# 【ハンズオン】エネミーに視野角を実装してみる。

## step-1 敵兵の前方方向(enemyForward)を求める。

Sample\_04/Sample\_04.slnをダブルクリックして、プロジェクトを起動してEnemy.cppを開いて、リスト－1のプログラムを入力してください。

リスト－１

|  |
| --- |
| //step-1 敵兵の前方方向(enemyForward)を求める。  //前方方向は{0, 0, 1}のベクトルをm\_rotationで回して求める。  CVector3 enemyForward = CVector3::AxisZ();  m\_rotation.Multiply(enemyForward); |

　このサンプルの敵キャラクターは、全く回転していないときはX軸の方向を向いています。ですので、敵キャラの回転クォータニオンを使って、X軸のベクトルを回してやれば、現在の敵キャラクターの前ベクトルを求めることができます。

## step-2 敵兵からプレイヤーに伸びるベクトル(toPlayerDir)を求める。

　続いて、敵兵からプレイヤーに向かって伸びるベクトルを計算します。リスト－２のプログラムを入力してください。

リスト－２

|  |
| --- |
| //step-2 敵兵からプレイヤーに伸びるベクトル(toPlayerDir)を求める。  CVector3 toPlayerDir = m\_player->GetPosition() - m\_position;  //正規化を行う前に、プレイヤーまでの距離を求めておく。  float toPlayerLen = toPlayerDir.Length();  //正規化！  toPlayerDir.Normalize(); |

　step-3で内積を利用して、敵兵とプレイヤーの角度を調べるプログラムを作成するのですが、この時に必要なベクトルは単位ベクトルなので、正規化を行っています。また、step-5で敵兵とプレイヤーの距離も必要になるので、正規化を行う前に距離計算も行っています。

## step-3 enemyForwardとtoPlayerDirの内積を計算する。

　続いて、敵兵の前方ベクトルと、プレイヤーに向かって伸びるベクトルで内積を計算します。リスト３のプログラムを入力してください。

リスト－３

|  |
| --- |
| //step-3 enemyForwardとtoPlayerDirの内積を計算する。  float d = enemyForward.Dot(toPlayerDir); |

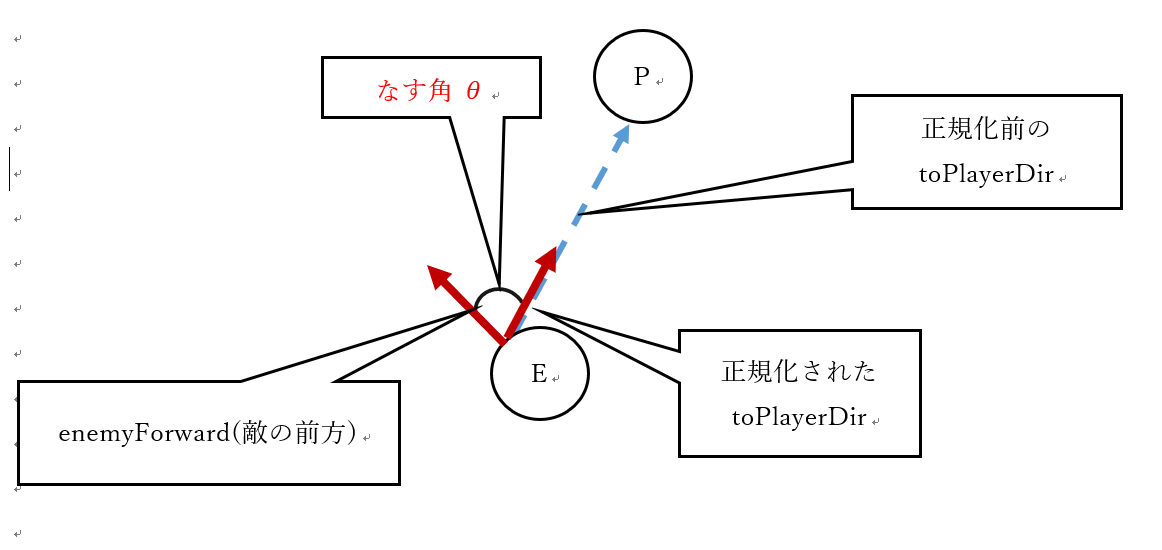
## Step-4 内積の結果をacos関数に渡して、enemyForwardとtoPlayerDirのなす角度を求める。

　内積の結果は二つのベクトルのなす角度のcosθとなっています。この値を逆三角関数に渡して、角度を求めましょう。リスト－４のプログラムを入力してください。

リスト－４

|  |
| --- |
| //step-4 内積の結果をacos関数に渡して、enemyForwardとtoPlayerDirのなす角度を求める。  float angle = acos(d); |

　このプログラムで下記の図のような角度を求めることができました。



## step-5 視野角判定

　では、最後のハンズオンです。ここまでで求めた結果を利用して視野角判定を行います。リスト－５のプログラムを入力してください。

リスト－５

|  |
| --- |
| //fabsfは絶対値を求める関数！  //角度はマイナスが存在するから、絶対値にする。  if ( fabsf(angle) < CMath::DegToRad(45.0f) //角度の判定  && toPlayerLen < 200.0f //距離の判定  ) {  //状態を追跡状態にする。  m\_state = eState\_Tracking;  } |

　視野角判定は角度だけではなく、距離の判定も必要になります。

## step-6 実行

　ここまで入力出来たら、プログラムを実行してみてください。正しく入力出来ていると敵兵がプレイヤーを発見すると追いかけてくるようになります。

【ハンズオン】エネミーを進行方向に向ける。

## step-1　atan2関数を使って、進行方向に向けるための角度を求める。

ここでは、敵の進行方向を利用して、回転させる角度を求めていきます。この角度を求めるためにatan2関数を利用します。  
　リスト－１のプログラムを入力してください。

リスト－１

|  |
| --- |
| //step-1 atan2を使って、進行方向に向けるための角度を求める。  float angle = atan2(m\_moveSpeed.x, m\_moveSpeed.z); |

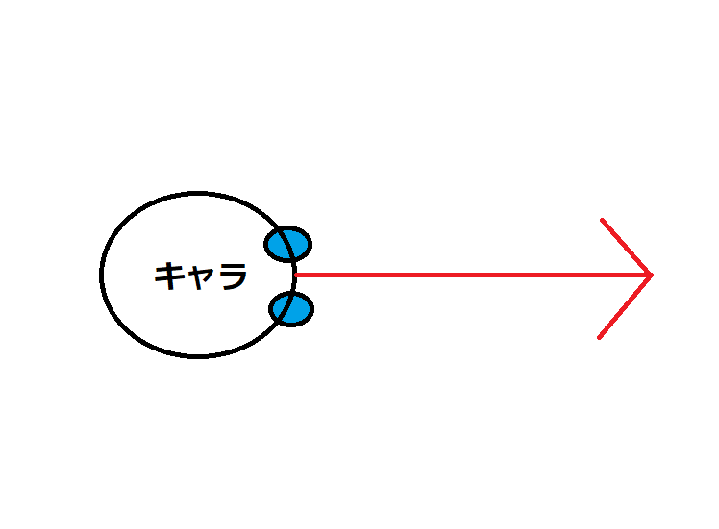
## step-2　求めた角度を使って、回転クォータニオンを求める。

回転させる角度を求めることができたら、あとは回転クォータニオンを作成するだけです。リスト－２のプログラムを入力してください。

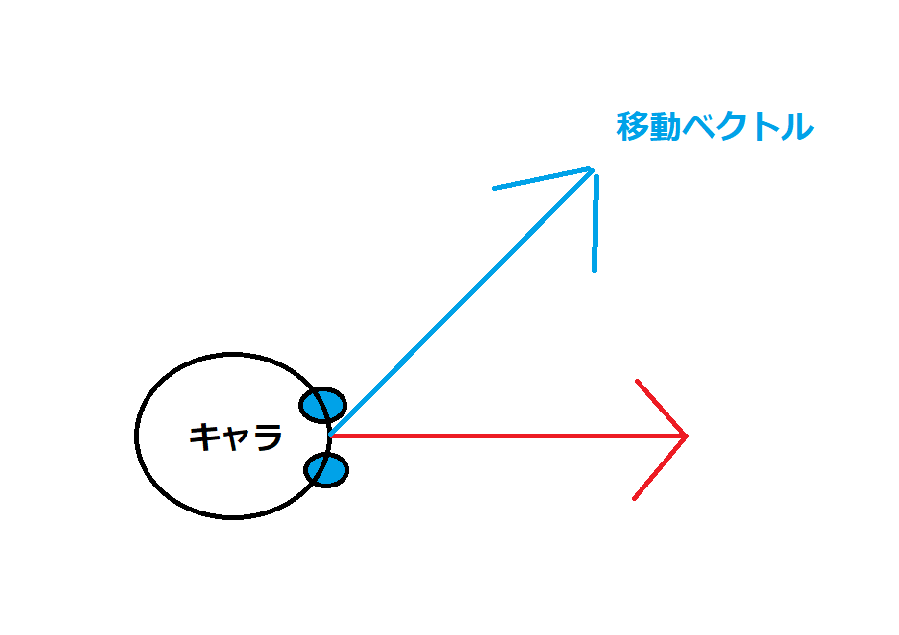
リスト－２

|  |
| --- |
| //step-2 求めた角度を使って、回転クォータニオンを求める。  m\_rotation.SetRotation(  CVector3::AxisY(),  angle  ); |

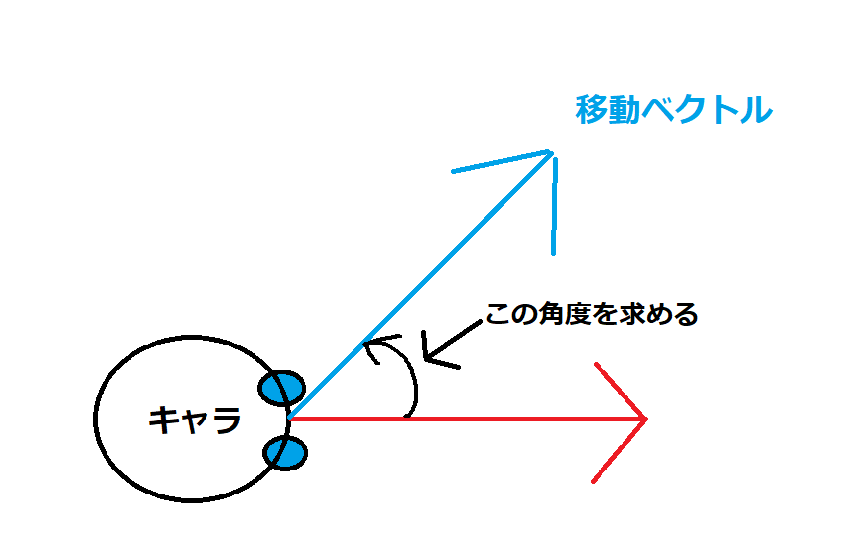
　では、なぜこのプログラムでキャラが進行方向を向くのか考えてみましょう。例えばキャラクターが回転していないときはX軸を向いている場合を考えます。



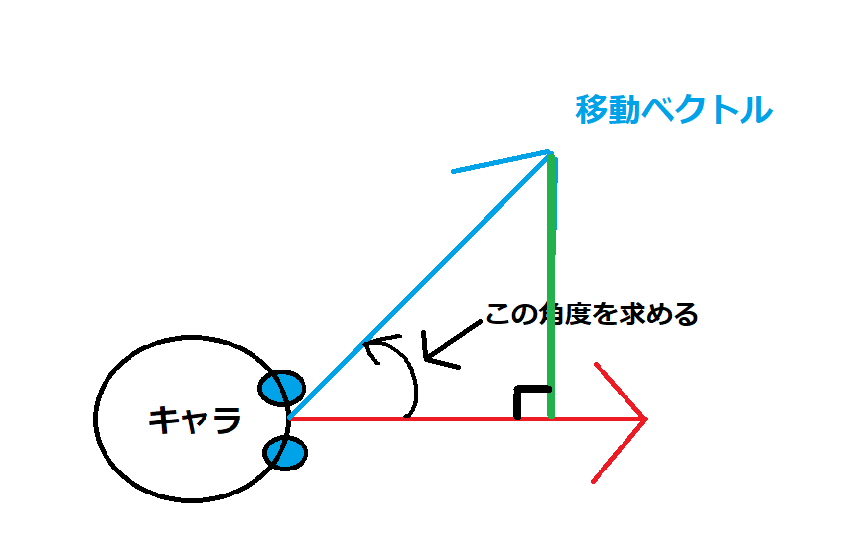
　このキャラを青い矢印の方向に進ませるケースを考えます。



　この時、キャラを進行方向に向けるためには赤い矢印を青い矢印に回すための回転角度を求めればいいことになります。



　ここで補助線を引いてみると、赤い矢印と青い矢印とで直角三角形が作れることが分かります。



　この直角三角形の２辺の長さは、ちょうど移動ベクトルのX成分とZ成分の値になります。直角三角形の２辺の長さが求まれば、この直角三角形のcosθ、sinθ、tanθのどれかの値を求めることができます。今回はZ ÷ Xを計算することで、tanθを求めることができます。tanθが求まれば、その値をatan関数に渡してやれば、角度を求めることができます。あとは、この角度を使って、Y軸周りの回転クォータニオンを作成するだけです。

## 【実習】プレイヤーを進行方向に向かせる

　Sample\_04/Player.cppを改造して、プレイヤーを進行方向に向かせるプログラムを実装しなさい。