Lecture 1 of 41

Computer Graphics (CG) Basics: Transformation Matrices & Coordinate Systems

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE

Public mirror web site: http://www.kddresearch.org/Courses/CIS636

Instructor home page: http://www.cis.ksu.edu/~bhsu

Readings:

Wikipedia: vectors (**http://bit.ly/eBrl09**), matrices (**http://bit.ly/fwpDwd**)

Sections 2.1 – 2.2, 13.2, 14.1 – 14.4, 17.1, Eberly *2^e* – see **http://bit.ly/ieUq45**

Appendices 1-4, Foley, J. D., VanDam, A., Feiner, S. K., & Hughes, J. F. (1991). *Computer Graphics, Principles and Practice, Second Edition in C.*

McCauley (Senocular.com) tutorial: **http://bit.ly/2yNPD**

Lecture Outline

- CG Basics 1: Basic Precalculus and Linear Algebra for CG
  - Matrices and vectors: definitions, basic operations
  - Vector spaces and affine spaces
  - Translation, Rotation, Scaling *aka* T, R, S transformations
  - Parametric equations (of lines, rays, line segments)
- Importance to Computer Graphics
  - Points as vectors, transformation matrices
  - Homogeneous coordinates
  - TRS in viewing/normalizing transformation
  - Intersections: clipping, ray tracing, *etc.*
- Looking Forward
  - The week ahead: Viewing (Part 1 of 4), Lab 0
  - Lab exercise: C/Linux, basic OpenGL setup (see KSOL)

Where We Are

Lecture	Topic	Primary Source(s)
0	Course Overview	Chapter 1, Eberly 2^e
1	**CG Basics: Transformation Matrices; Lab 0**	**Sections (§) 2.1, 2.2**
2	Viewing 1: Overview, Projections	§ 2.2.3 – 2.2.4, 2.8
3	Viewing 2: Viewing Transformation	§ 2.3 esp. 2.3.4; FVFH slides
4	**Lab 1a: Flash & OpenGL Basics**	**Ch. 2, 16[1], Angel *Primer***
5	Viewing 3: Graphics Pipeline	§ 2.3 esp. 2.3.7; 2.6, 2.7
6	Scan Conversion 1: Lines, Midpoint Algorithm	§ 2.5.1, 3.1; FVFH slides
7	**Viewing 4: Clipping & Culling; Lab 1b**	**§ 2.3.5, 2.4, 3.1.3**
8	Scan Conversion 2: Polygons, Clipping Intro	§ 2.4, 2.5 esp. 2.5.4, 3.1.6
9	Surface Detail 1: Illumination & Shading	§ 2.5, 2.6.1 – 2.6.2, 4.3.2, 20.2
10	**Lab 2a: Direct3D / DirectX Intro**	**§ 2.7, Direct3D handout**
11	Surface Detail 2: Textures; OpenGL Shading	§ 2.6.3, 20.3 – 20.4, *Primer*
12	Surface Detail 3: Mappings; OpenGL Textures	§ 20.5 – 20.13
13	**Surface Detail 4: Pixel/Vertex Shad.; Lab 2b**	**§ 3.1**
14	Surface Detail 5: Direct3D Shading; OGLSL	§ 3.2 – 3.4, Direct3D handout
15	Demos 1: CGA, Fun; Scene Graphs: State	§ 4.1 – 4.3, **CGA handout**
16	**Lab 3a: Shading & Transparency**	**§ 2.6, 20.1, *Primer***
17	**Animation 1: Basics, Keyframes; HW/Exam**	**§ 5.1 – 5.2**
	Exam 1 review; Hour Exam 1 (evening)	**Chapters 1 – 4, 20**
18	**Scene Graphs: Rendering; Lab 3b: Shader**	**§ 4.4 – 4.7**
19	**Demos 2: SFX; Skinning, Morphing**	**§ 5.3 – 5.5, CGA handout**
20	Demos 3: Surfaces; B-reps/Volume Graphics	§ 10.4, 12.7, **Mesh handout**

Lightly-shaded entries denote the due date of a written problem set; heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review; and the green-shaded entry, that of the term project.

Green, **blue** and **red** letters denote **exam review**, **exam**, and **exam solution review** dates.

Online Recorded Lectures for CIS 536/636 (Intro to CG)

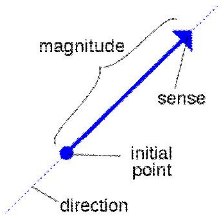
- **Project Topics for CIS 536/636**
- **Computer Graphics Basics (10)**
  - **1. Mathematical Foundations – Week 1 - 2**
  - **2. OpenGL Primer 1 of 3: Basic Primitives and 3-D – Weeks 2-3**
  - **3. Detailed Introduction to Projections and 3-D Viewing – Week 3**
  - **4. Fixed-Function Graphics Pipeline – Weeks 3-4**
  - **5. Rasterizing (Lines, Polygons, Circles, Ellipses) and Clipping – Week 4**
  - **6. Lighting and Shading – Week 5**
  - **7. OpenGL Primer 2 of 3: Boundaries (Meshes), Transformations – Weeks 5-6**
  - **8. Texture Mapping – Week 6**
  - **9. OpenGL Primer 3 of 3: Shading and Texturing, VBOs – Weeks 6-7**
  - **10. Visible Surface Determination – Week 8**
- **Recommended Background Reading for CIS 636**
- **Shared Lectures with CIS 736 (*Computer Graphics*)**
  - **Regular in-class lectures (30) and labs (7)**
  - **Guidelines for paper reviews – Week 6**
  - **Preparing term project presentations, CG demos – Weeks 11-12**

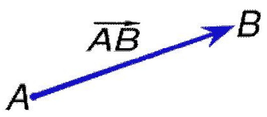
Background Expected

- Both Courses
  - Proficiency in C/C++ or *strong* proficiency in Java and ability to learn
  - Strongly recommended: matrix theory or linear algebra (e.g., Math 551)
  - At least 120 hours for semester (up to 150 depending on term project)
  - Textbook: *3D Game Engine Design*, *Second Edition* (2006), Eberly
  - Angel's *OpenGL: A Primer* recommended
- CIS 536 & 636 *Introduction to Computer Graphics*
  - Fresh background in precalculus: Algebra 1-2, Analytic Geometry
  - Linear algebra basics: matrices, linear bases, vector spaces
  - Watch background lectures
- CIS 736 *Computer Graphics*
  - Recommended: first course in graphics (background lectures as needed)
  - OpenGL experience helps
  - Read up on shaders and shading languages
  - Watch advanced topics lectures; see list before choosing project topic

Matrix and Vector Notation

- Vector: Geometric Object with Length (Magnitude), Direction
- Vector Notation (General Form)
  - Row vector $\mathbf{v} = (v_1, v_2, \dots, v_{n-1}, v_n)$
  - Column vector $$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix}$$

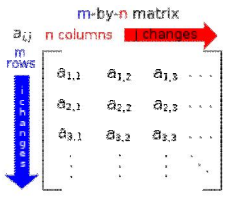




Wikipedia: *Vector*
http://bit.ly/eBrl09

- Coordinates in $\mathbb{R}^3$ (Euclidean Space)
  - Cartesian (see http://bit.ly/f5z1UC) $\mathbf{a} = (a_x, a_y, a_z)$.
  - Cylindrical (see http://bit.ly/gt5v3u) $\mathbf{v} = (r, \angle\theta, h)$
  - Spherical (see http://bit.ly/f4CvMZ) $\mathbf{v} = (\rho, \angle\theta, \angle\phi)$
- Matrix: Rectangular Array of Numbers

$$\mathbf{A} = \begin{bmatrix} 0 & -1 & -2 & -3 \\ 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 \end{bmatrix}$$


m-by-n matrix
$a_{i,j}$
n columns
j changes
m rows
i changes
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$

Wikipedia: *Matrix (mathematics)*
http://bit.ly/fwpDwd

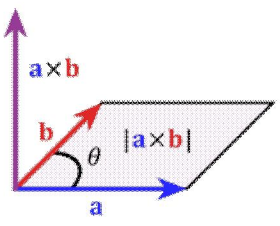
Wikimedia Commons, 2011 – Creative Commons License

Vector Operations: Dot & Cross Product, Arithmetic

- **Dot Product *aka* Inner Product *aka* Scalar Product**

$$\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}a_ib_i=a_1b_1+a_2b_2+\cdots+a_nb_n$$

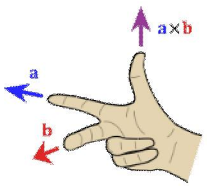
-



$$\mathbf{a}\times\mathbf{b}=\begin{vmatrix}\mathbf{i}&\mathbf{j}&\mathbf{k}\\a_1&a_2&a_3\\b_1&b_2&b_3\end{vmatrix}.$$

$$\mathbf{a}\times\mathbf{b}=\begin{vmatrix}a_2&a_3\\b_2&b_3\end{vmatrix}\mathbf{i}-\begin{vmatrix}a_1&a_3\\b_1&b_3\end{vmatrix}\mathbf{j}+\begin{vmatrix}a_1&a_2\\b_1&b_2\end{vmatrix}\mathbf{k}$$

$$\mathbf{a}\times\mathbf{b}=\mathbf{i}a_2b_3+\mathbf{j}a_3b_1+\mathbf{k}a_1b_2-\mathbf{i}a_3b_2-\mathbf{j}a_1b_3-\mathbf{k}a_2b_1.$$



-

$c\mathbf{v}$

Vector addition $\mathbf{u}+\mathbf{v}$

Matrix Operations [2]: Addition & Multiplication

- **Scalar Multiplication, Transpose**

$$2 \cdot \begin{bmatrix} 1 & 8 & -3 \\ 4 & -2 & 5 \end{bmatrix} = \begin{bmatrix} 2\cdot 1 & 2\cdot 8 & 2\cdot -3 \\ 2\cdot 4 & 2\cdot -2 & 2\cdot 5 \end{bmatrix} = \begin{bmatrix} 2 & 16 & -6 \\ 8 & -4 & 10 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{1} & 2 & 3 \\ 0 & -6 & 7 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \\ 2 & -6 \\ 3 & 7 \end{bmatrix}$$

- **M**

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{1}+0 & 3+0 & 1+5 \\ \mathbf{1}+7 & 0+5 & 0+0 \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{3} & 6 \\ 8 & \mathbf{5} & 0 \end{bmatrix}$$

- **Matrix Multiplication**

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix} \qquad B = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,p} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,p} \end{bmatrix} = \begin{bmatrix} B_1 & B_2 & \cdots & B_p \end{bmatrix}$$

$$A_i = \begin{bmatrix} a_{i,1} & a_{i,2} & \cdots & a_{i,n} \end{bmatrix} \qquad B_i = \begin{bmatrix} b_{1,i} & b_{2,i} & \cdots & b_{n,i} \end{bmatrix}^T.$$

$$AB = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix} \begin{bmatrix} B_1 & B_2 & \dots & B_p \end{bmatrix} = \begin{bmatrix} (A_1 \cdot B_1) & (A_1 \cdot B_2) & \dots & (A_1 \cdot B_p) \\ (A_2 \cdot B_1) & (A_2 \cdot B_2) & \dots & (A_2 \cdot B_p) \\ \vdots & \vdots & \ddots & \vdots \\ (A_m \cdot B_1) & (A_m \cdot B_2) & \dots & (A_m \cdot B_p) \end{bmatrix}.$$

Wikimedia Commons, 2011 – Creative Commons License

Linear Systems of Equations

- Definition: Linear System of Equations (LSE)
  - Collection of linear equations (see http://bit.ly/dNa2MO)
  - Each of form $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b.$
  - System shares same set of variables x_i

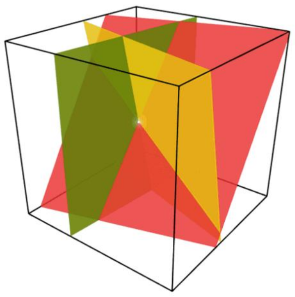
$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad &\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m. \end{aligned}$$

- Example
  - 3 equations in 3 unknov

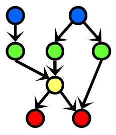
$$\begin{aligned} 3x + 2y - z &= 1 \\ 2x - 2y + 4z &= -2 \\ -x + \tfrac{1}{2}y - z &= 0 \end{aligned}$$

  - Solution

$$\begin{aligned} x &= 1 \\ y &= -2 \\ z &= -2 \end{aligned}$$



Wikimedia Commons, 2011 – Creative Commons License



Vector Spaces and Affine Spaces

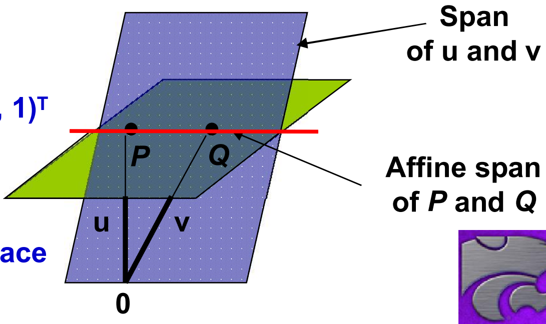
- Vector Space: Set of Points with Addition, Multiplication by Constant
  - Components
    - Set *V* (of vectors **u**, **v**, **w**) over which addition, scalar multiplication defined
    - Vector addition: $\mathbf{v} + \mathbf{w}$
    - Scalar multiplication: $\alpha\mathbf{v}$
  - Properties (necessary and sufficient conditions)
    - Addition: associative, commutative, identity ($\mathbf{0}$ vector such that $\forall\, \mathbf{v}\ .\ \mathbf{0} + \mathbf{v} = \mathbf{v}$), admits inverses ($\forall\, \mathbf{v}\ .\ \exists\mathbf{w}\ .\ \mathbf{v} + \mathbf{w} = \mathbf{0}$)
    - Scalar multiplication: satisfies $\forall\, \alpha, \beta, \mathbf{v}\ .\ (\alpha\beta)\mathbf{v} = \alpha(\beta\mathbf{v})$, $\forall\mathbf{v}\ .\ 1\mathbf{v} = \mathbf{v}$, $\forall\, \alpha, \beta, \mathbf{v}\ .\ (\alpha + \beta)\mathbf{v} = \alpha\mathbf{v} + \beta\mathbf{v}$, $\forall\, \alpha, \beta, \mathbf{v}\ .\ \alpha(\mathbf{v} + \mathbf{w}) = \alpha\mathbf{v} + \alpha\mathbf{w}$
  - Linear combination: $\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n$
- Affine Space: Set of Points with Geometric Operations (No "Origin")
  - Components
    - Set *V* (of points *P*, *Q*, *R*) *and* associated vector space
    - Operators: vector difference, point-vector addition
  - Affine combination (of *P* and *Q* by $t \in \mathbb{R}$): $P + t(Q - P)$
  - *NB*: for any vector space $(V, +, \cdot)$ there exists affine space (points(V), V)

Linear and Planar Equations in Affine Spaces

- Equation of Line in Affine Space
  - Let *P*, *Q* be points in affine space
  - Parametric form (real-valued parameter *t*)
    - ⇨ Set of points of form $(1 - t)P + tQ$
    - ⇨ Forms line passing through *P* and *Q*
  - Example
    - ⇨ Cartesian plane of points (x, y) is an affine space
    - ⇨ Parametric line between (a, b) and (c, d):
      $L = \{((1 - t)a + tc, (1 - t)b + td) \mid t \in \mathbb{R}\}$
- Equation of Plane in Affine Space
  - Let *P*, *Q*, *R* be points in affine space
  - Parametric form (real-valued parameters *s*, *t*)
    - ⇨ Set of points of form $(1 - s)((1 - t)P + tQ) + sR$
    - ⇨ Forms plane containing *P*, *Q*, *R*

Vector Space Spans and Affine Spans

- **Vector Space Span**
  - Definition – set of all linear combinations of a set of vectors
  - Example: vectors in $\mathbb{R}^3$
    - Span of single (nonzero) vector **v**: line through the origin containing **v**
    - Span of pair of (nonzero, noncollinear) vectors: plane through the origin containing both
    - Span of 3 of vectors in general position: all of $\mathbb{R}^3$
- **Affine Span**
  - Definition – set of all affine combinations of a set of points $P_1, P_2, \ldots, P_n$ in an affine space
  - Example: vectors, points in $\mathbb{R}^3$
    - Standard affine plan of points $(x, y, 1)^T$
    - Consider points P, Q
    - Affine span: line containing P, Q
    - Also intersection of span, affine space

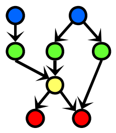


Subspaces

- **Intuitive Idea**
  - $\mathbb{R}^n$: vector or affine space of "equal or lower dimension"
  - Closed under constructive operator for space
- **Linear Subspace**
  - Definition
    - ⇨ Subset S of vector space $(V, +, \cdot)$
    - ⇨ Closed under addition (+) and scalar multiplication ($\cdot$)
  - Examples
    - ⇨ Subspaces of $\mathbb{R}^3$: origin (0, 0, 0), line through the origin, plane containing origin, $\mathbb{R}^3$ itself
    - ⇨ For vector $\mathbf{v}$, $\{\alpha\mathbf{v} \mid \alpha \in \mathbb{R}\}$ is a subspace (why?)
- **Affine Subspace**
  - Definition
    - ⇨ Nonempty subset S of vector space $(V, +, \cdot)$
    - ⇨ Closure S' of S under point subtraction is a linear subspace of V
  - *Important affine subspace of* $\mathbb{R}^4$: $\{(x, y, z, 1)\}$
  - Foundation of homogeneous coordinates, 3-D transformations

Bases

- <u>Spanning Set</u> (of Set *S* of Vectors)
  - Definition: set of vectors for which any vector in Span(*S*) can be expressed as linear combination of vectors in spanning set
  - Intuitive idea: spanning set “covers” Span(*S*)
- <u>Basis</u> (of Set *S* of Vectors)
  - Definition
    - Minimal spanning set of *S*
    - <u>Minimal</u>: any smaller set of vectors has smaller span
  - Alternative definition: linearly independent spanning set
- Exercise
  - Claim: basis of subspace of vector space is always linearly independent
  - Proof: by contradiction (suppose basis is dependent… not minimal)
- Standard Basis for $\mathbb{R}^3$: **i**, **j**, **k**
  - E = {e_1, e_2, e_3}, $e_1 = (1, 0, 0)^T$, $e_2 = (0, 1, 0)^T$, $e_3 = (0, 0, 1)^T$
  - *How to use this as coordinate system?*



Coordinates and Coordinate Systems

- Coordinates Using Bases
  - Coordinates
    - Consider basis **B** = {$\mathbf{v}_1$, $\mathbf{v}_2$, …, $\mathbf{v}_n$} for vector space
    - Any vector **v** in the vector space can be expressed as linear combination of vectors in **B**
    - Definition: coefficients of linear combination are coordinates
  - Example
    - **E** = {$\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$}, $\mathbf{i} \equiv \mathbf{e}_1 = (1, 0, 0)^T$, $\mathbf{j} \equiv \mathbf{e}_2 = (0, 1, 0)^T$, $\mathbf{k} \equiv \mathbf{e}_3 = (0, 0, 1)^T$
    - Coordinates of (a, b, c) with respect to **E**: $(a, b, c)^T$
- Coordinate System
  - Definition: set of independent points in affine space
  - Affine span of coordinate system is entire affine space
- Exercise
  - Derive basis for associated vector space of arbitrary coordinate system
  - (Hint: consider definition of affine span…)

Using the Dot Product: Length/Norm & Distance

- **Length**
  - Definition
    - $\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v} \bullet \boldsymbol{v}}$
    - $\mathbf{v} \bullet \mathbf{v} = \sum_i v_i^2$
  - *aka* Euclidean norm
- **Applications of the Dot Product**
  - Normalization of vectors: division by scalar length || v || converts to unit vector
  - Distances
    - Between points: $\| Q - P \|$
    - From points to planes
  - Generating equations (e.g., point loci): circles, hollow cylinders, etc.
  - Ray / object intersection equations
  - See A.3.5, FVD

Orthonormal Bases

- Orthogonality
  - Given: vectors **u** = $(u_1, u_2, \ldots, u_n)^T$, **v** = $(v_1, v_2, \ldots, v_n)^T$
  - Definition
    - ⇨ **u**, **v** are orthogonal if **u** • **v** = 0
    - ⇨ In $\mathbb{R}^2$, angle between orthogonal vectors is 90°
- Orthonormal Bases
  - Necessary and sufficient conditions
    - ⇨ **B** = $\{b_1, b_2, \ldots, b_n\}$ is basis for given vector space
    - ⇨ Every pair (b_i, b_j) is orthogonal
    - ⇨ Every vector b_i is of unit magnitude ($\| v_i \| = 1$)
  - Convenient property: can just take dot product **v** • b_i to find coefficients in linear combination (coordinates with respect to **B**) for vector **v**

Cumulative Transformation Matrices: Basic T, R, S

- T: Translation (see http://en.wikipedia.org/wiki/Translation_matrix)
  - Given
    - Point to be moved – e.g., vertex of polygon or polyhedron
    - Displacement vector (also represented as point)
  - Return: new, displaced (translated) point of rigid body
- R: Rotation (see http://en.wikipedia.org/wiki/Rotation_matrix)
  - Given
    - Point to be rotated about axis
    - Axis of rotation
    - Degrees to be rotated
  - Return: new, displaced (rotated) point of rigid body
- S: Scaling (see http://en.wikipedia.org/wiki/Scaling_matrix)
  - Given
    - Set of points centered at origin
    - Scaling factor
  - Return: new, displaced (scaled) point
- General: http://en.wikipedia.org/wiki/Transformation_matrix

Translation

- **Rigid Body Transformation**
- **To Move p Distance and Magnitude of Vector v:**

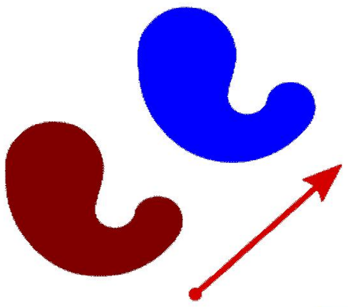
$$T_{\mathbf{v}}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} = \mathbf{p} + \mathbf{v}.$$

- **Invertibility**

$$T_{\mathbf{v}}^{-1} = T_{-\mathbf{v}}.$$

- **Compositionality**

$$T_{\mathbf{u}}T_{\mathbf{v}} = T_{\mathbf{u}+\mathbf{v}}.$$



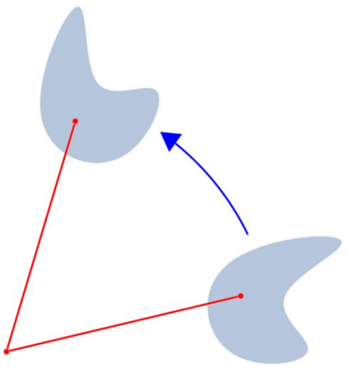
Rotation

- **Rigid Body Transformation**
- **Properties: Inverse ≡ Transpose**

$$Q^TQ = I = QQ^T$$
$$\det Q = +1$$

- **Idea: Define New (Relative) Coordinate System**
- **Example**

$$Q = \begin{bmatrix} 0.6 & -0.8 & 0 \\ 0.8 & 0.6 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

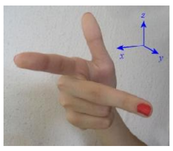


- **Rotations about x, y, and z Axes (using Plain 3-D Coordinates)**

$$Q_{\mathbf{x}}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad Q_{\mathbf{y}}(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \quad Q_{\mathbf{z}}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

Rotation as Change of Basis

- 3 x 3 rotation matrices
- 3 x 3 matrices that "rotate" world (leaving out *w* for simplicity)
- 3 unit vectors originally along *x*, *y*, *z* axes: moved to new positions
- Because of rigid-body rotation, new vectors are still:
  - unit vectors
  - perpendicular to each other
  - compliant with "right hand rule"
- *Any* such matrix transformation = rotation
  - about *some* axis
  - by *some* amount
- Let's call these *x*, *y*, and *z*-axis-aligned unit vectors e_1, e_2, e_3
- Writing out (these are also called **i**, **j**, **k**):



$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Scaling

- Not Rigid Body Transformation
- Idea: Move Points Toward/Away from Origin

$$S_v p = \begin{bmatrix} v_x & 0 & 0 & 0 \\ 0 & v_y & 0 & 0 \\ 0 & 0 & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} v_x p_x \\ v_y p_y \\ v_z p_z \\ 1 \end{bmatrix}$$

- Homogeneous Coordinates Make It Easier

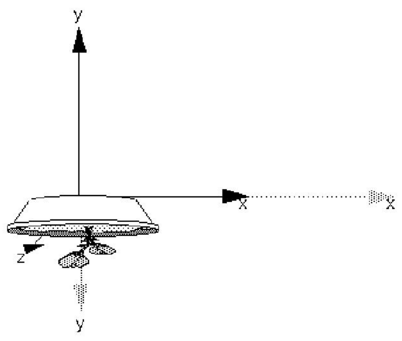
$$S_v p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{s} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \frac{1}{s} \end{bmatrix}$$

- Result

$$\begin{bmatrix} sp_x \\ sp_y \\ sp_z \\ 1 \end{bmatrix}$$

- Ratio Need Not Be Uniform in *x*, *y*, *z*

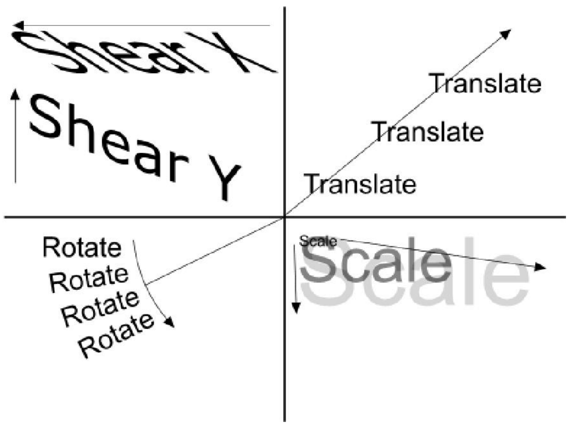
Results of glScalef(2.0, -0.5, 1.0)
© 1993 Neider, Davis, Woo
http://fly.cc.fer.hr/~unreal/theredbook/



Wikimedia Commons, 2008 – Creative Commons License

Other Transformations

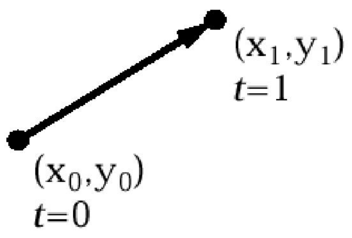
- Shear *aka* Skew (http://bit.ly/hZfx3W): "Tilting", Oblique Projection
- Perspective to Parallel View Volume ("*D*" in Foley *et al.*)
- See also
  - http://en.wikipedia.org/wiki/Transformation_matrix
  - http://www.senocular.com/flash/tutorials/transformmatrix/



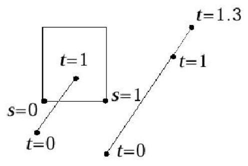
http://www.bobpowell.net/transformations.htm

Parametric Equation of a Line Segment

- Parametric form for line segment
  - $X = x_0 + t(x_1 - x_0) \qquad 0 \le t \le 1$
  - $Y = y_0 + t(y_1 - y_0)$
  - $P(t) = P_0 + t(P_1 - P_0)$

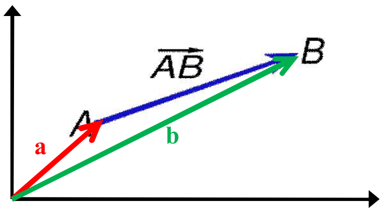


- Line in general: $t \in [-\infty, \infty]$
- Later: used for clipping (other intersection calculations)

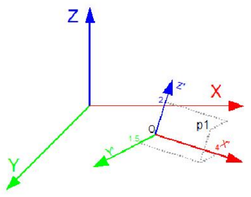


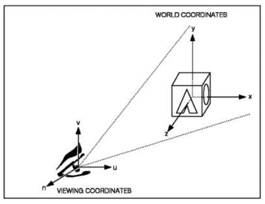
Importance to CG [1]: Vectors and Matrices

- Points as Vectors (w.r.t. Origin)



- Local Coordinate Systems (Spaces)





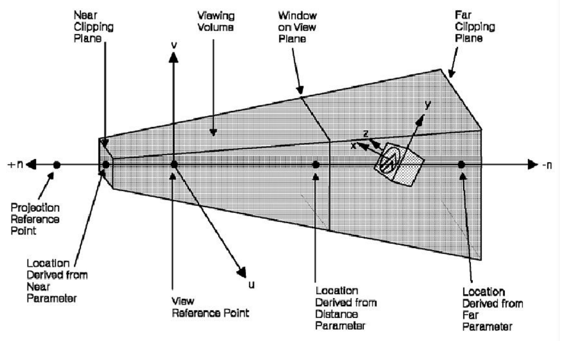
© 2009 Koen Samyn
http://knol.google.com/k/matrices-for-3d-applications-view-transformation

© 2007 IBM
http://bit.ly/cS4h7g

- Modelview transformation (MVT): model coordinates to world coordinates
- Viewing transformation: world coordinates to camera coordinates
- Several more to be covered in this course

Importance to CG [2]: Homogeneous Coordinates

- **Problem: Need to Support Non-Linear Transformations**
  - **Affine but not linear: *e.g.*, translation**
  - **Non-affine projections: *e.g.*, perspective**

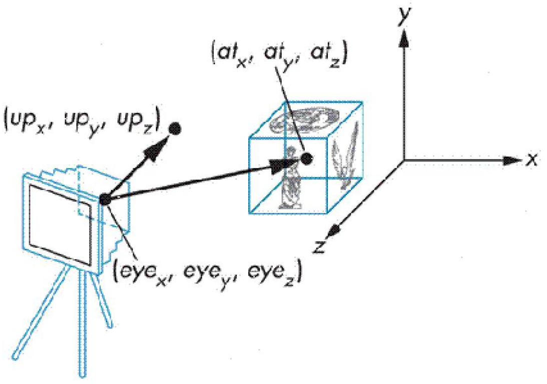

Near Clipping Plane
Viewing Volume
Window on View Plane
Far Clipping Plane
v
y
z
x
+n
-n
Projection Reference Point
Location Derived from Near Parameter
View Reference Point
u
Location Derived from Distance Parameter
Location Derived from Far Parameter

The GraPHIGS Programming Interface: Understanding Concepts
© 2007 IBM
http://bit.ly/cS4h7g

- **Solution: Use 4th Coordinate *w***
  - **Coordinates look like: $(x, y, z, w)^T$ with w kept normalized to 1**
  - **Homogeneous coordinates (Wikipedia: http://bit.ly/fG7RSk)**
  - **Specific case: barycentric (defined w.r.t. simplex, *e.g.,* polygon) http://en.wikipedia.org/wiki/Barycentric_coordinates_(mathematics)**

Importance to CG [3]: T, R, S in Viewing Transformation

- Want to
  - Specify arbitrary (user-defined) camera view (camera space *aka* CS)
  - Take picture of standard world space (WS), from eye point towards at point



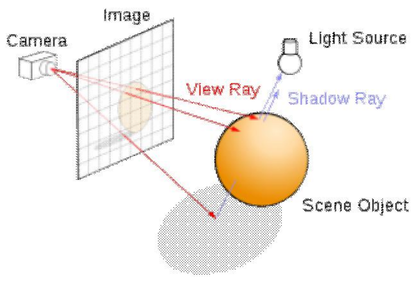
```
GLvoid gluLookAt( GLdouble eyeX, GLdouble eyeY, GLdouble eyeZ,
                                                    GLdouble centerX, GLdouble centerY, GLdouble centerZ,
                                                    GLdouble upX, GLdouble upY, GLdouble upZ)
```

© 2009 Roberto Toldo
http://bit.ly/hvAZAe

- Need to: Map CS to WS (Normalizing Transformation)

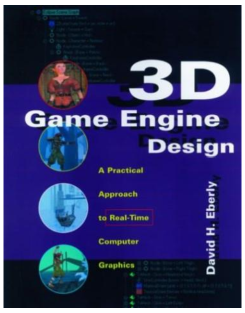
Importance to CG [4]: Intersections, Clipping

- **Problem: Need to Find Intersection between Objects**
  - Clipping: line segments – edge of polygon (model) with clip edge
  - Ray tracing: ray – from eye, through "screen" pixel, into scene



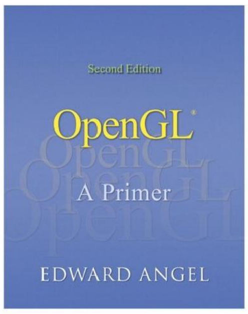
  - Many other intersections in computer graphics!
- **Solution: Represent Objects using Parametric Equations**
  - Moving object or object being traced (*e.g.*, ray): $\mathbf{P}(t)$
  - Find point where $\mathbf{P}(t) = \mathbf{Q}$ (boundary of second object)
  - May have multiple solutions (as polynomials may have > 1 zero)
  - Usually want closest one

Textbook and Recommended Books

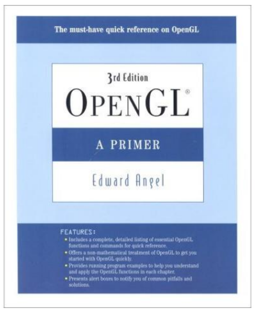


1st edition (outdated)

2nd edition



2nd edition (OK to use)



3rd edition

Required Textbook

Eberly, D. H. (2006). *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics, second edition*. San Francisco, CA: Morgan Kauffman.

Recommended References

Angel, E. O. (2007). *OpenGL: A Primer, third edition*. Reading, MA: Addison-Wesley. [2nd edition on reserve]

Shreiner, D., Woo, M., Neider, J., & Davis, T. (2009). *OpenGL® Programming Guide: The Official Guide to Learning OpenGL®, Versions 3.0 and 3.1, seventh edition*.

["The Red Book": use 7th ed. or later]

Lab 0

- Warm-Up Lab
  - Account set-up
  - Linux environment
  - Simple OpenGL exercise
- Basic Account Set-Up
  - See http://support.cis.ksu.edu to understand KSU Department of CIS setup
  - Make sure your CIS department account is set up
  - If not, use SelfServ: https://selfserv.cis.ksu.edu/selfserv/requestAccount
- Linux Environment
  - Make sure your CIS department account is set up
  - Learn how to navigate, set your shell (see KSOL, http://unixhelp.ed.ac.uk)
  - Lab 1 and first homeworks will ask you to render to local XWindows server
- Simple OpenGL exercise
  - Watch OpenGL Primer Part 1 as needed
  - Follow intro tutorials on “NeHe” (http://nehe.gamedev.net) as instructed
  - Turn in: source code, screenshot as instructed in Lab 0 handout

Summary

- Cumulative Transformation Matrices (CTM): T, R, S
  - Translation
  - Rotation
  - Scaling
  - Setup for Shear/Skew, Perspective to Parallel – see Eberly, Foley *et al.*
- "Matrix Stack" in OpenGL: Premultiplication of Matrices
- Coming Up
  - Parametric equations in clipping
  - Intersection testing: ray-cube, ray-sphere, implicit equations (ray tracing)
- Homogeneous Coordinates: What Is That 4th Coordinate?
  - http://en.wikipedia.org/wiki/Homogeneous_coordinates
  - Crucial for ease of normalizing T, R, S transformations in graphics
  - See: Slide 14 of this lecture
  - Note: Slides 20 & 23 (T, S) versus 21 (R)
  - Read about them in Eberly *2e*, Angel *3e*
  - Special case: barycentric coordinates

Terminology

- Cumulative Transformation Matrices (CTM): Translation, Rotation, Scaling
- Some Basic Analytic Geometry and Linear Algebra for CG
  - Vector space (VS) – set of vectors: addition, scalar multiplication; VS axioms
  - Affine space (AS) – set of points with associated VS: vector difference, point-vector addition; AS axioms
  - Linear subspace – nonempty subset *S* of VS (*V*, +, ·) closed under + and ·
  - Affine subspace – nonempty subset *S* of VS (*V*, +, ·) such that closure *S*' of *S* under point subtraction is a linear subspace of *V*
  - Dot product – scalar-valued inner product $\langle u, v \rangle \equiv u \bullet v \equiv u_1v_1 + u_2v_2 + \ldots + u_nv_n$
  - Orthogonality – property of vectors u, v that $u \bullet v = 0$
  - Orthonormality – basis containing pairwise-orthogonal unit vectors
  - Length (Euclidean norm) – $\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v} \bullet \boldsymbol{v}}$
  - Rigid body transformation – one that preserves distance between points
  - Homogeneous coordinates (esp. barycentric coordinates) – allow affine, projective transformations; "4-D" space for 3-D CG