ソフトゼミ B　第 8 回　スクロールとファイル分割

前回の制作でようやくブロック判定を付けることが出来ました。しかしながら 1 画面だけではアクションゲームとしては物足りないと思います。そこで今回は画面をスクロールさせる処理を追加していきます。また、シューティングと同じようにファイル分割をしていこうとおもいます。

スクロール

初めに、スクロールの追加をします。と、言っても当たり判定のときよりも随分簡単に追加することが出来ます。これは前回の判定は、一から作っていたのですが、今回のスクロールは内部的には端に来ると止まる処理さえ消せばいくらでも横に移動することができるからです。よってスクロールのメインは主人公が移動した際に、今までは画面を固定して主人公を動かしていたのを逆にして、主人公を固定して周りの画面を移動させるようにすればいいのです。そのため今回のスクロールは描画面で全面改定をします。

プログラムの追加

● マップチップの描画を拡張

続いては描画の変更に移ります。まずは固定だった背景マップを主人公の座標によって動くようにします。そのために以下のコードを削除してください。

```
//第7回追加区画
for(int j=0;j<15;j++)
{
  for(int i=0;i<32;i++)
  {
    if(i<SIDESIZE&&i>=0)
    {
      switch(map[i][j]){
        case 0:
        case 3:break;
        case 1:
```

```
        case 5: DrawGraph(i*32,j*32,img.kabe,TRUE);break;
        case 2: DrawGraph(i*32,j*32,img.yuka,TRUE);break;
        case 4: DrawGraph(i*32,j*32,img.goal,TRUE);break;
        case 6: DrawGraph(i*32,j*32,img.toge,TRUE); break;
        default: DrawGraph(i*32,j*32,img.null,TRUE);break;
      }
    }
  }
}
//第7回追加区画ここまで
```

続いて、次のコードに差し換えてください。

```
//第8回追加区画
//マップチップの描画
for(int j=0;j<15;j++)
{
  for(int i=【穴埋め】-9;i<【穴埋め】+12;i++)
  {
    if(i<SIDESIZE&&i>=0)
    {
      switch(map[i][j]){
        case 0:break;
        case 1: DrawGraph(i*32-player.x+CENTER,j*32,img.kabe,TRUE);break;
        case 2: DrawGraph(i*32-player.x+CENTER,j*32,img.yuka,TRUE);break;
        case 4: DrawGraph(i*32-player.x+CENTER,j*32,img.goal,TRUE);break;
        default: DrawGraph(i*32-player.x+CENTER,j*32,img.null,TRUE);break;
      }
    }
    else if(j==14){
      DrawGraph(i*32-player.x+CENTER,j*32,img.yuka,TRUE);
    }
  }
}
//第8回追加区画ここまで
```

4行目の”for(int i=(player.x/32)-9;i<(player.x/32)+12;i++)”というのは、今まではマッ

プの左端から右端までを描画していたのを、マップチップ単位で主人公の左マイナス9マスから右はプラス12マスまでを出力しているということです。何故右と左で4マスも差が出るのか、それは主人公の左は9マスあるのに対し、右のマスは10マスあることや、player.xは主人公の左端の座標を指示しているためです。またplayer.xを32で割っているのは単純にマップチップが32pixで構成されているからで、もしもマップチップの大きさを変更するときはここも変更しなければいけません。

● 主人公の描画（変更点）

ここまでで周囲の背景をスクロールさせるよう変更しましたが、今度は逆に主人公を固定させる処理を加えます。そのためにマップチップの描画の直後にある以下の部分を削除してください。

```
//主人公の描画
if(player.dire==0){
  DrawGraph(player.x,player.y,img.player_r,TRUE);
}
else{
  DrawGraph(player.x,player.y,img.player_l,TRUE);
}
```

そして以下に変更してください。

```
//主人公の描画
if(player.dire==0){
  DrawGraph(【穴埋め】,player.y,img.player_r,TRUE);
}
else{
  DrawGraph(【穴埋め】,player.y,img.player_l,TRUE);
}
```

背景を移動させるとは対照的に、今度は固定座標にするだけなのでわかりやすいと思います。

穴埋めはどちらもプレイヤーのx座標を表しています。両方とも中心位置を表す定数値（#defineで定義した数）を入れてください。

● 移動制限の変更

初めに今まで右端、左端に来るとそれ以上進めなくなる処理の内、右側の壁を撤去します。

```
//端に来ると止まる
if(player.x<0){
  player.vx=0;
  player.x=0;
}
/*************************************ここから*************************************/
else if(player.x>608){
  player.vx=0;
  player.x=608;
}
/*************************************ここまで*************************************/
```

コメントで示された範囲を削除し、以下のコードを追加します。

```
else if(player.x>【穴埋め】*32-32){
  player.vx=0;
  player.x=【穴埋め】*32-32;
}
```

これは、今まではマップのはし横ウィンドウの大きさ640から、プレイヤーの大きさである32pixを引いたx=602を、SIDESIZEという横のマップ数×32-プレイヤーの大きさという値に変更することで、拡張後の右端を定義します。

これでスクロールは出来るようになったと思いますので、次にファイル分割をしてみましょう。

ファイル分割

それでは続いてファイル分割に移ります。ファイル分割の基本的な説明はゼミ B 第 5 回でやりましたので今回は割愛して、実際の作業だけを進めていきます。

- ソースファイル
  - ・draw.cpp
  - ・judge.cpp
  - ・load.cpp
  - ・move.cpp
- ヘッダーファイル
  - ・define.h
  - ・struct.h

・global.h

・extern.h

・function.h

ヘッダーファイルの解説

- define.h

```
#define WINDOW_WIDTH 640//横ウィンドウの大きさ
#define WINDOW_HEIGHT 480//縦ウィンドウの大きさ
#define GRA 1//重力加速度
#define MAXSPEEDY 20//自由落下時の最高速度
#define BLOCK 32 //タイルの大きさ[px]
#define CENTER 288 //主人公の中央座標
#define SIDESIZE 50 //スクロールの大きさ
```

- struct.h

```
#include "define.h"
struct SPlayer
{
  /*---------------省略---------------*/
};
struct SImg
{
  /*---------------省略---------------*/
};
```

- global.h

```
#include "struct.h"
//変数定義
struct SPlayer player;
struct SImg img;
int map[SIDESIZE][15];
char keyState[256];
int i,j;
```

- extern.h

```
#include "struct.h"
extern struct SPlayer player;
extern struct SEnemy enemy;
extern int map[20][15];
extern char keyState[256];
```

- function.h

```
/*judge.cpp*/
void judge1(int blx,int bly);
void judge2(int blx,int bly);

/*load.cpp*/
void load(void);
void imgload(void);
void init(void);

/*move.cpp*/
void dead(void);
void move(void);

/*draw.cpp*/
void title(void);
void draw(void);
```

ソースファイルの解説

- draw.cpp

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
void title(void){
        ClearDrawScreen();//画面の内容を消去
        /*----------------省略----------------*/
```

```
}
void draw(void){
	ClearDrawScreen();//画面の内容を消去
	//背景の描画
	/*---------------省略---------------*/
	//描画
	ScreenFlip();
}
```

- judge.cpp

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
//第7回追加区画
//壁判定１（横判定）
void judge1(int blx,int bly){
	/*---------------省略---------------*/
}
//壁判定２（上下判定）
void judge2(int blx,int bly){
	/*---------------省略---------------*/
}
//第7回追加区画ここまで
```

- load.cpp

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
//第7回追加区画
//マップチップ読み込み
void load(void){
	/*---------------省略---------------*/
}
//第7回追加区画ここまで
//画像のロード
void imgload(void){
```

```
        /*---------------省略---------------*/
}
//変数の初期化
void init(void){
        /*---------------省略---------------*/
}
```

- move.cpp

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
//死亡判定
void dead(void){
        /*---------------省略---------------*/
}

void move(void){
        /*---------------省略---------------*/
}
```

- main.cpp

最後に main.cpp の先頭にある#include <stdlib.h> の下に以下の 2 行を追加してください。

```
#include "global.h"
#include "function.h"
```

お疲れ様でした。これでアクションゲームのサンプルは以上になります。
マップを改造したい人向けにアクションゲームのマップメーカーを作成しましたので、ほしい方はソフト班長まで声をかけてください。