

第3回b 敵の表示

○敵の出現・移動・描画

敵に関する部分を作成します。ソースファイルに新しく enemy.cpp を作成してください。その後、各々のソースファイル・ヘッダファイルの末尾に下記のような敵に関する記述を加えていきます。

・define.h 画面上の敵の最大数

```
// 画面上の敵の最大数
#define ENEMY_MAX      ( 128 )
```

・function.h 敵の設置、移動、描画関数のプロトタイプ宣言

```
// enemy.cpp
void SetEnemy( void );    // 敵設置関数
void MoveEnemy( void );  // 敵移動関数
void DrawEnemy( void );  // 敵描画関数
```

・struct.h 敵の情報をまとめた構造体 SEnemy の定義

```
// 敵の情報
struct SEnemy{
    int life;          // ライフ(0 または負の時この敵は存在しない)
    int count;         // 登場してから何フレーム経過したか
    int pos[ 2 ];      // X座標とY座標
    int type;          // 敵の種類
    float v;           // 速度
    float a;           // 加速度
    float th;          // 角度
};
```

・global.h 構造体変数の配列 enemy の宣言

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

・extern.h 構造体変数の配列 enemy のextern

```
extern struct SEnemy enemy[ ENEMY_MAX ]; // 敵
```

これで下準備は整いました。この時点でのソースコードはフォルダ「a」にまとめてあります(まだプロトタイプ宣言を行った関数の定義を記述していないのでコンパイルエラーとなります)。いよいよ先程プロトタイプ宣言を行った3つの関数を作成していきます。ソースコードは下記のようになります。これを enemy.cpp に記述して下さい。

```
/*
*****
enemy.cpp      敵関連関数
*****
*/
#define _USE_MATH_DEFINES
// インクルード
// C 言語標準関数
#include "math.h"
// DX ライブラリ
#include "DxLib.h"
// 自作ヘッダ
#include "define.h"
#include "struct.h"
#include "extern.h"
#include "function.h"
```

```

// 敵設置関数
void SetEnemy( void ){
    enemy[ 0 ].pos[ 0 ] = 400;
    enemy[ 0 ].pos[ 1 ] = 200;
    enemy[ 0 ].life = 50;
    enemy[ 0 ].type = 1;
    enemy[ 0 ].v = 2;
    enemy[ 0 ].a = 0.01f;
    enemy[ 0 ].count = 0;
    enemy[ 0 ].th = ( float )M_PI / 2;
}

// 敵描画関数
void DrawEnemy( void ){
    int i;
    for( i = 0; i < ENEMY_MAX; i++ ){ // 敵の構造体全てに対して
        if( enemy[ i ].life > 0 ){ // 敵が存在するときのみ描画
            switch( enemy[ i ].type ){ // 敵の種類に応じて処理を分岐可能
            default: // 今回は全てデフォルト
                // 敵の角度thで回転描画。1.0f = 拡大率
                DrawRotaGraph( enemy[ i ].pos[ 0 ], enemy[ i ].pos[ 1 ],
                    1.0f, enemy[ i ].th,
                    img.enemy[ enemy[ i ].type ], TRUE, FALSE );
                break;
            }
        }
    }
}

// 敵移動関数
void MoveEnemy( void ){
    int i;
    for( i = 0; i < ENEMY_MAX; i++ ){ // 敵の構造体全てに対して
        if( enemy[ i ].life > 0 ){ // 敵が存在するときのみ移動
            switch( enemy[ i ].type ){ // 敵の種類に応じて移動処理を分岐
            default: // 今回は全てデフォルト
                // 座標に角度thと速度vから計算した値を加算
                enemy[ i ].pos[ 0 ] +=
                    ( int )( enemy[ i ].v * cos( enemy[ i ].th ) );
                enemy[ i ].pos[ 1 ] +=
                    ( int )( enemy[ i ].v * sin( enemy[ i ].th ) );
                // 加速度を速度に加算
                enemy[ i ].v += enemy[ i ].a;
                // 画面上に存在したフレームを加算
                enemy[ i ].count++;
                // 敵が画面外にでたら消去
                if( enemy[ i ].pos[ 0 ] < -32 || enemy[ i ].pos[ 0 ] > 672 ||
                    enemy[ i ].pos[ 1 ] < -32 || enemy[ i ].pos[ 1 ] > 512 ){
                    enemy[ i ].life = 0;
                }
            }
        }
    }
}

```

```

        break;
    }
}
}
}

```

書面上で見やすいよう、一部の計算式と DrawRotaGraph 関数の呼び出しは途中で改行してあります。関数呼び出しや計算式の途中で改行した場合、VisualC++ は見やすいように自動的にインデントを加えてくれます。また、SetEnemy 関数は後により汎用的に使えるようにするので現在は仮組みの物です。

DrawRotaGraph はロードした画像を回転描画する関数です。今回は、敵の向きを保存するメンバ変数 th の値に応じて、回転描画を行っています。また、MoveEnemy 関数内で使われている関数 sin と cos は数学と同じ三角関数です。角度を渡してやると、それぞれ sin 値と cos 値を返り値として返します。これに速度をかけた値を座標に足してやれば、指定した方向に指定した速度で移動する、という寸法です。他はコメントに書いた通りです。

構造体変数の配列 enemy の各々の要素のメンバ変数の値を初期化します。Game 関数の先頭で int 型の変数 i を宣言し、以下のコードをプレイヤーの初期化後、画像のロードの前に加えて下さい。

```

// 敵の初期化
for( i = 0 ; i < ENEMY_MAX ; i++ ){
    enemy[ i ].life = 0 ;
    enemy[ i ].type = 1 ;
    enemy[ i ].pos[ 0 ] = 0 ;
    enemy[ i ].pos[ 1 ] = 0 ;
    enemy[ i ].v = 2 ;
    enemy[ i ].a = 0 ;
    enemy[ i ].count = 0 ;
}

```

最後に SetEnemy 関数をループ突入の直前で、MoveEnemy 関数をオブジェクトの移動部分で、DrawEnemy 関数を描画部分、ScreenFlip 関数の呼出より前で呼び出せば、敵が画面上に表示されます。この時点でのソースコードはフォルダ「b」を参照してください。

○練習問題

SetEnemy 関数を、使用していない(life が 0 以下で出現フラグが立っていない) 構造体変数を検索し、その構造体変数のメンバ変数に代入するように書き換えて下さい。また、Game 関数内で、300 回ループするごとに、1 度 SetEnemy 関数が呼び出されるように書き換えて下さい(ループ回数を数える変数の変数名は count として、Game 関数内におけるローカル変数として関数の先頭部分で宣言してください)。解凍はフォルダ「c」を参照してください。