```
//-------------------------------------------------------------------------------
// インベーダーゲーム
//-------------------------------------------------------------------------------
// Enemy.cpp
//
#include <DxLib.h>
#include "main.h"
#include "Enemy.h"

//-----------------
// 変数宣言
//-----------------
// 敵キャラ
int enemyImage;                                                 // 敵の画像ハンドル番号
int enemyPosxArray[ENEMY_DISP_YNUM][ENEMY_DISP_XNUM];           // 敵のX座標表示位置テーブル
int enemyPosyArray[ENEMY_DISP_YNUM][ENEMY_DISP_XNUM];           // 敵のY座標表示位置テーブル
bool enemyFlgArray[ENEMY_DISP_YNUM][ENEMY_DISP_XNUM];           // 敵の存在フラグテーブル

eEnemyMoveDirection enemyMoveDirection;             // 敵キャラの現在の移動方向
eEnemyMoveDirection enemyNextMoveDirection;         // 敵キャラの下移動後の移動方向
int enemyMoveIntervalCounter;                       // 敵キャラの移動する感覚を調整するカウンタ

/**
 *-------------------------------------------------
 * 敵キャラの起動時のみの初期化処理
 * Input:
 *    無し
 * Output:
 *    true = 正常終了 / false = エラー終了
 *-------------------------------------------------
 */
bool EnemySysInitProc(void)
{
    enemyImage = LoadGraph("image/enemy_1.png");
    if (enemyImage == -1)return false;

    return true;
}

/**
 *-------------------------------------------------
 * 敵キャラの初期化処理
 * Input:
 *    無し
 * Output:
 *    無し
 *-------------------------------------------------
 */
void EnemyInitProc(void)
{
    for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            enemyPosxArray[yy][xx] = (ENEMY_SIZE_WID + ENEMY_DISP_X_INTERVAL) * xx;
            enemyPosyArray[yy][xx] = (ENEMY_SIZE_HIG + ENEMY_DISP_Y_INTERVAL) * yy;
            enemyFlgArray[yy][xx] = true;
        }
    }

    enemyMoveDirection = ENEMY_DIR_RIGHT;
    enemyNextMoveDirection = ENEMY_DIR_NON;
    enemyMoveIntervalCounter = -1;
}
```

```
/**
 *-----------------------------------------------
 * 敵キャラの移動処理
 * Input:
 *    無し
 * Output:
 *    無し
 *-----------------------------------------------
 */
void EnemyMoveProc(void)
{
    enemyMoveIntervalCounter++;
    if (enemyMoveIntervalCounter > 0) {
        if (enemyMoveIntervalCounter > ENEMY_MOVE_INTERVAL_COUNT) {
            enemyMoveIntervalCounter = -1;
        }
        return;
    }

    for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            if (enemyMoveDirection == ENEMY_DIR_RIGHT) {
                // 右方向への移動(横移動だけなので、Y座標は触らない)
                enemyPosxArray[yy][xx] += ENEMY_MOVE_X_SPEED;
            }
            else if (enemyMoveDirection == ENEMY_DIR_LEFT) {
                // 左方向への移動(横移動だけなので、Y座標は触らない)
                enemyPosxArray[yy][xx] -= ENEMY_MOVE_X_SPEED;
            }
            else if (enemyMoveDirection == ENEMY_DIR_DOWN) {
                // 下方向への移動(縦移動だけなので、X座標は触らない)
                enemyPosyArray[yy][xx] += ENEMY_MOVE_Y_SPEED;
            }
        }
    }

    // エリアの端にぶつかっていないかのチェック
    eEnemyHitEdgeDir hitdir = CheckHitEdgeProc();
    if (hitdir != ENEMY_HIT_DIR_NON) {
        // いずれかの端に接触している
        if (hitdir != ENEMY_HIT_DIR_DOWN) {
            enemyMoveDirection = ENEMY_DIR_DOWN;
            if (hitdir == ENEMY_HIT_DIR_RIGHT) {
                enemyNextMoveDirection = ENEMY_DIR_LEFT;
            }
            else {
                enemyNextMoveDirection = ENEMY_DIR_RIGHT;
            }
        }
        else {
            // 敵が下端に達したので、プレイヤーの敗け
        }
    }
    else {
        if (enemyMoveDirection == ENEMY_DIR_DOWN) {
            enemyMoveDirection = enemyNextMoveDirection;
            enemyNextMoveDirection = ENEMY_DIR_NON;
        }
    }
}
```

```
/**
 *---------------------------------------------------
 * エリアの端にぶつかっていないかのチェック処理
 * Input:
 *    無し
 * Output:
 *    ENEMY_HIT_DIR_NON = ぶつかっていない
 *    それ以外 ぶつかっている方向(eEnemyHitEdgeDir 型)
 *---------------------------------------------------
 */
eEnemyHitEdgeDir CheckHitEdgeProc(void)
{
    eEnemyHitEdgeDir hitdir = ENEMY_HIT_DIR_NON;
    int restnum;
    if (enemyMoveDirection == ENEMY_DIR_RIGHT) {
        // 右方向に移動中なので、右端に接触していないかをチェックする
        for (int xx = ENEMY_DISP_XNUM-1; xx >= 0; xx--) {
            restnum = 0;
            for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
                if (enemyFlgArray[yy][xx] == true) {
                    restnum++;     // その列の残敵をカウント
                    if (enemyPosxArray[yy][xx] >= (WINDOW_SIZE_WID - ENEMY_SIZE_WID)) {
                        hitdir = ENEMY_HIT_DIR_RIGHT;
                        break;
                    }
                }
            }
            if (hitdir != ENEMY_HIT_DIR_NON || restnum > 0)break;
        }
    }
    else if (enemyMoveDirection == ENEMY_DIR_LEFT) {
        // 左方向に移動中なので、左端に接触していないかをチェックする
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            restnum = 0;
            for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
                if (enemyFlgArray[yy][xx] == true) {
                    restnum++;     // その列の残敵をカウント
                    if (enemyPosxArray[yy][xx] <= 0) {
                        hitdir = ENEMY_HIT_DIR_LEFT;
                        break;
                    }
                }
            }
            if (hitdir != ENEMY_HIT_DIR_NON || restnum > 0)break;
        }
    }
    else if (enemyMoveDirection == ENEMY_DIR_DOWN) {
        for (int yy = ENEMY_DISP_YNUM - 1; yy >= 0; yy--) {
            restnum = 0;
            for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
                if (enemyFlgArray[yy][xx] == true) {
                    restnum++;     // その列の残敵をカウント
                    if (enemyPosyArray[yy][xx] >= (WINDOW_SIZE_HIG - ENEMY_SIZE_HIG)) {
                        hitdir = ENEMY_HIT_DIR_DOWN;
                        break;
                    }
                }
            }
            if (hitdir != ENEMY_HIT_DIR_NON || restnum > 0)break;
        }
    }
    return hitdir;
}
```

```
/**
 *-----------------------------------------------
 * 敵キャラの描画処理
 * Input:
 *    無し
 * Output:
 *    無し
 *-----------------------------------------------
 */
void EnemyDrawProc(void)
{
    for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            if (enemyFlgArray[yy][xx] == true) {
                DrawGraph(enemyPosxArray[yy][xx], enemyPosyArray[yy][xx], enemyImage, true);
            }
        }
    }
}

/**
 *-----------------------------------------------
 * 敵キャラの解放処理
 * Input:
 *    無し
 * Output:
 *    true = 正常終了 / false = エラー終了
 *-----------------------------------------------
 */
bool EnemyReleaseProc(void)
{
    if (DeleteGraph(enemyImage) == -1)return false;

    return true;
}
```