ソフトゼミ B 2015 第 4 回

当たり判定

➢ 当たり判定とは

当たり判定とは、A と B(A や B は自機や敵、弾やアイテムなどのいずれかを指します)が 2 次元平面上で接しているか、または重なっているかを判定させるものです。今回のシューティングゲームでは、自機または敵の hp を弾の攻撃力だけ減らしたり、その時に弾を消したり、といった処理を行っていきます。

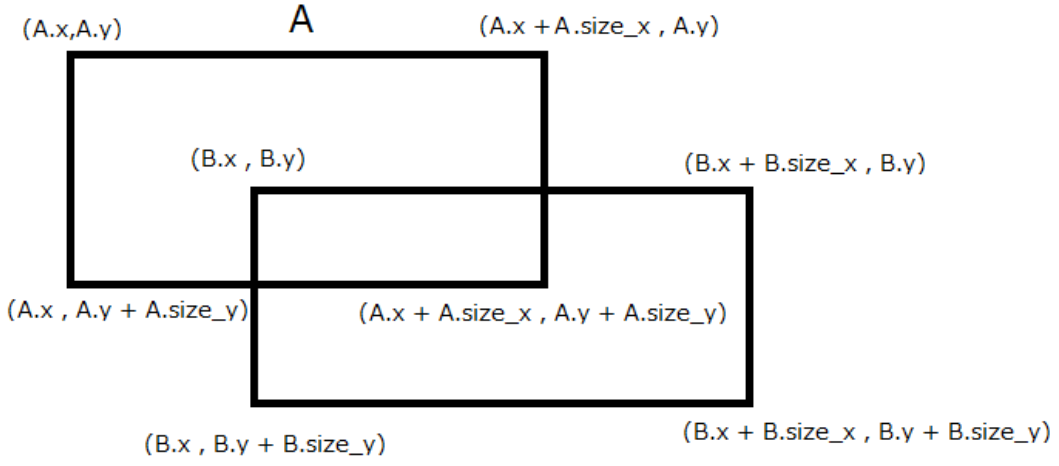
```
if(A と B が重なっているか判定する条件式)
{
        (処理する内容)
        //例えば、player.hp-=bullet.atk;など
}
```

➢ 当たり判定の条件式

今回は自機、敵、弾、アイテムすべてが四角形なので、四角形で説明します。2 つの四角形が重なっているかどうかの条件式は下の図で考えると、次のようになります。

A.x < B.x + B.size_x &&
A.x + A.size_x > B.x &&
A.y < By + B.size_y &&
A.y + A.size_y > B.y

このような文になる理由はゼミでも説明しますが、自分で具体的に考えてみるとさらに理解が深まると思います。



➢ プログラムの穴埋め

では実際にプログラムの穴埋めを行っていきましょう。

```
/*以下、4 つの関数が今回追加されたものです。
 「自機と敵」「自機の弾と敵」「敵の弾と自機」「敵の弾と自機」「自機とアイテム」の 4 つの場合に分けて処理を行います。
 あたり判定において共通している条件が、
        「A の左端が B の右端よりも左にある」かつ
        「A の右端が B の左端よりも右にある」かつ
        「A の上端が B の下端よりも上にある」かつ
        「A の下端が B の上端よりも下にある」
 ということです。
        「左端」= x 座標
        「上端」= y 座標
        「右端」= 「左端」+ 「x 方向の大きさ」
        「下端」= 「上端」+ 「y 方向の大きさ」
 です。*/

//あたり判定、自機と敵
void judge_player_to_enemy()
{
        //すべての敵の
        for(int i=0;i<【穴埋め 1】;i++)
        {
                //存在しているものが
                if(enemy[i].hp!=0)
                {
                        //自機と重なっていたら
                        if(     enemy[i].x < 【穴埋め 2】 &&
                                enemy[i].x + enemy[i].size_x > player.x &&
                                enemy[i].y < player.y + player.size_y &&
                                 【穴埋め 3】 > player.y )
                        {
                                player.hp -= enemy[i].hp; //自機の hp から敵の hp 分減らして
```

```
                                        if(player.hp<0) player.hp=0; //自機の hp がマイナスに
ならないように
                                        enemy[i].hp=0; //敵の hp を 0 にする
                                }
                        }
                }
}

//あたり判定、自機の弾と敵
void judge_playerbullet_to_enemy()
{
        //すべての敵で
        for(int i=0;i<ENEMY;i++)
        {
                //存在しているやつと
                if(enemy[i].hp!=0)
                {
                        //自機のすべての弾で
                        for(int j=0;j<BULLET;j++)
                        {
                                //存在しているものが
                                if(【穴埋め 4】)
                                {
                                        //重なっていたら
                                        if(     playerbullet[j].x < enemy[i].x +
enemy[i].size_x &&
                                                【穴埋め 5】  > enemy[i].x &&
                                                playerbullet[j].y < enemy[i].y +
enemy[i].size_y &&
                                                【穴埋め 6】 )
                                        {

                                                playerbullet[j].hp=0; //弾の存在を消
し
                                                enemy[i].hp -= playerbullet[j].atk; //
敵の hp から弾の atk 分減らして
```

```
                                                    if(enemy[i].hp<0) enemy[i].hp=0; //
敵の hp がマイナスにならないように
                                                    if(enemy[i].hp==0)
create_item( enemy[i].x,enemy[i].y,enemy[i].size_x,enemy[i].size_y ); //敵を倒したらアイ
テムを作る
                                                }
                                            }
                                        }
                                    }
                                }
}

//あたり判定、敵の弾と自機
void judge_enemybullet_to_player()
{
        //すべての敵の
        for(int i=0;i<ENEMY;i++)
        {
                //すべての弾で
                for(int j=0;j<BULLET;j++)
                {
                        //存在しているものが
                        if(enemybullet[i][j].hp)
                        {
                                //enemybullet は 2 次元配列になっていることに注意し
てください。
                                //自機と重なっていたら
                                if(     【穴埋め 7】 < player.x + player.size_x &&
                                        【穴埋め 8】> player.x &&
                                        【穴埋め⑨】 < player.y + player.size_y &&
                                        【穴埋め 10】 > player.y )
                                {

                                        enemybullet[i][j].hp=0; //弾の存在を消し
                                        player.hp -= enemybullet[i][j].atk; //自機の
hp から弾の atk 分減らして
```

```
                                        if(player.hp<0)
                                        {
                                                player.hp=0;//自機の hp がマイナス
にならないように
                                        }
                                }
                        }
                }
        }
}




//あたり判定、自機とアイテム
void judge_player_to_item()
{
        //すべてのアイテムで
        for(int i=0;i<ITEM;i++)
        {
                //存在しているものが
                if(item[i].hp!=0)
                {
                        //あたり判定の条件文を自分ですべて書いてみましょう。
                        //自機と重なっていたら
                        if(【穴埋め 11】)
                        {

                                player.score +=100;//得点を加算
                                player.hp+=3;//体力回復
                                if(player.hp>200)
                                {
                                        player.hp=200;
                                }
                                item[i].hp=0;   //アイテムを消す
```

```
                        }
                    }
                }
}
/*今回新たに上記の 4 つの関数が追加されたので、しっかりと処理を行うように update 関数に記述しておく必要があります。
*/

void update()
{
        //キーボード入力状況を取得
        GetHitKeyStateAll(key);

        //flag の中に何があるかで処理を変える
        switch(flag)
        {
            case title: //title のとき
                //Enter が入力されたら
                if(key[KEY_INPUT_RETURN])
                {
                    flag=gameplay; //flag の中身を gameplay に変更
                    initialize(); //初期化関数を呼び出す
                }
                break;

            case gameplay:  //gameplay のとき
                //移動の関数
                move_player();
                move_enemy();
                move_playerbullet();
                move_enemybullet();
                move_item();
                move_background();
                //いろいろと作る関数
                create_enemy();
                create_playerbullet();
```

```
                create_enemybullet();
                /*関数を書く順番は順不同です*/
                【穴埋め 12】
                【穴埋め 13】
                【穴埋め 14】
                【穴埋め 15】
                //自機の hp が 0 になったら
                if(player.hp==0)
                {
                        flag=gameover;  //flag の中身を gameover に変える
                }
                break;

        case gameover: //gameover のとき
                //X が入力されたら
                if(key[KEY_INPUT_X])
                {
                        flag=title; //flag の中身を title に変える
                }
                break;
    }
}
```