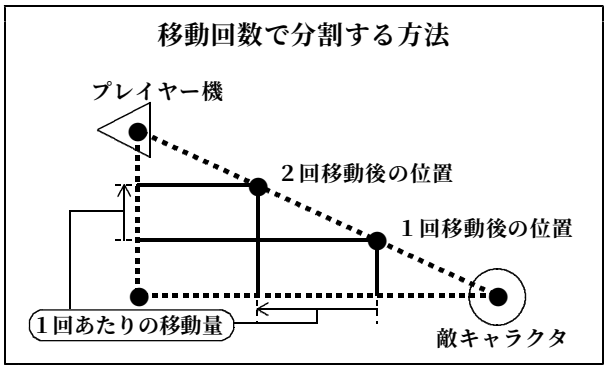
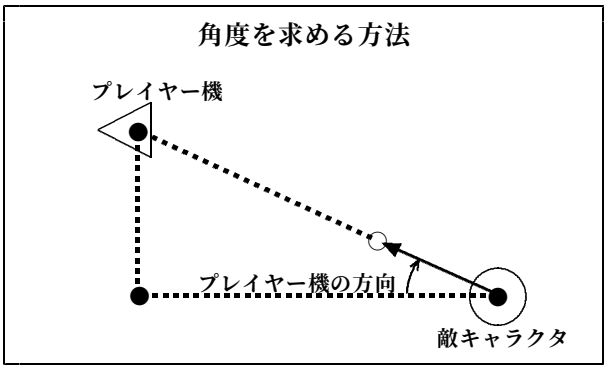
# オブジェクト指向と ゲームプログラミング

## §　アルゴリズム編 － 第４回　プレイヤーを狙う弾　§

### プレイヤーを狙う弾

プレイヤー機の方向へ弾を撃つ場合、その狙い方は、大きく分けて２とおりあります。
１つは、プレイヤー機と敵キャラクターとの座標の差を移動回数または１回あたりの移動量で分割して、速度とするものです。実際にキャラクターを表示するグラフィック画面は、横方向のｘ座標と縦方向のｙ座標を利用しているため、この方式は比較的簡単に使えます。
もう１つは、敵キャラクターからプレイヤー機を見た場合の方向(角度)を求めて、その方向に移動する速度を計算するという方法があります。これは、実際に人間が何かを狙うときの発想と同じなので、考え方としてなじみやすいはずです。しかし、この方式をプログラムへ適用しようとすると、"方向"というものを取り扱わなければならないために、プログラミングがやや難しくなります。ですが、この「方向」の概念を利用すると、最初のアルゴリズムでは難しい弾の移動が実現できるようになります。




### "角度"を計算で求める

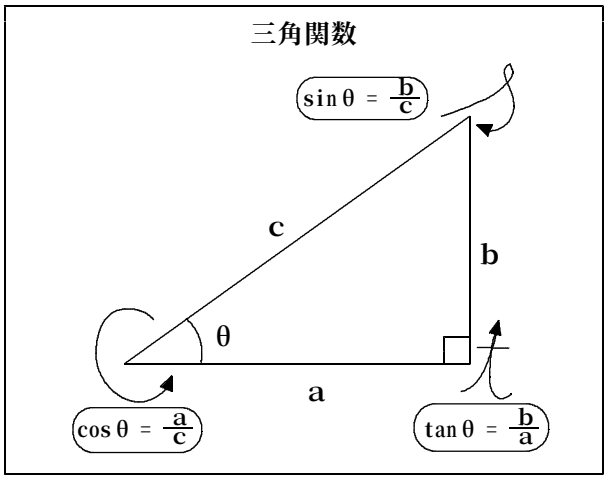
方向の概念を利用してターゲットを狙うアルゴリズムには、「ターゲットがどの方向に見えるか」という要素が欠かせません。
２次元の場合は、方向というのは"角度"のパラメータを利用すれば数値として処理できます。よって、プログラム的には「角度を計算してその方向を狙う」という処理になります。
ここで、敵キャラクタからプレイヤー機を狙う弾の移動速度を計算することを考えてみましょう。２次元のゲームを実際に表現するのは画面座標系なので、最終的にはｘ座標方向への速度とｙ座標方向への速度の２つの数値を計算するのが目的になります。
ほとんどの場合、プレイヤー機と敵キャラクタの座標しかありません。これから角度を計算するためには"三角関数"と呼ばれる関数を利用しなければなりません。

### 三角関数

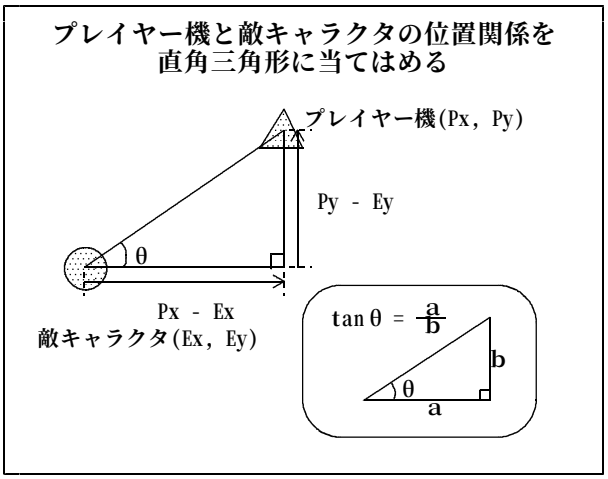
三角関数には、

- sin（サイン）
- cos（コサイン）
- tan（タンジェント）

の３種類があります。これらはすべて「直角三角形の特定の頂点の角度から、辺の長さの比を計算するもの」です。




▲角度θに対して、sin/cos/tanは
このような辺の比率になります




▲位置関係に直角三角形を当てはめて、
三角関数から角度を求めましょう

この中で、今回使えそうなものは、tanです。ｘ座標の差とｙ座標の差、そして角度θが、

$$\tan\theta = \frac{\text{プレーヤー機のｙ座標} - \text{敵キャラクターのｙ座標}}{\text{プレーヤー機のｘ座標} - \text{敵キャラクターのｘ座標}}$$

というようにtanによって関連づけられます。

ただしtanは、あくまでも「角度を入力すると辺の長さの比が出力される」という関数です。プレイヤー機の方向を調べるためには、座標の差から計算した「辺の長さの比」から角度を計算しなくてはならないため、逆の動作が必要です。

このために、tanの"逆関数"を利用します。逆関数というのは入力と出力の関係をひっくり返した関数です。

tan自体は、角度を入力すると辺の長さの比が出力されます。tanの逆関数は、辺の長さの比を入力すると、角度が出力されます。このような逆関数があれば、

$$\theta = \text{tanの逆関数}\left\{\frac{\text{プレーヤー機のｙ座標} - \text{敵キャラクターのｙ座標}}{\text{プレーヤー機のｘ座標} - \text{敵キャラクターのｘ座標}}\right\} \quad \cdots ①$$

というように、ｙ座標の差をｘ座標の差で割った数値から、角度θを計算できます。つまり、プレイヤー機の方向が角度θとして得られるわけです。

Ｃ言語では、tanの逆関数がatan関数(arctan:アークタンジェント)、atan2関数として用意されているので、これをそのまま利用できます。これらの関数の返値の単位はラジアンです。範囲は、atan関数が「π／２ ～ －π／２」(９０度～－９０度)、atan2関数が「π ～ －π」(１８０度～－１８０度)となります。

Javaでは、Math.atanメソッド(π／２ ～ －π／２)とMath.atan2メソッド(π ～ －π)が用意されています。これらは、CLDCに含まれていないため、携帯Javaでは使用できませんが、DoJa4.0以降では、FastMath.atanメソッド(９０度～－９０度)、FastMath.atan2メソッド(０度～１８０度)が用意されています。

arctanの計算上、分母(ｘ座標の差)が０になる場合があります。この場合は計算できないのでＣ言語の場合、処理系によってはエラーになってしまいます。この場合、分母が０の場合の専用の処理が必要になります。

なお、ほとんどのプログラミング言語の三角関数は、角度をすべて"ラジアン"という単位で処理します。ここでの説明や図では"度"の単位になっていますが、実際にプログラムを作成する場合にはラジアンに対応させる必要があります。

ラジアンは、１回転を「２π」で表します。度は「３６０度」で１回転ですが、この３６０という数値には、実は根拠がありません。これに対してラジアンは、半径が１の円の円周と同じ２πが１回転なので、明確な根拠があります。この特徴を利用した近似計算が、いたるところで使われています。

## ｘ／ｙ方向の移動速度を計算する

これまで説明したように、プレイヤー機の方向は角度という形で得られます。次は、この角度から実際に画面上をキャラクタが移動するのに必要な、

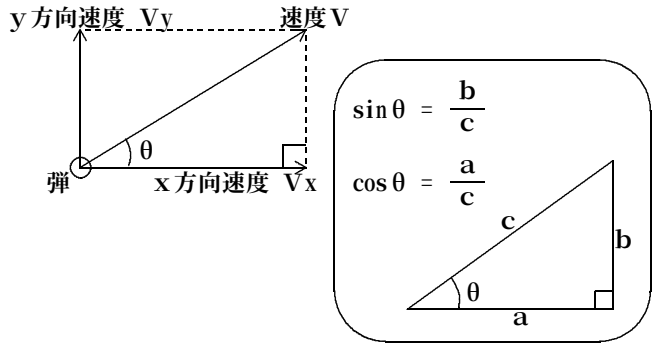
・ｘ方向の速度　Vx
・ｙ方向の速度　Vy

の２つを計算してみましょう。

このような処理には、三角関数のうちsinとcosを利用します。sinとcosともに、前述のように「角度から辺の長さの比を計算する」関数です。以下の図のように当てはめてやると、速度を１とした場合のｘ方向の速度はcosで、ｙ方向の速度はsinでそれぞれ計算できまることがわかります。
計算結果は、進行方向の速度を１としたものなので、これらの結果に実際の弾の速度Ｖを掛けてやれば、それぞれの座標軸方向への実際の速度が求まります。
よって、

$$Vx = V \cdot \cos\theta$$
$$Vy = V \cdot \sin\theta$$

とすることで、各座標軸方向への移動速度が計算できるわけです。




▲ｘ方向／ｙ方向への移動量

## 放射状に拡散する弾を実現しよう

方向の概念を使って実現しやすくなる弾の移動表現として、放射状に広がっていく弾があります。プレイヤー機を直接狙う弾の角度θをもとに、プラス方向とマイナス方向に角度αだけずらした弾を同時に発生させることによって、分裂して拡散する弾が作れます。
このような表現を「移動回数で分割する」アルゴリズムで実現しようとしても、ほとんど不可能です。「直接狙う弾をひとつだけ用意すればよい」という条件では、このアルゴリズムのほうが簡単です。しかし、ちょっとしたアレンジを付け加える場合には、角度を利用して取り扱うほうがはるかに有利になります。弾の個性に合わせて、アルゴリズムの選択をすればよいでしょう。