ソフトゼミ B　第 4 回

当たり判定

ソフトゼミ B 第 4 回です。今回は当たり判定についてです。

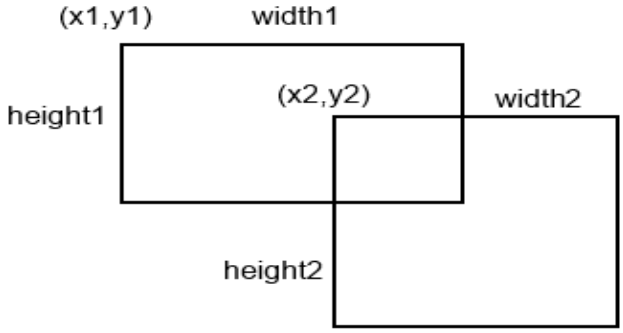
◆　当たり判定とは

今回は 2 次元平面上での当たり判定について解説します。そもそも当たり判定って何？という方もいると思います。簡単に言ってしまえば、2 つのもの（今回のシューティングゲームで言えば自分の弾と敵本体など）が重なっているかどうかを判定して、もし重なっていたら何か処理をする（自分の弾を消す、敵の HP を減らすなど）というものです。

```
if(何か 2 つの物体が重なっているかどうかを判定する条件式){
        //もし重なっていれば
        (片方について何か処理をする)
        (もう片方について何か処理をする)
        (重ならないように座標をずらす)
        //などなど、適切な処理を行う
}
```

◆　当たり判定の条件式（四角形と四角形の場合）

2 つの四角形が重なっているかどうかを判断する場合は、下の図を見ればなんとなく分かると思いますが、左上の座標だけではなく右下の座標も必要になります。プログラム上で扱われている自機や敵の座標を表す x,y は、**左上**の座標を表す変数です。では右下の座標はどうするかというと、これは簡単で、左上の座標に幅と高さを足せば OK です。



このような図の場合は、

条件式： x1+width1>x2&&x2+width2>x1&&y1+height1>y2&&y2+height2>y1

となります。前半が x 座標についての当たり判定、後半が y 座標についての当たり判定です。どうしてこのようになるのかは、ゼミ中でも解説しますが、自分で考えてみるとなお理解が深まると思います。また、余裕がある人は、今回のゼミでは扱いませんが、四角形以外の図形（円と円など）の当たり判定も考えてみるとよいでしょう。

◆ プログラムの穴埋め

では前回までと同様に、プログラムの穴埋めをしていきましょう。

```
//////////////////////////////////////////////////////省略//////////////////////////////////////////////////////

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

                        【穴埋め】=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)
                    【穴埋め】
                    //敵を倒したらアイテムを作る*/
                        }
                    }
                }
            }
        }
}

//あたり判定、敵の弾と自機
voidjudge_enemybullet_to_player(){
    //すべての敵の
    for(inti=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){
                //自機と重なっていたら
                if( enemybullet[i][j].x <【穴埋め】&&
                【穴埋め】>player.x&&
                enemybullet[i][j].y <【穴埋め】&&
                【穴埋め】>player.y ){

                    【穴埋め】=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、自機と敵
void judge_player_to_enemy(){
    //すべての敵の
    for(inti=0;i<ENEMY;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(【穴埋め】&&
             【穴埋め】&&
             【穴埋め】&&
             【穴埋め】 ){
                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
            }
        }
    }
}
        ////////////////////////////////////////////////////省略////////////////////////////////////////////////////
//数値的な更新をする関数
void update(){
    //……省略……
    //いろいろと作る関数
    create_enemy();
    create_playerbullet();
    create_enemybullet();
    //あたり判定関数
     【穴埋め】;
     【穴埋め】;
     【穴埋め】;
    //【穴埋め】;        //ここは追加要素なのでやりたい人は追加資料を見てください
        ////////////////////これ以降は前回までと変わらないので省略////////////////////////////
```