XNA Mathematics Class

305890
Spring 2011
3/28/2011
Kyoung Shin Park

XNA Math & D3DXMath

XNAMath Type	D3DXMath Type
HALF	D3DXFLOAT16
XMFLOAT2	D3DXVECTOR2
XMHALF2	D3DXVECTOR2_16F
XMFLOAT3	D3DXVECTOR3
XMFLOAT4	D3DXVECTOR4
XMHALF4	D3DXVECTOR4_16F
XMMATRIX (or XMFLOAT4x4A)	D3DXMATRIXA16
XMVECTOR (or XMFLOAT4)	D3DXQUATERNION D3DXPLANE D3DXCOLOR

XNA Math & D3DXMath

XNAMath Macro	D3DXMath Macro
XM_PI	D3DX_PI
XM_1DIVPI	D3DX_1BYPI
XMConvertToRadians	D3DXToRadian
XMConvertToDegrees	D3DXToDegree

XNA Math & D3DXMath

XNAMath Function	D3DXMath Function
XMVector3Length	D3DXVec3Length
XMVector3Dot	D3DXVec3Dot
XMVector3Cross	D3DXVec3Cross
XMVectorAdd	D3DXVec2Add D3DXVec3Add
XMVectorSubtract	D3DXVec2Subtract D3DXVec3Subtract
XMVectorMin	D3DXVec2Minimize D3DXVec3Minimize
XMVectorMax	D3DXVec2Maximize D3DXVec3Maximize
…….	…….

Vector – A Mathematical Definition

- Definition
  - A vector is a list of numbers
  - A vector is an array of numbers
- Vectors vs. Scalars
  - Scalar is not a vector quantity
  - Vector quantity: velocity, displacement
  - Scalar quantity: speed, distance
- Vector Dimension
  - Tell how many numbers the vector contains

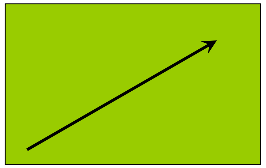
Vector – A Mathematical Definition

- Notation
  - Surround the list of numbers with square brackets, e.g.

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Vector – A Geometric Definition

- Geometrically speaking
  - A vector is a directed line segment that has **magnitude** and **direction**
  - The magnitude of a vector is the length of the vector
  - The direction of a vector describes which way the vector is pointing in space



A 2D vector

Position vs. Displacement

- Vectors do not have position
- ***Only magnitude and direction***
- Example:
  - Displacement – e.g. Take three steps forward
  - Velocity – e.g. Traveling northeast at 50 MPH

Vector

- XMVECTOR (xnamath.h)
  - A portable type used to represent a vector of four 32-bit floating-point or integer components, each aligned optimally and mapped to a hardware vector register.
  - Instances of XMVECTOR can be stored into an instance of XMFLOAT4 with XMStoreFloat4.

```
typedef struct _XMFLOAT4 {
   FLOAT x;
   FLOAT y;
   FLOAT z;
   FLOAT w;
} XMFLOAT4;
```

Vector

- XMFLOAT2, XMFLOAT3 (xnamath.h)

```
typedef struct _XMFLOAT2 {
   FLOAT x;
   FLOAT y;
} XMFLOAT2;

typedef struct _XMFLOAT3 {
   FLOAT x;
   FLOAT y;
   FLOAT z;
} XMFLOAT3;
```

Vector

- XNA Framework Math provides Vector2, Vector3, Vector4 class
  - **Vector2** (x, y)
    - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector2_members(v=XNAGameStudio.30).aspx
  - **Vector3** (x, y, z)
    - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector3_members(v=XNAGameStudio.30).aspx
  - **Vector4** (x, y, z, w)
    - http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector4_members(v=XNAGameStudio.30).aspx

3D Vector Operations

- Equality

```
Vector3 u(1.0f, 0.0f, 1.0f);
Vector3 v(0.0f, 1.0f, 0.0f);
if (u == v) return true;          // equal
if (u != v) return true;          // not equal
```

- Length(Magnitude) $\|\vec{v}\|$

```
Vector3 v(1.0f, 2.0f, 3.0f);
float magnitude = v.Length(); // =sqrt(14)
```

- Normalize $\frac{\vec{v}}{\|\vec{v}\|}$

```
Vector3 v(1.0f, 2.0f, 3.0f);
v = v.Normalize();    // After this line executes, vector will be unit-length
```

3D Vector Operations

- Addition: u + v

```
Vector3 u(2.0f, 0.0f, 1.0f);
Vector3 v(0.0f, -1.0f, 5.0f);
Vector3 sum = u + v;// (2.0+0.0, 0.0-1.0, 1.0+5.0) = (2.0, -1.0, 6.0)
```

- Subtraction: u - v

```
Vector3 u(2.0f, 0.0f, 1.0f);
Vector3 v(0.0f, -1.0f, 5.0f);
Vector3 diff = u - v;   // (2.0-0.0, 0.0+1.0, 1.0-5.0) = (2.0, 1.0, -4.0)
```

- Scalar multiplication: u * *k*

```
Vector3 u(2.0f, 0.0f, -1.0f);
Vector3 scaleVec = u * 10.0f;  // (2.0, 0.0, -1.0) *10.0 = (20.0, 0.0, -10.0)
```

3D Vector Operations

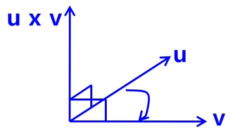
- Dot product

```
Vector3 u(1.0f, -1.0f, 0.0f);
Vector3 v(3.0f, 2.0f, 1.0f);
float out = Vector3.Dot(u, v); // 1.0*3.0 + -1.0*2.0 + 0.0*1.0 = 1.0
```

$$\vec{u}\cdot\vec{v} = \|\vec{u}\|\|\vec{v}\|\cos\theta = u_x v_x + u_y v_y + u_z v_z$$

- Cross product
  - u x v = -(v x u)

```
Vector3 u(1.0f, -1.0f, 0.0f);
Vector3 v(3.0f, 2.0f, 1.0f);
Vector3 out = Vector3.Cross(u, v);
```



$$\vec{u}\times\vec{v} = (u_y v_z - u_z v_y, u_z v_x - u_x v_z, u_x v_y - u_y v_x)$$

3D Vector Operations

- Zero

```
Vector3 v = Vector3.Zero;       // v=Vector3(0, 0, 0)
```

- Forward

```
Vector3 v= Vector3.Forward; // v=Vector3(0, 0, -1) RHS
```

- Right

```
Vector3 v= Vector3.Right;      // v=Vector3(1, 0, 0) RHS
```

- Up

```
Vector3 v= Vector3.Up;         // v=Vector3(0, 1, 0) RHS
```

- UnitX

```
Vector3 v= Vector3.UnitX;      // v=Vector3(1, 0, 0)
```

- UnitY

```
Vector3 v= Vector3.UnitY;      // v=Vector3(0, 1, 0)
```

- UnitZ

```
Vector3 v= Vector3.UnitZ;      // v=Vector3(0, 0, 1)
```

Matrix

- A rectangular grid of numbers arranged into *rows* and *columns*.
- Vector vs. Matrix
  - Vector is an array of scalars
  - Matrix is an array of vectors
- Dimensions and Notation : ***r*** x ***c*** matrix

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

r(3) rows

c(3) columns

XMMatrix

- XMMATRIX (xnamath.h)
  - $v' = v_{1x4} T_{4x4}$ (not $T_{4x4} v_{1x4}$)

- XMMATRIX
  - _ij element = ith row number, jth column number

```
typedef struct _XMMATRIX {
    union {
      XMVECTOR r[4];
      struct {
         FLOAT _11;   FLOAT _12;   FLOAT _13;   FLOAT _14;
         FLOAT _21;   FLOAT _22;   FLOAT _23;   FLOAT _24;
         FLOAT _31;   FLOAT _32;   FLOAT _33;   FLOAT _34;
         FLOAT _41;   FLOAT _42;   FLOAT _43;   FLOAT _44; // translation(x,y,z)
      };
      FLOAT m[4][4];
    };
} XMMATRIX;
```

Matrix Operations

- Matrix arithmetic operation: ==, +, -, *, /

```
Matrix A(1,0,0,0,
         0,1,0,0,
         0,0,1,0,
         1,2,3,1);            // A의 초기화
Matrix B(....);       // B의 초기화
Matrix C = A * B;     // C = AB
```

- Matrix element access

```
Matrix m;
m.M11 = 5.0f;          // _11 = 5.0f
m.M12 = 6.0f;          // _12 = 5.0f
```

Matrix Operations

- Transpose matrix M^T

```
Matrix A(....);           // A 초기화
Matrix B;
B = Matrix.Transpose(A);       // B = transpose(A)
```

- Inverse matrix M^{-1}

```
Matrix A(....);           // A 초기화
Matrix B;
B = Matrix.Invert(A);             // B = inverse(A)
float det = A.Determinant();      // Determinant of matrix A
```

- Identity matrix I

```
Matrix m = Matrix.Identity;    // identity matrix
```

Matrix Operations

- Translation matrix

```
Vector3 translationVector(1, 2, 3);
Matrix A = Matrix.CreateTranslation(translationVector);
```

- Rotation matrix

```
Matrix ROTX = Matrix.CreateRotationX(MathHelper.ToRadians(60)); // 60°
Matrix ROTY = Matrix.CreateRotationY(MathHelper.Pi);            // 180°
Matrix ROTZ = Matrix.CreateRotationZ(MathHelper.PiOver4);       // 45°
```

- Scale matrix

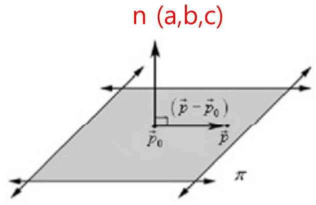
```
Vector3 scaleVector(2, 2, 2);
Matrix S = Matrix.CreateScale(scaleVector);
Matrix SS = Matrix.CreateScale(2, 2, 2);
```

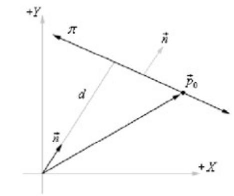
Plane

- XMMath Library Plane Functions
  - XMPlaneDot
  - XMPlaneDotCoord
  - XMPlaneDotNormal
  - XMPlaneEqual
  - XMPlaneFromPointNormal
  - XMPlaneFromPoints
  - XMPlaneIntersectLine
  - XMPlaneIntersectPlane
  - XMPlaneIsInfinite
  - XMPlaneIsNaN
  - XMPlaneNearEqual
  - XMPlaneNormalize
  - XMPlaneNormalizeEst
  - XMPlaneNotEqual
  - XMPlaneTransform
  - XMPlaneTransformStream

Plane

- A plane is defined by **a normal vector, n,** and **a point on the plane, p_0**:
  - ax + by + cz + d = 0
  - n•p + d = 0
    n = (a, b, c), p=[n, d]
    d = -n•p (the shortest signed distance)
  - n•(p - p_0) =0
    p_0 is a point on the plane





Plane Construction

- Normal vector, n, and the signed distance, d:
  - Plane p(a, b, c, d)

- Normal vector, n, and a point on the plane, p_0:
  - d = -n•p_0

```
Vector3 n(a, b, c);
Plane p(n, d);
```

- Three points on the plane, p_0, p_1, p_2:
  - u = p_1 - p_0; v = p_2 - p_0 ; n = u x v; d = -n•p_0

```
Vector3 p0, p1, p2;
Plane p(p0, p1, p2);
```

Plane Normalization

- Normalizing a Plane
  - Normalize the plane normal vector. But, recall the normal vector influences the constant, d.
  - Therefore, if we normalize the normal vector, we must also recalculate d.

$$\frac{1}{\|\mathrm{n}\|}(\mathrm{n},\mathrm{d}) = \left(\frac{n}{\|n\|}, \frac{d}{\|n\|}\right)$$

```
Vector3 n(a, b, c);
Plane p(n, d);
p.Normalize();
```

Point and Plane Spatial Relationship

- Point and Plane Spatial Relationship
  - If n•p + d = 0, then p is planar with the plane.
  - If n•p + d > 0, then p is in front of the plane and in the plane's positive half space.
  - If n•p + d < 0, then p is back of the plane and in the plane's negative half-space.
- PlaneDotCoord
  - Plane(a, b, c, d), Vector3(x, y, z), **a*x + b*y + c*z + d*1**

```
Plane p(0.0, 1.0, 0.0, 0.0); Vector3 v(3.0, 5.0, 2.0);
float x = p.DotCoord(v);
if (x approximately equal 0.0) // coplanar to the plane
if (x > 0) // in positive half-space
if (x < 0) // in negative half-space
```

```
-Approximately equal
const float EPSILON = 0.001f;
boolean Equals (float lhs, float rhs) { return fabs(lhs – rhs) < EPSILON ? true : false;}
```

Relationship between Point and Plane

- XMPlaneDot
  - Plane(a, b, c, d), Vector4(x, y, z, w), **a*x + b*y + c*z + d*w**

```
XMVECTOR XMPlaneDot(XMVECTOR P, XMVECTOR V);
```

- XMPlaneDotCoord
  - Plane(a, b, c, d), Vector3(x, y, z), **a*x + b*y + c*z + d*1**

```
XMVECTOR XMPlaneDotCoord(XMVECTOR P, XMVECTOR V);
```

- XMPlaneDotNormal
  - Plane(a, b, c, d), Vector3(x, y, z), **a*x + b*y + c*z + d*0**

```
XMVECTOR XMPlaneDotNormal(XMVECTOR P, XMVECTOR V);
```


Plane Transformation

- Plane Transformation
  - Transform a normalized plane, p=(n, d) by a matrix (or a quaternion): $p(T^{-1})^T$

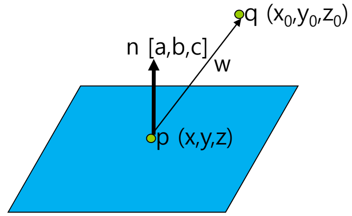
```
// rendering simple water reflection
Vector3 planeWaterNormal = new Vector3(0, 1, 0);
Plane planeWater = new Plane(planeWaterNormal, 1);
planeWater.Normalize();
planeWater = Plane.Transform(planeWater, camera.View);
planeWater = Plane.Transform(planeWater, camera.Projection);
this.GraphicsDevice.ClipPlanes[0].Plane = planeWater;
this.GraphicsDevice.ClipPlanes[0].IsEnabled = true;
```


Distance from a Point to a Plane

- **The shortest distance, D**, from a point, $q(x_0, y_0, z_0)$, to the plane, P(n, d):
  - Point, q, lies in the plane if and only if D=0

$$w = [x_0 - x, y_0 - y, z_0 - z]$$

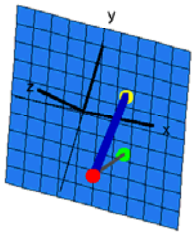
$$D = \frac{|n \bullet w|}{\|n\|} = \frac{|a(x_0 - x) + b(y_0 - y) + c(z_0 - z)|}{\sqrt{a^2 + b^2 + c^2}} = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}$$

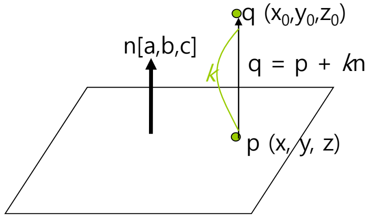


Projecting w onto n: $w_{\parallel} = n\frac{w \cdot n}{\|n\|^2}$ & $\|w_{\parallel}\| = \frac{|w \cdot n|}{\|n\|}$

Nearest Point on a Plane to a Particular Point

- Find a nearest point, **p(x, y, z)**, on the plane, P(n,d) to a particular point, q(x_0, y_0, z_0):
  - p = q – *k*n (*k*=the shortest signed distance from a point, q, to the plane, P)
  - If n is the unit vector, then *k* = n•q + d
  - **p = q – (n•q + d)n**





$$\text{Distance(q, plane)} = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}$$

$$where\ q(x_0, y_0, z_0)\ \ and\ \ Plane\ ax + by + cz + d = 0$$

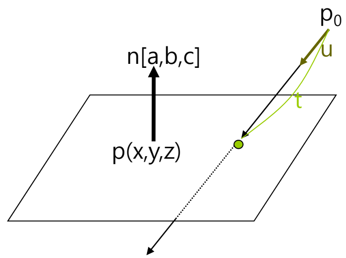
Intersection of Ray and Plane

- Ray: **p**(*t*) = **p**$_0$ + *t***u**
- Plane: p•n + d = 0
- Ray/Plane Intersection:

$$(p_0 + tu) \cdot n + d = 0$$

$$tu \cdot n = -d - p_0 \cdot n$$

$$t = \frac{-(p_0 \cdot n + d)}{u \cdot n}$$



- No intersection, if a ray is parallel to the plane, i.e., **u•n**=0
- No intersection, if *t* is not in [0, ∞), i.e., **t<0**
- When intersected,

$$p\left(\frac{-(p_0 \cdot n + d)}{u \cdot n}\right) = p_0 + \frac{-(p_0 \cdot n + d)}{u \cdot n} u$$

Rays, Lines, and Line Segments

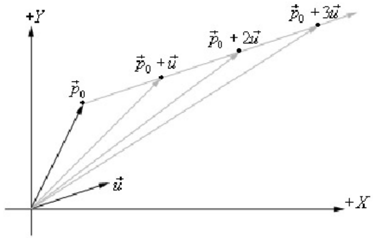
- Ray :

  p(*t*) = **p**$_0$ + *t***u** where p_0 is the origin of the ray, t∈ [0, ∞)

  u is a vector specifying the direction of the ray
- Line :

  p(*t*) = **p**$_0$ + *t***u** where t ∈ [-∞, ∞)
- Line segment:

  p(*t*) = **p**$_0$ + *t***u** where u = **p**$_1$ – **p**$_0$, t ∈ [0, 1]



Reference

- XNA Framework Math Overview (XNA Game Studio 3.0) http://msdn.microsoft.com/en-us/library/bb203910(v=XNAGameStudio.30).aspx
- http://www.math.umn.edu/~nykamp/m2374/readings/planedist/