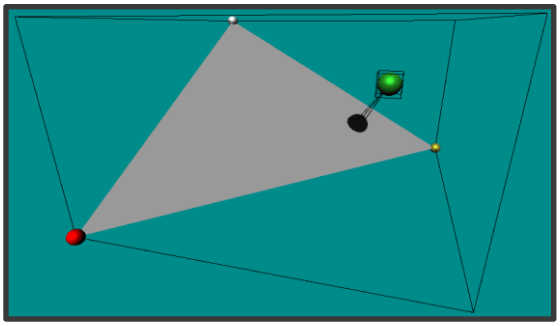
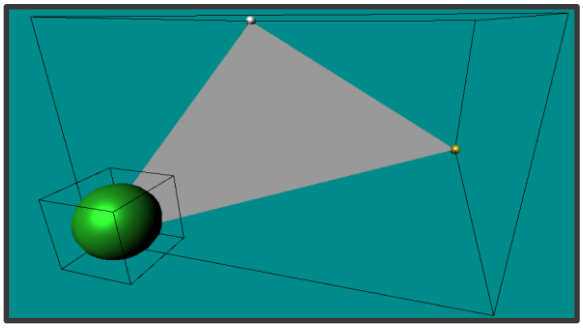
【衝突判定３Ｄ】高速化ＡＡＢＢ

"衝突判定３Ｄ_三角形と球体"でも解説させて頂いた通り、
数万、数十万回の処理を行いますので、0.016秒(60FPS)の間に処理が間に合わず、
処理落ちすることもあります。
当然、他の処理もありますので、できるだけ早く処理させないといけません。

そこで、３Ｄの最速衝突判定、立方体と立方体の衝突判定を使用して、
衝突判定の負荷を減らすということをやっていきます。
この時に作る立方体をAABB(axis-aligned bounding boxes)や、
バウンディングボックスなどと言います。

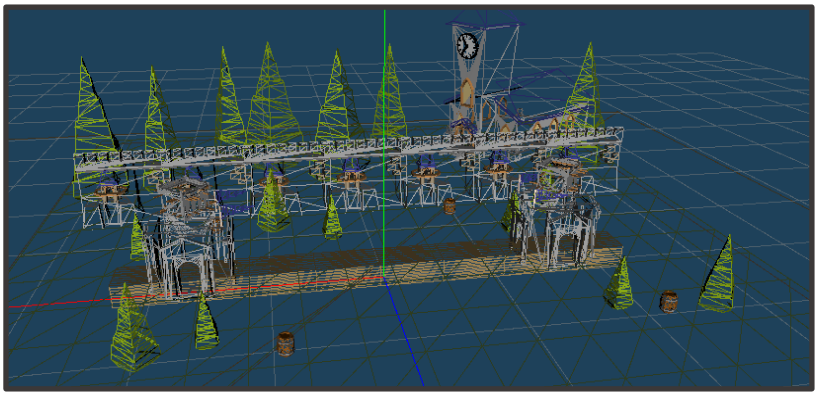
三角形と球体の衝突判定を例に上げると、下図のようなイメージになります。





この立方体同士が衝突していれば、「衝突している可能性がある」ため、
三角形と球体の衝突判定を行い、
立方体同士が衝突していなければ、三角形と球体は「衝突していない」という
判定になるため、三角形と球体の処理を行う必要が無くなり、
処理負荷を下げれる、という段取りになります。

２回処理を行うケースも出てくるので、本当に処理負荷されるのかな？
と、思われがちですが、その効果は抜群で、



約2万1千ポリゴンあるステージのモデルと、
球体1つの衝突判定を行った結果、50～60倍の処理時間の差が出ました。

ちなみに、DxLibの衝突判定関数 MV1CollCheck_Sphereも、
AABBによる衝突判定を行っています。

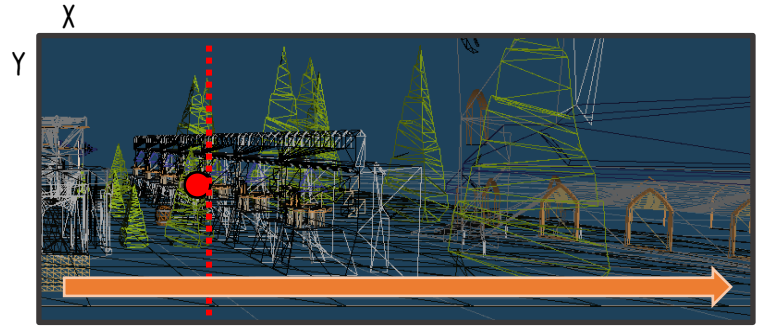
コードで書くと、以下のようなイメージです。

```
// AABB(立方体による簡易判定)
if ((shotMinPos.x > polygon->MaxPosition.x ||
    shotMinPos.y > polygon->MaxPosition.y ||
    shotMinPos.z > polygon->MaxPosition.z ||
    shotMaxPos.x < polygon->MinPosition.x ||
    shotMaxPos.y < polygon->MinPosition.y ||
    shotMaxPos.z < polygon->MinPosition.z) == false)
{

    // 三角形面上の最近接点との距離判定
    if (AsoUtility::IsHitTriangleSphere(
        refMesh.Vertexs[polygon->VIndex[0]].Position,
        refMesh.Vertexs[polygon->VIndex[1]].Position,
        refMesh.Vertexs[polygon->VIndex[2]].Position,
        shotPos, ShotBase::COL_RADIUS))
    {
        shot->Blast();
    }

}
```

DxLibでは、更に高速化を行うために、
頂点情報をソートして、処理回数を減らしたりしています。



左から順に処理していき、
赤の点線以降の
ポリゴンは、衝突判定を
行う必要がない。

ステージモデルを分割して、分割単位でAABB判定を行うのも良い方法です。

各区画のAABB判定を行うと、紫範囲のステージモデルとしか、
衝突判定を行わくて良いことがわかります。