CIS 636
Introduction to Computer Graphics

CG Basics 1 of 8:
Mathematical Foundations

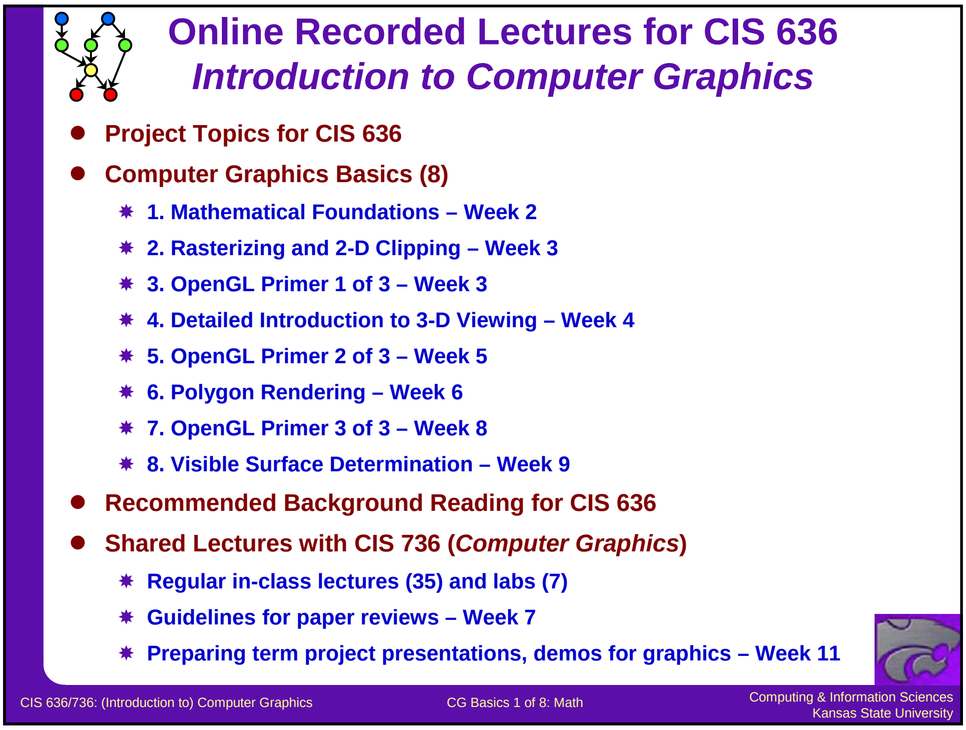
William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course pages: **http://snipurl.com/1y5gc**
Course web site: **http://www.kddresearch.org/Courses/CIS636**
Instructor home page: **http://www.cis.ksu.edu/~bhsu**

Readings:
Sections 2.1 – 2.2, 13.2, 14.1 – 14.4, 17.1, Eberly 2^e – see **http://snurl.com/1ye72**
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 tutorial: **http://www.senocular.com/flash/tutorials/transformmatrix/**

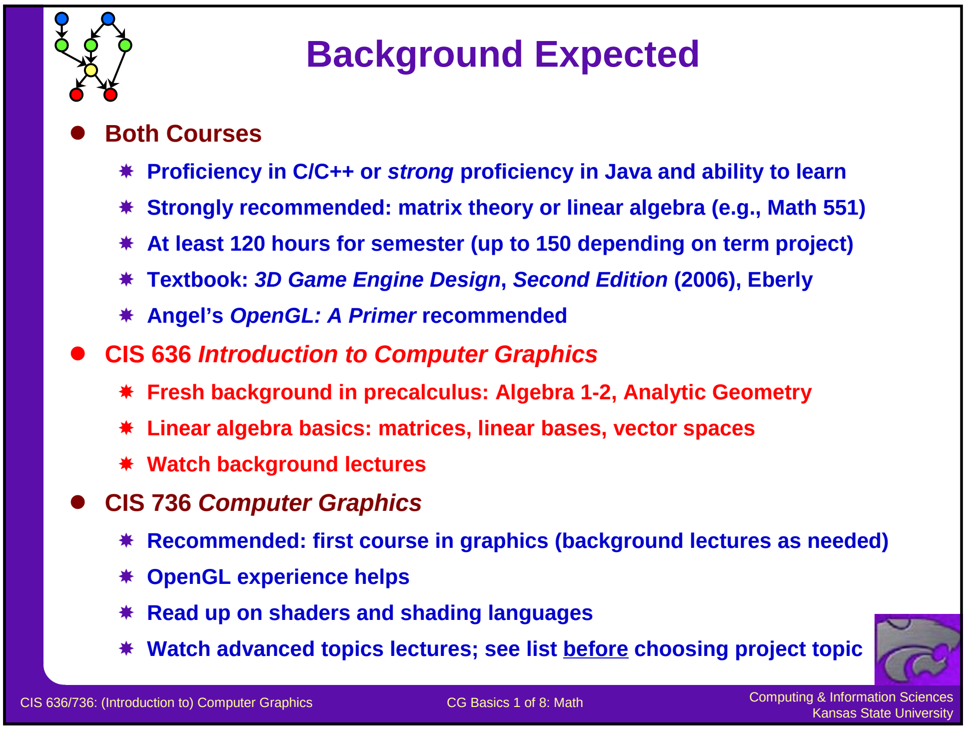
Lecture Outline

- Quick Review: Basic Precalculus and Linear Algebra for CG
- Precalculus: Analytic Geometry and Trigonometry
  - Dot products and distance measures (norms, equations)
  - Review of some basic trigonometry concepts
- Vector Spaces and Affine Spaces
  - Subspaces
  - Linear systems, linear independence, bases, orthonormality
  - Equations for objects in affine spaces
- Cumulative Transformation Matrices (CTM) *aka* “Composite”, “Current”
  - Translation
  - Rotation
  - Scale
- Parametric Equations
- Implicit Functions



Online Recorded Lectures for CIS 636
Introduction to Computer Graphics

- **Project Topics for CIS 636**
- **Computer Graphics Basics (8)**
    - **1. Mathematical Foundations – Week 2**
    - **2. Rasterizing and 2-D Clipping – Week 3**
    - **3. OpenGL Primer 1 of 3 – Week 3**
    - **4. Detailed Introduction to 3-D Viewing – Week 4**
    - **5. OpenGL Primer 2 of 3 – Week 5**
    - **6. Polygon Rendering – Week 6**
    - **7. OpenGL Primer 3 of 3 – Week 8**
    - **8. Visible Surface Determination – Week 9**
- **Recommended Background Reading for CIS 636**
- **Shared Lectures with CIS 736 (*Computer Graphics*)**
    - **Regular in-class lectures (35) and labs (7)**
    - **Guidelines for paper reviews – Week 7**
    - **Preparing term project presentations, demos for graphics – Week 11**



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 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**

Math Review for CIS 636

- Overview: First Month (Weeks 2-5 of Course)
  - Review of mathematical foundations of CG: analytic geometry, linear algebra
  - Line and polygon rendering
  - Matrix transformations
  - Graphical interfaces
- Line and Polygon Rendering (Week 3)
  - Basic line drawing and 2-D clipping
  - Bresenham's algorithm
  - Follow-up: 3-D clipping, z-buffering (painter's algorithm)
- Matrix Transformations (Week 4)
  - Application of linear transformations to rendering
  - Basic operations: translation, rotation, scaling, shearing
  - Follow-up: review of standard graphics libraries (e.g., *OpenGL*)
- Graphical Interfaces
  - Brief overview
  - Survey of windowing environments (MFC, Java AWT)

Quick Review: Basic Linear Algebra for CG

- Reference: Appendix A.1 – A.4, Foley *et al*
- A.1 Vector Spaces and Affine Spaces
  - Equations of lines, planes
  - Vector subspaces and affine subspaces
- A.2 Standard Constructions in Vector Spaces
  - Linear independence and spans
  - Coordinate systems and bases
- A.3 Dot Products and Distances
  - Dot product in $\mathbf{R}^n$
  - Norms in $\mathbf{R}^n$
- A.4 Matrices
  - Binary matrix operations: basic arithmetic
  - Unary matrix operations: transpose and inverse
- Application: Transformations and Change of Coordinate Systems

Affine transformations
© 2005 Trevor McCauley
(Senocular)

Cumulative Transformation Matrices: Basic T, R, S Transformations

- **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:**

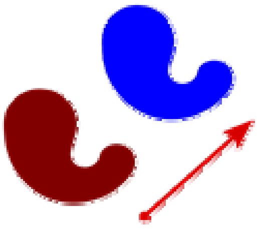
$$T_{\mathbf{v}}\mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} = \mathbf{p} + \mathbf{v}.$$

- **Invertibility**

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

- **Compositionality**

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

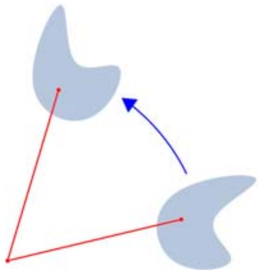


Wikimedia Commons, 2008 – Creative Commons License

Rotation

- Rigid Body Transformation
- Properties: Inverse ≡ Transpose

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



- 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)

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

Wikimedia Commons, 2008 – Creative Commons License

Scaling

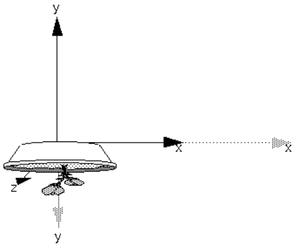
- Not Rigid Body Transformation
- Idea: Move Points Toward/Away from Origin

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

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

- 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

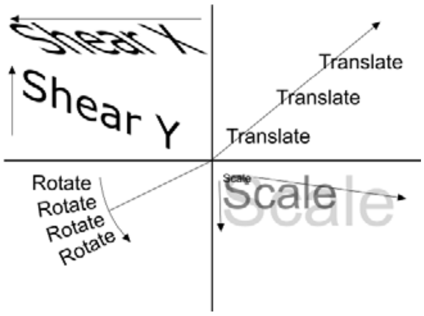
$$\begin{bmatrix} sp_x \\ sp_y \\ sp_z \\ 1 \end{bmatrix}$$

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

Wikimedia Commons, 2008 – Creative Commons License

Other Transformations

- **Shear: Used with Oblique Projections**
- **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

Vector Spaces and Affine Spaces

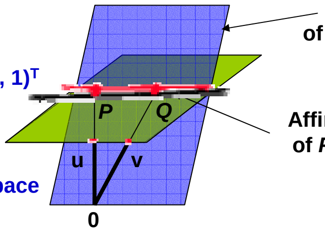
- **Vector Space: Set of Points with Addition, Multiplication by Constant**
  - **Components**
    - **Set *V* (of vectors u, v, w) over which addition, scalar multiplication defined**
    - **Vector addition: v + w**
    - **Scalar multiplication: αv**
  - **Properties (necessary and sufficient conditions)**
    - **Addition: associative, commutative, identity (0 vector such that $\forall$ v . 0 + v = v), admits inverses ($\forall$ v . $\exists$w . v + w = 0)**
    - **Scalar multiplication: satisfies $\forall$ α, β, v . ($\alpha\beta$)v = α(βv), $\forall$v . 1v = v, $\forall$ α, β, v . (α + β)v = αv + βv, $\forall$ α, β, v . α(v + w) = αv + αw**
  - **Linear combination: $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$**
- **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*, +, ·) 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
  - <u>Parametric form</u> (real-valued <u>parameter</u> *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
  - <u>Parametric form</u> (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

- **<u>Vector Space Span</u>**
  - Definition – set of all linear combinations of a set of vectors
  - Example: vectors in $\mathbb{R}^3$ $\mathbb{R}^3$
    - $\mathbb{R}$ Span of single (nonzero) vector **v**: line through the origin containing **v**
    - $\mathbb{R}^2$ Span of pair of (nonzero, noncollinear) vectors: plane through the origin containing both
    - Span of 3 of vectors in <u>general position</u>: all of $\mathbb{R}^3$
- **<u>Affine Span</u>**
  - 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 $\mathbb{R}^3$
    - [illegible] Standard affine plan of points $(x, y, 1)^T$
    - Consider points *P*, *Q*
    - <u>Affine span:</u> line containing *P*, *Q*
    - Also intersection of span, affine space



Span of u and v

Affine span of *P* and *Q*

Independence

- **Linear Independence**
  - Definition: (linearly) dependent vectors
    - Set of vectors $\{v_1, v_2, \ldots, v_n\}$ such that one lies in the span of the rest
    - $\exists\, v_i \in \{v_1, v_2, \ldots, v_n\} \,.\, v_i \in \text{Span}(\{v_1, v_2, \ldots, v_n\} \sim \{v_i\})$
  - (Linearly) independent: $\{v_1, v_2, \ldots, v_n\}$ not dependent

iff $\exists\, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} \alpha_j \vec{v}_j = \vec{v}_i$$

- **Affine Independence**
  - Definition: (affinely) dependent points
    - Set of points $\{v_1, v_2, \ldots, v_n\}$ such that one lies in the (affine) span of the rest
    - $\exists\, P_i \in \{P_1, P_2, \ldots, P_n\} \,.\, P_i \in \text{Span}(\{P_1, P_2, \ldots, P_n\} \sim \{P_i\})$
  - (Affinely) independent: $\{P_1, P_2, \ldots, P_n\}$ not dependent
- **Consequences of Linear Independence**
  - Equivalent condition: $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = 0 \Leftrightarrow \alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$
  - Dimension of span is equal to the number of vectors

basis

Subspaces

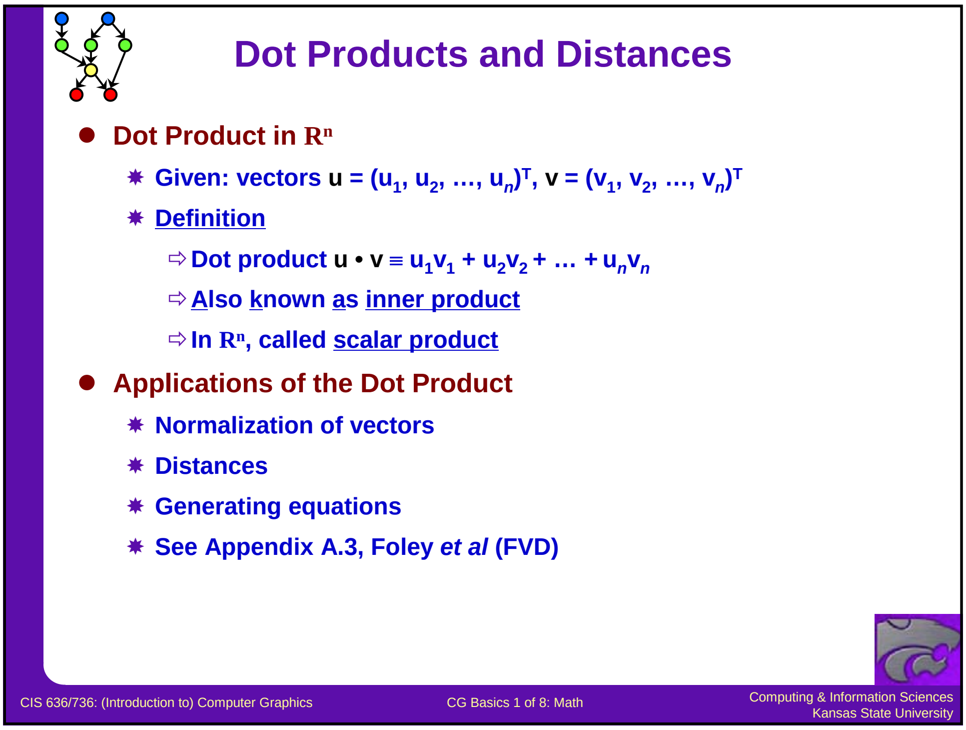
- **Intuitive Idea**
  - $\mathbb{R}^n$: vector or affine space of "equal or lower dimension"
  - Closed under constructive operator for space
- **Linear Subspace**
  - Definition
    - Subset S of vector space $(V, +, \cdot)$
    - Closed under addition (+) and scalar multiplication (·)
  - Examples
    - Subspaces of $\mathbb{R}^3$: origin (0, 0, 0), line through the origin, plane containing origin, $\mathbb{R}^3$ itself
    - For vector v, $\{\alpha v \mid \alpha \in \mathbb{R}\}$ 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

- **Spanning Set** (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*)
- **Basis** (of Set *S* of Vectors)
  - Definition
    - Minimal spanning set of *S*
    - Minimal: 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 $\mathbf{R}^3$
  - 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?*

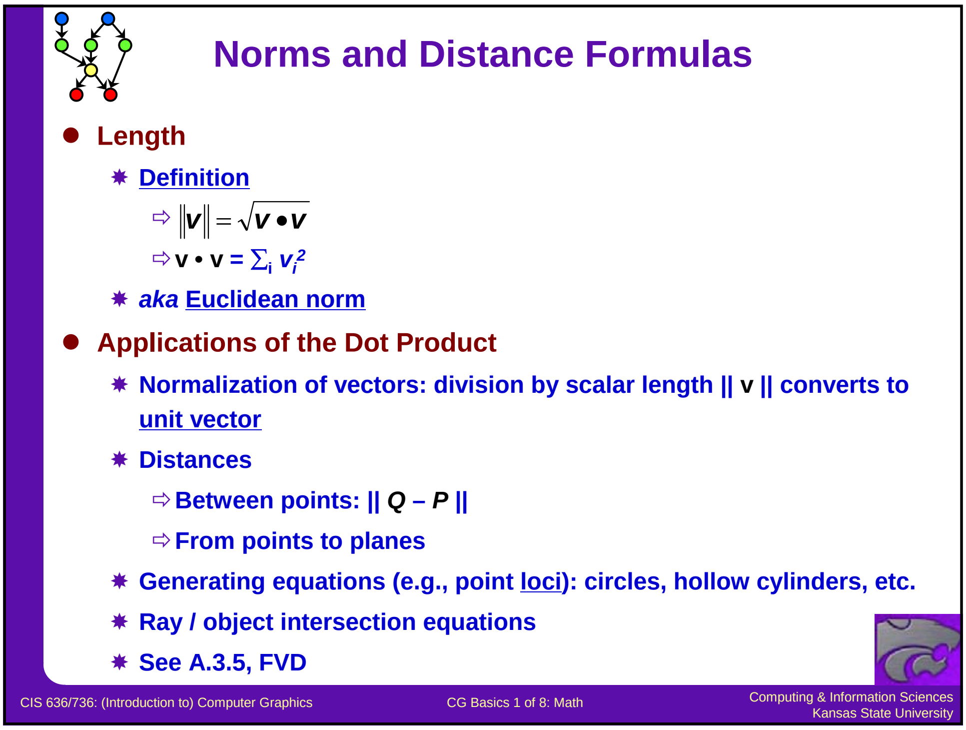
Coordinates and Coordinate Systems

- Coordinates Using Bases
  - Coordinates
    - Consider basis B = $\{v_1, v_2, \ldots, v_n\}$ 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 = $\{e_1, e_2, e_3\}$, $e_1 = (1, 0, 0)^T$, $e_2 = (0, 1, 0)^T$, $e_3 = (0, 0, 1)^T$
    - 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…)



Dot Products and Distances

- Dot Product in $\mathbb{R}^n$
  - Given: vectors $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$, $\mathbf{v} = (v_1, v_2, \ldots, v_n)^T$
  - Definition
    - ⇨ Dot product $\mathbf{u} \bullet \mathbf{v} \equiv u_1v_1 + u_2v_2 + \ldots + u_nv_n$
    - ⇨ Also known as inner product
    - ⇨ In $\mathbb{R}^n$, called scalar product
- Applications of the Dot Product
  - Normalization of vectors
  - Distances
  - Generating equations
  - See Appendix A.3, Foley *et al* (FVD)



Norms and Distance Formulas

- Length
  - Definition
    - ⇨ $\|\mathbf{v}\| = \sqrt{\mathbf{v} \bullet \mathbf{v}}$
    - ⇨ $\mathbf{v} \bullet \mathbf{v} = \sum_i v_i^2$
  - *aka* Euclidean norm
- Applications of the Dot Product
  - Normalization of vectors: division by scalar length $\| \mathbf{v} \|$ 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

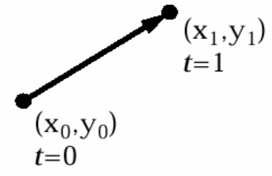
Orthonormal Bases

- **Orthogonality**
  - Given: vectors $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$, $\mathbf{v} = (v_1, v_2, \ldots, v_n)^T$
  - Definition
    - u, v are orthogonal if $\mathbf{u} \bullet \mathbf{v} = 0$
    - In $\mathbb{R}^2$, angle between orthogonal vectors is 90º
- **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 $\mathbf{v} \bullet b_i$ to find coefficients in linear combination (coordinates with respect to B) for vector v

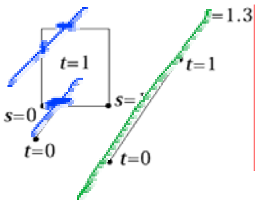
---

Parametric Line Formulation [1]: Basic Form

- Parametric form for line segment
  - $X = x_0 + t(x_1 - x_0)$ $\quad 0 \leqslant t \leqslant 1$
  - $Y = y_0 + t(y_1 - y_0)$
  - $P(t) = \underline{P_0} + t(\underline{P_1} - P_0)$



- "true," i.e., interior intersection, if *sedge* and *tline* in [0,1]



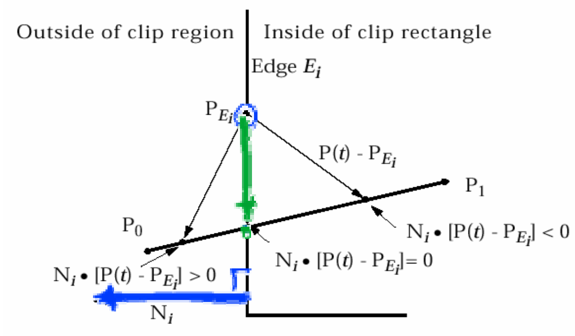
© 2003 – 2007 A. van Dam, Brown University

Parametric Line Formulation [2]
Clipping

- **Use parametric line formulation**

  $$\boxed{P(t) = P_0 + (P_1 - P_0)t}$$

- **Find the four *t*s for the four clip edges, then decide which form true intersections and calculate (*x*, *y*) for those only (≤ 2)**



- **For any point P_{E_i} on edge E_i**

Parametric Line Formulation [3]:
Clipping Formulas

Now we can solve for the value of t at the

Intersection of P0 P1 with the edge E_i:

$$N_i \bullet [P(t) - P_{E_i}] = 0$$

First, substitute for P(t):

$$N_i \bullet [P_0 + (P_1 - P_0)t - P_{E_i}] = 0$$

Next, group terms and distribute the dot product:

$$N_i \bullet [P_0 - P_{E_i}] + N_i \bullet [P_1 - P_0]t = 0$$

Let *D* be the vector from P_0 to P_1 = $\boxed{(P_1 - P_0)}$ and solve for *t*:

$$t = \frac{N_i \bullet [P_0 - P_{E_i}]}{-N_i \bullet \boxed{D}}$$

Independently discovered by

Cyrus & Beck and

Liang & Barsky

Note that this gives a valid value of *t* only if the denominator of the expression is nonzero. For this to be true, it must be the case that

Degeneracies

- ✓ **$N_i \neq 0$ (that is, the normal should not be 0; this could occur only as a mistake)** horiz. line
- ✓ **$D \neq 0$ (that is, $P_1 \neq P_0$)** point (collocated endpts)
- ✓ **$N_i \bullet D \neq 0$ (edge E_i and line *D* are not parallel; if they are, no intersection).** ‖ lines

The algorithm checks these conditions.

Rotation as Change of Basis

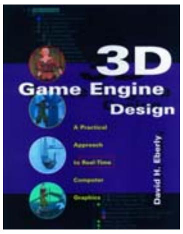
- **3 x 3 rotation matrices**
- **We learned about 3 x 3 matrices that "rotate" the world (we're leaving out the homogeneous coordinate for simplicity)**
- **When they do, the three unit vectors that used to point along the *x*, *y*, and *z* axes are moved to new positions**
- **Because it is a rigid-body rotation**
  - **the new vectors are still unit vectors**
  - **the new vectors are still perpendicular to each other**
  - **the new vectors still satisfy the "right hand rule"**
- ***Any* matrix transformation that has these three properties is a rotation about *some* axis by *some* amount!**
- **Let's call three *x*-axis, *y*-axis, and *z*-axis-aligned unit vectors *e1*, *e2*, *e3***
- **Writing out:**

$$e_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}$$

$$e_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}$$

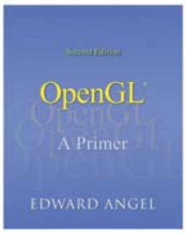
$$e_3 = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

Textbook and Recommended Books

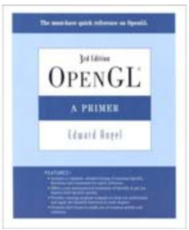


1st edition (outdated)

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. (2007). *OpenGL® Programming Guide: The Official Guide to Learning OpenGL®, Version 2.1, sixth edition.*

["The Red Book": use 5th ed. or later]

Summary

- Cumulative Transformation Matrices (CTM): T, R, S
  - Translation } rigid body
  - Rotation
  - Scaling ← non - RB
  - Setup for Shear, Perspective to Parallel – see Eberly, Foley *et al.* ← non - affine
- "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 16 of this lecture
  - Note: Slides 8 & 10 (T, S) versus 9 (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 admitting addition, scalar multiplication and observing VS axioms
  - Affine space (AS) – set of points with associated vector space admitting vector difference, point-vector addition and observing 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*
  - Span – set of all linear combinations of set of vectors
  - Linear independence – property of set of vectors that none lies in span of others
  - Basis – minimal spanning set of set of vectors
  - Dot product – scalar-valued inner product <u, v> ≡ u • v ≡ $u_1v_1 + u_2v_2 + \ldots + u_nv_n$
  - Orthogonality – property of vectors u, v that u • v = 0
  - Orthonormality – basis containing pairwise-orthogonal unit vectors
  - Length (Euclidean norm) – $\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v} \bullet \boldsymbol{v}}$