Jaakko Lehtinen, MIT CSAIL
with lots of material from Frédo Durand
Cool Submissions from Assn 2

- Fully rigged meshes
  - keyframing
  - IK

Pat Doyle

Szymon Jakubczak
Stop the Press

- Some very useful books are available on books24x7
  - Real-Time Rendering, 2nd ed. (Akenine-Möller, Haines)
  - Realistic Ray Tracing, 2nd ed. (Shirley, Morley)
  - Advanced Global Illumination, 2nd ed. (Dutré, Bala, Bekaert)
Tuesday Recap

• Intro to rendering
  – Producing a picture based on scene description
  – Main variants: Ray casting/tracing vs. rasterization
  – Ray casting vs. ray tracing (secondary rays)

• Ray Casting basics
  – Camera definitions
    • Orthographic, perspective
  – Ray representation
    • \( P(t) = \text{origin} + t \times \text{direction} \)
  – Ray generation
  – Ray/plane intersection
Ray-Plane Intersection

- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for $t$

\[ P(t) = R_0 + t \cdot R_d \]
\[ H(P) = n \cdot P + D = 0 \]
\[ n \cdot (R_0 + t \cdot R_d) + D = 0 \]
\[ t = \frac{-(D + n \cdot R_0)}{n \cdot R_d} \]

Done!
Ray Casting

- Ray Casting Basics
- Camera and Ray Generation
- Ray-Plane Intersection
- Ray-Sphere Intersection
Sphere Representation?

• Implicit sphere equation
  – Assume centered at origin (easy to translate)
  – $H(P) = ||P||^2 - r^2 = P \cdot P - r^2 = 0$
Ray-Sphere Intersection

\[ R_d \]

\[ R_o \]
Ray-Sphere Intersection

- Insert explicit equation of ray into implicit equation of sphere & solve for $t$

$$P(t) = R_o + t*R_d$$  ;  $$H(P) = P \cdot P - r^2 = 0$$
Ray-Sphere Intersection

- Insert explicit equation of ray into implicit equation of sphere & solve for $t$

\[ P(t) = R_o + t*R_d \quad ; \quad H(P) = P \cdot P - r^2 = 0 \]

\[
(R_o + tR_d) \cdot (R_o + tR_d) - r^2 = 0
\]

\[
R_d \cdot R_d t^2 + 2R_d \cdot R_o t + R_o \cdot R_o - r^2 = 0
\]
Ray-Sphere Intersection

- Quadratic: \( at^2 + bt + c = 0 \)
  - \( a = 1 \) (but only if \( ||R_d|| = 1 \)!!!)
  - \( b = 2R_d \cdot R_o \)
  - \( c = R_o \cdot R_o - r^2 \)

- with discriminant \( d = \sqrt{b^2 - 4ac} \)

- and solutions \( t_{\pm} = \frac{-b \pm d}{2a} \)
Ray-Sphere Intersection

• 3 cases, depending on the sign of $b^2 - 4ac$
• What do these cases correspond to?
Ray-Sphere Intersection

- 3 cases, depending on the sign of $b^2 - 4ac$
- What do these cases correspond to?
- Which root ($t+$ or $t-$) should you choose?
  - Closest positive!
Ray-Sphere Intersection

- It's so easy that all ray-tracing images have spheres!

:-)
Sphere Normal

• Simply $\mathbf{Q}/||\mathbf{Q}||$
  – $\mathbf{Q} = \mathbf{P}(t)$, intersection point
  – (for spheres centered at origin)
Questions?
Ray-Triangle Intersection
Ray-Triangle Intersection

- Use ray-plane intersection followed by in-triangle test
Ray-Triangle Intersection

- Use ray-plane intersection followed by in-triangle test
- Or try to be smarter
  - Use barycentric coordinates
Barycentric Definition of a Plane

• A (non-degenerate) triangle \((a,b,c)\) defines a plane
• Any point \(P\) on this plane can be written as
  \[ P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c, \]
  with \(\alpha + \beta + \gamma = 1\)

Why? How?

[Möbius, 1827]
Barycentric Coordinates
Barycentric Coordinates

- Since $\alpha + \beta + \gamma = 1$, we can write $\alpha = 1 - \beta - \gamma$

  $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$

  $P(\beta, \gamma) = (1-\beta-\gamma)a + \beta b + \gamma c$

  $= a + \beta(b-a) + \gamma(c-a)$
Barycentric Coordinates

• Since $\alpha + \beta + \gamma = 1$, we can write $\alpha = 1 - \beta - \gamma$

\[
P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c
\]

\[
P(\beta, \gamma) = (1 - \beta - \gamma) a + \beta b + \gamma c
\]

\[
= a + \beta(b - a) + \gamma(c - a)
\]

Vectors that lie on the triangle plane
Barycentric Coordinates

• Since $\alpha + \beta + \gamma = 1$, we can write $\alpha = 1 - \beta - \gamma$

$$P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$$

$$P(\beta, \gamma) = (1-\beta-\gamma)a + \beta b + \gamma c$$

$$= a + \beta (b-a) + \gamma (c-a)$$

Non-orthogonal coordinate system on the plane!
Barycentric Definition of a Plane

• \( P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c \)
  with \( \alpha + \beta + \gamma = 1 \)

• Is it explicit or implicit?

Fun to know:
\( P \) is the barycenter, the single point upon which the triangle would balance if weights of size \( \alpha, \beta, \) & \( \gamma \) are placed on points \( a, b \) & \( c \).
Barycentric Definition of a Triangle

- \( \mathbf{P}(\alpha, \beta, \gamma) = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c} \)
  with \( \alpha + \beta + \gamma = 1 \) parametrizes the entire plane
Barycentric Definition of a Triangle

- \( P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c \)
  with \( \alpha + \beta + \gamma = 1 \) parametrizes the entire plane

- If we require in addition that \( \alpha, \beta, \gamma \geq 0 \), we get just the triangle!
  - Note that with \( \alpha + \beta + \gamma = 1 \) this implies
    \[ 0 < \alpha < 1 \quad \& \quad 0 < \beta < 1 \quad \& \quad 0 < \gamma < 1 \]
  - Verify:
    - \( \alpha=0 \Rightarrow P \) lies on line \( b-c \)
    - \( \alpha,\beta=0 \Rightarrow P = c \)
    - etc.
How Do We Compute $\alpha$, $\beta$, $\gamma$?

- Ratio of opposite sub-triangle area to total area
  - $\alpha = A_a/A$  $\beta = A_b/A$  $\gamma = A_c/A$
- Use signed areas for points outside the triangle
How Do We Compute $\alpha, \beta, \gamma$?

- Or write it as a 2x2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$
  - $e_1 = (b-a), e_2 = (c-a)$
  
  $$a + \beta e_1 + \gamma e_2 - P = 0$$

This should be zero
How Do We Compute $\alpha, \beta, \gamma$?

- Or write it as a 2x2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$
  - $e_1 = (b-a)$, $e_2 = (c-a)$

\[ a + \beta e_1 + \gamma e_2 - P = 0 \]

This should be zero

Something’s wrong...
This is a linear system of 3 equations and 2 unknowns!
How Do We Compute $\alpha, \beta, \gamma$?

- Or write it as a 2x2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$
  
  $- e_1 = (b-a), e_2 = (c-a)$

$\langle e_1, a + \beta e_1 + \gamma e_2 - P \rangle = 0$

$\langle e_2, a + \beta e_1 + \gamma e_2 - P \rangle = 0$

These should be zero

Ha! We’ll take inner products of this equation with $e_1, e_2$
How Do We Compute $\alpha, \beta, \gamma$?

- Or write it as a 2x2 linear system
- $\mathbf{P}(\beta, \gamma) = \mathbf{a} + \beta \mathbf{e}_1 + \gamma \mathbf{e}_2$

  \[ \begin{align*}
  \mathbf{e}_1 &= (\mathbf{b} - \mathbf{a}), \quad \mathbf{e}_2 = (\mathbf{c} - \mathbf{a}) \\
  \langle \mathbf{e}_1, \mathbf{a} + \beta \mathbf{e}_1 + \gamma \mathbf{e}_2 - \mathbf{P} \rangle &= 0 \\
  \langle \mathbf{e}_2, \mathbf{a} + \beta \mathbf{e}_1 + \gamma \mathbf{e}_2 - \mathbf{P} \rangle &= 0
  \end{align*} \]

\[
\begin{pmatrix}
\langle \mathbf{e}_1, \mathbf{e}_1 \rangle & \langle \mathbf{e}_1, \mathbf{e}_2 \rangle \\
\langle \mathbf{e}_2, \mathbf{e}_1 \rangle & \langle \mathbf{e}_2, \mathbf{e}_2 \rangle
\end{pmatrix}
\begin{pmatrix}
\beta \\
\gamma
\end{pmatrix}
=
\begin{pmatrix}
c_1 \\
c_2
\end{pmatrix}
\]

where
\[
\begin{pmatrix}
c_1 \\
c_2
\end{pmatrix}
=
\begin{pmatrix}
\langle (\mathbf{P} - \mathbf{a}), \mathbf{e}_1 \rangle \\
\langle (\mathbf{P} - \mathbf{a}), \mathbf{e}_2 \rangle
\end{pmatrix}
\]

and $\langle \mathbf{a}, \mathbf{b} \rangle$ is the dot product.
Questions?
Intersection with Barycentric Triangle
Intersection with Barycentric Triangle

- Again, set ray equation equal to barycentric equation
Intersection with Barycentric Triangle

- Again, set ray equation equal to barycentric equation

\[ \mathbf{P}(t) = \mathbf{P}(\beta, \gamma) \]
Intersection with Barycentric Triangle

- Again, set ray equation equal to barycentric equation

\[ P(t) = P(\beta, \gamma) \]

\[ R_o + t * R_d = a + \beta(b-a) + \gamma(c-a) \]
Intersection with Barycentric Triangle

- Again, set ray equation equal to barycentric equation
  \[ P(t) = P(\beta, \gamma) \]
  \[ R_o + t \cdot R_d = a + \beta(b-a) + \gamma(c-a) \]
- Intersection if \( \beta + \gamma < 1 \) & \( \beta > 0 \) & \( \gamma > 0 \)
  
  (and \( t > t_{\text{min}} \ldots \))
Intersection with Barycentric Triangle
Intersection with Barycentric Triangle

- \( \mathbf{R}_o + t \mathbf{R}_d = \mathbf{a} + \beta (\mathbf{b} - \mathbf{a}) + \gamma (\mathbf{c} - \mathbf{a}) \)
Intersection with Barycentric Triangle

- \( \mathbf{R}_o + t \mathbf{R}_d = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \)

\[
\begin{align*}
\mathbf{R}_{ox} + t\mathbf{R}_{dx} &= a_x + \beta(b_x - a_x) + \gamma(c_x - a_x) \\
\mathbf{R}_{oy} + t\mathbf{R}_{dy} &= a_y + \beta(b_y - a_y) + \gamma(c_y - a_y) \\
\mathbf{R}_{oz} + t\mathbf{R}_{dz} &= a_z + \beta(b_z - a_z) + \gamma(c_z - a_z)
\end{align*}
\]
Intersection with Barycentric Triangle

- \( \mathbf{R}_o + t \, \mathbf{R}_d = \mathbf{a} + \beta (\mathbf{b} - \mathbf{a}) + \gamma (\mathbf{c} - \mathbf{a}) \)
  
  \[
  \begin{align*}
  \mathbf{R}_{ox} + t \mathbf{R}_{dx} &= a_x + \beta (b_x - a_x) + \gamma (c_x - a_x) \\
  \mathbf{R}_{oy} + t \mathbf{R}_{dy} &= a_y + \beta (b_y - a_y) + \gamma (c_y - a_y) \\
  \mathbf{R}_{oz} + t \mathbf{R}_{dz} &= a_z + \beta (b_z - a_z) + \gamma (c_z - a_z)
  \end{align*}
  \]

3 equations, 3 unknowns
Intersection with Barycentric Triangle

- $R_o + t \cdot R_d = a + \beta (b-a) + \gamma (c-a)$

\[
\begin{align*}
R_{ox} + tR_{dx} &= a_x + \beta (b_x-a_x) + \gamma (c_x-a_x) \\
R_{oy} + tR_{dy} &= a_y + \beta (b_y-a_y) + \gamma (c_y-a_y) \\
R_{oz} + tR_{dz} &= a_z + \beta (b_z-a_z) + \gamma (c_z-a_z)
\end{align*}
\]

- Regroup & write in matrix form $Ax = b$

\[
\begin{bmatrix}
\beta \\
\gamma \\
t
\end{bmatrix}
= 
\begin{bmatrix}
\beta \\
\gamma \\
t
\end{bmatrix}
\]

3 equations, 3 unknowns
Intersection with Barycentric Triangle

- \( \mathbf{R}_o + t \times \mathbf{R}_d = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \)

\[
\begin{align*}
\mathbf{R}_{ox} + t \mathbf{R}_{dx} &= a_x + \beta(b_x - a_x) + \gamma(c_x - a_x) \\
\mathbf{R}_{oy} + t \mathbf{R}_{dy} &= a_y + \beta(b_y - a_y) + \gamma(c_y - a_y) \\
\mathbf{R}_{oz} + t \mathbf{R}_{dz} &= a_z + \beta(b_z - a_z) + \gamma(c_z - a_z)
\end{align*}
\]

- Regroup & write in matrix form \( \mathbf{A}\mathbf{x} = \mathbf{b} \)

\[
\begin{bmatrix}
    a_x - b_x & a_x - c_x & R_{dx} \\
    a_y - b_y & a_y - c_y & R_{dy} \\
    a_z - b_z & a_z - c_z & R_{dz}
\end{bmatrix}
\begin{bmatrix}
    \beta \\
    \gamma \\
    t
\end{bmatrix}
= \begin{bmatrix}
    a_x - R_{ox} \\
    a_y - R_{oy} \\
    a_z - R_{oz}
\end{bmatrix}
\]
Cramer’s Rule

- Used to solve for one variable at a time in system of equations

\[
\beta = \frac{\begin{vmatrix} a_x - R_{ox} & a_x - c_x & R_{dx} \\ a_y - R_{oy} & a_y - c_y & R_{dy} \\ a_z - R_{oz} & a_z - c_z & R_{dz} \end{vmatrix}}{|A|} \\
\gamma = \frac{\begin{vmatrix} a_x - b_x & a_x - R_{ox} & R_{dx} \\ a_y - b_y & a_y - R_{oy} & R_{dy} \\ a_z - b_z & a_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|} \\
t = \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}
\]

\(| | \) denotes the determinant

Can be copied mechanically into code
Barycentric Intersection Pros

- Efficient
- Stores no plane equation
- Get the barycentric coordinates for free
  - Useful for interpolation, texture mapping
Barycentric Interpolation

- Values $v_1$, $v_2$, $v_3$ defined at $a$, $b$, $c$
  - Colors, normal, texture coordinates, etc.
- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$ is the point...
- $v(\alpha, \beta, \gamma) = \alpha v_1 + \beta v_2 + \gamma v_3$ is the barycentric interpolation of $v_1$-$v_3$ at point $P$
  - Sanity check: $v(1,0,0) = v_1$, etc.
- I.e., once you know $\alpha$, $\beta$, $\gamma$, $v_1$ you can interpolate values using the same weights.
  - Convenient!
Questions?

- Image computed using the RADIANCE system by Greg Ward
Ray Casting: Object oriented design

For every pixel
   Construct a ray from the eye
   For every object in the scene
      Find intersection with the ray
      Keep if closest
Object-Oriented Design

- We want to be able to add primitives easily
  - Inheritance and virtual methods
- Even the scene is derived from Object3D!

```
Object3D
  bool intersect(Ray, Hit, tmin)
```

- Plane
  ```
  bool intersect(Ray, Hit, tmin)
  ```
- Sphere
  ```
  bool intersect(Ray, Hit, tmin)
  ```
- Triangle Mesh
  ```
  bool intersect(Ray, Hit, tmin)
  ```
- Group
  ```
  bool intersect(Ray, Hit, tmin)
  ```

- Also cameras are abstracted (perspective/ortho)
  - Methods for generating rays for given image coordinates
Assignment 4 & 5: Ray Casting/Tracing

• Write a basic ray caster
  – Orthographic and perspective cameras
  – Spheres and triangles
  – 2 Display modes: color and distance

• We provide classes for
  – Ray: origin, direction
  – Hit: t, Material, (normal)
  – Scene Parsing

• You write ray generation, hit testing, simple shading
Books

- Peter Shirley et al.: *Fundamentals of Computer Graphics* AK Peters

- Ray Tracing
  - Jensen
  - Shirley
  - Glassner

Remember the ones at books24x7 mentioned in the beginning!
Constructive Solid Geometry (CSG)

• A neat way to build complex objects from simple parts using Boolean operations
  – Very easy when ray tracing
• We used this in the Max Payne games for modeling the environments
  – Not so easy when not ray tracing :)}
CSG Examples
Constructive Solid Geometry (CSG)

Given overlapping shapes A and B:

Union
Intersection
Subtraction

Should only “count” overlap region once!
How can we implement CSG?

4 cases

Union

Intersection

Subtraction
Collect Intersections

Each ray processed separately!
Implementing CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A

This would certainly work, but would need to determine if points are inside solids...

:-(
Implementing CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A

2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - How? Just keep an “entry” / “exit” bit for each intersection
     - Easy to determine from intersection normal and ray direction
   - Compute union/intersection/subtraction of the intervals
Implementing CSG

2. Overlapping intervals:
   • Find the intervals of "inside" along the ray for A and B
   • How? Just keep an “entry” / “exit” bit for each intersection
     • Easy to determine from intersection normal and ray direction
   • Compute union/intersection/subtraction of the intervals

Problem reduces to 1D for each ray
CSG is Easy with Ray Casting...

• ...but very hard if you actually try to compute an explicit representation of the resulting surface as a triangle mesh

• In principle very simple, but floating point numbers are not exact
  – E.g., points do not lie exactly on planes...
  – Computing the intersection A vs B is not necessarily the same as B vs A...
  – The line that results from intersecting two planes does not necessarily lie on either plane...
  – etc., etc.
CSG Raytraced Image à la Fredo
Questions?
Precision

- What happens when
  - Ray Origin lies on an object?
  - Grazing rays?

- Problem with floating-point approximation
The evil $\varepsilon$

- In ray tracing, do NOT report intersection for rays starting at the surface
  - Secondary rays will start at the surfaces
  - Requires epsilons
  - Best to nudge the starting point off the surface e.g., along normal
The evil $\varepsilon$

- Edges in triangle meshes
  - Must report intersection (otherwise not watertight)
  - Hard to get right
Questions?

Image by Henrik Wann Jensen
Transformations and Ray Casting

- We have seen that transformations such as affine transforms are useful for modeling & animation
- How do we incorporate them into ray casting?
Incorporating Transforms

1. Make each primitive handle any applied transformations and produce a camera space description of its geometry

   Transform {
       Translate { 1 0.5 0 }
       Scale { 2 2 2 }
       Sphere {
           center 0 0 0
           radius 1
       }
   }

2. ...Or Transform the Rays
Primitives Handle Transforms

```plaintext
Sphere {
    center 3 2 0
    z_rotation 30
    r_major 2
    r_minor 1
}
```

- Complicated for many primitives
Transform the Ray

• Move the ray from *World Space* to *Object Space*

\[
\mathbf{p}_{WS} = \mathbf{M} \mathbf{p}_{OS}
\]

\[
\mathbf{p}_{OS} = \mathbf{M}^{-1} \mathbf{p}_{WS}
\]
Transform Ray

- New origin:

- New direction:
Transform Ray

- New origin:
  \[ \text{origin}_{OS} = M^{-1} \text{origin}_{WS} \]
- New direction:

\[ \begin{align*} q_{WS} &= \text{origin}_{WS} + t_{WS} \times \text{direction}_{WS} \\
q_{OS} &= \text{origin}_{OS} + t_{OS} \times \text{direction}_{OS} \end{align*} \]
Transform Ray

- New origin:
  \[ \text{origin}_{OS} = M^{-1} \text{origin}_{WS} \]

- New direction:
  \[ \text{direction}_{OS} = M^{-1} (\text{origin}_{WS} + 1 \times \text{direction}_{WS}) - M^{-1} \text{origin}_{WS} \]
Transform Ray

- New origin:
  \[ \text{origin}_{OS} = \mathbf{M}^{-1} \text{origin}_{WS} \]
- New direction:
  \[ \text{direction}_{OS} = \mathbf{M}^{-1} (\text{origin}_{WS} + 1 \times \text{direction}_{WS}) - \mathbf{M}^{-1} \text{origin}_{WS} \]
  \[ \text{direction}_{OS} = \mathbf{M}^{-1} \text{direction}_{WS} \]

Note that the w component of direction is 0!

World Space

Object Space
What about $t$?

- If $M$ includes scaling, $\text{direction}_{OS}$ ends up NOT be normalized after transformation

- Two solutions
  - Normalize the direction
  - Don't normalize the direction
1. Normalize direction

World Space

Object Space

1_{WS}

1_{OS}
1. Normalize direction

- \( t_{OS} \neq t_{WS} \)
  and must be rescaled after intersection

==> One more possible failure case...
2. Don't normalize direction
2. Don't normalize direction

- $t_{OS} = t_{WS} \rightarrow$ convenient!
2. Don't normalize direction

- \( t_{OS} = t_{WS} \) \( \rightarrow \) convenient!
- But you should not rely on \( t_{OS} \) being true distance in intersection routines (e.g. \( a \neq 1 \) in ray-sphere test)
2. Don't normalize direction

- \( t_{OS} = t_{WS} \) \( \Rightarrow \) convenient!
- But you should not rely on \( t_{OS} \) being true distance in intersection routines (e.g. \( a\neq1 \) in ray-sphere test)
Transforming Points & Directions

• Transform point

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} = \begin{bmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix} = \begin{bmatrix}
ax+by+cz+d \\
ex+fy+gz+h \\
iy+jy+kz+l \\
1
\end{bmatrix}
\]

• Transform direction

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
0
\end{bmatrix} = \begin{bmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
0
\end{bmatrix} = \begin{bmatrix}
ax+by+cz \\
ex+fy+gz \\
iy+jy+kz \\
0
\end{bmatrix}
\]

Homogeneous Coordinates:

\((x,y,z,w)\)

\(w = 0\) is a point at infinity (direction)
Transforming Points & Directions

• Transform point

\[
\begin{pmatrix}
x' \\
y' \\
z' \\
1
\end{pmatrix} = \begin{pmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
z \\
1
\end{pmatrix} = \begin{pmatrix}
ax+by+cz+d \\
ex+fy+gz+h \\
iy+jz+kz+l \\
1
\end{pmatrix}
\]

• Transform direction

\[
\begin{pmatrix}
x' \\
y' \\
z' \\
0
\end{pmatrix} = \begin{pmatrix}
a & b & c & d \\
e & f & g & h \\
i & j & k & l \\
0 & 0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
x \\
y \\
z \\
0
\end{pmatrix} = \begin{pmatrix}
ax+by+cz \\
ex+fy+gz \\
iy+jz+kz \\
0
\end{pmatrix}
\]

Homogeneous Coordinates: \((x,y,z,w)\)

\(w = 1\) is a point at infinity (direction)

\(w = 0\) is a point at infinity (direction)

• If you do not store \(w\) you need different routines to apply \(M\) to a point and to a direction ==> Store everything in 4D!
Recap: How to Transform Normals?

World Space

Object Space

$n_{WS}$

$n_{OS}$
Transformation for shear and scale

Incorrect Normal Transformation

Correct Normal Transformation
Transformation for shear and scale

Incorrect Normal Transformation

Correct Normal Transformation
So how do we do it right?

- Think about transforming the tangent plane to the normal, not the normal vector.

Pick any vector $v_{OS}$ in the tangent plane, how is it transformed by matrix $M$?

$$v_{WS} = M v_{OS}$$
Transform tangent vector $\nu$

$\nu$ is perpendicular to normal $n$:

Dot product

$$n_{OS}^T \nu_{OS} = 0$$

$$n_{OS}^T (M^{-1} M) \nu_{OS} = 0$$

$$(n_{OS}^T M^{-1}) (M \nu_{OS}) = 0$$

$$(n_{OS}^T M^{-1}) \nu_{WS} = 0$$

$\nu_{WS}$ is perpendicular to normal $n_{WS}$:

$$n_{WS}^T = n_{OS}^T (M^{-1})$$

$$n_{WS} = (M^{-1})^T n_{OS}$$

$$n_{WS}^T \nu_{WS} = 0$$
Position, direction, normal

• Position
  – transformed by the full homogeneous matrix \( M \)

• Direction
  – transformed by \( M \) except the translation component

• Normal
  – transformed by \( M^{-T} \), no translation component
Questions?

- Further reading
  - *Realistic Ray Tracing, 2nd ed.* (Shirley, Morley)