第７回　画像のスクロールとテキスト読み込み

前回までのプログラムでゲームとしてはほぼ完成形になったといっても過言ではありません。今回はそれに付け加える形となります。

テキストの読み込み

今までは、マップチップがあるのかないのかというデータをプログラム上に直接書き込んでいました。動きとしてはそれで問題はないのですが、マップを編集したりするのに非常に不便です。そこで外部のテキストデータからデータを読み込んで、よりわかりやすいプログラムにしてみましょう。

FileRead～

外部からファイルを読み込むのに使う関数は主に３つです。

・FileRead_open(“”『テキストファイルの場所』”);

テキストファイルを読み込むためのハンドルを持ってきてくれる関数です。

int 型の値を返してくるので、int 型の変数に格納しましょう。

・FileRead_scanf(“”『ハンドルをしまった変数』”, 書式指定文字, &変数名);

ハンドルをしまった変数が表わすテキストファイルからデータを読み込む関数です。

最初に読み込むファイルのハンドルが必要な以外、scanf と同じ感じです

使い方も scanf と同じです。

・FileRead_close(“”『ハンドルをしまった変数』”);

FileRead_open で開いたファイルを閉じる関数です。開けたら閉じる、生活の基本です。

では実際にプログラムに組み込んでみましょう。

まずはプログラムの中で読み込むためのテキストファイルを用意しましょう。テキストの書式は自分で決めてかまわないのですが、今回は**x 座標の値, y 座標の値, フラグ**という形で書いていきたいと思います。

```
1, 1, 0
2, 1, 0
3, 1, 0
……
TIP_MAX_X, 1, 0
1, 2, 0
2, 3, 0
……
TIP_MAX_X, TIP_MAX_Y, 0
```

今度はそれを読みこむプログラムを実際に書いてみようと思います。

```
void g_format( void){
        //ゲームの初期化

  int i;
  int j;
  int fp;
  ～中略～
  for( i = 0; i <= TIP_MAX_X; i++){
        for( j = 0; j <= TIP_MAX_Y; j++){
                tip[i][j] = 0;
        }
  }
  //チップを初期化
  //tip[i][j] == 1でチップが存在する。０ならなし。

  fp = FileRead_open( "data/stage.txt");
  //テキストファイルの読み込み
  if( fp != 0){
  //fp=0の時がエラー、エラーなのに無理やり読み込むと不吉なので一応if文で不吉を回避
        int x;
        int y;
        int value;
        //ここで宣言してもいいよね？一応Cの変数宣言の位置としてはOKな場所だしｗ
        while( FileRead_eof( fp) == 0){
                //行末まで読み込む
                FileRead_scanf( fp, "¥t%d,¥t%d,¥t%d", &x, &y, &value);
                tip[x][y] = value;
                //'¥t'を使っているのはテキストファイルを読みやすくしているため
                //エスケープ符号はあんま使わないほうがいいかなぁ？
        }
        FileRead_close( fp);
        //ここ重要！！
  }
～以下略～
```

こんな感じです、テキストファイルにマップの状況を書き込んで、プログラムでそれを読み込んでいます。こうしておくと、あとでマップだけの編集という形がとれるので便利です。

プログラムの可読性もあがって一石二鳥です。

画面のスクロール

画面のスクロールは非常に簡単です。

ゲームの内部座標を、画面に対しての相対座標に変えてやるのです。言葉では非常にわかりにくいですね。例えば動いている電車から外を見ると風景が後ろに流れて見えますよね。本来風景が後ろに流れているのではなく、自分が前に動いているのですが、自分からは風景が動いているように見えます。スクロールも似た感じです。キャラクターの動作にあわせて、ゲーム全体の位置を動かしてやるのです。ＰＣの画面はうごかないですしね。

まず、画面に対してゲーム全体がどれだけ動いたかを現す変数を導入します。

game.cpp のグローバル変数で **int wallx, wally** を宣言します。次に **g_format(void)**の中で以下二つを初期化します。

wallx=0; wally=0;

基本的に、自機が画面に対しての座標である一定以上動いたら、wallx・wally を動かしてゲーム全体の座標を動かしてやることになります。その一定ラインを決める数値の定義として **HASI_RIGHT** と **HASI_LEFT** を加えます。

この２つはゲームを動かす最中には変わらないので#define で宣言してあげましょう。

```
#define HASI_RIGHT 400
#define HASI_LEFT 200
```

次に **g_tip_draw(void)**をいじります。DrawGraph の座標をちょっと変えてあげます。

実際には、以下の太字の部分が追加した部分になります。

```
DrawGraph( i * TIP_HABA - TIP_HABA / 2 - wallx, j * TIP_TAKASA - TIP_TAKASA / 2 -
wally,tip_handle, false);
```

次に g_teki(void) をいじります。当たり判定の座標も画面に対しての相対座標にしましょう。

ついでに描画も相対にしてやりましょう。

```
if( g_atari( jiki.x, jiki.haba, teki[i].x - wallx, teki[i].haba)){
  if( g_atari( jiki.y, jiki.takasa, teki[i].y - wally, teki[i].takasa)){
    //自分が敵に触れたら敵が消滅する
    teki[i].flag = 0;
  }
}
DrawGraph( (teki[i].x - teki[i].haba / 2) - wallx, (teki[i].y - teki[i].takasa / 2) - wally,
teki[i].handle, true);
//敵を描画する
```

次に **g_jiki(void)**をいじりましょう。自機が一定ラインに達したら wallx と wally を増減させてゲーム全体の相対座標を動かしましょう。同時に一定ラインを超えないように座標を制御するのを忘れずに。この辺は丸々追加ですね。

```
if( jiki.x < HASI_LEFT){
  wallx += jiki.x - HASI_LEFT;
  jiki.x = HASI_LEFT;
}
else if( jiki.x > HASI_RIGHT){
  wallx += jiki.x - HASI_RIGHT;
  jiki.x = HASI_RIGHT;
}
```

では当たり判定を相対座標に直してあげましょう。for 文のところは変わらないので省略。

ソースは以下に載せます。

```
if( g_atari( jiki.y + jiki.sokudo_y, jiki.takasa, j * TIP_TAKASA - wally, TIP_TAKASA)){
  if( g_atari( jiki.x, jiki.haba, i * TIP_HABA - wallx, 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 - wally, 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(要はジャンプ中)
        for( k = 0; ; k--){
          if( g_atari( jiki.y + k, jiki.takasa, j * TIP_TAKASA - wally, TIP_TAKASA)){
            jiki.sokudo_y = k + 1;
            break;
          }
        }
      }
    }
  }
  if( g_atari( jiki.x + jiki.sokudo_x, jiki.haba, i * TIP_HABA - wallx, TIP_HABA)){
    if( g_atari( jiki.y, jiki.takasa, j * TIP_TAKASA - wally, 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 - wallx, 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 - wallx, TIP_HABA)){
            jiki.sokudo_x = k + 1;
            break;
          }
        }
      }
    }
  }
}
                }
```

以上でマップ・敵・自機のすべてを画面に対する相対座標に直すことができました。

では実際に動かしてみましょう。正常に動けば成功です。動かなかったら打ち込みミスがあるか、余計なことが書かれているかのどちらかです。

以上でソフトゼミ B は終わりです。お疲れ様でした。