自機のコリジョンについて

2013/06/28

高橋友也

●コリジョン判定の流れ PlayerGrid::Update

1. 入力情報の取得
   1. PlayerGrid::GetStickDir
   2. PlayerGrid:: GetStickDirBehindCamera
2. 次の行動に入れる状態かチェック
   1. PlayerGrid::IsStartMove
3. 入力情報から、移動後の座標を求め、コリジョン判定し、今フレームの行動を決定
   1. FieldUnit::GetMoveOrder
4. 動作開始前の初期化
   1. FieldUnit::InitMove
5. 動作の設定 アクションコマンドの発行を行います。
   1. FieldUnit::SetMove

●③の挙動の詳細

（一番複雑なアナログ移動 自転車で解説　ソースを追いながら読んでください。）

1. FieldUnit::GetMoveOrder
   1. Formごとの処理分岐をします。
2. FieldUnit:: GetMoveOrderCycle
   1. 直前の行動ごとに処理分岐をします。
3. FieldUnit:: GetMoveOrderCycle\_Walk
   1. GetMoveOrderCycle\_WalkBase内で、入力情報から行動を決定し、
   2. GetMoveOrderCycle\_HitCheck内で、コリジョンチェックします。
4. FieldUnit::GetMoveOrderCycle\_HitCheck
   1. GetMoveOrder\_HitCheckAnalogNormalが実態
   2. 引数で渡す、FieldUnit::GetOrderCycleDirは、そのフレームの移動量と方向を取得するための関数
5. FieldUnit::GetMoveOrder\_HitCheckAnalogNormal
   1. FieldUnit::GetMoveOrder\_HitCheckCoreは、アトリビュートなどにより、ジャンプなど特殊な移動に変換する処理です。
6. FieldUnit::GetMoveOrder\_HitCheckAnalog
   1. m\_pGetMoveOrderFuncで、入力状態から、移動量、方向を取得します。
   2. HitCheckMove関数で、コリジョンチェックを行います。コリジョンに引っかかった場合、引数に渡した、slice\_check\_tblに壁ずり補間の候補が返ってきます。
   3. 壁ずり動作の場合。
      1. FieldUnit::SetSliceMoveParameterで、移動スピードなどを、壁ずり補完時ように変換します。
      2. FieldUnit:: SetMoveForceWayTempSave、FieldUnit::SetMoveForceWayで、MoveModelBaseが保持している移動スピード、方向を壁ずり動作時の初期値に変更します。
      3. M\_pGetMoveOrderFuncで、壁ずり時の今フレームの移動スピード、方向を取得します。
      4. FieldUnit::SetMoveForceWayTempRecoverで、MoveModelBaseのパラメータをもとに戻します。
      5. FieldUnit::HitCheckMoveとm\_pHitCheckCoreFuncで、壁ずり移動で、移動できるかのチェックを行います。
      6. ⑤までの工程を、壁ずり動作が可能になるか、slice\_check\_tblの要素がなくなるまでループします。

●HitCheckMove関数内の解説

　FieldUnit::HitCheckMoveには、グリッド移動用とアナログ移動用がありますが、

　より複雑なアナログ移動用の解説を行います。

1. 大雑把に下記の工程に分かれます。
   1. ①　オリジナルポジションの更新
   2. ②　オリジナルの移動方向と、半グリッド先まで進めるかのチェック用移動量の算出
   3. ③　オリジナルの移動先と半グリッド分の移動先に侵入可能かチェック
   4. ④　斜め方向移動時の３点チェック
   5. ⑤　移動のリミットチェック　（壁に食い込む場合に、どこまで移動できるのかのチェック）
   6. ⑥　壁にぶつかった場合の、次にチェックする方向の算出
2. ①オリジナルポジションの更新
   1. 関数の引数で受け取った、wayとdistそのままの方向、量移動したポジションを求めます。
3. ②オリジナルの移動方向と、半グリッド先まで進めるかのチェック用移動量の算出
   1. オリジナルの方向、移動量から、移動成分を半グリッドのサイズにした移動量のベクトルを生成します。
   2. このベクトルは、人物のサイズを考慮した移動先のチェックのために使用します。
4. ③オリジナルの移動先と半グリッド分の移動先に侵入可能かチェック
   1. オリジナル移動量のポジションと半グリッド移動量のポジション両方で、コリジョンチェックを行い、両方が移動可能かチェックします。
   2. ここで、移動不可能な場合、もう、この先のチェックは行わず、⑥壁にぶつかった場合の、次にチェックする方向の算出　に進みます。
5. ④斜め方向移動時の３点チェック
   1. X方向にだけ移動した場合とZ方向にだけ移動した場合のコリジョンチェックを行います。
   2. ここで、どちらかが移動不可能になった場合は、⑥壁にぶつかった場合の、次にチェックする方向の算出　に進みます。
6. ⑤移動のリミットチェック　（壁に食い込む場合に、どこまで移動できるのかのチェック）
   1. オリジナル移動先を基準にした8方向の半グリッド先に食い込まないか詳細なチェックを行います。
   2. チェックには、斜め壁と通常壁の2系統があり、どちらかを通ります。
      1. checkSliceLimitCheck　ななめ壁
      2. getNearWallMask　と　getNearWallMaskLimit　通常壁
   3. 求めた移動limit　の値から、すでに壁にめり込んでいるパターンを洗い出します。

●checkSliceLimitCheck関数内の解説

1. GetSliceHitchWay
   1. 斜めアトリビュートとHitckアトリビュートをチェックし、斜め壁の方向を求めます。
2. PosSliceLine
   1. Original\_posとさらに半グリッド進んだ先で、壁ベクトルとの交点を求め、好転がある場合は、壁までの距離をp\_limitに格納します。
3. PosSliceWall
   1. 壁の方向にさらに進んだ場合に、壁があるかどうかをチェックし、壁がある場合は、その壁に食い込まないように、p\_limitを再度調整する。

●getNearWallMask　と　getNearWallMaskLimitの解説

1. FieldUnit::getNearWallMask
   1. Original\_posがグリッドを4つに分割した場合の、どのセクターに所属するのかを確認し、さらに1グリッド先が壁かどうかをチェックする。
   2. ヒットした方向をhitとp\_hit\_result\_alldirに格納
2. FieldUnit:: getNearWallMaskLimit
   1. getNearWallMaskで求めたhitとp\_hit\_result\_alldirから、移動制限までの距離を求める。

以上です。