ソフトゼミ B　第 6 回(アクションゲーム)

キャラの移動と重力

ゼミ B もそろそろ終盤に入ります。シューティングゲームの創作法も学び終わり、なんとなくゲーム制作の感覚がつかめてきた頃合いではないでしょうか？

とまあ前置きもほどほどに、本題に入っていこうと思います。

✤アクションゲームとは

さて、今回から皆さんにお伝えする課題は、「アクションゲームの制作」です(とは言っても「アッ、ハイ」ってなる方も多いと思うので、漠然と「マ○オ的なナニか」を作るととらえて頂けると幸いです)。

初めに、次の項目に目を通してください。

● 今回お触りする主な関数

✧ void Map(void)・・・マップのファイルを開き、ファイルの内容を読み込みます。
(ファイルの開き方は後述します)

✧ void move(void)・・・主人公の移動に関する関数。

✧ void enemymove(void)・・・敵の移動に関する関数。

✧ void draw(void)・・・ゲームの描画に関する関数。

以上、4 つの関数について今回は解説していきます。

１、マップチップ

最初に、アクションゲームにおいて重要な要素となるマップチップというものについて説明します。

まず初めに、HEIGHT_SIZE(縦の長さ)×WIDTH_SIZE(横の長さ)の長方形のマップを作るため、map[WIDTH_SIZE][HEIGHT_SIZE]という配列を用意しておきます。

次に、void Map(void)関数を見てください。

この関数では、簡単に言うと fopen(“/*ファイル名*/”,”/*モード(r)*/”)関数で指定したファイルを開き、fscanf(/*ファイルポインタ*/,”%d”,&/*変数*/)関数で開いたファイルを読み込み、その中身を配列 map に入れていくという作業を行っております。

(fopen のモードについてここで説明すると、混乱を招く恐れがあるので、どうしても気になって夜しか眠れないという方がいましたら、個人的に聞きに来るか自分で ggr なりしてください。)

さてさて、そんなこんなで今回お配りしたマップファイルを読み込んでいくと、配列 map には、以下のように数字が入ったことになります。

map[i][j]の中身

map配列の縦の値(j) ＼ map配列の横の値(i)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

ここまでご理解いただけたら、次は **draw(void)**関数の「マップチップからの描画」に目を移してください。

ここでは switch 文により、**map** 配列の要素の値それぞれに対応した画像(ブロック)を表示するように書かれています。この switch 文を for 文で回すことにより、1 マス大きさ 32(px)の 15(縦)×35(横)マスのマップを作ることができます。

複数のステージを作るアクションゲームでは、マップチップの使い方を覚えておくと便利です。

2、キャラクターの移動

次は主人公と敵の移動についてご説明します。

アクションのキャラクターの移動で、シューティングと異なる点があるとすれば、それは**重力がかかる**ということです。一見難しく感じるかもしれませんが、要は**キャラクターが空中にいる時に、y 軸のプラス方向の速度(vy)に加速度(今回は１)を加える**という作業を追加するだけです。

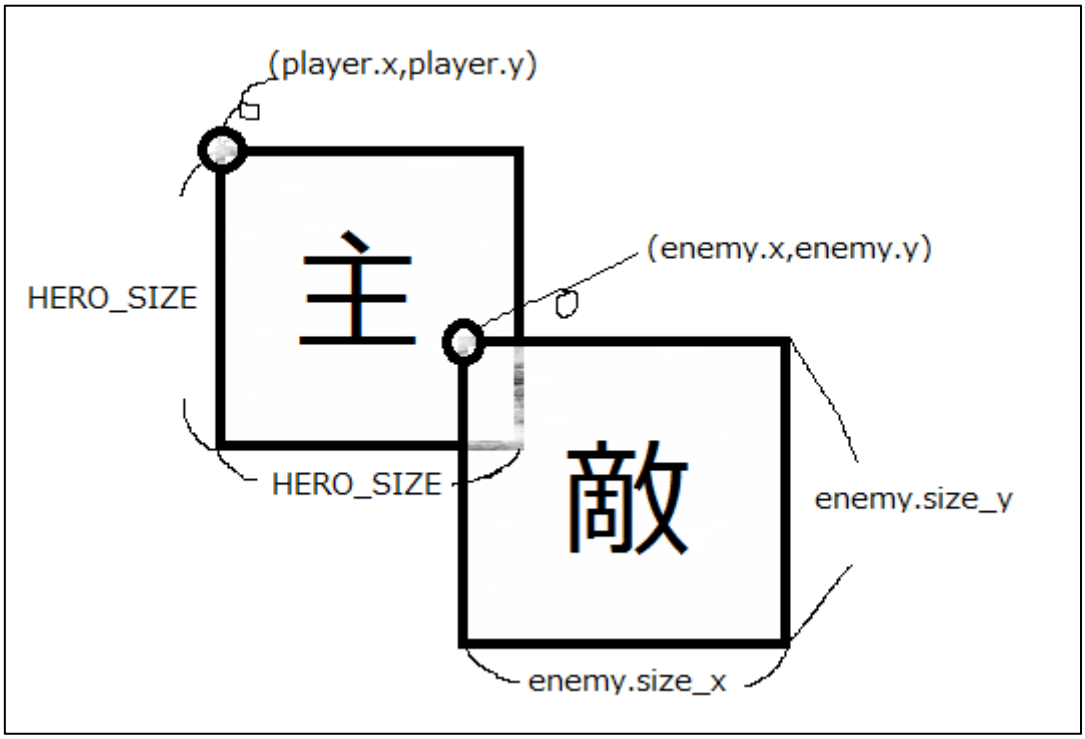
move(void)関数を見てください。ここでは、プレイヤーが操作するキャラクターの移動について書かれています。穴埋めが用意してあるので、上記のことを踏まえた上で 6－1～6～3 を解いてみてください。(わからないことがあれば、先輩たちに遠慮なく聞いてください)。

穴埋めが終わったら、次は **enemymove(void)**関数を見てください。今回の敵は、横方向にのみ移動し、一定間隔(50)移動したら折り返し往復する仕様になっています。ここにも穴埋めを用意してあるので、6－4 を解いてみてください。

3、敵との接触判定

ここで今回の折り返し地点、敵との当たり判定です。

これに関しては絵で説明する方が早いと思うので、以下の図を見てください。



上の図で主人公と敵の座標の関係がつかめたら、**move(void)**関数の中にある穴埋め 6－5、6－6 を解いてみてください。

穴埋め問題は以上となります。全て埋まったら、いよいよアクションゲームの起動です。ドキドキの瞬間ですね()。実行したらしばらく動かして遊んでみてください。

４、画面のスクロール

ゲームは実行できたでしょうか？実行できた方は、キャラクターを動かしているうちにあることに気づいたのではないでしょうか。

そう、主人公の動きに合わせて画面がスクロールしています。

draw(void)のマップチップの for 文が、以下のようになっていると思います。

```
for (int j = 0; j < HEIGHT_SIZE; j++) {
        for (int i = (player.x / BLOCK) - 9; i < (player.x / BLOCK) + 12; i++)
                ・・・(略)
```

j(縦)の範囲については特に問題はないのですが、重要なのは i(横)の動く範囲です。

ここで言っていることを端的に表現すると、「主人公の存在するブロックの 9 マス前から 11 マス先まで」となります。

慣れないと少し難しかもしれませんが、画面スクロールが使えるようになると作れるステージの幅が広がるので、紙などに自分で絵を描いて座標を書き込んでみたりするとわかりやすくなると思います。（一人ではお手上げ！という方は先輩に気軽に聞いてください）

以上で今回のゼミは終わりになります。

「時間が余って暇！」という人は伝えてくれれば追加課題を教えるので、ぜひお申し付けください