<<大幅改訂版>>

ソフトゼミＢ 第４回 キー入力 II ～弾の発射～ ＋敵複数化

■ はじめに

自機・敵機ともに動けるようになったので、今回は、いよいよ弾を発射させてみます。もちろん、普通シューティングゲームには弾は複数個存在します。ということは、必然的に同じ型の構造体が複数必要となってきます。ゼミＢでは、**構造体の配列**を使うことによって、複数の弾を実現します。(C++を本格的に学んでいくと、もっと効率の良い方法もあるのですが。) なお、今回の後半では、同じ手法で、今まで１体だった敵を複数出るように改造します。

この資料は、Linux の diff コマンドで第３回のソースとの差分をとって作ったものなので、正確だとは思います。さらに、第３回のソースから改造して試験もしたので大丈夫だと思いますが、それでも動かない！って場合には報告してください。

■ 前半で追加・削除する箇所

前半では、以下の箇所を追加したり削除したりします。

マクロ(#define)

```
/* 弾関連 */
// プレイヤーが撃った弾の最大数
#define PLAYER_SHOT_MAX 256
// 敵が撃った弾の最大数
#define ENEMY_SHOT_MAX 1024
// 弾のサイズ
#define BULLET_SIZE 16

/* キー関連 */
//ショットボタン
#define SHOT_BUTTON KEY_INPUT_Z
```

前回は、#define で**自機・敵機のサイズの設定**をしました。この#define は**その下**あたりにでも書いてください。ここでは、弾が存在できる最大数と、ショットボタンを定義しています。

ショットボタンの KEY_INPUT_Z は既に DX ライブラリの中で定義されている数ですが、このように、define で定義されたものに対しても define をすることができます。これで、ショットに使うボタンを簡単に変更できます。

プロトタイプ宣言

```
void movePlayer( void );
void setPlayerShot( void );
void movePlayerShots( void );
void moveEnemy( void );
void setEnemyShot( void );
void moveEnemyShots( void );
```

灰色で塗りつぶした部分を追加してください。今までのプロトタイプ宣言のところに入れてください。なお、(C＋＋なので)引数の方の void は省略して、void setPlayerShot();のように宣言してもOK です。

構造体

ゲームデータ

```
// ゲームデータ
struct SGame{
	/*プレイヤー*/
	int dPlayerX, dPlayerY; //デフォルトのプレイヤー座標
	int dPlayerLife;		//デフォルトのプレイヤーライフ
	int dPlayerVelocity;	//デフォルトのプレイヤーの速度

	/*敵*/
	int dEnemyLife;			//デフォルトの敵機ライフ
	int dEnemyVX, dEnemyVY;	//デフォルトの敵機の速度(x 成分, y 成分)

	/*プレイヤーの弾*/
	int dPlayerShotLife; //デフォルトのプレイヤーの弾のライフ
	int dPlayerShotVX, dPlayerShotVY; //デフォルトのプレイヤーの弾の速度(x 成分, y 成分)

	/*敵の弾*/
	int dEnemyShotLife;	//デフォルトの敵の弾のライフ
	int dEnemyShotVX, dEnemyShotVY; //デフォルトの敵の弾の速度(x 成分, y 成分)
};
```

上記のような新しい構造体、「SGame」を宣言してください。この構造体は、ゲームに関するあらゆる情報を網羅します。構造体が宣言されている部分の一番上で宣言してください。

SimgList 構造体

SimgList 構造体(読み込む画像の一覧)に、新しいメンバ

```
int playerShot;
int enemyShot;
```

を追加してください。(自機・敵機から発射された弾の画像の識別番号を格納する変数です。)

プレイヤーの弾を扱う構造体

新たな構造体「SPlayerShot」を

```
//プレイヤーの弾
struct SPlayerShot{
	int x, y;
	int vx, vy;
	int life;
};
```

のように追加してください。場所は SPlayer か SEnemy の下あたりが望ましいでしょう。

敵の弾を扱う構造体

上に同じく「SEnemyShot」を、

```
//敵の弾
struct SEnemyShot{
	int x, y;
	int vx, vy;
	int life;
};
```

のように追加してください。場所は、SPlayerShot か、SEnemy の下あたりが望ましいでしょう。

グローバル変数

ゲームデータ　<<改訂で修正>>

上で定義したゲームデータを実際に格納する構造体の実体を定義します。

```
struct SGame game;
```

追加する場所としては、他のグローバル変数(char keyState[256];や、struct SPlayer player;など)が宣言されているあたりが望ましいでしょう。

弾

```
struct SPlayerShot playerShot[ PLAYER_SHOT_MAX ];
int playerShotNum;
struct SEnemyShot  enemyShot[ ENEMY_SHOT_MAX ];
int enemyShotNum;
```

弾の実体と、弾の個数を格納します。 やはり、他のグローバル変数があるあたりに宣言してください。

ループ用変数の宣言　<<改訂で追加>>

ここからは、WinMain 関数の内部を変更してゆきます。

```
int i;
```

for 文のループで使うループ変数です。C++なので、for 文の初期化部で宣言しても問題はありませんが、C 風に書いているので、WinMain 関数の一番上で宣言してください。

初期化

削除する部分

以下は**削除**してください。

```
player.x = 240;
player.y = 240;
player.velocity = 4;
player.life = player.maxLife;
player.score = 0;
```

値を変更して遊んだ場合は、この値とは限りません。値の初期化は、これ以降直接指定するのではなく、ゲームデータ経由で指定するので削除します。ゲームデータの操作はこれから示していきます。

追加する部分　その１

長いです。覚悟しましょう。

```
// 初期値の設定
game.dPlayerX = 240;
game.dPlayerY = 240;
game.dPlayerLife = 100;
game.dPlayerVelocity = 4;

game.dEnemyLife = 5;
game.dEnemyVX = 0;
game.dEnemyVY = 2;

game.dPlayerShotLife = 1;
game.dPlayerShotVX = 0;
game.dPlayerShotVY = -15;

game.dEnemyShotLife = 1;
game.dEnemyShotVX = 0;
game.dEnemyShotVY = 15;

// 初期化
player.x = game.dPlayerX;
player.y = game.dPlayerY;
player.maxLife = game.dPlayerLife;
player.life = player.maxLife;
player.velocity = game.dPlayerVelocity;
player.score = 0;

enemy.x = 120;
enemy.y = 50;
enemy.vx = game.dEnemyVX;
```

```
enemy.vy = game.dEnemyVY;
enemy.maxLife = game.dEnemyLife;
enemy.life = enemy.maxLife;

playerShotNum = 0;
enemyShotNum = 0;

for(i = 0; i < PLAYER_SHOT_MAX; i++){
        playerShot[ i ].life = 0;
}
for(i = 0; i < ENEMY_SHOT_MAX; i++){
        enemyShot[ i ].life = 0;
}
```

さっき削除した跡にこれを書いてください。

灰色で塗った、敵の初期 x,y 座標の初期化はゲームデータに含めませんでした。理由は、後半で複数体の敵をランダムに配置するからです。したがって、今は灰色の部分は前回の状態のままにしておいてください。

追加する部分　その２

```
// 画像のロード
imgList.player = LoadGraph( "img/player.png" );
imgList.playerShot = LoadGraph( "img/playerShot.png" );
imgList.enemy  = LoadGraph( "img/enemy.png" );
imgList.enemyShot  = LoadGraph( "img/enemyShot.png" );
imgList.sidebar  = LoadGraph( "img/sidebar.png" );
```

灰色で示した行を追加してください。

新たに、弾の画像をロードするようにします。ちなみに、弾は上の define で定義した数四方の正方形で、この資料通りに作ると１６[px] * 16[px]となります。

メインの処理

自機からの弾の発射

もし、前回までのソースに「//プレイヤーの弾発射追加予定」と書いてあれば、そのコメントを消

して、そこに書いてください。ない場合には、movePlayer()の次あたりに書いてください。

```
setPlayerShot();
movePlayerShots();
```

敵機からの弾の発射

もし、前回までのソースに「//敵の弾発射追加予定」と書いてあれば、そのコメントを消して、そこに書いてください。ない場合には、moveEnemy()の次あたりに書いてください。

```
setEnemyShot();
moveEnemyShots();
```

自機からの弾の描画

場所は、自機の描画(DrawGraph(player.x,～とかいうやつ)の次です。

```
// 自機の弾の描画
for( i = 0; i < PLAYER_SHOT_MAX; i++ ){
 if( playerShot[ i ].life == 0 ){ continue; }
 DrawGraph( playerShot[ i ].x,playerShot[ i ].y, imgList.playerShot, TRUE );
}
```

配列にある弾のうち、ライフが 0 以外のものを描画します。

敵機からの弾の描画

場所は、敵機の描画(DrawGraph(enemy.x,～とかいうやつ)の次です。

```
// 敵機の弾の描画
for( i = 0; i < ENEMY_SHOT_MAX; i++ ){
 if( enemyShot[ i ].life == 0 ){ continue; }
 DrawGraph( enemyShot[ i ].x, enemyShot[ i ].y, imgList.enemyShot, TRUE );
}
```

関数

弾を動かす処理を実際に書いていきます。追加する箇所はソースの末尾など。これまた長いです。

```
void setPlayerShot( void ){
 int i;
```

```
 if( keyState[ SHOT_BUTTON ] && playerShotNum < PLAYER_SHOT_MAX ){
 // 未使用の添字を探索
 for( i = 0; i < PLAYER_SHOT_MAX; i++ ){
    if( playerShot[ i ].life == 0 ){ break; }
 }
 playerShot[ i ].x = player.x + PLAYER_SIZE / 4;
 playerShot[ i ].y = player.y;
 playerShot[ i ].life = 1;
 playerShot[ i ].vx = game.dPlayerShotVX;
 playerShot[ i ].vy = game.dPlayerShotVY;
 playerShotNum ++;
 }
}

void movePlayerShots( void ){
 int i;
 for ( i = 0; i < PLAYER_SHOT_MAX; i++ ){
  //死んだ弾, 未使用の弾は移動させない。
  if( playerShot[ i ].life == 0 ){ continue; }

  //速度分だけすすめる。
  playerShot[ i ].x += playerShot[ i ].vx;
  playerShot[ i ].y += playerShot[ i ].vy;

  // 弾が完全に画面外に出たら、その弾は死ぬ。
  if( playerShot[ i ].y < - BULLET_SIZE ){
    playerShot[ i ].life = 0;
    playerShotNum--;
  }
 }
}

void setEnemyShot( void ){
 int i;
```

```
  // 未使用の添字を探索
  for( i= 0; i < ENEMY_SHOT_MAX; i++ ) {
    if( enemyShot[ i ].life == 0 ){ break; }
  }
  enemyShot[ i ].x = enemy.x + ENEMY_SIZE / 4;
  enemyShot[ i ].y = enemy.y + ENEMY_SIZE - enemy.vy;
  enemyShot[ i ].life = 1;
  enemyShot[ i ].vx = game.dEnemyShotVX;
  enemyShot[ i ].vy = game.dEnemyShotVY;
  enemyShotNum ++;
}

void moveEnemyShots( void ){
  int i;
  for ( i = 0; i < ENEMY_SHOT_MAX; i++ ){
    //死んだ弾, 未使用の弾は移動させない。
    if( enemyShot[ i ].life == 0 ){ continue; }

    //速度分だけすすめる
    enemyShot[ i ].x += enemyShot[ i ].vx;
    enemyShot[ i ].y += enemyShot[ i ].vy;

    // 弾が完全に画面外に出たら、その弾は死ぬ。
    if( enemyShot[ i ].y > WINDOW_HEIGHT + BULLET_SIZE ){
      enemyShot[ i ].life = 0;
      enemyShotNum --;
    }
  }
}
```

書き終えたら動作確認してみましょう。Z キーで弾が発射できるようになっていて、敵が弾を撃ってくるようになっていれば成功です。

■ 後半で追加する箇所

後半では、敵を配列にする作業がほとんどです。

マクロ(#define)　<<改訂で追加>>

```
// 敵の最大数
#define ENEMY_MAX 256
```

を、#define ENEMY_SIZE の下あたりに宣言してください。ENEMY_MAX は、一度に存在できる、敵の最大数を示します。

プロトタイプ宣言　<<改訂で追加>>

```
void setEnemy( void );
```

を、プロトタイプ宣言の部分に追加してください。場所は void moveEnemy(void);の上あたりが多分見やすいと思います。

構造体

構造体「struct SGame」に新しいメンバ、

```
int enemyTime; //敵を出す間隔[frame]
```

を追加してください。

グローバル変数

敵の配列化

```
struct SEnemy enemy;
```

を、

```
struct SEnemy enemy[ ENEMY_MAX ];
```

に変更してください。単独の構造体であった敵を、**構造体の配列**にすることによって、複数存在できるようにします。上限は先程定義した「ENEMY_MAX(=256)」です。なお、ここを修正すると、配列になるため、enemy に添字がない部分を中心に大量のエラーが発生します。それらをこれから直していきます。

細かい変数＜＜改訂で追加＞＞

```
int enemyNum;
int enemyTrigger;
```

を、グローバル変数として宣言してください。enemyNum は、敵の数を監視し、enemyTrigger は、敵の発生間隔をカウントします。場所としては、さきほど配列化した enemy の宣言の下あたりが望ましいかと思います。

初期化

変更・削除

ゲームデータの初期化内、

```
game.enemyTime = 50;
game.dEnemyLife = 5;
game.dEnemyVX = 0;
game.dEnemyVY = 1;
```

灰色の文を追加してください。

また、

```
enemy.x = 120;
enemy.y = 50;
enemy.vx = game.dEnemyVX;
enemy.vy = game.dEnemyVY;
enemy.maxLife = game.dEnemyLife;
enemy.life = enemy.maxLife;
```

を、**削除**してください。(単一の敵に対する初期化パラメータなので)

追加　＜＜改訂で追加＞＞

この他、playerShotNum = 0;の上に、

```
enemyNum = 0;

enemyTrigger = game.enemyTime;
```

を追加してください。

playerShot の初期化 for 文と enemyShot の初期化 for 文の間に、

```
for(i = 0; i < ENEMY_MAX; i++){
        enemy[ i ].life = 0;
}
```

を追加してください。

メインの処理

自機・自機の弾関連の操作の直後(movePlayerShots();の次)に

```
/* トリガーが 0 なら敵を配置 0 でないならカウントを減らす */
enemyTrigger--;
if( enemyTrigger <= 0 ){
        setEnemy();
        enemyTrigger = game.enemyTime;
}
```

を追加してください。(敵を配置する処理)。

単一の敵機描画

```
DrawGraph( enemy.x , enemy.y , imgList.enemy , TRUE );
```

を、複数の敵機描画

```
for( i = 0; i < ENEMY_MAX; i++ ){
        if( enemy[ i ].life == 0 ){ continue; }
        DrawGraph( enemy[ i ].x, enemy[ i ].y, imgList.enemy, TRUE );
}
```

に変更してください。

関数

setEnemy 関数

以下の内容をもった setEnemy 関数を新たに作成してください。場所は moveEnemy 関数の上あたりが見やすいと思います。

```
void setEnemy ( void ){
  int i;
```

```
 //敵数 MAX の時、配置なし
 if(enemyNum == ENEMY_MAX){ return; }
 //x 座標ランダムな位置に敵を配置。 端っこ 1 マス分には配置しない。
 int x = GetRand(WINDOW_WIDTH - SIDEBAR_WIDTH - ENEMY_SIZE * 2) + ENEMY_SIZE;
 int y = - ENEMY_SIZE;
 // 未使用の添字を探索
 for(i = 0; i < ENEMY_MAX; i++){
  if(enemy[ i ].life == 0){ break; }
 }
 enemy[ i ].x = x;
 enemy[ i ].y = y;
 enemy[ i ].life = game.dEnemyLife;
 enemy[ i ].vx = game.dEnemyVX;
 enemy[ i ].vy = game.dEnemyVY;
 enemyNum++;
}
```

moveEnemy 関数

以下のように改造します。

```
void moveEnemy( void ){//全体的に複数化。enemy → enemy[ i ]
 int i;
 for(i = 0; i < ENEMY_MAX; i++){
  if(enemy[ i ].life == 0){ continue; }
  enemy[ i ].x += enemy[ i ].vx;
  enemy[ i ].y += enemy[ i ].vy;
  // 敵が完全に画面外に出たら、その敵は死ぬ。
  if( enemy[ i ].x < - ENEMY_SIZE ||
    enemy[ i ].x > WINDOW_WIDTH - SIDEBAR_WIDTH + ENEMY_SIZE ||
    enemy[ i ].y > WINDOW_HEIGHT + ENEMY_SIZE
  ){
    enemy[ i ].life = 0;
    enemyNum --;
  }
 }
```

```
}
```

灰色で塗った位置に for 文を追加して、すべての enemy を enemy[i]に書き換えてください。

setEnemyShot 関数

以下のように改造します。

```
void setEnemyShot( void ){//全体的に複数化。enemy → enemy[ i ]
	int i, j;
	for( i = 0; i < ENEMY_MAX; i++ ){
		if( enemy[ i ].life == 0 ){ continue; }
		// 未使用の添字を探索
		for( j = 0; j < ENEMY_SHOT_MAX; j++ ) {
			if( enemyShot[ j ].life == 0 ){ break; }
		}
		enemyShot[ j ].x = enemy[ i ].x + ENEMY_SIZE / 4;
		enemyShot[ j ].y = enemy[ i ].y + ENEMY_SIZE;
		enemyShot[ j ].vx = game.dEnemyShotVX;
		enemyShot[ j ].vy = game.dEnemyShotVY;
		enemyShot[ j ].life = 1;
		enemyShotNum ++;
	}
}
```

for 文がひとつ外側に増えます。内側の for 文は、ループカウンタが j に変更になったので、影響する部分は書き換えてください。

以上で終わりです。敵がうじゃうじゃ出てくれば成功です。余裕があれば、弾を発射させるタイミングや向きなどを調整してみるのはいかがでしょう。

これで今回の部分は終了です。お疲れ様でした。