線形補間による移動

線形補間とは？

Wikipediaより。

座標(x0, y0)と(x1, y1)があるとする。
ここで、 [x0, x1]の間にあるxが与えられたときに、
この線上にある点を得たいとする。

$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0}$$

簡略し過ぎかもしれませんが、点と点を結ぶ線を補間していくことです。

(0, 0)

xが値が5の時、yの値はいくつでしょう？という問題解く時に、
上記の式を使います。

$$\frac{y - 2}{(4 - 2)} = \frac{(5 - 3)}{(7 - 3)}$$

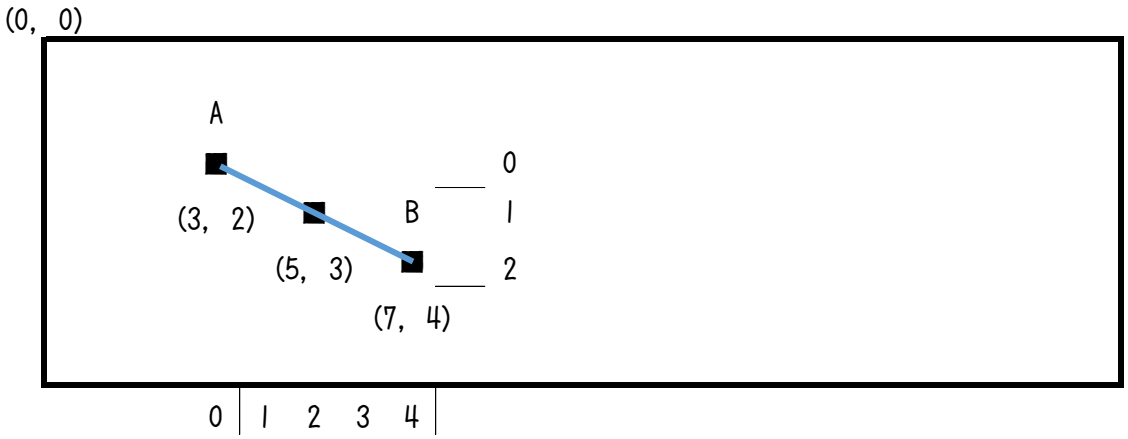
$$\frac{y - 2}{2} = \frac{2}{4}$$

$$y - 2 = 1$$

$$y = 3$$

移動に例えると、移動開始座標と、移動終了座標と、その間のxがわかれば、
等速で移動した時のyが導きだせますよ！という意味です。
(xとyは逆でも可)

このままだと、ゲームで何に使って良いのかよくわかりませんので、
式をもう少しかみ砕いていくと、
前述の例では、xを2としていますので、xの総移動量4の半分になります。
だったら、yも移動量(2)の半分(1)移動させるよ、ということをやっています。



これを応用すると、xもyも足並み揃えて割合(率)で一緒に移動させよう、
ということができます。

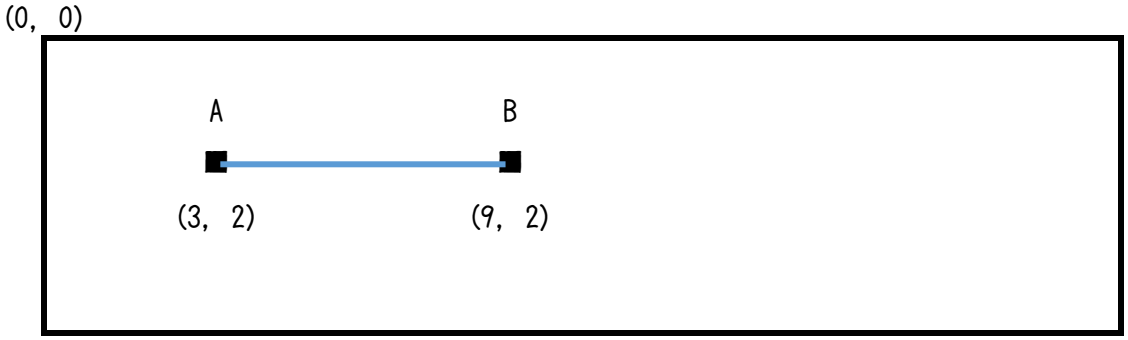
```
x  = x0 + (t * (x1 - x0))
y  = y0 + (t * (y1 - y0))
```

tは、0から1までの割合。
tが、0.5だったら50%、
tが、0.8だったら80%、
tが、1.0だったら100%(x1, y1に到着した)

xとか、片方の座標が決まってなくても、
開始地点と、終了地点があって、割合で移動できる。

まだ、ゲームへの活用方法がピンときませんが、
線形補間自体は、CGやアニメーションなどでも使われている、
メジャーな技術になりますので、もう少し続きを頑張ってみましょう。

これまでの移動との違い



上記でいくと最初は、X3の座標から、Xを徐々に増やしていって、
X9まで移動したら、ゴール、という意味です。

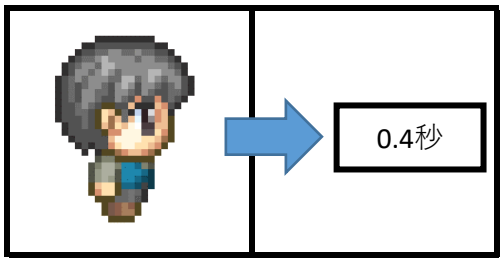
今回は、等速(慣性移動ではない)でユニットを移動させますので、
スピード(移動量)が1だとしたら、6回移動させると、ゴールに辿りつきます。
(60FPSだとすると、6回 ⇒ 6フレーム ⇒ 6 * 0.016秒 = 0.096秒)

スピード(移動量)を調整して、ちょうど良い速さにするのが、
これまでの移動のやり方でしたが、『 時間 』や『 距離 』を重要視したい、
という場面にも、遭遇する機会があるでしょう。

- イベントシーンで、ユニットを特定の場所まで、３秒で移動させて！
- 追従NPCの実装で、プレイヤーとの距離に応じて移動速度を変えて！
- 3Dキャラクターの向きを１秒くらいかけて、滑らかに90度回転させて！

それでは実装していきましょう

AsoBaseでは、
ユニットの１マスあたりの移動時間を0.4秒としたいと思います。



① 応用式を関数化していきます。

```
// 線形補間
static Vector2 Lerp(Vector2 start, Vector2 end, float t)
{
    Vector2 ret = start;
    ret.x += t * (end.x - start.x);
    ret.y += t * (end.y - start.y);
    return ret;
}
```

② 時間を定数で定義

```
static constexpr float TIME_MOVE = 0.4f;
```

③ Update内で、Lerp関数に
移動開始座標、移動終了座標、時間の経過割合を渡して、移動先の座標を求める

```
mStepMove += mSceneManager->GetDeltaTime();
float t = mStepMove / TIME_MOVE;
mPos = Vector2::Lerp(mMvSPos, mMvEPos, t);
if (t >= 1.0f)
{
    mPos = mMvEPos;
    //mIsPushing = false;
    ChangeState(STATE::IDLE);
}
```