第５回　当たり判定２

今回はマップチップとの当たり判定について解説します。

マップチップとの当たり判定は、

①押されているボタンから自機の速度が決まる。

②その速度を自機の座標にたした場合マップチップと当たるか調べる。

③当たってしまう場合自機がマップチップに当たらないように速度を調整する。

④自機の座標に調整された速度をたす。

という流れの中の②と③の部分です。

上の様に文章で書くとたった 2 行ですみますがこれをプログラムで表そうとすると、上下左右どの方向から当たったかによって速度の調整の仕方が変わってしまうので、当たり判定で縦方向から当たったのか横方向から当たったのかを判定を、速度の±で上下(左右)どっちから当たったかを判定を、さらに 4 種類それぞれの速度の調整を書かないといけないのでとても長い文になってしまいます。

それを書いたのが下のプログラムです。

```
void g_jiki_tip( void){
    //自機とチップとの当たり判定と、当たっている場合はその補正をする関数
    //for文で全てのチップとの当たり判定を行う頭の悪い方法です
    //
    //具体的な方法は
    //1.(現在の座標)+(速度)がチップと当たるのならば
    //2.仮の速度を一旦ゼロにし、for文を使い、１ずつ速度を増やしていき実際に当たる最小値を探します
    //3.その最小の値マイナス１がギリギリあたらないラインなので､そこで速度を寸止めさせます
    //4.そしてその速度を実際の速度に代入させます
    //これをx座標、y座標ともに計算させます
    //ちなみに斜めにチップにあたるとバグりますｗ

    int i;
    int j;
    for( i = 0; i <= TIP_MAX_X; i++){
        for( j = 0; j <= TIP_MAX_Y; j++){
            if( tip[i][j] == 1){
                //for文の二重ループとif文でチップがあるかを探す

                int k;
                if( g_atari( jiki.y + jiki.sokudo_y, jiki.takasa, j * TIP_TAKASA,
                TIP_TAKASA)){
                    if( g_atari( jiki.x, jiki.haba, i * TIP_HABA, TIP_HABA)){
                        //現在のy座標+y成分の速度が当たって、尚且つ今のx座標で当たるのならば
                        if( jiki.sokudo_y > 0){
                            //下に向かって速度があるとき
                            for( k = 0; ; k++){
```

```
                                if( g_atari( jiki.y + k, jiki.takasa, j * TIP_TAKASA,
                                TIP_TAKASA)){
                                    jiki.sokudo_y = k - 1;
                                    jiki.flag = 1;
                                    break;
                                    //最小の値マイナス１を代入しブレイクします
                                }
                            }
                        }
                        else if( jiki.sokudo_y < 0){
                            //こちらは上に向かっているとき
                            for( k = 0; ; k--){
                                if( g_atari( jiki.y + k, jiki.takasa, j * TIP_TAKASA,
                                TIP_TAKASA)){
                                    jiki.sokudo_y = k + 1;
                                    break;
                                }
                            }
                        }
                    }
                }

                if( g_atari( jiki.x + jiki.sokudo_x, jiki.haba, i * TIP_HABA, TIP_HABA)){
                    if( g_atari( jiki.y, jiki.takasa, j * TIP_TAKASA, TIP_TAKASA)){
                        //こちらはx座標の判定。チップの側面ってことね。構成は変わらず
                        //つーかfor文のifの機能使えばもっときれいになるね
                        if( jiki.sokudo_x > 0){
                            for( k = 0; ; k++){
                                if( g_atari( jiki.x + k, jiki.haba, i * TIP_HABA,
                                TIP_HABA)){
                                    jiki.sokudo_x = k - 1;
                                    break;
                                }
                            }
                        }
                        else if( jiki.sokudo_x < 0){
                            for( k = 0; ; k--){
                                if( g_atari( jiki.x + k, jiki.haba, i * TIP_HABA,
                                TIP_HABA)){
                                    jiki.sokudo_x = k + 1;
                                    break;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

コメントアウトでほとんど説明されていますが、この文では、現在の y 座標+y 成分の速度が当たって、尚且つ今の x 座標で当たるのならば縦方向から当たった判定し、縦方向から当たってる場合、自機の縦方向の速度を調べ速度が＋(下方向に進んでいる)の場合上から、－(上方向に進んでいる)の場合下から当たったと判定しています。同様にして横方向の当たり判定もしています。

また自機がマップチップに上から当たっていると判定されたときに出てきている jiki.flag というのは、自機が地面に接しているかいないかを表す変数で、1 のときに接していて、0 のときには接していないことを表しています。

それでは実際にこの関数を実行させるために、**g_jiki 内に g_jiki_tip();を追加しましょう。**

ここで注意しなければいけないのは最初に書いた①～④の流れの通りにするために、

①の処理を行う

g_jiki_sousa();**と**

④の処理を行う

```
jiki.x += jiki.sokudo_x;
jiki.y += jiki.sokudo_y;
```

の間に g_jiki_tip();**を追加しないといけないので気をつけてください。**