Fundamentals of Grassmann Algebra

Eric Lengyel, PhD
Terathon Software
Math used in 3D programming

- Dot / cross products, scalar triple product
- Planes as 4D vectors
- Homogeneous coordinates
- Plücker coordinates for 3D lines
- Transforming normal vectors and planes with the inverse transpose of a matrix
Math used in 3D programming

- These concepts often used without a complete understanding of the big picture
  - Can be used in a way that is not natural
  - Different pieces used separately without knowledge of the connection among them
There is a bigger picture

- All of these arise as part of a single mathematical system
  - Understanding the big picture provides deep insights into seemingly unusual properties
  - Knowledge of the relationships among these concepts makes better 3D programmers
History

- Hamilton, 1843
  - Discovered quaternion product
  - Applied to 3D rotations
  - Not part of Grassmann algebra
History

- Grassmann, 1844
  - Formulated progressive and regressive products
  - Understood geometric meaning
  - Published “Algebra of Extension”
History

- Clifford, 1878
  - Unified Hamilton’s and Grassmann’s work
  - Basis for modern geometric algebra and various algebras used in physics
History

- Quaternion Algebra
- Grassmann Algebra
- Geometric Algebra
- Spacetime Algebra

Separate Components
Special Cases
Outline

- Grassmann algebra in 3-4 dimensions
  - Wedge product, bivectors, trivectors...
  - Transformations
  - Homogeneous model
  - Geometric computation
  - Programming considerations
The wedge product

- Also known as:
  - The progressive product
  - The exterior product

- Gets name from symbol:
  \[ a \wedge b \]

- Read “a wedge b”
The wedge product

- Operates on scalars, vectors, and more
  - Ordinary multiplication for scalars $s$ and $t$:
    \[
    s \wedge t = t \wedge s = st
    \]
    \[
    s \wedge \mathbf{v} = \mathbf{v} \wedge s = s\mathbf{v}
    \]
- The square of a vector $\mathbf{v}$ is always zero:
  \[
  \mathbf{v} \wedge \mathbf{v} = 0
  \]
Wedge product anticommutativity

- Zero square implies vectors anticommute

\[(a + b) \wedge (a + b) = 0\]
\[a \wedge a + a \wedge b + b \wedge a + b \wedge b = 0\]
\[a \wedge b + b \wedge a = 0\]
\[a \wedge b = -b \wedge a\]
Bivectors

- Wedge product between two vectors produces a “bivector”
  - A new mathematical entity
  - Distinct from a scalar or vector
  - Represents an oriented 2D area
    - Whereas a vector represents an oriented 1D direction
    - Scalars are zero-dimensional values
Bivectors

- Bivector is two directions and magnitude
Bivectors

- Order of multiplication matters

\[ \mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a} \]
Bivectors in 3D

- Start with 3 orthonormal basis vectors:

\[ \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \]

- Then a 3D vector \( \mathbf{a} \) can be expressed as

\[ a_1 \mathbf{e}_1 + a_2 \mathbf{e}_2 + a_3 \mathbf{e}_3 \]
Bivectors in 3D

\[ \mathbf{a} \wedge \mathbf{b} = (a_1 \mathbf{e}_1 + a_2 \mathbf{e}_2 + a_3 \mathbf{e}_3) \wedge (b_1 \mathbf{e}_1 + b_2 \mathbf{e}_2 + b_3 \mathbf{e}_3) \]

\[ \mathbf{a} \wedge \mathbf{b} = a_1 b_2 (\mathbf{e}_1 \wedge \mathbf{e}_2) + a_1 b_3 (\mathbf{e}_1 \wedge \mathbf{e}_3) + a_2 b_1 (\mathbf{e}_2 \wedge \mathbf{e}_1) \]
\[ + a_2 b_3 (\mathbf{e}_2 \wedge \mathbf{e}_3) + a_3 b_1 (\mathbf{e}_3 \wedge \mathbf{e}_1) + a_3 b_2 (\mathbf{e}_3 \wedge \mathbf{e}_2) \]

\[ \mathbf{a} \wedge \mathbf{b} = (a_2 b_3 - a_3 b_2)(\mathbf{e}_2 \wedge \mathbf{e}_3) + (a_3 b_1 - a_1 b_3)(\mathbf{e}_3 \wedge \mathbf{e}_1) \]
\[ + (a_1 b_2 - a_2 b_1)(\mathbf{e}_1 \wedge \mathbf{e}_2) \]
Bivectors in 3D

- The result of the wedge product has three components on the basis

\[ \mathbf{e}_2 \wedge \mathbf{e}_3, \quad \mathbf{e}_3 \wedge \mathbf{e}_1, \quad \mathbf{e}_1 \wedge \mathbf{e}_2 \]

- Written in order of which basis vector is missing from the basis bivector
Bivectors in 3D

- Do the components look familiar?

\[ a \wedge b = (a_2 b_3 - a_3 b_2) (e_2 \wedge e_3) + (a_3 b_1 - a_1 b_3) (e_3 \wedge e_1) + (a_1 b_2 - a_2 b_1) (e_1 \wedge e_2) \]

- These are identical to the components produced by the cross product \( a \times b \).
Shorthand notation

\[ e_{12} = e_1 \land e_2 \]

\[ e_{23} = e_2 \land e_3 \]

\[ e_{31} = e_3 \land e_1 \]

\[ e_{123} = e_1 \land e_2 \land e_3 \]
Bivectors in 3D

\[ \mathbf{a} \wedge \mathbf{b} = (a_2 b_3 - a_3 b_2) \mathbf{e}_{23} + (a_3 b_1 - a_1 b_3) \mathbf{e}_{31} + (a_1 b_2 - a_2 b_1) \mathbf{e}_{12} \]
Comparison with cross product

- The cross product is not associative:
  \[(a \times b) \times c \neq a \times (b \times c)\]
- The cross product is only defined in 3D
- The wedge product is associative, and it’s defined in all dimensions
Trivectors

- Wedge product among three vectors produces a “trivector”
  - Another new mathematical entity
  - Distinct from scalars, vectors, and bivectors
  - Represents a 3D oriented volume
Trivectors

\[ \mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} \]
Trivectors in 3D

- A 3D trivector has one component:

\[ \mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} = (a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_1b_3c_2 - a_2b_1c_3 - a_3b_2c_1) \cdot (\mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3) \]

- The magnitude is \( \det([\mathbf{a} \quad \mathbf{b} \quad \mathbf{c}]) \)
Trivectors in 3D

- 3D trivector also called *pseudoscalar* or *antiscalar*
  - Only one component, so looks like a scalar
  - Flips sign under reflection
Scalar Triple Product

- The product
  \[
  \mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c}
  \]
  produces the same magnitude as
  \[
  (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}
  \]
  but also extends to higher dimensions
Grading

- The *grade* of an entity is the number of vectors wedged together to make it
  - Scalars have grade 0
  - Vectors have grade 1
  - Bivectors have grade 2
  - Trivectors have grade 3
  - Etc.
3D multivector algebra

- 1 scalar element
- 3 vector elements
- 3 bivector elements
- 1 trivector element
- No higher-grade elements
- Total of 8 multivector basis elements
Multivectors in general dimension

- In $n$ dimensions, the number of basis $k$-vector elements is $\binom{n}{k}$
- This produces a nice symmetry
- Total number of basis elements always $2^n$
# Multivectors in general dimension

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Graded elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1</td>
</tr>
<tr>
<td>2</td>
<td>1 2 1</td>
</tr>
<tr>
<td>3</td>
<td>1 3 3 1</td>
</tr>
<tr>
<td>4</td>
<td>1 4 6 4 1</td>
</tr>
<tr>
<td>5</td>
<td>1 5 10 10 5 1</td>
</tr>
</tbody>
</table>
Four dimensions

- Four basis vectors $e_1, e_2, e_3, e_4$
- Number of basis bivectors is $\binom{4}{2} = 6$
- There are 4 basis trivectors
Vector / bivector confusion

- In 3D, vectors have three components
- In 3D, bivectors have three components
- Thus, vectors and bivectors *look like* the same thing!
- This is a big reason why knowledge of the difference is not widespread
Cross product peculiarities

- Physicists noticed a long time ago that the cross product produces a *different* kind of vector
  - They call it an “axial vector”, “pseudovector”, “covector”, or “covariant vector”
  - It transforms differently than ordinary “polar vectors” or “contravariant vectors”
Cross product transform

- Simplest example is a reflection:

\[
M = \begin{pmatrix}
-1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]
Cross product transform

\[(1,0,0) \times (0,1,0) = (0,0,1)\]

\[M(1,0,0) \times M(0,1,0) = (-1,0,0) \times (0,1,0) = (0,0,-1)\]

- Not the same as \[M(0,0,1) = (0,0,1)\]
Cross product transform
Cross product transform

- In general, for 3 x 3 matrix \( M \),

\[
M(a_1 e_1 + a_2 e_2 + a_3 e_3) = a_1 M_1 + a_2 M_2 + a_3 M_3
\]

\[
Ma \times Mb = (a_1 M_1 + a_2 M_2 + a_3 M_3) \times (b_1 M_1 + b_2 M_2 + b_3 M_3)
\]
Cross product transform

\[ \mathbf{M}_a \times \mathbf{M}_b = \]
\[
(a_2 b_3 - a_3 b_2)(\mathbf{M}_2 \times \mathbf{M}_3)
\]
\[
+ (a_3 b_1 - a_1 b_3)(\mathbf{M}_3 \times \mathbf{M}_1)
\]
\[
+ (a_1 b_2 - a_2 b_1)(\mathbf{M}_1 \times \mathbf{M}_2)
\]
Products of matrix columns

\[
(\mathbf{M}_2 \times \mathbf{M}_3) \cdot \mathbf{M}_1 = \det \mathbf{M}
\]
\[
(\mathbf{M}_3 \times \mathbf{M}_1) \cdot \mathbf{M}_2 = \det \mathbf{M}
\]
\[
(\mathbf{M}_1 \times \mathbf{M}_2) \cdot \mathbf{M}_3 = \det \mathbf{M}
\]

- Other dot products are zero
Matrix inversion

- Cross products as rows of matrix:

\[
M = \begin{bmatrix}
M_2 \times M_3 \\
M_3 \times M_1 \\
M_1 \times M_2
\end{bmatrix}
\begin{bmatrix}
det M & 0 & 0 \\
0 & det M & 0 \\
0 & 0 & det M
\end{bmatrix}
\]
Cross product transform

- Transforming the cross product requires the inverse matrix:

\[
\begin{bmatrix}
M_2 \times M_3 \\
M_3 \times M_1 \\
M_1 \times M_2
\end{bmatrix} = (\det M)M^{-1}
\]
Cross product transform

- Transpose the inverse to get right result:

\[
(\det \mathbf{M}) \mathbf{M}^{-T} \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} = \]

\[
(a_2 b_3 - a_3 b_2)(\mathbf{M}_2 \times \mathbf{M}_3) + (a_3 b_1 - a_1 b_3)(\mathbf{M}_3 \times \mathbf{M}_1) + (a_1 b_2 - a_2 b_1)(\mathbf{M}_1 \times \mathbf{M}_2)
\]
Cross product transform

- Transformation formula:

\[ Ma \times Mb = (\det M)M^{-T} (a \times b) \]

- Result of cross product must be transformed by inverse transpose times determinant
Cross product transform

- If $\mathbf{M}$ is orthogonal, then inverse transpose is the same as $\mathbf{M}$
- If the determinant is positive, then it can be left out if you don’t care about length
- Determinant times inverse transpose is called adjugate transpose
Cross product transform

- What’s really going on here?
- When we take a cross product, we are really creating a bivector
- Bivectors are not vectors, and they don’t behave like vectors
Normal “vectors”

- A triangle normal is created by taking the cross product between two tangent vectors
- A normal is a bivector and transforms as such
Normal “vector” transformation
Classical derivation

- Standard proof for inverse transpose for transforming normals:
  - Preserve zero dot product with tangent
  - Misses extra factor of det $M$

\[
N \cdot T = 0 \\
UN \cdot MT = 0 \\
N^T U^T MT = 0 \\
U^T = M^{-1} \\
U = M^{-T}
\]
Matrix inverses

- In general, the $i$-th row of the inverse of $M$ is $1/\det M$ times the wedge product of all columns of $M$ except column $i$. 
Higher dimensions

- In $n$ dimensions, the $(n-1)$-vectors have $n$ components, just as 1-vectors do.
- Each 1-vector basis element uses exactly one of the spatial directions $\mathbf{e}_1\ldots\mathbf{e}_n$.
- Each $(n-1)$-vector basis element uses all except one of the spatial directions $\mathbf{e}_1\ldots\mathbf{e}_n$. 
Symmetry in three dimensions

- Vector basis and bivector \((n-1)\) basis

\[
\begin{align*}
\mathbf{e}_1 & \quad \mathbf{e}_2 \wedge \mathbf{e}_3 \\
\mathbf{e}_2 & \quad \mathbf{e}_3 \wedge \mathbf{e}_1 \\
\mathbf{e}_3 & \quad \mathbf{e}_1 \wedge \mathbf{e}_2
\end{align*}
\]
Symmetry in four dimensions

- Vector basis and trivector \((n-1)\) basis

\[
\begin{align*}
\mathbf{e}_1 & \quad \mathbf{e}_2 \wedge \mathbf{e}_3 \wedge \mathbf{e}_4 \\
\mathbf{e}_2 & \quad \mathbf{e}_1 \wedge \mathbf{e}_4 \wedge \mathbf{e}_3 \\
\mathbf{e}_3 & \quad \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_4 \\
\mathbf{e}_4 & \quad \mathbf{e}_1 \wedge \mathbf{e}_3 \wedge \mathbf{e}_2
\end{align*}
\]
Dual basis

- Use special notation for wedge product of all but one basis vector:

\[ \overline{e}_1 = e_2 \wedge e_3 \wedge e_4 \]

\[ \overline{e}_2 = e_1 \wedge e_4 \wedge e_3 \]

\[ \overline{e}_3 = e_1 \wedge e_2 \wedge e_4 \]

\[ \overline{e}_4 = e_1 \wedge e_3 \wedge e_2 \]
Dual basis

• Instead of saying \((n-1)\)-vector, we call these “antivectors”

• In \(n\) dimensions, antivector always means a quantity expressed on the basis with grade \(n-1\)
Vector / antivector product

- Wedge product between vector and antivector is the origin of the dot product

\[(a_1 e_1 + a_2 e_2 + a_3 e_3) \wedge (b_1 \bar{e}_1 + b_2 \bar{e}_2 + b_3 \bar{e}_3)\]

\[= (a_1 b_1 + a_2 b_2 + a_3 b_3)(e_1 \wedge e_2 \wedge e_3)\]

- They complement each other, and “fill in” the volume element
Vector / antivector product

- Many of the dot products you take are actually vector / antivector wedge products
- For instance, $\mathbf{N} \cdot \mathbf{L}$ in diffuse lighting
- $\mathbf{N}$ is an antivector
- Calculating volume of extruded bivector
Diffuse Lighting
The regressive product

- Grassmann realized there is another product symmetric to the wedge product
- Not well-known at all
  - Most books on geometric algebra leave it out completely
- Very important product, though!
The regressive product

- Operates on antivectors in a manner symmetric to how the wedge product operates on vectors
- Uses an upside-down wedge:

$$\bar{e}_1 \vee \bar{e}_2$$

- We call it the “antiwedge” product
The antiwedge product

- Has same properties as wedge product, but for antivectors
- Operates in complementary space on dual basis or “antibasis”
The antiwedge product

- Whereas the wedge product increases grade, the antiwedge product decreases it.
- Suppose, in $n$-dimensional Grassmann algebra, $A$ has grade $r$ and $B$ has grade $s$.
- Then $A \wedge B$ has grade $r + s$.
- And $A \vee B$ has grade
  
  \[ n - (n - r) - (n - s) = r + s - n \]
Antiwedge product in 3D

\[ \bar{e}_1 \lor \bar{e}_2 = (e_2 \land e_3) \lor (e_3 \land e_1) = e_3 \]

\[ \bar{e}_2 \lor \bar{e}_3 = (e_3 \land e_1) \lor (e_1 \land e_2) = e_1 \]

\[ \bar{e}_3 \lor \bar{e}_1 = (e_1 \land e_2) \lor (e_2 \land e_3) = e_2 \]
Similar shorthand notation

\[ \overline{e}_{12} = \overline{e}_1 \lor \overline{e}_2 \]
\[ \overline{e}_{23} = \overline{e}_2 \lor \overline{e}_3 \]
\[ \overline{e}_{31} = \overline{e}_3 \lor \overline{e}_1 \]
\[ \overline{e}_{123} = \overline{e}_1 \lor \overline{e}_2 \lor \overline{e}_3 \]
Join and meet

- Wedge product joins *vectors* together
  - Analogous to union
- Antiwedge product joins *antivectors*
  - Antivectors represent absence of geometry
  - Joining antivectors is like removing vectors
  - Analogous to intersection
  - Called a meet operation
Homogeneous coordinates

- Points have a 4D representation:

  \[ P = (x, y, z, w) \]

- Conveniently allows affine transformation through 4 x 4 matrix

- Used throughout 3D graphics
Homogeneous points

- To project onto 3D space, find where 4D vector intersects subspace where $w = 1$

$$P = (x, y, z, w)$$

$$P_{3D} = \left( \frac{x}{w}, \frac{y}{w}, \frac{z}{w} \right)$$
Homogeneous model

- With Grassmann algebra, homogeneous model can be extended to include 3D points, lines, and planes
- Wedge and antiwedge products naturally perform union and intersection operations among all of these
4D Grassmann Algebra

- Scalar unit
- Four vectors: \( e_1, e_2, e_3, e_4 \)
- Six bivectors: \( e_{12}, e_{23}, e_{31}, e_{41}, e_{42}, e_{43} \)
- Four antivectors: \( \bar{e}_1, \bar{e}_2, \bar{e}_3, \bar{e}_4 \)
- Antiscalar unit (quadvector)
Homogeneous lines

- Take wedge product of two 4D points

\[ P = (P_x, P_y, P_z, 1) = P_x e_1 + P_y e_2 + P_z e_3 + e_4 \]

\[ Q = (Q_x, Q_y, Q_z, 1) = Q_x e_1 + Q_y e_2 + Q_z e_3 + e_4 \]
Homogeneous lines

\[ \mathbf{P} \wedge \mathbf{Q} = (Q_x - P_x) \mathbf{e}_{41} + (Q_y - P_y) \mathbf{e}_{42} + (Q_z - P_z) \mathbf{e}_{43} \]
\[ + (P_y Q_z - P_z Q_y) \mathbf{e}_{23} + (P_z Q_x - P_x Q_z) \mathbf{e}_{31} + (P_x Q_y - P_y Q_x) \mathbf{e}_{12} \]

- This bivector spans a 2D plane in 4D
- In subspace where \( w = 1 \), this is a 3D line
Homogeneous lines

- The 4D bivector no longer contains any information about the two points used to create it.
- Contrary to parametric origin / direction representation.
Homogeneous lines

- The 4D bivector can be decomposed into two 3D components:
  - A tangent vector and a moment bivector
  - These are perpendicular

\[
P \wedge Q = (Q_x - P_x) e_{41} + (Q_y - P_y) e_{42} + (Q_z - P_z) e_{43} \\
+ (P_y Q_z - P_z Q_y) e_{23} + (P_z Q_x - P_x Q_z) e_{31} + (P_x Q_y - P_y Q_x) e_{12}
\]
Homogeneous lines

- Tangent vector is $\mathbf{T} = \mathbf{Q}^{3D} - \mathbf{P}^{3D}$
- Moment bivector is $\mathbf{M} = \mathbf{P}^{3D} \wedge \mathbf{Q}^{3D}$

$$\mathbf{P} \wedge \mathbf{Q} = (Q_x - P_x)\mathbf{e}_{41} + (Q_y - P_y)\mathbf{e}_{42} + (Q_z - P_z)\mathbf{e}_{43}$$

$$+ (P_y Q_z - P_z Q_y)\mathbf{e}_{23} + (P_z Q_x - P_x Q_z)\mathbf{e}_{31} + (P_x Q_y - P_y Q_x)\mathbf{e}_{12}$$
Moment bivector
Plücker coordinates

- Origin of Plücker coordinates revealed!
  - They are the coefficients of a 4D bivector
- A line \( \mathbf{L} \) in Plücker coordinates is

\[
\mathbf{L} = \{ \mathbf{Q} - \mathbf{P} : \mathbf{P} \times \mathbf{Q} \}
\]

- A bunch of seemingly arbitrary formulas in Plücker coordinates will become clear
Homogeneous planes

- Take wedge product of three 4D points

\[ P = (P_x, P_y, P_z, 1) = P_x e_1 + P_y e_2 + P_z e_3 + e_4 \]
\[ Q = (Q_x, Q_y, Q_z, 1) = Q_x e_1 + Q_y e_2 + Q_z e_3 + e_4 \]
\[ R = (R_x, R_y, R_z, 1) = R_x e_1 + R_y e_2 + R_z e_3 + e_4 \]
Homogeneous planes

\[ P \wedge Q \wedge R = N_x \vec{e}_1 + N_y \vec{e}_2 + N_z \vec{e}_3 + D\vec{e}_4 \]

- \( N \) is the 3D normal bivector
- \( D \) is the offset from origin in units of \( N \)

\[ N = P_{3D} \wedge Q_{3D} + Q_{3D} \wedge R_{3D} + R_{3D} \wedge P_{3D} \]

\[ D = -P_{3D} \wedge Q_{3D} \wedge R_{3D} \]
A homogeneous plane is a 4D antivector. It transforms by the inverse of a 4 x 4 matrix. Just like a 3D antivector transforms by the inverse of a 3 x 3 matrix. Orthogonality is not common here due to translation in the matrix.
Projective geometry

<table>
<thead>
<tr>
<th>4D Entity</th>
<th>3D Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector (1-space)</td>
<td>Point (0-space)</td>
</tr>
<tr>
<td>Bivector (2-space)</td>
<td>Line (1-space)</td>
</tr>
<tr>
<td>Trivector (3-space)</td>
<td>Plane (2-space)</td>
</tr>
</tbody>
</table>

- We always project onto the 3D subspace where \( w = 1 \)
Geometric computation in 4D

- Wedge product
  - Multiply two points to get the line containing both points
  - Multiply three points to get the plane containing all three points
  - Multiply a line and a point to get the plane containing the line and the point
Geometric computation in 4D

- Antiwedge product
  - Multiply **two planes** to get the line where they intersect
  - Multiply **three planes** to get the point common to all three planes
  - Multiply **a line and a plane** to get the point where the line intersects the plane
Geometric computation in 4D

- Wedge or antiwedge product
  - Multiply a point and a plane to get the signed minimum distance between them in units of the normal magnitude
  - Multiply two lines to get a special signed crossing value
Product of two lines

- Wedge product gives an antiscalar (quadvector or 4D volume element)
- Antiwedge product gives a scalar
- Both have same sign and magnitude
- Grassmann treated scalars and antiscalars as the same thing
Product of two lines

- Let \( L_1 \) have tangent \( T_1 \) and moment \( M_1 \)
- Let \( L_2 \) have tangent \( T_2 \) and moment \( M_2 \)
- Then,

\[
L_1 \lor L_2 = -(T_1 \lor M_2 + T_2 \lor M_1)
\]
\[
L_1 \land L_2 = -(T_1 \land M_2 + T_2 \land M_1)
\]
Product of two lines

- The product of two lines gives a “crossing” relation
  - Positive value means clockwise crossing
  - Negative value means counterclockwise
  - Zero if lines intersect
Crossing relation

$L_1 \lor L_2 > 0$

$L_1 \lor L_2 < 0$
Distance between lines

- Product of two lines also relates to signed minimum distance between them

\[
d = \frac{\mathbf{L}_1 \lor \mathbf{L}_2}{\|\mathbf{T}_1 \land \mathbf{T}_2\|}
\]

- (Here, numerator is 4D antiwedge product, and denominator is 3D wedge product.)
Ray-triangle intersection

- Application of line-line product
- Classic barycentric calculation difficult due to floating-point round-off error
  - Along edge between two triangles, ray can miss both or hit both
  - Typical solution involves use of ugly epsilons
Ray-triangle intersection

- Calculate 4D bivectors for triangle edges and ray
  - Take antiwedge products between ray and three edges
  - Same sign for all three edges is a hit
  - Impossible to hit or miss both triangles sharing edge
  - Need to handle zero in consistent way
Weighting

- Points, lines, and planes have “weights” in homogeneous coordinates

<table>
<thead>
<tr>
<th>Entity</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>(w) coordinate</td>
</tr>
<tr>
<td>Line</td>
<td>Tangent component (T)</td>
</tr>
<tr>
<td>Plane</td>
<td>(x, y, z) component</td>
</tr>
</tbody>
</table>
Weighting

- Mathematically, the weight components can be found by taking the antiwedge product with the antivector \((0,0,0,1)\)
- We would never really do that, though, because we can just look at the right coefficients
Normalized lines

- Tangent component has unit length
  - Magnitude of moment component is perpendicular distance to the origin
Normalized planes

- \((x,y,z)\) component has unit length
- Wedge product with (normalized) point is perpendicular distance to plane
Programming considerations

- Convenient to create classes to represent entities of each grade
  - Vector4D
  - Bivector4D
  - Antivector4D
Programming considerations

- Fortunate happenstance that C++ has an overloadable operator $\wedge$ that looks like a wedge
- But be careful with operator precedence if you overload $\wedge$ to perform wedge product
  - Has lowest operator precedence, so get used to enclosing wedge products in parentheses
Combining wedge and antiwedge

- The same operator can be used for wedge product and antiwedge product
  - Either they both produce the same scalar and antiscalar magnitudes with the same sign
  - Or one of the products is identically zero
  - For example, you would always want the antiwedge product for two planes because the wedge product is zero for all inputs
## Summary

<table>
<thead>
<tr>
<th>Old school</th>
<th>New school</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross product → axial vector</td>
<td>Wedge product → bivector</td>
</tr>
<tr>
<td>Dot product</td>
<td>Antiwedge vector / antivector</td>
</tr>
<tr>
<td>Scalar triple product</td>
<td>Triple wedge product</td>
</tr>
<tr>
<td>Plücker coordinates</td>
<td>4D bivectors</td>
</tr>
<tr>
<td>Operations in Plücker coordinates</td>
<td>4D wedge / antiwedge products</td>
</tr>
<tr>
<td>Transform normals with inverse transpose</td>
<td>Transform antivectors with adjugate transpose</td>
</tr>
</tbody>
</table>
• Slides available online at
  • http://www.terathon.com/lengyel/

• Contact
  • lengyel@terathon.com
Supplemental Slides
Example application

- Calculation of shadow region planes from light position and frustum edges
  - Simply a wedge product
Points of closest approach

- Wedge product of line tangents gives complement of direction between closest points
Points of closest approach

- Plane containing this direction and first line also contains closest point on second line
Two dimensions

- 1 scalar unit
- 2 basis vectors
- 1 bivector / antiscalar unit
- No cross product
- All rotations occur in plane of 1 bivector
One dimension

- 1 scalar unit
- 1 single-component basis vector
  - Also antiscalar unit
- Equivalent to “dual numbers”
- All numbers have form $a + be$
  - Where $e^2 = 0$
Explicit formulas

- Define points \( P, Q \) and planes \( E, F \), and line \( L \):

\[
P = (P_x, P_y, P_z, 1) = P_x e_1 + P_y e_2 + P_z e_3 + e_4
\]

\[
Q = (Q_x, Q_y, Q_z, 1) = Q_x e_1 + Q_y e_2 + Q_z e_3 + e_4
\]

\[
E = (E_x, E_y, E_z, E_w) = E_x e_1 + E_y e_2 + E_z e_3 + E_w e_4
\]

\[
F = (F_x, F_y, F_z, F_w) = F_x e_1 + F_y e_2 + F_z e_3 + F_w e_4
\]

\[
L = T_x e_{14} + T_y e_{24} + T_z e_{34} + M_x e_{23} + M_y e_{31} + M_z e_{12}
\]
Explicit formulas

- Product of two points

\[
P \land Q = (Q_x - P_x)e_{41} + (Q_y - P_y)e_{42} + (Q_z - P_z)e_{43} \\
+ (P_y Q_z - P_z Q_y)e_{23} + (P_z Q_x - P_x Q_z)e_{31} + (P_x Q_y - P_y Q_x)e_{12}
\]
Explicit formulas

- Product of two planes

\[ E \lor F = (E_z F_y - E_y F_z) e_{41} + (E_x F_z - E_z F_x) e_{42} + (E_y F_x - E_x F_y) e_{43} \]
\[ + (E_x F_w - E_w F_x) e_{23} + (E_y F_w - E_w F_y) e_{31} + (E_z F_w - E_w F_z) e_{12} \]
Explicit formulas

- Product of line and point

\[
L \wedge P = (T_y P_z - T_z P_y + M_x) \bar{e}_1 + (T_z P_x - T_x P_z + M_y) \bar{e}_2 \\
+ (T_x P_y - T_y P_x + M_z) \bar{e}_3 + (-P_x M_x - P_y M_y - P_z M_z) \bar{e}_4
\]
Explicit formulas

- Product of line and plane

\[
L \lor E = (M_z E_y - M_y E_z - T_x E_w) e_1 + (M_x E_z - M_z E_x - T_y E_w) e_2 \\
+ (M_y E_x - M_x E_y - T_z E_w) e_3 + (E_x T_x + E_y T_y + E_z T_z) e_4
\]