3D アクションゲームプログラミング

○評価要件

☑壁に衝突した状態で移動すると壁に沿って移動するようにする
☑壁との衝突で弾丸を反射させる

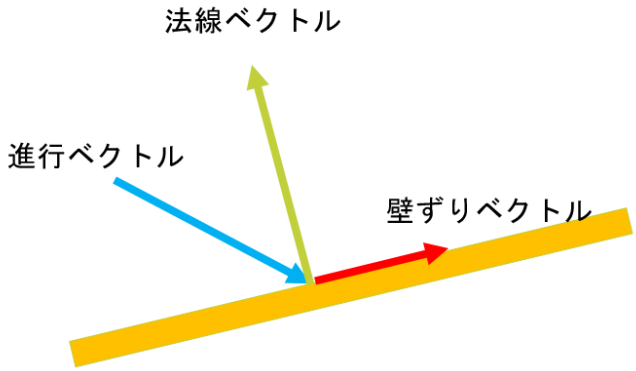
○概要

今回はレイキャストを利用して壁との衝突処理を実装します。
前回は下方向へのレイキャストにより、地面との衝突処理を実装しました。

今回は移動中に進行ベクトル方向にレイキャストをすることで壁との衝突処理を実装します。
単純に衝突した位置に補正するのではなく、壁をズリズリとずるような感じで移動させるのがポイントです。

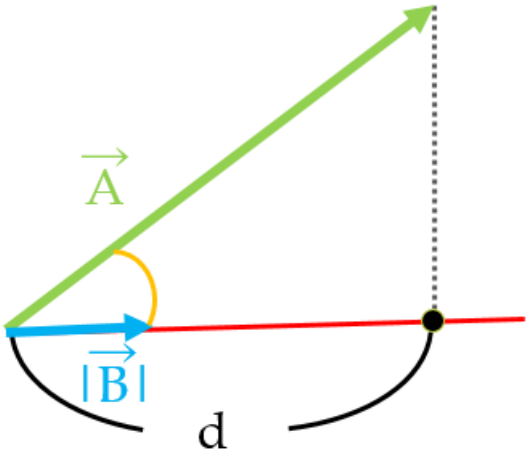
○壁ずり移動

キャラクターが進行ベクトル方向に移動し、壁に衝突した際、壁ずりベクトル方向に移動するようにするのが今回の目的です。
壁ずり移動を実現するには今回も三角関数を使います。



○三角関数

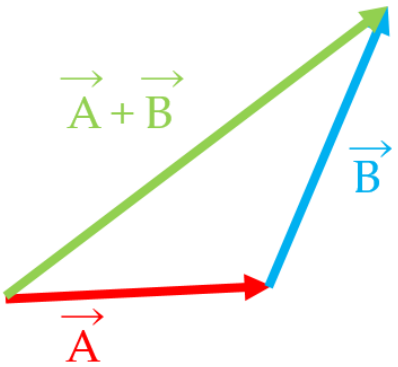
前回も学習しましたが、三角関数について復習しておきましょう。



- ２つのベクトルの内積を求めることで
  角度、または距離を算出できる。
- 両方のベクトルが単位ベクトルの場合、
  角度を求めることができる。
- 片方のベクトルが単位ベクトルの場合、
  距離を求めることができる。
  ベクトルＡと単位ベクトルＢの内積で
  距離ｄが算出できる

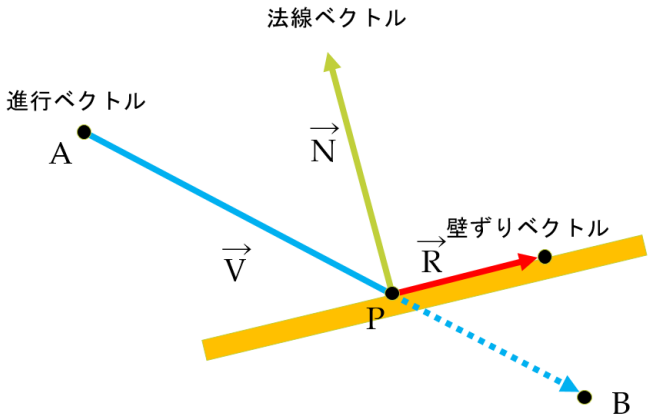
○ベクトルの合成

今回はベクトルの合成についても知っておきましょう。



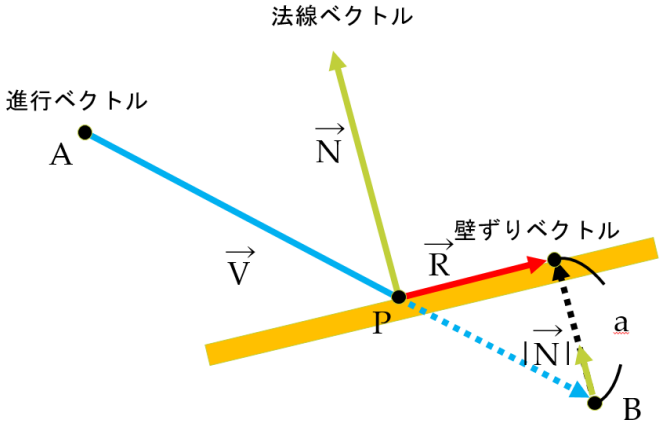
- ２つのベクトルを足し算することで
  合成ベクトルを求めることができる。

○壁ずりベクトルの求め方①



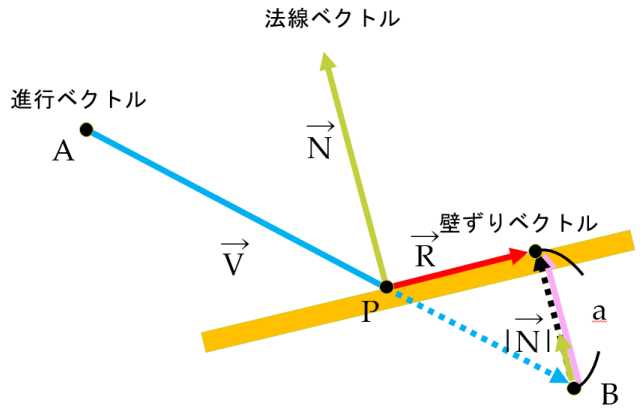
- レイキャストで交点 P と
  法線ベクトル N を算出する

○壁ずりベクトルの求め方②



- 三角関数で a を算出する

○壁ずりベクトルの求め方③



- 進行ベクトルと終点から法線方向へのベクトルの合成で算出できる。

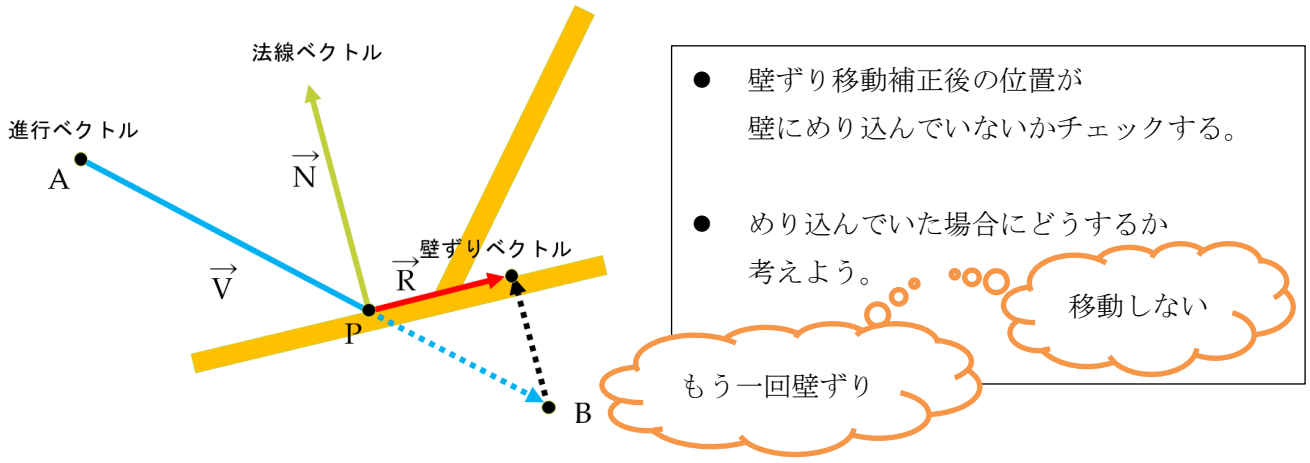
$$\vec{PB} = B - P$$

$$\vec{R} = \vec{PB} + (|\vec{N}| * a)$$

- 壁ずり後の座標は終点から法線方向へ a 進んだ位置で算出できる

$$O = B + (|\vec{N}| * a)$$

○壁ずり後の位置が壁にめり込んでいた場合



○壁ずり移動実装

理屈がわかったところでプログラムを実装しましょう。

Character.cpp

```
---省略---

// 水平移動更新処理
void Character::UpdateHorizontalMove(float elapsedTime)
{
    ---省略---

    // 移動処理
    position.x += velocity.x * elapsedTime;
    position.z += velocity.z * elapsedTime;

    // 水平速力量計算
    float velocityLengthXZ = sqrtf(velocity.x * velocity.x + velocity.z * velocity.z);
    if (velocityLengthXZ > 0.0f)
    {
        // 水平移動値
        float mx = velocity.x * elapsedTime;
        float mz = velocity.z * elapsedTime;

        // レイの開始位置と終点位置
        DirectX::XMFLOAT3 start = { position.x, position.y + stepOffset, position.z };
        DirectX::XMFLOAT3 end   = { position.x + mx, position.y + stepOffset, position.z + mz };

        // レイキャストによる壁判定
        HitResult hit;
        if (Stage::Instance().RayCast(start, end, hit))
        {
            // 壁までのベクトル
            DirectX::XMVECTOR Start = DirectX::XMLoadFloat3(&hit.position);
            DirectX::XMVECTOR End   = DirectX::XMLoadFloat3(&end);
            DirectX::XMVECTOR Vec   = DirectX::XMVectorSubtract(End, Start);

            // 壁の法線
```

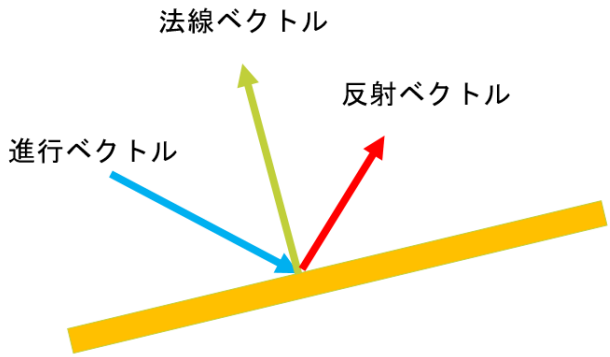
```
            DirectX::XMVECTOR Normal = DirectX::XMLoadFloat3(&hit.normal);

            // 入射ベクトルを法線に射影
            DirectX::XMVECTOR Dot = DirectX::XMVector3Dot(DirectX::XMVectorNegate(Vec), Normal);
            Dot = DirectX::XMVectorScale(Dot, 1.1f);  // 壁にめり込まないように少しだけ補正

            // 補正位置の計算
            DirectX::XMVECTOR CollectPosition = DirectX::XMVectorMultiplyAdd(Normal, Dot, End);
            DirectX::XMFLOAT3 collectPosition;
            DirectX::XMStoreFloat3(&collectPosition, CollectPosition);

            // 壁ずり方向へレイキャスト
            HitResult hit2;
            if (!Stage::Instance().RayCast(start, collectPosition, hit2))
            {
                // 壁ずり方向で壁に当たらなかったら補正位置に移動
                position.x = collectPosition.x;
                position.z = collectPosition.z;
            }
            else
            {
                position.x = hit2.position.x;
                position.z = hit2.position.z;
            }
        }
        else
        {
            // 移動
            position.x += mx;
            position.z += mz;
        }
    }
    ---省略---
}
```

実装できたらキャラクターを移動操作してみて壁に衝突してみましょう。
壁をズリズリと移動出来たら OK です。

○反射

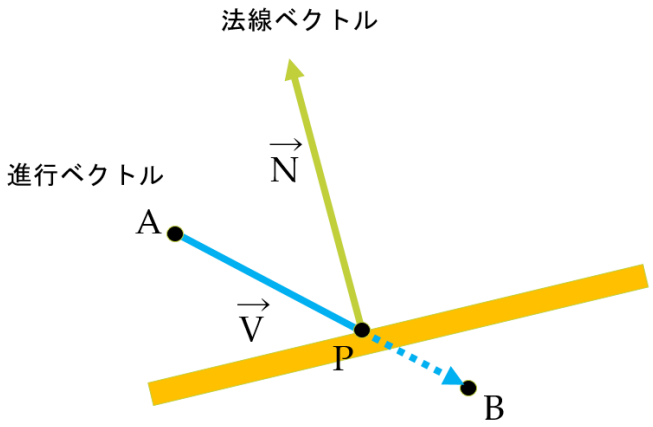
今回、課題では取り扱いませんでしたが、壁ずりベクトルとほぼ同じ方法で反射ベクトルを算出できます。
ボールを壁に反射させるなど使いどころは結構あるので覚えておきましょう。

○反射ベクトル



- 進行ベクトル方向に進んだ物体が
  壁に当たって跳ね返った方向。

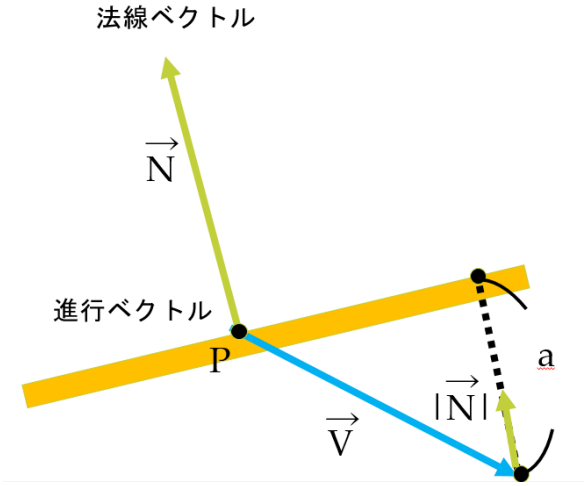
○反射ベクトルの求め方①



- まずはレイキャストで交点 P と
  法線ベクトル N を算出する。。

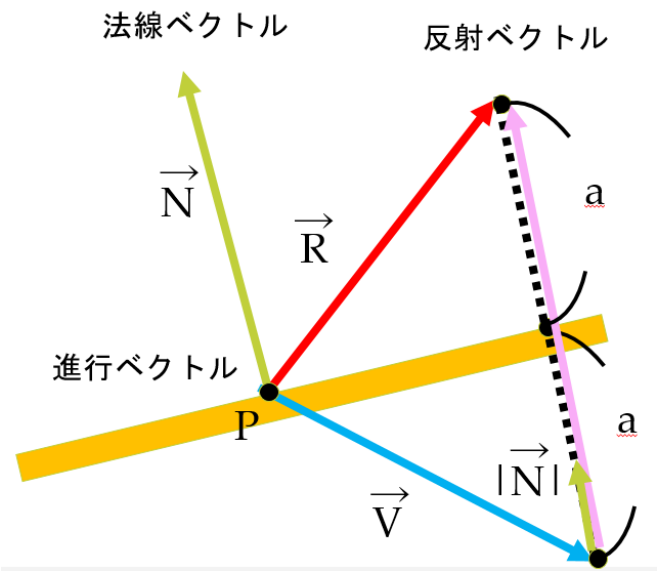
$$\overrightarrow{V} = B - A$$

○反射ベクトルの求め方②



- 三角関数で a を算出する。

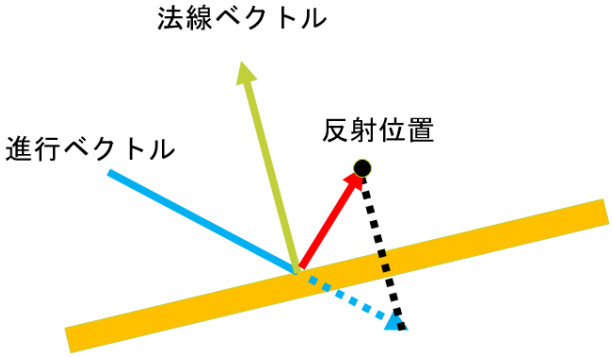
○反射ベクトルの求め方③



- 逆進行ベクトルと終点から
  法線方向へのベクトルで合成できる。

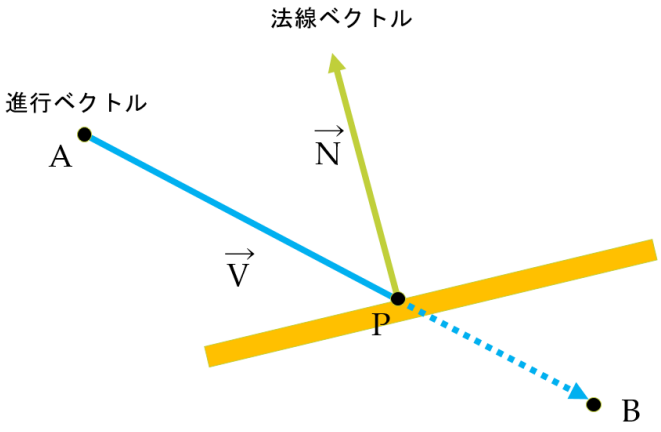
$$\vec{R} = \vec{V} + (|\vec{N}| * a * 2)$$

○反射位置



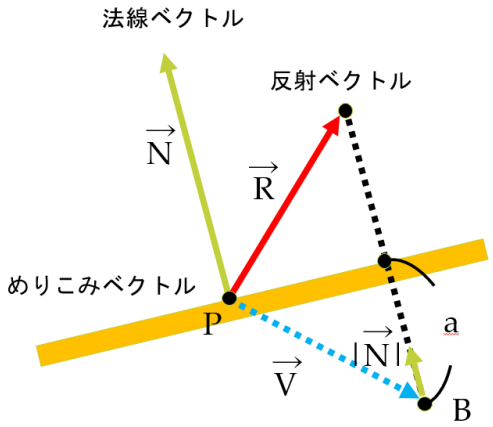
- 進行ベクトル方向に進んだ物体が
  壁にめり込んだ分の跳ね返った時の位置。

○反射位置の求め方①



- レイキャストで交点Pと法線ベクトルNを
  算出する。

○反射位置の求め方②

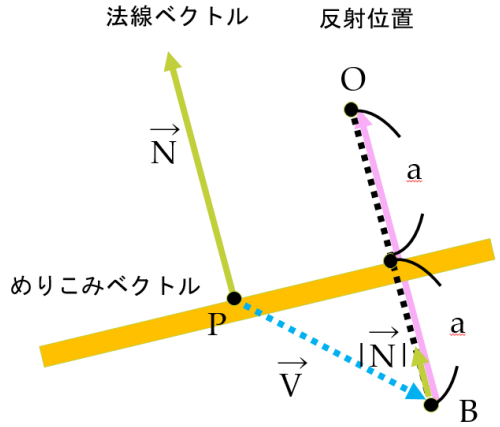


- 三角関数で a を算出する

$$\vec{V} = B - P$$

$$a = -\vec{V} \cdot |\vec{N}|$$

○反射位置の求め方③



- 終点から法線方向へ長さ a の
  2 倍先が反射位置。

$$O = B + (|\vec{N}| * a * 2)$$

弾丸が壁に衝突した時に弾丸を反射させるなどのプログラムを実装してみましょう。
お疲れ様でした。