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

//-----------------
// 変数定義
//-----------------
int nowStageNo;           // 現在のステージ番号
// テスト用のキーの状態
int nowWKeyPress;         // 条件１
int prevWKeyPress;
int nowLKeyPress;         // 条件２
int prevLKeyPress;
int nowOKeyPress;         // 条件３
int prevOKeyPress;
stSaveData saveDat;       // セーブデータ
int nowF1KeyPress;
int prevF1KeyPress;
int nowF2KeyPress;
int prevF2KeyPress;

/**
 *-------------------------------------------------
 * ゲームシーンの初期化処理
 * Input:
 *    無し
 * Output:
 *    true = 正常終了 / false = エラー終了
 *-------------------------------------------------
 */
// プレイヤーが勝利した時の再初期化処理
void SceneGameNextStageInit(void)
{
    // 敵キャラの関連の初期化
    EnemyInitProc();
    EnemyShotInitProc();
    // プレイヤー関連の初期化
    // プレイヤーのY座標は常に最下段なので、再設定の必要はない
    // プレイヤーが勝利した時に呼ばれるので、プレイヤーの存在フラグの再設定の必要もない
    playerPosX = WINDOW_SIZE_WID / 2 - PLAYER_SIZE_WID / 2;                // 横方向の画面中央
}

// 敵キャラに最下段まで到達された時の再初期化処理
void SceneGameStageOccupationInit(void)
{
    // 敵キャラのY座標を４段上に戻す(X座標はそのまま)
    for(int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            if (enemyFlgArray[yy][xx] == true) {
                enemyPosyArray[yy][xx] -= ENEMY_MOVE_Y_SPEED * ENEMY_OCCUPATION_GO_UP_STEPS;
                if (enemyPosyArray[yy][xx] < 0) {
                    enemyFlgArray[yy][xx] = false;
                }
            }
        }
    }
}
```

```
/**
 *-----------------------------------------------------
 * ゲームシーンのメイン処理
 * Input:
 *    無し
 * Output:
 *    無し
 *-----------------------------------------------------
 */
void SceneGameMainProc(void)
{
    SceneGameUpdate();
    SceneGameDraw();

    // ステージが終了したかどうか
    eStageExitConditionsKind exit = StageExitConditionCheck();
    if (exit != ESTAGE_EXIT_NON) {
        switch (exit) {
        case ESTAGE_EXIT_CLEAR:
            // 条件１：プレイヤーが勝った
            //   ステージ番号を１加算(シーンは変更無し)
            //   ステージ番号が最大ステージ数を超えていたら、ゲーム終了
            //   次のステージに進む際は、敵の位置や敵の存在フラグを再初期化してやる
            nowStageNo++;
            if (nowStageNo > STAGE_NUM_MAX) {
                // 最大ステージ数を超えたらゲーム終了
                nowScene = SCENE_KIND_GAMEOVER;
            } else {
                SceneGameNextStageInit();
            }
            break;
        case ESTAGE_EXIT_ENEMY_SHOT:
            // 条件２：敵弾にプレイヤーが撃たれた
            //   敵キャラはそのままで、プレイヤーのみ初期化
            //     ・プレイヤーの初期位置の再設定
            //     ・プレイヤーの存在フラグをtrueに初期化
            playerRestPlaneNum--;
            if (playerRestPlaneNum < 0) {
                // 残機が無くなったので、ゲーム終了
                nowScene = SCENE_KIND_GAMEOVER;
            } else {
                // 敵キャラはそのままでいいので、プレイヤーのみ初期化する
                playerPosX = WINDOW_SIZE_WID / 2 - PLAYER_SIZE_WID / 2;           // 横方向の画面中央
                playerFlg = true;
            }
            break;
        case ESTAGE_EXIT_ENEMY_OCCUPATION:
            // 条件３：敵に最下段まで到達された
            //   残機を減らして、残機が残っていれば、敵の位置を数段上にずらして再開する
            playerRestPlaneNum--;
            if (playerRestPlaneNum >= 0) {
                SceneGameStageOccupationInit();
                playerPosX = WINDOW_SIZE_WID / 2 - PLAYER_SIZE_WID / 2;           // 横方向の中央
                playerFlg = true;
                enemyMoveDirection = enemyNextMoveDirection;
            } else {
                // 残機がなくなったので、ゲーム終了
                nowScene = SCENE_KIND_GAMEOVER;
            }
            break;
        default:
            break;
        }
    }
}
```

```
/**
 *---------------------------------------------------
 * ゲームシーンの更新処理
 * Input:
 *    無し
 * Output:
 *    無し
 *---------------------------------------------------
 */
void SceneGameUpdate(void)
{
    // プレイヤーの移動処理
    PlayerMoveProc();
    // プレイヤー弾の発射処理
    PlayerBulletShotProc();
    // 敵弾の発射処理
    EnemyBulletShotProc();
    // プレイヤー弾の移動処理
    PlayerBulletMove();
    // 敵弾の移動処理
    EnemyBulletMove();
    // 敵キャラの移動処理
    EnemyMoveProc();

    // 当たり判定
    if (PlayerBulletHitCheck() == true) {

    }
    EnemyBulletHitCheck();                // 敵弾の当たり判定
    GameDataSaveProc();

    //---------------------------------------------------------------
    // テスト用処理
    //---------------------------------------------------------------
    if (nowWKeyPress == 0 && prevWKeyPress == 1) {
        // 条件１：プレイヤーが勝った
        // 敵キャラを無条件に全てやられた状態にする
        for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
            for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
                enemyFlgArray[yy][xx] = false;
            }
        }
    } else if (nowLKeyPress == 0 && prevLKeyPress == 1) {
        // 条件２：敵弾にプレイヤーが撃たれた
        playerFlg = false;
    } else if (nowOKeyPress == 0 && prevOKeyPress == 1) {
        // 条件３：敵に最下段まで到達された
        // 敵キャラのうち一番下に残っているキャラをひとつだけ最下段にY座標を変更する
        int restnum;
        int chkpos = WINDOW_SIZE_HIG - ENEMY_MOVE_Y_SPEED;
        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++;          // その行の残機をカウント
                    enemyPosyArray[yy][xx] = chkpos - (ENEMY_MOVE_Y_SPEED * (ENEMY_DISP_YNUM - (yy+1)));
                }
            }
        }
        enemyNextMoveDirection = enemyMoveDirection;
        enemyMoveDirection = ENEMY_DIR_DOWN;
    }
    //---------------------------------------------------------------
}
```

```
/**
 *-----------------------------------------------------
 * ゲームシーンの描画処理
 * Input:
 *    無し
 * Output:
 *    無し
 *-----------------------------------------------------
 */
void SceneGameDraw(void)
{
    SetDrawScreen(DX_SCREEN_BACK);        // 描画する画面を裏の画面に設定する
    ClearDrawScreen();                    // 描画する画面の内容を消去する

    // プレイヤーの描画
    PlayerDrawProc();
    // プレイヤー弾の描画
    PlayerBulletDraw();
    // 敵の描画
    EnemyDrawProc();
    // 敵弾の描画
    EnemyBulletDraw();
    // 残機数の文字列表示
    DrawFormatString(10, 10, 0xff00ff, "プレイヤーの残機数：%d機", playerRestPlaneNum);
    DrawFormatString(10, 28, 0xff00ff, "現在のステージ番号：ステージ %d", nowStageNo);

    ScreenFlip();                         // 裏の画面を表の画面に瞬間コピー
}

/**
 *-----------------------------------------------------
 * ゲームシーンの解放処理
 * Input:
 *    無し
 * Output:
 *    true = 正常終了 / false = エラー終了
 *-----------------------------------------------------
 */
bool SceneGameReleaseProc(void)
{
    return true;
}

/**
 *-----------------------------------------------------
 * ステージ終了状況のチェック
 * Input:
 *    無し
 * Output:
 *    eStageExitConditionsKind ステージ終了条件種別
 *-----------------------------------------------------
 */
eStageExitConditionsKind StageExitConditionCheck(void)
{
    eStageExitConditionsKind exitCondition = ESTAGE_EXIT_NON;

    // 敵の残機数を取得する
    int enemyNum = 0;
    for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            if (enemyFlgArray[yy][xx] == true)enemyNum++;
        }
    }
```

```
    if (enemyNum == 0) {
        // 敵の残機数が0なので、プレイヤーがステージをクリアした
        exitCondition = ESTAGE_EXIT_CLEAR;
    }
    else if (playerFlg == false) {
        // プレイヤーがやられた
        exitCondition = ESTAGE_EXIT_ENEMY_SHOT;
    }
    else if (CheckHitEdgeProc() == ENEMY_HIT_DIR_DOWN) {
        // 敵が最下段に到達した
        exitCondition = ESTAGE_EXIT_ENEMY_OCCUPATION;
    }

    return exitCondition;
}


/**
 *-----------------------------------------------------
 * セーブ処理
 * Input:
 *    無し
 * Output:
 *    true = 正常終了 / false = エラー終了
 *-----------------------------------------------------
 */
bool GameDataSaveProc(void)
{
    // アップトリガーでキーの押下判定を行う
    if (prevF1KeyPress == 1 && nowF1KeyPress == 0) {
        if (SaveProc() == false)return false;
    }
    return true;
}

bool SaveProc(void)
{
    // 保存するデータのセット
    saveDat.pRestNum = playerRestPlaneNum;
    saveDat.stageNo  = nowStageNo;
    for (int yy = 0; yy < ENEMY_DISP_YNUM; yy++) {
        for (int xx = 0; xx < ENEMY_DISP_XNUM; xx++) {
            saveDat.ePosxArray[yy][xx] = enemyPosxArray[yy][xx];
            saveDat.ePosyArray[yy][xx] = enemyPosyArray[yy][xx];
            saveDat.eFlgArray[yy][xx] = enemyFlgArray[yy][xx];
        }
    }
    saveDat.eMoveDir = enemyMoveDirection;
    saveDat.eNextMoveDir = enemyNextMoveDirection;

    FILE* fp;
    int err;
    if ((err = fopen_s(&fp, "SaveData.dat", "wb")) != 0) {
        printf("File Open error!! err_code = %d¥n", err);
        return false;
    }

    fwrite((char*)&saveDat, sizeof(char), sizeof(saveDat), fp);

    fclose(fp);

    return true;
}
```