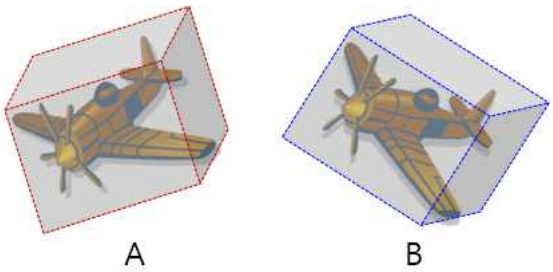
1. 3D 그래픽 기초(3D Graphics Fundamentals)

(9) DirectXMath 충돌 검사(Collision Detection)

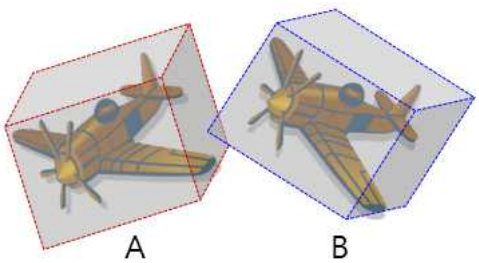
① 게임 객체의 교차(Game Object Intersection) 판정

게임 객체들의 교차(충돌)를 판정하는 여러 가지 방법이 있다. 가장 쉽게 생각할 수 있는 방법은 게임 객체를 구성하는 메쉬의 다각형(프리미티브)을 사용하여 직접 검사하는 것이다. 예를 들어, 광선과 게임 객체의 교차를 검사할 때, 광선이 게임 객체의 메쉬를 구성하는 모든 다각형들과 교차하는 가를 검사하는 것이다. 그러나 이 방법은 모든 점들 또는 다각형을 사용하여 직접 충돌(교차)을 검사하기 때문에 다각형이 많은 경우 계산 시간이 너무 많이 걸릴 수 있다.

일반적으로 게임 객체를 완전히 포함하는 간단한(근사하는) 입체(바운딩 객체: Bounding object)를 사용하여 충돌 검사를 한다. 바운딩 객체가 게임 객체를 포함하기 때문에, 다음 그림과 같이 게임 객체 A의 바운딩 객체와 게임 객체 B의 바운딩 객체와 교차하지 않으면 게임 객체 A와 게임 객체 B는 교차(충돌)할 수 없기 때문이다.



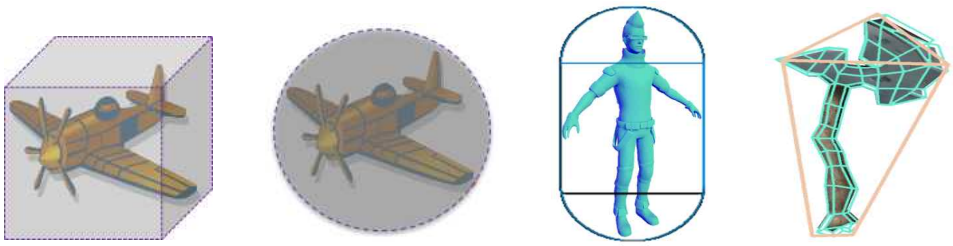
바운딩 객체를 사용한 충돌 검사는 충돌을 판정하는 시간이 적게 걸리는 장점이 있지만 충돌의 정확도가 떨어질 수 있다(False alarm). 다음 그림에서 게임 객체 A의 바운딩 객체와 게임 객체 B의 바운딩 객체는 교차하지만 게임 객체 A와 게임 객체 B는 교차하지 않는다.



일반적으로 게임 프로그램에서 많이 사용하는 바운딩 객체는 다음과 같이 큐브(Cube), 구(Sphere), 캡슐(Capsule), 원기둥(Cylinder), 메쉬 등이다. 어떤 바운딩 객체를 사용하는

가는 게임 객체의 모양, 충돌 검사의 시간, 충돌 검사의 정확도를 고려해야 한다.

- ▫ 바운딩 큐브(Bounding Cube, 바운딩 박스: Bounding Box)
  게임 객체를 둘러싸는 최소 외접 직육면체이다.
- ▫ 바운딩 구(Bounding Sphere)
  게임 객체를 둘러싸는 최소 외접 구이다.
- ▫ 바운딩 캡슐(Bounding Capsule)
  게임 객체를 둘러싸는 최소 외접 캡슐이다.
- ▫ 바운딩 메쉬(Bounding Mesh)
  게임 객체의 메쉬의 다각형을 합쳐서 아주 적은 개수의 다각형으로 메쉬의 모양을 유지하도록 재구성한 메쉬이다.



- ▪ 다단계(계층적) 검사 방법
  충돌 검사를 여러 단계로 구성하여 수행하는 방법이다.
  ❶ 간단한 바운딩 객체를 사용한 충돌 검사를 수행한다. 바운딩 객체가 충돌이 되지 않으면 게임 객체는 충돌하지 않은 것이다.
  ❷ 바운딩 객체가 충돌하면 게임 객체의 메쉬를 사용하여 세밀한 충돌 검사를 수행한다.

게임 객체들의 충돌 검사를 위하여 동일한 유형의 바운딩 객체들 사이의 교차 판정을 많이 사용한다. 픽킹(Picking) 또는 게임 객체의 가시성(Visibility) 판단을 위하여 광선(Ray)과 바운딩 객체의 교차 판정을 사용한다. 평면(Plane)과 바운딩 객체의 교차 판정은 절두체 컬링(Frustum Culling) 또는 게임 객체의 벽체 충돌 판정에 사용한다.

② 바운딩 박스(Bounding Box)

게임 객체를 둘러싸는 최소 외접 직육면체를 바운딩 객체로 사용한다. 바운딩 박스는 축-정렬 바운딩 박스(Axis-Aligned Bounding Box: AABB)와 객체-정렬 바운딩 박스(Object-Oriented Bounding Box: OOBB)로 나뉘어 진다.

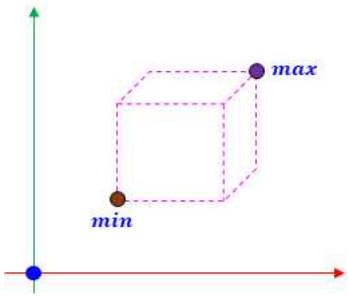
바운딩 박스의 충돌(교차) 검사 알고리즘으로 유명한 분리축 정리(SAT: Separating Axis Theorem)를 사용할 수 있다. 이 알고리즘에 대한 설명은 생략하고 바운딩 박스에 대한 개념만을 이해하도록 하자.

- 축-정렬 바운딩 박스(Axis-Aligned Bounding Box: AABB)

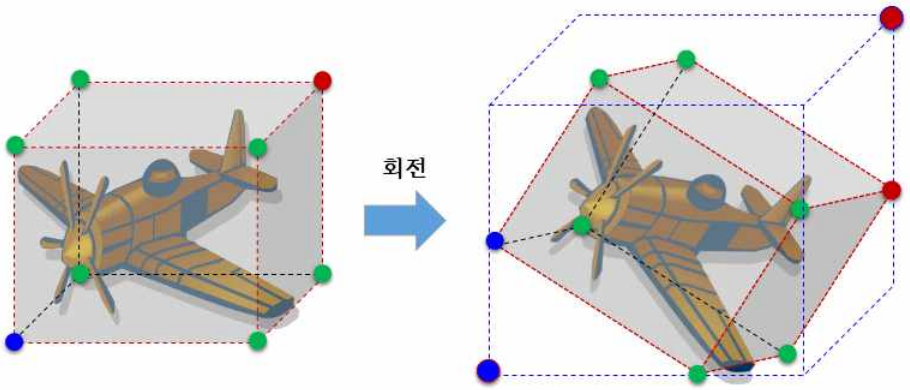
  AABB 바운딩 박스는 월드 좌표계의 각 축에 정렬된(평행한) 변을 갖는 바운딩 박스(직육면체)이다. 다음 그림과 같이 바운딩 박스(직육면체)의 각 변(지역 좌표계의 x-축, y-축, z-축)은 월드 좌표계의 x-축, y-축, z-축과 항상 정렬된다(방향이 같다). 메쉬를 구성하는 모든 정점(월드 좌표계)에 x-좌표의 최소(a)와 최대(A), y-좌표의 최소(b)와 최대(B), z-좌표의 최소(c)와 최대(C)를 구하면 AABB 바운딩 박스를 생성할 수 있다. 벡터 (a, b, c)를 AABB 바운딩 박스의 최소점(Min Vector, $\boldsymbol{min}$ 벡터)이라 하고, 벡터 (A, B, C)를 최대점(Max Vector, $\boldsymbol{max}$ 벡터)이라 한다. 바운딩 박스의 중심점(Center) $\boldsymbol{C}$와 대각선(Extent) 벡터 $\boldsymbol{e}$는 다음과 같이 구할 수 있다.

$$\boldsymbol{C} = \frac{1}{2}(\boldsymbol{max} + \boldsymbol{min})$$

$$\boldsymbol{e} = \frac{1}{2}(\boldsymbol{max} - \boldsymbol{min})$$

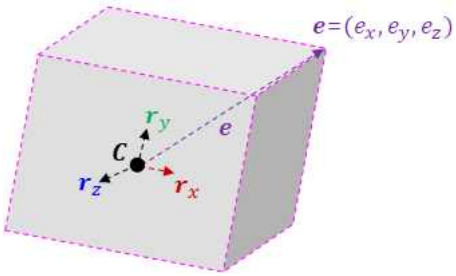


AABB 바운딩 박스는 기본적으로 게임 객체의 메쉬에서 생성할 수 있다. 메쉬의 정점들은 모델 좌표계로 표현되므로 메쉬의 정점에서 생성한 AABB 바운딩 박스는 모델 좌표계로 표현되며, 충돌 검사를 하기 위한 게임 객체의 바운딩 박스는 월드 좌표계로 표현된 바운딩 박스이다. 게임 객체가 이동하면 AABB 모델 바운딩 박스의 최소점과 최대점을 게임 객체가 이동한 만큼 이동하면 AABB 바운딩 박스를 구할 수 있다. 게임 객체의 크기가 변하면 AABB 바운딩 박스의 최소점과 최대점을 크기 변환하면 된다. 그러나 게임 객체가 회전하면 AABB 바운딩 박스의 최소점과 최대점을 다시 계산해야 한다. 게임 객체가 회전을 하더라도 그 게임 객체의 AABB 바운딩 박스는 회전하지 않는다(바운딩 박스가 회전하면 월드 좌표계의 축과 바운딩 박스가 정렬되지 않으므로).



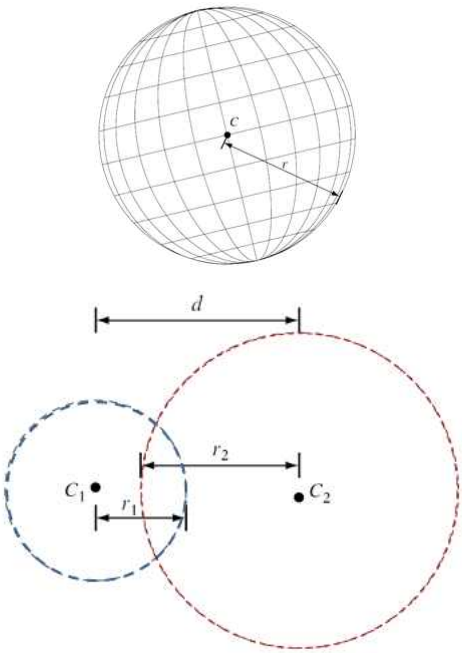
▪ 객체-정렬 바운딩 박스(Object-Oriented Bounding Box: OOBB)

OOBB 바운딩 박스는 객체의 로컬 좌표축에 정렬된(평행한) 변을 갖는 바운딩 박스(직육면체)이다. OOBB 바운딩 박스는 중심(Center) $\boldsymbol{C}$, 대각선 벡터 $\boldsymbol{e}$, 방향 벡터 $\boldsymbol{r}_x$, $\boldsymbol{r}_y$, $\boldsymbol{r}_z$로 표현할 수 있다. 게임 객체의 회전에 따라 바운딩 박스의 방향 벡터 $\boldsymbol{r}_x$, $\boldsymbol{r}_y$, $\boldsymbol{r}_z$을 회전시킨다.



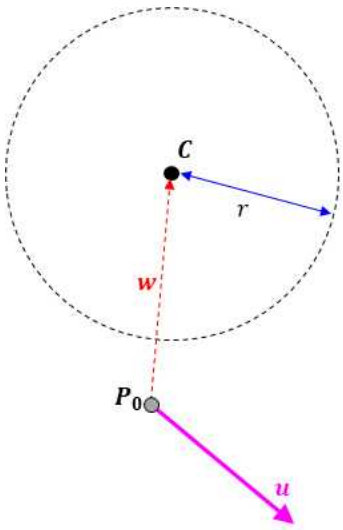
③ 바운딩 구(Bounding Sphere)

바운딩 구는 중심 $\boldsymbol{C}$와 반지름(r)로 표현할 수 있다. 바운딩 구를 사용한 게임 객체의 충돌 검사는 가장 쉽고 빠르다. 바운딩 구와 바운딩 구의 교차는 구의 중심 사이의 거리와 반지름의 합을 비교하여 계산할 수 있다.

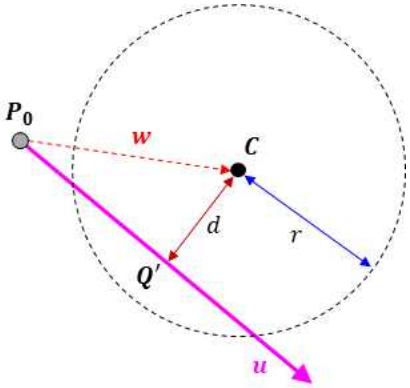


- 구와 광선의 교차

  구의 중심 $\boldsymbol{C}=(C_x, C_y, C_z)$에서 광선까지 거리 d를 계산하여 구의 반지름(r)과 비교를 한다. 광선의 시작점이 $\boldsymbol{P}_0$이고 방향 벡터가 $\boldsymbol{u}$, $\boldsymbol{w}=(\boldsymbol{C}-\boldsymbol{P}_0)$일 때, $(\boldsymbol{w}\bullet\boldsymbol{u}<0)$이고 $(\boldsymbol{w}\bullet\boldsymbol{w}>r^2)$이면 다음 그림과 같이 구가 광선 뒤에 있다(교차하지 않는다).



  $\{(\boldsymbol{u}\bullet\boldsymbol{u})(\boldsymbol{w}\bullet\boldsymbol{w})-(\boldsymbol{w}\bullet\boldsymbol{u})^2\}\le(\boldsymbol{u}\bullet\boldsymbol{u})r^2$이면 구와 광선은 교차한다.



$$Q' = P_0 + \frac{w \cdot u}{|u|^2} u$$

$$d^2 = (w \cdot w) - \left| \frac{(w \cdot u)}{(u \cdot u)} \right|^2 = (w \cdot w) - \left(\frac{w \cdot u}{u \cdot u} \right)^2 (u \cdot u) = (w \cdot w) - \frac{(w \cdot u)^2}{(u \cdot u)}$$

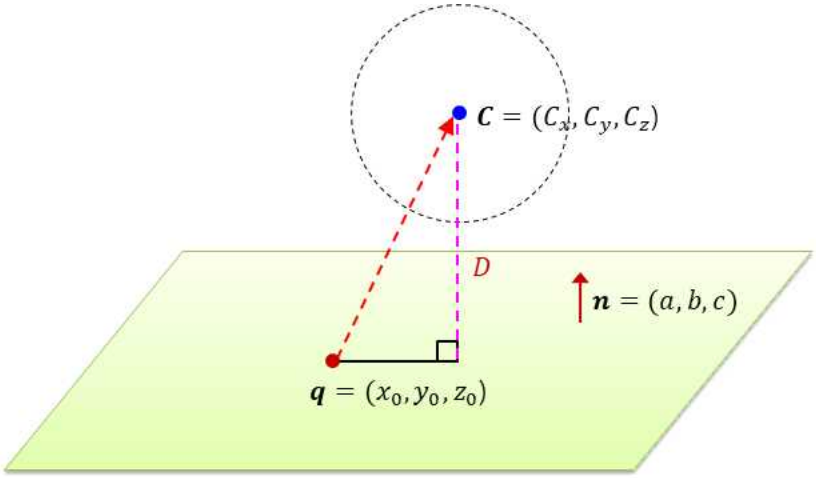
$$d^2 = (w \cdot w) - \frac{(w \cdot u)^2}{(u \cdot u)} \leq r^2$$

$$(u \cdot u)(w \cdot w) - (w \cdot u)^2 \leq (u \cdot u) r^2$$

- 구와 평면의 교차

  구의 중심 $C = (C_x, C_y, C_z)$에서 정규화된 평면 $ax + by + cz + d = 0$까지 거리 D를 계산하여 구의 반지름(r)과 비교를 한다. ($D > r$)이면 구가 평면 앞에 있다. ($D < -r$)이면 구가 평면 뒤에 있다. ($-r \leq D \leq r$)이면 구와 평면은 교차한다.

$$D = aC_x + bC_y + cC_z + d$$



④ DirectXMath 충돌(Collision) 구조체

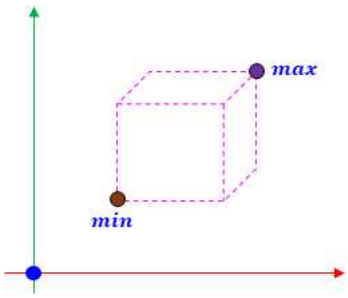
DirectXMath 수학 라이브러리는 "DirectXCollision.h"와 "DirectXCollision.inl" 파일을

통하여 아주 유용한 충돌 구조체와 함수들을 제공한다.

- BoundingBox
- BoundingOrientedBox
- BoundingSphere
- BoundingFrustum

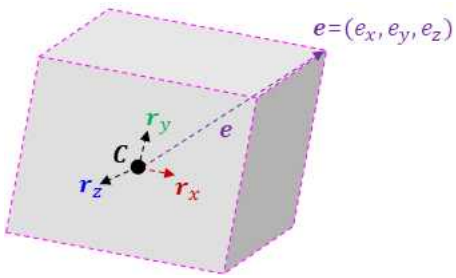
BoundingBox 구조체는 다음과 같이 바운딩 박스의 중심이 Center이고 대각선이 Extents인 AABB 바운딩 박스이다.

```
struct BoundingBox {
   XMFLOAT3 Center;
   XMFLOAT3 Extents;
   size_t CORNER_COUNT = 8;
   ...
};
```



BoundingOrientedBox 구조체는 다음과 같이 바운딩 박스의 중심이 Center, 대각선이 Extents, 방향이 Orientation인 OOBB 바운딩 박스이다. Orientation은 쿼터니언(Quaternion)으로 표현된다. 쿼터니언은 회전축이 $\boldsymbol{n}=(a,b,c)$이고 회전 각도가 θ인 회전을 나타내는 4차원 벡터 $\boldsymbol{q}=(x,y,z,w)$이다.

$$\boldsymbol{q}=(x,y,z,w)=\left(sin\left(\frac{\theta}{2}\right)a,\ sin\left(\frac{\theta}{2}\right)b,\ sin\left(\frac{\theta}{2}\right)c,\ cos\left(\frac{\theta}{2}\right)\right)$$

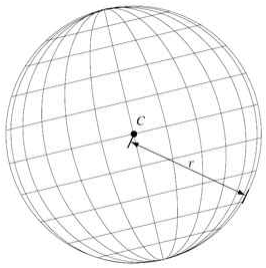



```
struct BoundingOrientedBox {
   XMFLOAT3 Center;
   XMFLOAT3 Extents;
   XMFLOAT4 Orientation;
   size_t CORNER_COUNT = 8;
   ...
};
```

BoundingSphere 구조체는 다음과 같이 구의 중심이 Center이고 반지름이 Radius인

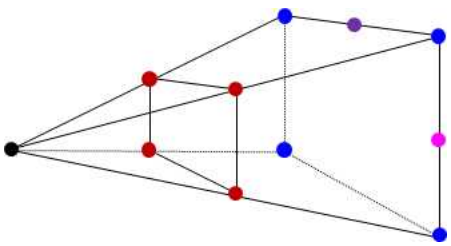
바운딩 구이다.

```
struct BoundingSphere {
   XMFLOAT3 Center;
   float Radius;
   ...
};
```



BoundingFrustum 구조체는 다음과 같이 원점이 Origin, 방향이 Orientation, 근평면까지 거리가 Near, 원평면까지 거리가 Far, 왼쪽 평면 기울기가 LeftSlope, 오른쪽 평면 기울기가 RightSlope, 위쪽 평면 기울기가 TopSlope, 아래쪽 평면 기울기가 BottomSlope인 절두체(육면체)를 나타내는 바운딩 객체이다.

```
struct BoundingFrustum {
   XMFLOAT3 Origin;
   XMFLOAT4 Orientation;
   float Near;
   float Far;
   float LeftSlope;
   float RightSlope;
   float TopSlope;
   float BottomSlope;
   size_t CORNER_COUNT = 8;
   ...
};
```



⑤ DirectXMath 삼각형(Triangle) 충돌 검사

삼각형과 다른 바운딩 객체와의 교차를 검사할 수 있는 다음과 같은 전역 함수를 제공한다. 이 함수들은 “TriangleTests” 네임스페이스를 통하여 제공된다. 함수를 사용하려면 “stdafx.h” 파일에 “using namespace TriangleTests;”를 추가하거나 TriangleTests::함수()의 형태로 사용해야 한다.

- `ContainmentType ContainedBy(...);`
- `PlaneIntersectionType Intersects(...);`

▫ bool Intersects(...);

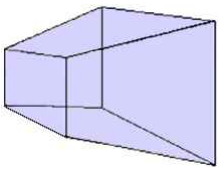
자료형 ContainmentType의 “DISJOINT”는 바운딩 객체들 사이의 교차가 없는(만나지 않는) 것이고, “CONTAINS”는 하나의 바운딩 객체가 다른 바운딩 객체를 완전히 포함하는 것이며, “INTERSECTS”는 바운딩 객체들의 부분끼리 교차하는(만나는) 것을 의미한다.

```
enum ContainmentType {
   DISJOINT,
   INTERSECTS,
   CONTAINS
};
```

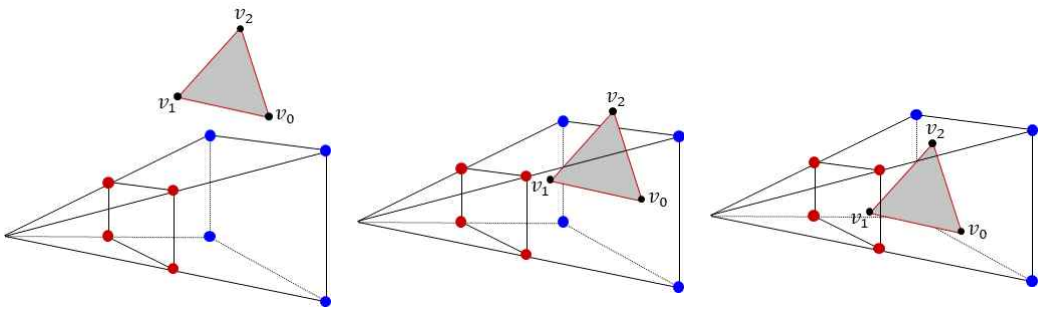
자료형 PlaneIntersectionType은 평면과 바운딩 객체의 위치 관계를 표현한다. “FRONT”는 바운딩 객체가 평면 앞에 있는 것이고, “BACK”은 바운딩 객체가 평면 뒤에 있는 것이며, “INTERSECTING”는 바운딩 객체가 평면과 교차하는(만나는) 것을 의미한다.

```
enum PlaneIntersectionType {
   FRONT,
   INTERSECTING,
   BACK
};
```

ContainedBy() 함수는 삼각형(세 꼭지점 v0, v1, v2)이 절두체(6개 평면 plane0,..., plane5)에 포함되는 가를 검사한다. 검사의 결과는 삼각형과 절두체(Frustum)의 위치 관계를 나타내는 ContainmentType이다. 절두체는 6개의 사각형(직사각형이 아님)으로 표현할 수 있는 입체임에 유의하라.



```
ContainmentType ContainedBy(XMVECTOR v0, XMVECTOR v1, XMVECTOR v2,
XMVECTOR plane0, XMVECTOR plane1, XMVECTOR plane2, XMVECTOR plane3,
XMVECTOR plane4, XMVECTOR plane5);
```

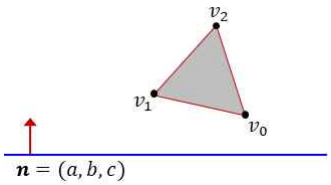


DISJOINT　　INTERSECTS　　CONTAINS

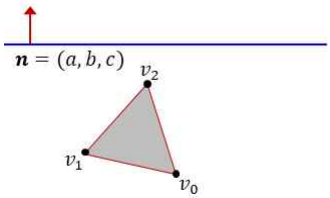
다음 Intersects() 함수는 삼각형(세 꼭지점 v0, v1, v2)과 평면(plane)의 교차를 검사한

다. 검사의 결과는 삼각형과 평면의 위치 관계를 나타내는 PlaneIntersectionType이다.

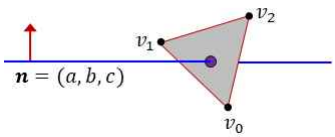
```
PlaneIntersectionType Intersects(XMVECTOR v0, XMVECTOR v1, XMVECTOR
v2, XMVECTOR plane);
```



FRONT



BACK

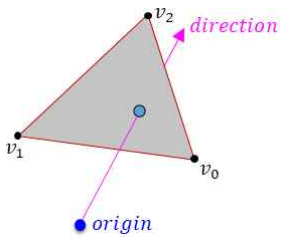


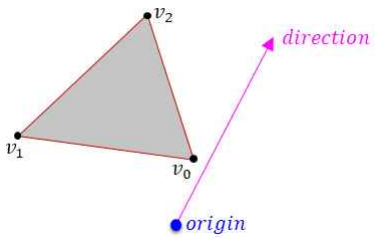
INTERSECTING

```
XMVECTOR v0, v1, v2, plane;
…
PlaneIntersectionType result = TriangleTests::Intersects(v0, v1, v2,
plane);
```

다음 Intersects() 함수는 삼각형(세 꼭지점 v0, v1, v2)과 광선(시작점 origin, 방향 direction)의 교차를 검사한다. 광선이 삼각형과 교차하면 참(true)을 반환하고, 광선의 시작점에서 교점까지의 거리를 distance로 넘겨준다.

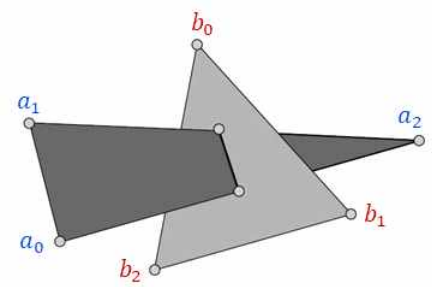
```
bool Intersects(XMVECTOR origin, XMVECTOR direction, XMVECTOR v0,
XMVECTOR v1, XMVECTOR v2, float &distance);
```





다음 Intersects() 함수는 삼각형(세 꼭지점 a0, a1, a2)과 삼각형(세 꼭지점 b0, b1, b2)의 교차를 검사한다.

```
bool Intersects(XMVECTOR a0, XMVECTOR a1, XMVECTOR a2, XMVECTOR b0,
XMVECTOR b1, XMVECTOR b2);
```



⑥ BoundingBox 구조체 충돌 검사 함수

BoundingBox 구조체는 바운딩 박스의 중심이 Center이고 대각선이 Extents인 AABB 바운딩 박스이며, 다음과 같은 멤버 함수를 제공한다. BoundingBox는 구조체이므로 구조체 객체를 통하여 멤버 함수를 호출함에 주의하라.

- 생성자(Constructor)

```
BoundingBox(BoundingBox &box);
BoundingBox(XMFLOAT3 &center, XMFLOAT3 &extents);
```

다음은 BoundingBox 구조체의 생성자를 사용하는 예이다.

```
XMFLOAT3 center(0.0f, 0.0f, 0.0f);
XMFLOAT3 extents(2.0f, 2.0f, 2.0f);
BoundingBox aabb(center, extents);
XMVECTOR point = XMVectorSet(1.0f, 2.0f, 3.0f);
ContainmentType result = aabb.Contains(point);
```

- 초기화 함수(Initialization function, 생성 함수: Creation function)
  다음은 BoundingBox 구조체를 초기화할 수 있는 정적 함수(Static function)이다.

다음 CreateFromPoints() 멤버 함수는 최소점(min 벡터)과 최대점(max 벡터)로부터 AABB 바운딩 박스를 초기화한다.

```
static void CreateFromPoints(BoundingBox &out, XMVECTOR min,
XMVECTOR max);
```

위의 예를 다음과 같이 작성할 수 있다.

```
XMVECTOR min = XMVectorSet(-2.0f, -2.0f, -2.0f, 1.0f);
XMVECTOR max = XMVectorSet(+2.0f, +2.0f, +2.0f, 1.0f);
BoundingBox aabb;
BoundingBox::CreateFromPoints(aabb, min, max);
XMVECTOR point = XMVectorSet(1.0f, 2.0f, 3.0f);
ContainmentType result = aabb.Contains(point);
```

다음 CreateFromPoints() 멤버 함수는 메쉬의 정점들의 배열(pPoints), 정점의 개수(count), 정점 하나의 크기(바이트 수, stride)로부터 AABB 바운딩 박스를 초기화한다. 정점의 첫 번째 요소는 정점의 위치 벡터라고 가정한다. 정점의 개수가 n, i-번째 정점의 위치 벡터가 $\boldsymbol{p}_i = (x_i, y_i, z_i)$이라고 가정할 때 AABB 바운딩 박스의 중점 $\boldsymbol{C}$는 다음과 같이 계산할 수 있다.

$$\boldsymbol{C} = (\bar{x}, \bar{y}, \bar{z})$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i,\ \ \bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$$

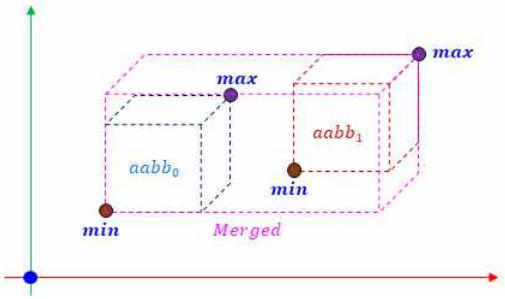
```
static void CreateFromPoints(BoundingBox &out, size_t count,
XMFLOAT3 *pPoints, size_t stride);
```

다음 CreateFromSphere() 멤버 함수는 바운딩 구(sphere)에 외접하는 최소 정육면체로 AABB 바운딩 박스를 초기화한다.

```
static void CreateFromSphere(BoundingBox &out, BoundingSphere
&sphere);
```

다음 CreateMerged() 멤버 함수는 두 개의 바운딩 박스(aabb0, aabb1)를 포함하는 바운딩 박스로 AABB 바운딩 박스를 초기화한다.

```
static void CreateMerged(BoundingBox &out, BoundingBox &aabb0,
BoundingBox &aabb1);
```



- 꼭지점 반환 함수

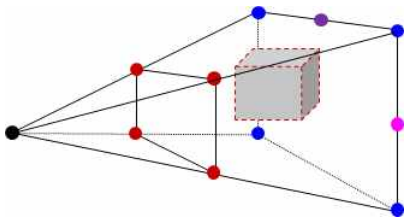
다음 GetCorners() 멤버 함수는 AABB 바운딩 박스의 꼭지점 8개의 벡터(XMFLOAT3)를 반환한다.

```
void GetCorners(XMFLOAT3 *pCorners);
```

- 포함(Contains) 함수

다음 ContainedBy() 멤버 함수는 AABB 바운딩 박스가 평면 6개로 표현되는 절두체에 포함(교차)되는 가를 검사한다. 반환 자료형은 ContainmentType이다.

```
ContainmentType ContainedBy(XMVECTOR plane0, XMVECTOR plane1,
XMVECTOR plane2, XMVECTOR plane3, XMVECTOR plane4, XMVECTOR plane5);
```



다음 Contains() 멤버 함수는 AABB 바운딩 박스가 점(point)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR point);
```

다음 Contains() 멤버 함수는 AABB 바운딩 박스가 삼각형(v0, v1, v2)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR v0, v1, v2);
```

다음 Contains() 멤버 함수는 AABB 바운딩 박스가 바운딩 구(sphere)를 포함(교차)

하는 가를 검사한다.

```
ContainmentType Contains(BoundingSphere &sphere);
```

다음 Contains() 멤버 함수는 AABB 바운딩 박스가 다른 AABB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingBox &box);
```

다음 Contains() 멤버 함수는 AABB 바운딩 박스가 OOBB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingOrientedBox &box);
```

다음 Contains() 멤버 함수는 AABB 바운딩 박스가 절두체(frustum)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingFrustum &frustum);
```

- 교차(Intersects) 함수

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 평면(plane)과 교차하는 가를 검사한다. 반환 자료형은 평면과의 위치 관계를 나타내는 PlaneIntersectionType이다.

```
PlaneIntersectionType Intersects(XMVECTOR plane);
```

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 광선(origin, direction)과 교차하는 가를 검사한다. 광선이 AABB 바운딩 박스와 교차하면 광선의 원점에서 교점까지의 거리를 distance로 전달한다.

```
bool Intersects(XMVECTOR origin, direction, float &distance);
```

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 삼각형(v0, v1, v2)과 교차하는 가를 검사한다.

```
bool Intersects(XMVECTOR v0, v1, v2);
```

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 바운딩 구(sphere)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingSphere &sphere);
```

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 다른 AABB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingBox &box);
```

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 OOBB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingOrientedBox &box);
```

다음 Intersects() 멤버 함수는 AABB 바운딩 박스가 절두체(frustum)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingFrustum &frustum);
```

- 변환(Transform) 함수

함수 Transform()은 AABB 바운딩 박스를 변환한다. AABB 바운딩 박스의 중심은 벡터 Center이고 대각선이 벡터 Extents이다. 그러므로 AABB 바운딩 박스를 행렬을 사용하여 변환하는 것은 벡터 Center와 벡터 Extents를 행렬과 곱셈을 하는 것과 같다.

```
struct BoundingBox {
   XMFLOAT3 Center;
   XMFLOAT3 Extents;
   size_t CORNER_COUNT = 8;
   ...
};
```

다음 Transform() 멤버 함수는 행렬(m)을 사용하여 AABB 바운딩 박스를 변환하고 변환된 AABB 바운딩 박스를 반환한다(out). 일반적으로 행렬(m)은 게임 객체의 월드 변환 행렬이다. 즉, 모델 좌표계의 메쉬의 점들로부터 구한 AABB 바운딩 박스를 월드 좌표계의 AABB 바운딩 박스로 변환한다.

```
void Transform(BoundingBox &out, XMMATRIX m);
```

다음 Transform() 멤버 함수는 AABB 바운딩 박스를 SRT(Scale, Rotation, Translation) 벡터를 사용하여 변환한다. 벡터 rotation은 회전을 표현하는 쿼터니언이고 벡터 translation은 $(x, y, z, 1)$ 형태로 평행이동의 양을 나타낸다. 이 값들로 변환 행렬을 구성할 수 있으므로 위의 Transform() 멤버 함수와 같은 함수이다.

```
void Transform(BoundingBox &out, float scale, XMVECTOR rotation,
XMVECTOR translation);
```

- 예제

다음은 게임 객체가 카메라에 보이는 가를 AABB 바운딩 박스를 사용하여 검사하는 예제이다. 게임 객체가 카메라의 절두체에 포함되면 렌더링을 하고 그렇지 않으면 렌더링하지 않는 것을 절두체 컬링(Frustum culling)이라고 한다.

CMesh::m_xmBoundingBox는 모델 좌표계의 바운딩 박스이다.

```
class CMesh
{
   ...
   BoundingBox    m_xmBoundingBox;
};

CCubeMesh::CCubeMesh(float fWidth, float fHeight, float fDepth)
{
   float fx = fWidth*0.5f, fy = fHeight*0.5f, fz = fDepth*0.5f;
```

```
   ...
   m_xmBoundingBox = BoundingBox(XMFLOAT3(0.0f, 0.0f, 0.0f),
XMFLOAT3(fx, fy, fz));
}

class CCamera
{
   XMFLOAT4X4            m_xmf4x4View;
   XMFLOAT4X4            m_xmf4x4Projection;
   BoundingFrustum       m_xmFrustum;
   ...
   void GenerateFrustum();
   bool IsInFrustum(BoundingBox& xmbbWorld);
};

void CCamera::GenerateFrustum()
{
   m_xmFrustum.CreateFromMatrix(m_xmFrustum,
XMLoadFloat4x4(&m_xmf4x4Projection));
   XMMATRIX xmmtxInversView = XMMatrixInverse(NULL,
XMLoadFloat4x4(&m_xmf4x4View));
   m_xmFrustum.Transform(m_xmFrustum, xmmtxInversView);
}

bool CCamera::IsInFrustum(BoundingBox& xmbbWorld)
{
   return(m_xmFrustum.Contains(xmbbWorld) != DirectX::DISJOINT);
}

class CGameObject
{
   XMFLOAT4X4     m_xmf4x4World;
   CMesh          *m_pMesh = NULL;
   ...
   bool IsVisible(CCamera *pCamera);
};

bool CGameObject::IsVisible(CCamera *pCamera)
{
   BoundingBox xmbbModel = m_pMesh->m_xmBoundingBox;
   xmbbModel.Transform(xmbbModel, XMLoadFloat4x4(&m_xmf4x4World));
   bool bIsVisible = pCamera->IsInFrustum(xmbbModel);

   return(bIsVisible);
}

void CScene::Render(CCamera *pCamera)
{
   for (int j = 0; j < m_nObjects; j++)
   {
```

```
            bool bIsVisible = m_ppObjects[j]->IsVisible(pCamera);
            if (bIsVisible) m_ppObjects[j]->Render();
        }
    }
```

⑦ BoundingOrientedBox 구조체 충돌 검사 함수

BoundingOrientedBox 구조체는 OOBB 바운딩 박스를 표현하며, 다음과 같은 멤버 함수를 제공한다. BoundingOrientedBox는 구조체이므로 구조체 객체를 통하여 멤버 함수를 호출함에 주의하라.

- 생성자(Constructor)

```
BoundingOrientedBox(BoundingOrientedBox &box);
BoundingOrientedBox(XMFLOAT3 &center, XMFLOAT3 &extents, XMFLOAT4
&orientation);
```

- 초기화 함수(Initialization function, 생성 함수: Creation function)

  다음은 BoundingOrientedBox 구조체를 초기화할 수 있는 정적 함수(Static function)이다.

  다음 CreateFromBoundingBox() 멤버 함수는 AABB 바운딩 박스(box)로부터 OOBB 바운딩 박스(out)를 초기화한다.

```
static void CreateFromBoundingBox(BoundingOrientedBox &out,
BoundingBox &box);
```

  다음 CreateFromPoints() 멤버 함수는 메쉬의 정점들의 배열(pPoints), 정점의 개수(count), 정점 하나의 크기(바이트 수, stride)로부터 OOBB 바운딩 박스를 초기화한다. 정점의 첫 번째 요소는 정점의 위치 벡터라고 가정한다.

```
static void CreateFromPoints(BoundingOrientedBox &out, size_t count,
XMFLOAT3 *pPoints, size_t stride);
```

  메쉬의 정점들의 배열(pPoints)에서 OOBB 바운딩 박스의 중심 Center와 대각선 Extents 벡터를 계산하는 것은 AABB 바운딩 박스와 같다. OOBB 바운딩 박스의 방향은 정점들의 공분산 행렬(Covariance matrix) $\boldsymbol{C}$를 구하고 이 행렬의 고유벡터(Eigen vector)를 계산하여 최대 분산의 방향으로 결정할 수 있다. 정점의 개수가 n, i-번째 정점의 위치 벡터가 $\boldsymbol{p}_i = (x_i, y_i, z_i)$이라고 가정한다.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i,\ \ \bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$$

$$\boldsymbol{C} = \begin{bmatrix} C_{xx} & C_{cy} & C_{xz} \\ C_{yx} & C_{yy} & C_{yz} \\ C_{zx} & C_{zy} & C_{zz} \end{bmatrix}$$

$$C_{xx} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2,\ C_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}),\ C_{xz} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(z_i - \bar{z})$$

$$C_{yx} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x}),\ C_{yy} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2,\ C_{yz} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})(z_i - \bar{z})$$

$$C_{zx} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})(x_i - \bar{z}),\ C_{zy} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})(y_i - \bar{y}),\ C_{zz} = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2$$

- 꼭지점 반환 함수

  다음 GetCorners() 멤버 함수는 OOBB 바운딩 박스의 꼭지점 8개의 벡터(XMFLOAT3)를 반환한다.

```
void GetCorners(XMFLOAT3 *pCorners);
```

- 포함(Contains) 함수

  다음 ContainedBy() 멤버 함수는 OOBB 바운딩 박스가 평면 6개로 표현되는 절두체에 포함(교차)되는 가를 검사한다. 반환 자료형은 ContainmentType이다.

```
ContainmentType ContainedBy(XMVECTOR plane0, XMVECTOR plane1,
XMVECTOR plane2, XMVECTOR plane3, XMVECTOR plane4, XMVECTOR plane5);
```

  다음 Contains() 멤버 함수는 OOBB 바운딩 박스가 점(point)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR point);
```

  다음 Contains() 멤버 함수는 OOBB 바운딩 박스가 삼각형(v0, v1, v2)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR v0, v1, v2);
```

  다음 Contains() 멤버 함수는 OOBB 바운딩 박스가 바운딩 구(sphere)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingSphere &sphere);
```

  다음 Contains() 멤버 함수는 OOBB 바운딩 박스가 AABB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingBox &box);
```

  다음 Contains() 멤버 함수는 OOBB 바운딩 박스가 다른 OOBB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingOrientedBox &box);
```

  다음 Contains() 멤버 함수는 OOBB 바운딩 박스가 절두체(frustum)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingFrustum &frustum);
```

- 교차(Intersects) 함수

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 평면(plane)과 교차하는 가를 검사한다. 반환 자료형은 평면과의 위치 관계를 나타내는 PlaneIntersectionType이다.

```
PlaneIntersectionType Intersects(XMVECTOR plane);
```

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 광선(origin, direction)과 교차하는 가를 검사한다. 광선이 OOBB 바운딩 박스와 교차하면 광선의 원점에서 교점까지의 거리를 distance로 전달한다.

```
bool Intersects(XMVECTOR origin, direction, float &distance);
```

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 삼각형(v0, v1, v2)과 교차하는 가를 검사한다.

```
bool Intersects(XMVECTOR v0, v1, v2);
```

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 바운딩 구(sphere)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingSphere &sphere);
```

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 AABB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingBox &box);
```

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 다른 OOBB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingOrientedBox &box);
```

  다음 Intersects() 멤버 함수는 OOBB 바운딩 박스가 절두체(frustum)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingFrustum &frustum);
```

- 변환(Transform) 함수

  함수 Transform()은 행렬을 사용하여 OOBB 바운딩 박스를 변환한다. OOBB 바운딩 박스의 중심은 벡터 Center이고 대각선이 벡터 Extents이며 방향이 Orientation이다. 그러므로 행렬을 사용하여 OOBB 바운딩 박스를 변환하는 것은 벡터 Center, 벡터 Extents, 벡터 Orientation을 행렬과 곱셈하는 것과 같다.

```
struct BoundingOrientedBox {
   XMFLOAT3 Center;
   XMFLOAT3 Extents;
   XMFLOAT4 Orientation;
   size_t CORNER_COUNT = 8;
   ...
```

```
};
```

다음 Transform() 멤버 함수는 OOBB 바운딩 박스를 행렬(m)을 사용하여 변환하고 변환된 OOBB 바운딩 박스를 반환한다(out). 일반적으로 행렬(m)은 게임 객체의 월드 변환 행렬이다. 즉, 모델 좌표계의 메쉬의 점들로부터 구한 OOBB 바운딩 박스를 월드 좌표계의 OOBB 바운딩 박스로 변환한다.

```
void Transform(BoundingOrientedBox &out, XMMATRIX m);
```

다음 Transform() 멤버 함수는 OOBB 바운딩 박스를 SRT(Scale, Rotation, Translation) 벡터를 사용하여 변환한다. 벡터 rotation은 회전을 표현하는 쿼터니언이고 벡터 translation은 $(x, y, z, 1)$ 형태로 평행이동의 양을 나타낸다. 이 값들로 변환 행렬을 구성할 수 있으므로 위의 Transform() 멤버 함수와 같은 함수이다.

```
void Transform(BoundingOrientedBox &out, float scale, XMVECTOR
rotation, XMVECTOR translation);
```

⑧ BoundingSphere 구조체 충돌 검사 함수

BoundingSphere 구조체는 바운딩 구를 표현하며, 다음과 같은 멤버 함수를 제공한다.

- 생성자(Constructor)

```
BoundingSphere(XMFLOAT3 &center, float radius);
BoundingSphere(BoundingSphere &sphere);
```

- 초기화 함수(Initialization function, 생성 함수: Creation function)

다음은 BoundingSphere 구조체를 초기화할 수 있는 정적 함수(Static function)이다.

다음 CreateFromBoundingBox() 멤버 함수는 AABB 바운딩 박스(box)로부터 바운딩 구(out)를 초기화한다.

```
static void CreateFromBoundingBox(BoundingSphere &out, BoundingBox
&box);
```

다음 CreateFromBoundingBox() 멤버 함수는 OOBB 바운딩 박스(box)로부터 바운딩 구(out)를 초기화한다.

```
static void CreateFromBoundingBox(BoundingSphere &out,
BoundingOrientedBox &box);
```

다음 CreateFromPoints() 멤버 함수는 메쉬의 정점들의 배열(pPoints), 정점의 개수(count), 정점 하나의 크기(바이트 수, stride)로부터 바운딩 구를 초기화한다. 정점의 첫 번째 요소는 정점의 위치 벡터라고 가정한다.

```
static void CreateFromPoints(BoundingSphere &out, size_t count,
```

```
XMFLOAT3 *pPoints, size_t stride);
```

다음 CreateFromFrustum() 멤버 함수는 바운딩 절두체(frustum)로부터 바운딩 구(out)를 초기화한다.

```
static void CreateFromFrustum(BoundingSphere &out, BoundingFrustum
&frustum);
```

다음 CreateMerged() 멤버 함수는 두 개의 바운딩 구(sphere1, sphere2)를 포함하는 바운딩 구로 바운딩 구(out)를 초기화한다.

```
static void CreateMerged(BoundingSphere &out, &sphere1, &sphere2);
```

- 포함(Contains) 함수

다음 ContainedBy() 멤버 함수는 바운딩 구가 평면 6개로 표현되는 절두체에 포함(교차)되는 가를 검사한다. 반환 자료형은 ContainmentType이다.

```
ContainmentType ContainedBy(XMVECTOR plane0, XMVECTOR plane1,
XMVECTOR plane2, XMVECTOR plane3, XMVECTOR plane4, XMVECTOR plane5);
```

다음 Contains() 멤버 함수는 바운딩 구가 점(point)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR point);
```

다음 Contains() 멤버 함수는 바운딩 구가 삼각형(v0, v1, v2)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR v0, v1, v2);
```

다음 Contains() 멤버 함수는 바운딩 구가 다른 바운딩 구(sphere)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingSphere &sphere);
```

다음 Contains() 멤버 함수는 바운딩 구가 AABB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingBox &box);
```

다음 Contains() 멤버 함수는 바운딩 구가 OOBB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingOrientedBox &box);
```

다음 Contains() 멤버 함수는 바운딩 구가 절두체(frustum)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingFrustum &frustum);
```

- 교차(Intersects) 함수

다음 Intersects() 멤버 함수는 바운딩 구가 평면(plane)과 교차하는 가를 검사한다. 반환 자료형은 평면과의 위치 관계를 나타내는 PlaneIntersectionType이다.

```
PlaneIntersectionType Intersects(XMVECTOR plane);
```

다음 Intersects() 멤버 함수는 바운딩 구가 광선(origin, direction)과 교차하는 가를 검사한다. 광선이 바운딩 구와 교차하면 광선의 원점에서 교점까지의 거리를 distance로 전달한다.

```
bool Intersects(XMVECTOR origin, direction, float &distance);
```

다음 Intersects() 멤버 함수는 바운딩 구가 삼각형(v0, v1, v2)과 교차하는 가를 검사한다.

```
bool Intersects(XMVECTOR v0, v1, v2);
```

다음 Intersects() 멤버 함수는 바운딩 구가 다른 바운딩 구(sphere)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingSphere &sphere);
```

다음 Intersects() 멤버 함수는 바운딩 구가 AABB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingBox &box);
```

다음 Intersects() 멤버 함수는 바운딩 구가 OOBB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingOrientedBox &box);
```

다음 Intersects() 멤버 함수는 바운딩 구가 절두체(frustum)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingFrustum &frustum);
```

- 변환(Transform) 함수

  함수 Transform()은 행렬을 사용하여 바운딩 구를 변환한다. 바운딩 구의 중심은 벡터 Center이고 반지름이 Radius이다. 그러므로 행렬을 사용하여 바운딩 구를 변환하는 것은 벡터 Center를 행렬과 곱셈하는 것이다.

```
struct BoundingSphere {
   XMFLOAT3 Center;
   float Radius;
   ...
};
```

다음 Transform() 멤버 함수는 바운딩 구를 행렬(m)을 사용하여 변환하고 변환된 바운딩 구를 반환한다(out). 일반적으로 행렬(m)은 게임 객체의 월드 변환 행렬이다. 즉, 모델 좌표계의 메쉬의 점들로부터 구한 바운딩 구를 월드 좌표계의 바운딩 구로 변

환한다.

```
void Transform(BoundingSphere &out, XMMATRIX m);
```

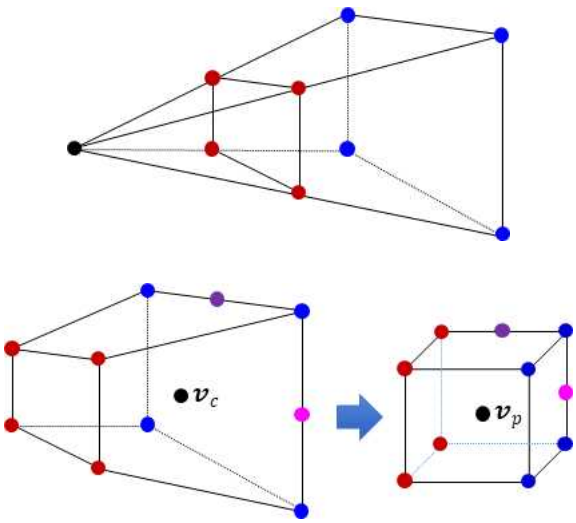
다음 Transform() 멤버 함수는 바운딩 구를 SRT(Scale, Rotation, Translation) 벡터를 사용하여 변환한다. 벡터 rotation은 회전을 표현하는 쿼터니언이고 벡터 translation은 $(x, y, z, 1)$ 형태로 평행이동의 양을 나타낸다. 이 값들로 변환 행렬을 구성할 수 있으므로 위의 Transform() 멤버 함수와 같은 함수이다.

```
void Transform(BoundingSphere &out, float scale, XMVECTOR rotation,
XMVECTOR translation);
```

⑨ BoundingFrustum 구조체 충돌 검사 함수

BoundingFrustum 구조체는 바운딩 절두체를 표현하며, 다음과 같은 멤버 함수를 제공한다. 바운딩 절두체는 절두체의 원점(Origin)과 방향(Orientation), 그리고 6개의 평면으로 구성된다. 카메라 절두체의 경우 절두체의 원점은 카메라 좌표계의 원점 $(0, 0, 0)$, 방향은 카메라 좌표계의 z-축이다. 그리고 6개의 평면은 원근 투영 변환 행렬 $\boldsymbol{P}$로부터 생성할 수 있다.

```
struct BoundingFrustum
{
   XMFLOAT3 Origin;
   XMFLOAT4 Orientation;
   float Near;
   float Far;
   float LeftSlope;
   float RightSlope;
   float TopSlope;
   float BottomSlope;
   size_t CORNER_COUNT = 8;
   ...
};
```



원근 투영 변환 행렬 $\boldsymbol{P}$는 카메라 좌표계의 점 $\boldsymbol{v}_c = (x_c, y_c, z_c, 1)$을 $\boldsymbol{v}_p = (x_p, y_p, z_p, w_p)$로 변환한다. 행렬 $\boldsymbol{P}$를 다음과 같이 열벡터의 형태로 표현하자.

$$\boldsymbol{P} = \begin{pmatrix} xScale & 0 & 0 & 0 \\ 0 & yScale & 0 & 0 \\ 0 & 0 & zf/(zf-zn) & 1 \\ 0 & 0 & -zn*zf/(zf-zn) & 0 \end{pmatrix}$$

$$\boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3, \boldsymbol{P}_4) \quad \boldsymbol{P}_j = (P_{1j}, P_{2j}, P_{3j}, P_{4j})$$

$$\boldsymbol{v}_c\boldsymbol{P} = (x_c, y_c, z_c, 1)\boldsymbol{P} = \boldsymbol{v}_p = (x_p, y_p, z_p, w_p) = (\boldsymbol{v}_c \bullet \boldsymbol{P}_1, \boldsymbol{v}_c \bullet \boldsymbol{P}_2, \boldsymbol{v}_c \bullet \boldsymbol{P}_3, \boldsymbol{v}_c \bullet \boldsymbol{P}_4)$$

$$(x_c, y_c, z_c, 1)\boldsymbol{P} = (x_p, y_p, z_p, z_c) = (x_c P_{11}, y_c P_{22}, z_c P_{33} + P_{43}, z_c)$$

$$(x_p, y_p, z_p, w_p) \rightarrow \left(\frac{x_p}{w_p}, \frac{y_p}{w_p}, \frac{z_p}{w_p}, 1\right) \rightarrow \left(\frac{x_p}{w_p}, \frac{y_p}{w_p}, \frac{z_p}{w_p}\right)$$

카메라 좌표계의 점 $\boldsymbol{v}_c = (x_c, y_c, z_c, 1)$가 카메라 절두체 내부의 점이면 다음이 성립한다.

$$-1 \le \frac{x_p}{w_p} \le 1, \ -1 \le \frac{y_p}{w_p} \le 1, \ 0 \le \frac{z_p}{w_p} \le 1$$

$$-w_p \le x_p \le w_p, \ -w_p \le y_p \le w_p, \ 0 \le z_p \le w_p$$

$$(-w_p \le x_p \le w_p) = (-(\boldsymbol{v}_c \bullet \boldsymbol{P}_4) \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_1) \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_4))$$

$$0 \le \boldsymbol{v}_c \bullet (\boldsymbol{P}_1 + \boldsymbol{P}_4), \ 0 \le \boldsymbol{v}_c \bullet (-\boldsymbol{P}_1 + \boldsymbol{P}_4)$$

$$(-w_p \le y_p \le w_p) = (-(\boldsymbol{v}_c \bullet \boldsymbol{P}_4) \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_2) \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_4))$$

$$0 \le \boldsymbol{v}_c \bullet (\boldsymbol{P}_2 + \boldsymbol{P}_4), \ 0 \le \boldsymbol{v}_c \bullet (-\boldsymbol{P}_2 + \boldsymbol{P}_4)$$

$$(0 \le z_p \le w_p) = (0 \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_3) \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_4))$$

$$0 \le (\boldsymbol{v}_c \bullet \boldsymbol{P}_3), \ 0 \le \boldsymbol{v}_c \bullet (-\boldsymbol{P}_3 + \boldsymbol{P}_4)$$

카메라 절두체의 왼쪽 평면의 방정식은 다음과 같다.

$$\boldsymbol{v}_c \bullet (\boldsymbol{P}_1 + \boldsymbol{P}_4) = 0$$

$$\boldsymbol{v}_c \bullet (\boldsymbol{P}_1 + \boldsymbol{P}_4) = (x_c, y_c, z_c, 1) \bullet \{(P_{11}, P_{21}, P_{31}, P_{41}) + (P_{14}, P_{24}, P_{34}, P_{44})\} = 0$$

$$\boldsymbol{v}_c \bullet (\boldsymbol{P}_1 + \boldsymbol{P}_4) = (x_c, y_c, z_c, 1) \bullet (P_{11} + P_{14}, P_{21} + P_{24}, P_{31} + P_{34}, P_{41} + P_{44}) = 0$$

$$\boldsymbol{v}_c \bullet (\boldsymbol{P}_1 + \boldsymbol{P}_4) = (P_{11} + P_{14})x_c + (P_{21} + P_{24})y_c + (P_{31} + P_{34})z_c + (P_{41} + P_{44}) = 0$$

$$(P_{11} + P_{14})x_c + (P_{21} + P_{24})y_c + (P_{31} + P_{34})z_c + (P_{41} + P_{44}) = 0$$

$$ax_c + by_c + cz_c + d = 0$$

위 식은 카메라 절두체의 왼쪽 평면의 법선 벡터가 방정식은 원근 투영 변환 행렬 $\boldsymbol{P}$의 첫 번째 열과 네 번째 열로부터 구할 수 있음을 보여준다. 절두체의 나머지 평면의 방정식도 유사한 방식으로 유도할 수 있다. 원근 투영 변환 행렬 $\boldsymbol{P}$에서 구한 평면의 방정식은 카메라 좌표계로 표현됨에 주의하라.

- 생성자(Constructor)

```
BoundingFrustum(XMMATRIX projection);
BoundingFrustum(BoundingFrustum &frustum);
BoundingFrustum(XMFLOAT3 &origin, XMFLOAT4 &orientation, float
right, float left, float top, float bottom, float near, float far);

{
   BoundingFrustum frustum(gmtxProjection);
   XMVECTOR point = XMVectorSet(1.0f, 2.0f, 3.0f);
   ContainmentType result = frustum.Contains(point);
}
```

- 초기화 함수(Initialization function, 생성 함수: Creation function)

  다음은 BoundingFrustum 구조체를 초기화할 수 있는 정적 함수(Static function)이다.

  다음 CreateFromMatrix() 멤버 함수는 원근 투영 변환 행렬(projection)로부터 바운딩 절두체(out)를 초기화한다.

```
static void CreateFromMatrix(BoundingFrustum &out, XMMATRIX
projection);
```

- 꼭지점, 평면 반환 함수

  다음 GetCorners() 멤버 함수는 바운딩 절두체의 꼭지점 8개의 벡터(XMFLOAT3)를 반환한다.

```
void GetCorners(XMFLOAT3 *pCorners);
```

  다음 GetPlanes() 멤버 함수는 바운딩 절두체의 6개의 평면(XMVECTOR)을 반환한다.

```
void GetPlanes(XMVECTOR *pNear, *pFar, *pRight, *pLeft, *pTop,
*pBottom);
```

- 포함(Contains) 함수

  다음 ContainedBy() 멤버 함수는 바운딩 절두체가 평면 6개로 표현되는 다른 절두체에 포함(교차)되는 가를 검사한다. 반환 자료형은 ContainmentType이다.

```
ContainmentType ContainedBy(XMVECTOR plane0, XMVECTOR plane1,
XMVECTOR plane2, XMVECTOR plane3, XMVECTOR plane4, XMVECTOR plane5);
```

  다음 Contains() 멤버 함수는 바운딩 절두체가 점(point)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR point);
```

  다음 Contains() 멤버 함수는 바운딩 절두체가 삼각형(v0, v1, v2)을 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(XMVECTOR v0, v1, v2);
```

다음 Contains() 멤버 함수는 바운딩 절두체가 다른 바운딩 구(sphere)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingSphere &sphere);
```

다음 Contains() 멤버 함수는 바운딩 절두체가 AABB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingBox &box);
```

다음 Contains() 멤버 함수는 바운딩 절두체가 OOBB 바운딩 박스(box)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingOrientedBox &box);
```

다음 Contains() 멤버 함수는 바운딩 절두체가 다른 절두체(frustum)를 포함(교차)하는 가를 검사한다.

```
ContainmentType Contains(BoundingFrustum &frustum);
```

- 교차(Intersects) 함수

다음 Intersects() 멤버 함수는 바운딩 절두체가 평면(plane)과 교차하는 가를 검사한다. 반환 자료형은 평면과의 위치 관계를 나타내는 PlaneIntersectionType이다.

```
PlaneIntersectionType Intersects(XMVECTOR plane);
```

다음 Intersects() 멤버 함수는 바운딩 절두체가 광선(origin, direction)과 교차하는 가를 검사한다. 광선이 바운딩 절두체와 교차하면 광선의 원점에서 교점까지의 거리를 distance로 전달한다.

```
bool Intersects(XMVECTOR origin, direction, float &distance);
```

다음 Intersects() 멤버 함수는 바운딩 절두체가 삼각형(v0, v1, v2)과 교차하는 가를 검사한다.

```
bool Intersects(XMVECTOR v0, v1, v2);
```

다음 Intersects() 멤버 함수는 바운딩 절두체가 바운딩 구(sphere)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingSphere &sphere);
```

다음 Intersects() 멤버 함수는 바운딩 절두체가 AABB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingBox &box);
```

다음 Intersects() 멤버 함수는 바운딩 절두체가 OOBB 바운딩 박스(box)와 교차하는 가를 검사한다.

```
bool Intersects(BoundingOrientedBox &box);
```

다음 Intersects() 멤버 함수는 바운딩 절두체가 다른 절두체(frustum)와 교차하는가를 검사한다.

```
bool Intersects(BoundingFrustum &frustum);
```

- 변환(Transform) 함수

다음 Transform() 멤버 함수는 바운딩 절두체를 행렬(m)을 사용하여 변환하고 변환된 바운딩 절두체를 반환한다(out). 일반적으로 행렬(m)은 카메라 변환 행렬의 역행렬이다.

```
void Transform(BoundingFrustum &out, XMMATRIX m);
```

다음 Transform() 멤버 함수는 바운딩 절두체를 SRT(Scale, Rotation, Translation) 벡터를 사용하여 변환한다. 벡터 rotation은 회전을 표현하는 쿼터니언이고 벡터 translation은 $(x, y, z, 1)$ 형태로 평행이동의 양을 나타낸다. 이 값들로 변환 행렬을 구성할 수 있으므로 위의 Transform() 멤버 함수와 같은 함수이다.

```
void Transform(BoundingFrustum &out, float scale, XMVECTOR rotation,
XMVECTOR translation);
```