ポジションについて

2013/06/28

高橋友也

●概要

ポジションを扱うクラス、Gridクラス、Positionクラスは、

グリッドマップ、レールマップ双方での、座標の扱いを同じインターフェースで行えるように作成したクラスになります。

このクラスを利用することで、

グリッドマップ、レールマップを意識せずに、処理を実装することができます。

●クラスの説明

Grid　　 ：　グリッド座標での操作を行う。

Position　：　通常スケールでの座標の操作を行います。

●構造体の説明

PositionData　：　Gridクラス、Positionクラスで管理するデータの塊

　　　　　　　　　この構造体をGrid,Positionクラスは操作する。

　Gridが保持ているときには、グリッド座標が入っているので注意！

　また、この構造体の中身を直接アクセスすることは原則禁止とする。

　　　　　　　　　直接アクセスする場合には、必ずtypeをチェックして、

　　　　　　　　　グリッドマップ、レールマップどちらのデータなのかを確認する。

●注意ポイント

　▽RoundOffについて

　　Positonクラスでは基本使用しないことを推奨する。

　　　小数点単位でのコリジョンチェックが必要なマップでは、

　　　これにより、侵入不可能な座標にジャンプしてしまうこともある。

　▽AddPosForPlayer

　　こちらは、C02用自機移動用の関数になる。

　　レール＋ビハインドマップ以外では使用しない。

　　引数に渡す、値のマトリクスに違いがある。

　　通常＝レールローカル空間

　　ForPlayer＝ワールド空間

　▽Fast系関数

　　これらの関数は、ポジションを進めた後に、front side方向の再計算をカットしている。

　▽保持しているfront upについて

　　ここで保持しているfront upは、若干ややこしい持ち方をしている。

　　グリッドマップ＝front upは一律（0,0,1）(0,1,0)

　　レールマップ =　ラインのfront方向　up方向

　　が保持されている。

　▽ポジションタイプがRailになっている際のVector型の座標取得

　　こちらは、処理負荷が高くなるので注意が必要

　　関数内で毎回Pailポジションから３D座標計算をしている。

　　逆にも同じことがいえる。

　▽３D座標からレールポジションへの変更

　　こちらはさらに注意が必要で、処理負荷が高いだけでなく、

　　実際には３D座標の位置にレールが無い可能性もある。

　　また、精度も高いとは言えず、変換先がコネクトラインである必要がある。

以上。