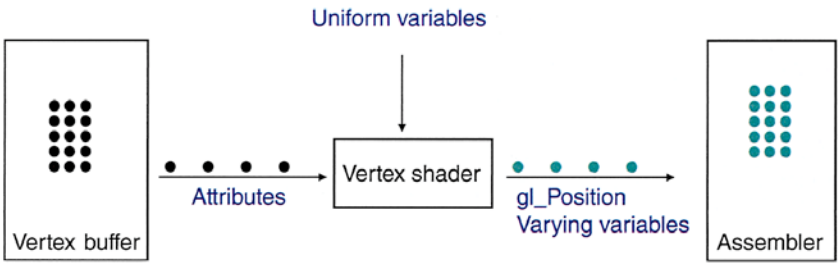
KAIST School of Computing

RECAP

GLSL PIPELINE

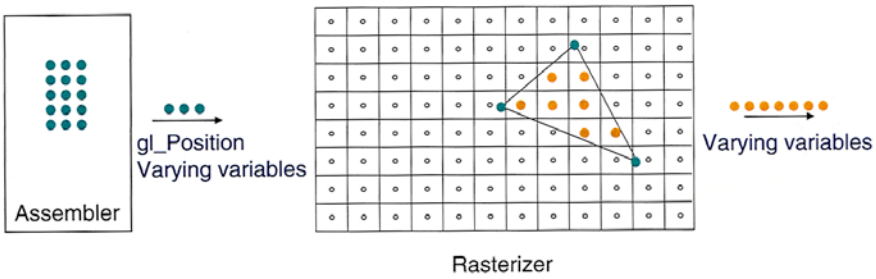


GLSL Pipeline: Vertex Shader



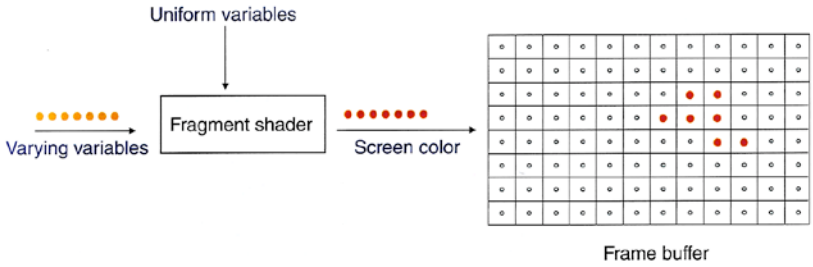
- Vertices are stored in a vertex buffer.
- When a draw call is issued, each of the vertices passes through the vertex shader
- On input to the vertex shader, each vertex (black) has associated attributes.
- On output, each vertex (cyan) has a value for gl_Position and for its varying variables.

GLSL Pipeline: Rasterization



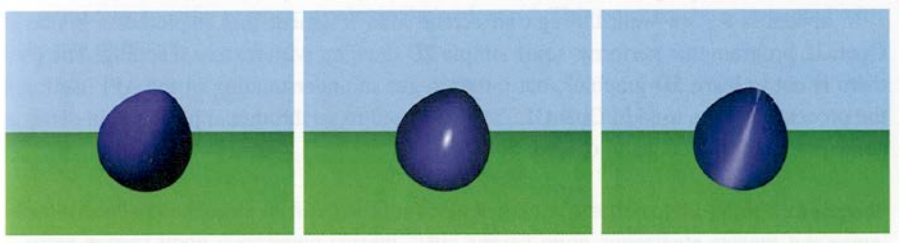
- The data in gl_Position is used to place the three vertices of the triangle on a virtual screen.
- The rasterizer figures out which pixels (orange) are inside the triangle and interpolates the varying variables from the vertices to each of these pixels.

GLSL Pipeline: Fragment Shader



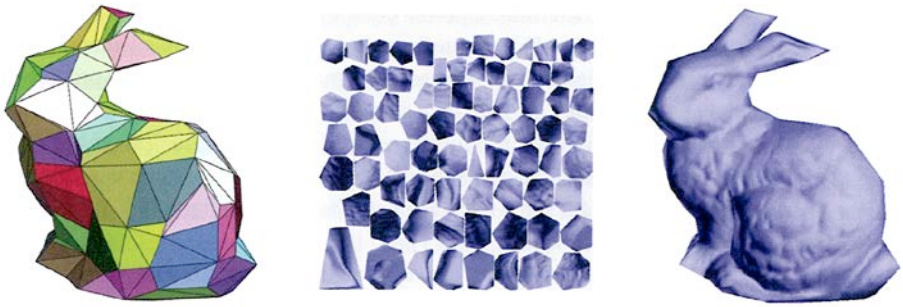
- Each pixel (orange) is passed through the fragment shader, which computes the final color of the pixel (pink).
- The pixel is then placed in the frame buffer for display.

GLSL Pipeline: Fragment Shader



- By changing the fragment shader, we can simulate light reflecting of different kinds of materials.

Texture Mapping



- A simple geometric object described by a small number of triangles.
- An auxiliary image called a texture.
- Parts of the texture are glued onto each triangle giving a more complicated appearance.

Chapter 2

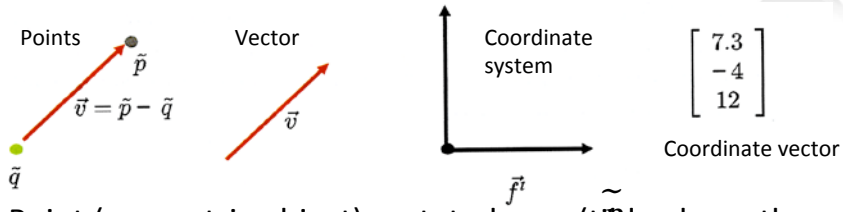
LINEAR TRANSFORMATION

Point vs. Vector

- Represent Points using coordinates
- To perform geometric transformations to these points
- Vectors: 3D motion via linear transformations
- Coordinate vector: the position of the point

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Point vs. Coordinate Vector



1. Point (geometric object): notated as $\tilde{p}$ (tilde above the letter), non-numerical object.
2. Vector (motion): notated as $\vec{a}$ (arrow above the letter), non-numerical object.
3. Coordinate system: denoted as $\vec{f}^t$ (bold: column vector, *t* makes it transpose), non-numerical object
   basis for vector; frame for point
4. Coordinate vector: noted as $\mathbf{c}$ (bold letter), numerical object

Vector Space

- A vector space V is some set of elements $\vec{v}$
- NB: Vector (motion) is NOT just a set of three numbers!!!
- If a set of vectors is not linearly dependent, we call linearly independent.
- If $\vec{b}_1 \ldots \vec{b}_n$ are linearly independent, all vectors $\vec{v}$ of V can be expressed with coordinates c_i of a basis of V (a set of $\vec{b}_i$).

$$\vec{v} = \sum_i c_i \vec{b}_i \, .$$

- n is the dimension of the basis/space

Vector Space

- Free motion in space, 3 dimensional vector
- In vector algebra notation:

$$\vec{v} = \sum_i c_i \vec{b}_i = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} .$$

- a vector $\vec{v}$
- row basis vectors $\vec{\mathbf{b}}^t$
- column coordinate vector $\mathbf{c}$

$$\vec{v} = \vec{\mathbf{b}}^t \mathbf{c}.$$

Linear Transformation

- Linear transformation follows these two properties:

$$L(\vec{v}+\vec{u}) = L(\vec{v}) + L(\vec{u})$$

$$L(\alpha\vec{v}) = \alpha L(\vec{v}) .$$

- Vector transformation (such that the basis is linearly independent):

$$\vec{v} \Rightarrow L(\vec{v}) = L\left(\sum_i c_i \vec{b}_i\right) = \sum_i c_i L(\vec{b}_i) .$$

3-by-3 Transformation

- Rewrite the linear transform

$$\begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \begin{bmatrix} L(\vec{b_1}) & L(\vec{b_2}) & L(\vec{b_3}) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} .$$

- $L(\vec{b_1})$ is actually a linear combination of the original basis vectors.

$$L(\vec{b_1}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} M_{1,1} \\ M_{2,1} \\ M_{3,1} \end{bmatrix}$$

3-by-3 Transformation

- 3-by-3 matrix:

$$\begin{bmatrix} L(\vec{b}_1) & L(\vec{b}_2) & L(\vec{b}_3) \end{bmatrix} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}.$$

- Putting all together:

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}.$$

- A matrix to transform one vector to another:

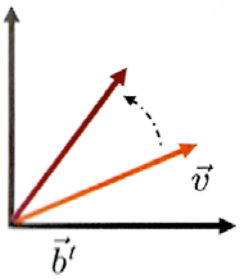
$$\vec{v} = \vec{\mathbf{b}}^t \mathbf{c} \Rightarrow \vec{\mathbf{b}}^t M\mathbf{c}$$

Linear transform of a vector

- A vector undergoes a linear transformation

$$\vec{v} = \vec{\mathbf{b}}^t \mathbf{c} \Rightarrow \vec{\mathbf{b}}^t M\mathbf{c}$$

- The matrix *M* depends on the chosen linear transformation.



$$\vec{v} \Rightarrow L(\vec{v})$$

Inverse transform

- Identity matrix

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

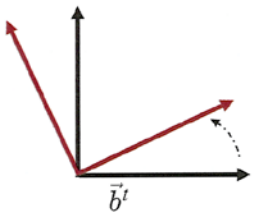
$$MM^{-1} = M^{-1}M = I\ .$$

- In 3D graphics, while moving objects around in space, it will seldom make sense to use an non-invertible transform.

Linear transform of a basis

- A basis undergoes a linear transformation

$$\vec{\mathbf{b}}^t \Rightarrow \vec{\mathbf{b}}^t M$$



- Valid to multiply a matrix times a coordinate vector
- change a basis of a vector $\vec{\mathbf{b}}^t$ to $\vec{\mathbf{a}}^t$

$$\vec{\mathbf{a}}^t = \vec{\mathbf{b}}^t M\ , \qquad \vec{v} = \vec{\mathbf{b}}^t \mathbf{c} = \vec{\mathbf{a}}^t M^{-1}\mathbf{c}\ .$$

Dot product

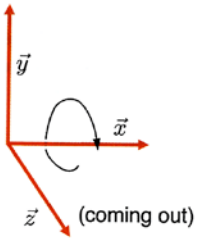
- Input: two vectors $\vec{v} \cdot \vec{w}$
- Output: a real number
- dot product = the squared length $\left\|\vec{v}\right\|^2 := \vec{v} \cdot \vec{v}$
- The angle between the two vectors: $\theta \in [0..\pi]$

$$\cos\theta = \frac{\vec{v} \cdot \vec{w}}{\left\|\vec{v}\right\| \left\|\vec{w}\right\|}.$$

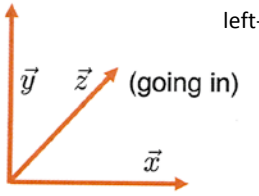
3D orthogonal basis

- Orthogonal vectors: $\vec{v} \cdot \vec{w} = 0$
- A right-handed orthogonal coordinate system. The z axis comes out of the screen (OpenGL).
- A left-handed orthogonal coordinate system. The z axis goes into the screen (DirectX).

right-handed



left-handed

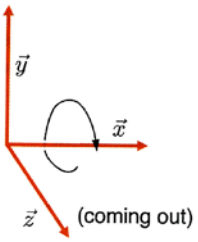


Cross product

- Input: two vectors
- Output: a vector

$$\vec{v} \times \vec{w} := \left\|\vec{v}\right\| \left\|\vec{w}\right\| \sin\theta \vec{n} ,$$

- where $\vec{n}$ is a unit vector that is orthogonal to the plane spanned by $\vec{v}$ and $\vec{w}$
- $[\vec{v}, \vec{w}, \vec{n}]$ forms a right-handed basis



Cross product

- In a right-handed orthogonal basis $\vec{\mathbf{b}}^t$
- We can compute a cross-product as

$$(\vec{\mathbf{b}}^t \mathbf{c}) \times (\vec{\mathbf{b}}^t \mathbf{d}) = \begin{bmatrix} c_2 d_3 - c_3 d_2 \\ c_3 d_1 - c_1 d_3 \\ c_1 d_2 - c_2 d_1 \end{bmatrix}$$

2D Rotation

- Let $\vec{\mathbf{b}}^t$ be a 2D right-handed orthonormal (orthogonal and unit vectors) basis

$$\vec{v} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

- Rotated vector

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta.$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

3D Rotation

- Every rotation fixes an axis of rotation and rotates by some angle about that axis.
- Rotation around the z axis:

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

3D Rotation

- Rotation around the x axis

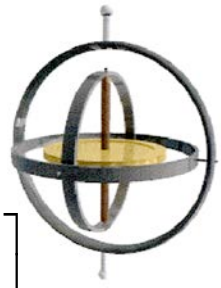
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

- Rotation around the y axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

xyz-Euler angle rotation

- Axis of rotation $\vec{k} = \left[k_x, k_y, k_z\right]^t$



- xyz-Euler angle rotation matrix

$$\begin{bmatrix} k_x^2 v + c & k_x k_y v - k_z s & k_x k_z v + k_y s \\ k_y k_x v + k_z s & k_y^2 v + c & k_y k_z v - k_x s \\ k_z k_x v - k_y s & k_z k_y v + k_x s & k_z^2 v + c \end{bmatrix}$$

where $c := \cos\theta, s := \sin\theta, v := 1 - c.$

Scales

- Scaling operations

$$\begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$