```
//-------------------------------------------------------------------    kadai1_Vol04_Sample
// ゲーム制作：shooting
// 氏名　　：0000000　苗字　名前
//-------------------------------------------------------------------
#include <DxLib.h>
#include <time.h>

//-------------------------
// 定数定義
//-------------------------
const int WINDOW_WID = 640;                     // ゲーム画面の横サイズ
const int WINDOW_HIG = 480;                     // ゲーム画面の縦サイズ

const int PLAYER_WID = 32;                      // プレイヤー画像の横サイズ
const int PLAYER_HIG = 32;                      // プレイヤー画像の縦サイズ
const int PLAYER_MOVE_X = 8;                    // プレイヤー機体のX方向の移動量
const int PLAYER_MOVE_Y = 8;                    // プレイヤー機体のY方向の移動量

const int BULLET_WID = 32;                      // 弾の画像の横サイズ
const int BULLET_HIG = 32;                      // 弾の画像の縦サイズ
const int BULLET_SPEED = 16;                    // 弾の移動量

const int ENEMY_WID = 64;                       // 敵画像の横サイズ
const int ENEMY_HIG = 64;                       // 敵画像の縦サイズ
const int ENEMY_SPEED = 4;                      // 敵の移動量

const int HAIKEI_WID = 640;                     // 背景画像の横サイズ
const int HAIKEI_HIG = 960;                     // 背景画像の縦サイズ
const int HAIKEI_MOVE_SPEED = 3;                // 背景の移動量

const int BLAST_WID = 96;                       // 爆発画像の横サイズ
const int BLAST_HIG = 96;                       // 爆発画像の縦サイズ
const int BLAST_ANIM_XNUM = 6;                  // 爆発画像ファイル内の横方向のパターン数
const int BLAST_ANIM_YNUM = 4;                  // 爆発画像ファイル内の縦方向のパターン数
const int BLAST_ANIM_MAX = BLAST_ANIM_XNUM * BLAST_ANIM_YNUM;
                                                // 爆発アニメーションのパターン数
//-------------------------
// 変数宣言
//-------------------------
int playerImage;            // 自機の画像のハンドル番号
int playerPosX;             // 自機のX座標
int playerPosY;             // 自機のY座標
bool playerFlg;             // 自機の表示状態管理用フラグ(true:表示、false:非表示)

int bulletImage;            // 弾の画像のハンドル番号
int bulletPosX;             // 弾のX座標
int bulletPosY;             // 弾のY座標
bool shotFlg;               // 弾の発射状態を管理する(true=発射中, false=未発射)

int enemyImage;             // 敵の画像を格納する領域
int enemyPosX;              // 敵のX座標
int enemyPosY;              // 敵のY座標
bool enemyFlg;              // 敵の表示状態管理用フラグ(true:表示、false:非表示)
```

```
int haikeiImage;              // 背景画像のハンドル番号
int haikeiPosX;               // 背景のX座標
int haikeiPosY;               // 背景のY座標

int blastImage;               // 爆発画像ファイルのハンドル番号
int blastImageArray[BLAST_ANIM_MAX];   // 爆発アニメーション用画像毎のハンドル番号テーブル
int blastPosX;                // 爆発の表示X座標
int blastPosY;                // 爆発の表示Y座標
int blastAnimCounter;         // 爆発アニメーションの表示カウンター(0～)
bool blastAnimFlg;            // 爆発の表示状態管理フラグ(true=表示, false=非表示)

bool blastEnemyFlg;           // 敵の爆発フラグ(true=爆発, false=爆発していない)
bool blastPlayerFlg;          // プレイヤー爆発フラグ(true=爆発, false=爆発していない)

// テスト用
bool playerHitFlg;            // プレイヤーと敵が衝突したかどうか(true=衝突, false=非接触)


//-------------------------
// WinMain関数
//-------------------------
int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrevInst, LPSTR lpCmdLine, int nCmdShow)
{
  //-------------------------
  // システム処理
  //-------------------------
  SetWindowText("学籍番号 氏名");   // ゲームウィンドウのタイトル
  SetGraphMode(640, 480, 32);       // ゲームウィンドウのサイズと色モードを設定
  ChangeWindowMode(true);           // ゲームウィンドwの表示方法(false = フルスクリーン)

  if (DxLib_Init() == -1) {         // DXライブラリの初期化処理
    return -1;              // DXライブラリの初期化失敗の為システム終了
  }

  // 乱数の初期化
  SRand((unsigned int)time(NULL));

  //-------------------------
  // グラフィックの登録
  //-------------------------
  // 自機画像の読込み
  playerImage = LoadGraph("image/player.png");
  if (playerImage == -1) {
    return -1;              // 画像読み込みに失敗したので、エラー終了
  }

  // 弾画像の読み込み
  bulletImage = LoadGraph("image/shot.png");
  if (bulletImage == -1) {
    return -1;              // 画像読み込みに失敗したので、エラー終了
  }
```

```
// 敵画像の読込み
enemyImage = LoadGraph("image/enemy1.png");
if (enemyImage == -1) {
  return -1;               // 画像読み込みに失敗したので、エラー終了
}

// 背景画像の読み込み
haikeiImage = LoadGraph("image/haikei.png");
if (haikeiImage == -1) {
  return -1;               // 画像読み込みに失敗したので、エラー終了
}

// 爆発画像ファイルを読み込む
int err;
err = LoadDivGraph("image/blast.png", BLAST_ANIM_MAX,
  BLAST_ANIM_XNUM, BLAST_ANIM_YNUM,
  BLAST_WID, BLAST_HIG, blastImageArray);
if (err == -1) {
  return -1;               // 画像読み込みに失敗したので、エラー終了
}

//--------------------------
// 変数の初期化
//--------------------------
//プレイヤー
playerPosX = WINDOW_WID / 2 - PLAYER_WID / 2;    // ウィンドウの中央X座標
playerPosY = WINDOW_HIG / 2 - PLAYER_HIG / 2;    // ウィンドウの中央Y座標
playerFlg = true;

// プレイヤーの弾
bulletPosX = 0;
bulletPosY = 0;
shotFlg = false;             // 弾が未発射状態

// 敵
enemyPosX = WINDOW_WID / 2 - ENEMY_WID / 2;      // ウィンドウの横中央
enemyPosY = ENEMY_HIG;                           // ウィンドウ最上部
enemyFlg = true;

// 背景
haikeiPosX = 0;
haikeiPosY = 0 - (HAIKEI_HIG - WINDOW_HIG);

// 爆発アニメーション
blastPosX = 0;
blastPosY = 0;
blastAnimCounter = 0;
blastAnimFlg = false;
blastEnemyFlg = false;
blastPlayerFlg = false;

// テスト用
playerHitFlg = false;
```

```
//-------------------------
// ゲームループ
//-------------------------
while (ProcessMessage() == 0 && CheckHitKey(KEY_INPUT_ESCAPE) == 0) {
  //-------------------------
  // ゲームのメイン処理
  //-------------------------
  // 背景のスクロール処理
  haikeiPosY += HAIKEI_MOVE_SPEED;
  if (haikeiPosY > HAIKEI_HIG) {
    haikeiPosY -= HAIKEI_HIG;
  }

  //-------------------------
  // プレイヤー処理
  //-------------------------
  if (playerFlg == false) {
    // プレイヤーが非表示になっているので、再表示する
    playerPosX = WINDOW_WID / 2 - PLAYER_WID / 2;
    playerPosY = WINDOW_HIG - PLAYER_HIG;
    playerFlg = true;
  }

  if (CheckHitKey(KEY_INPUT_RIGHT) == 1) {
    // 右キーが押されているので、機体を右に移動させる
    playerPosX = playerPosX + PLAYER_MOVE_X;
  }
  if (CheckHitKey(KEY_INPUT_LEFT) == 1) {
    // 左キーが押されているので、機体を左に移動させる
    playerPosX = playerPosX - PLAYER_MOVE_X;
  }
  if (CheckHitKey(KEY_INPUT_UP) == 1) {
    // 上キーが押されているので、機体を上に移動させる
    playerPosY = playerPosY - PLAYER_MOVE_Y;
  }
  if (CheckHitKey(KEY_INPUT_DOWN) == 1) {
    // 下キーが押されているので、機体下に移動させる
    playerPosY = playerPosY + PLAYER_MOVE_Y;
  }

  // 自機の現在位置がウィンドウ外に飛び出していないかのチェックを行う。
  // 左端のチェック
  if (playerPosX < 0) {
    playerPosX = 0;
  }
  // 右端のチェック
  if (playerPosX > (WINDOW_WID - PLAYER_WID)) {
    playerPosX = (WINDOW_WID - PLAYER_WID);
  }
  // 上端のチェック
  if (playerPosY < 0) {
    playerPosY = 0;
  }
```

```
// 下端のチェック
if (playerPosY > (WINDOW_HIG - PLAYER_HIG)) {
  playerPosY = (WINDOW_HIG - PLAYER_HIG);
}

//-------------------------
// 弾の処理
//-------------------------
if (shotFlg == false) {
  // 弾が未発射状態なので、弾を撃つ事ができる
  if (CheckHitKey(KEY_INPUT_SPACE) == 1) {
    // スペースキーが押されたので、弾を発射する
    bulletPosX = playerPosX;
    bulletPosY = playerPosY - BULLET_HIG;

    shotFlg = true;
  }
}
else {
  // 弾が発射されている状態
  // 弾を移動させる
  bulletPosY = bulletPosY - BULLET_SPEED;

  if (bulletPosY < 0) {
    // 弾がウィンドウ外に出たので、未発射状態にする
    shotFlg = false;
  }
}

//-------------------------
// 敵の処理
//-------------------------
if (enemyFlg == true) {
  // 敵の機体は表示中なので、移動させる
  enemyPosY = enemyPosY + ENEMY_SPEED;

  if (enemyPosY > WINDOW_HIG) {
    // 敵がウィンドウの外に飛び出したので、消去する
    enemyFlg = false;
  }
}
else {
  // 敵が非表示になっているので、再度画面上部に表示する
  enemyPosX = GetRand(WINDOW_WID - ENEMY_WID);     // X座標はランダムに変化させる
  enemyPosY = 0 - ENEMY_HIG;
  enemyFlg = true;                                 // フラグを表示に設定しなおす
}
```

```
//--------------------------
// 爆発アニメーション処理
//--------------------------
if (blastAnimFlg == true) {
  blastAnimCounter++;

  if (blastAnimCounter >= BLAST_ANIM_MAX) {
    blastAnimFlg = false;
    if (blastEnemyFlg == true) {
      blastEnemyFlg = false;
    }
    if (blastPlayerFlg == true) {
      blastPlayerFlg = false;
    }
  }
}

//--------------------------
//当たり判定
//--------------------------
// 弾と敵
if (enemyFlg == true && shotFlg == true) {
  // 敵と弾の両方ともが表示されている
  if (enemyPosY + ENEMY_WID > bulletPosY            // ①敵の下 ＞ 弾の上
    && enemyPosY < bulletPosY + BULLET_HIG          // ②敵の上 ＜ 弾の下
    && enemyPosX < bulletPosX + BULLET_WID          // ③敵の左 ＜ 弾の右
    && enemyPosX + ENEMY_WID > bulletPosX           // ④敵の右 ＞ 弾の左
    ) {
    enemyFlg = false;             // 敵を消す
    shotFlg = false;              // 弾を消す

    // 爆発アニメーション用設定
    blastPosX = (enemyPosX + ENEMY_WID / 2) - (BLAST_WID / 2);
    blastPosY = (enemyPosY + ENEMY_HIG / 2) - (BLAST_HIG / 2);
    blastAnimCounter = 0;
    blastAnimFlg = true;
    blastEnemyFlg = true;
  }
}
```

```
// プレイヤーと敵
if (playerFlg == true && enemyFlg == true) {
  // プレイヤーと敵の両方ともが表示されている
  if (enemyPosY + ENEMY_WID > playerPosY              // ①敵の下 ＞ 自機の上
    && enemyPosY < playerPosY + PLAYER_HIG            // ②敵の上 ＜ 自機の下
    && enemyPosX < playerPosX + PLAYER_WID            // ③敵の左 ＜ 自機の右
    && enemyPosX + ENEMY_WID > playerPosX             // ④敵の右 ＞ 自機の左
    ) {
    playerFlg = false;              // プレイヤーを消す

    // 爆発アニメーション用設定
    blastPosX = (playerPosX + PLAYER_WID / 2) - (BLAST_WID / 2);
    blastPosY = (playerPosY + PLAYER_HIG / 2) - (BLAST_HIG / 2);
    blastAnimCounter = 0;
    blastAnimFlg = true;
    blastPlayerFlg = true;

    playerHitFlg = true;
  }
  else {
    playerHitFlg = false;
  }
}
```

```
    //-------------------------
    // 描画処理
    //-------------------------
    SetDraw// 描画する画面を裏の画面に設定する
    ClearDrawScree// 描画する画面の内容を消去する

    // 画面上に対角線を引く
    DrawLine(0, 0, WINDOW_WID - 1, WINDOW_HIG - 1, GetColor(255, 255, 255));
    DrawLine(WINDOW_WID - 1, 0, 0, WINDOW_HIG - 1, GetColor(255, 255, 255));

    // 背景画像を表示する。
    DrawGraph(haikeiPosX, haikeiPosY, haikeiImage, true);
    DrawGraph(haikeiPosX, haikeiPosY - HAIKEI_HIG, haikeiImage, true);
    // 背景の表示開始Y座標が０よりも大きくなって裏の黒い部分が見える様になった
    // 時の為に、表示開始座標の直上にもう一枚背景画像を表示しておいてやる
    // 背景が切れ目なく表示が続く様にする為。

    // 敵画像を描画する
    if (enemyFlg == true) {
      DrawGraph(enemyPosX, enemyPosY, enemyImage, true);
    }

    // 自機画像を描画する
    if (playerFlg == true) {
      DrawGraph(playerPosX, playerPosY, playerImage, true);
    }

    // 弾画像を描画する
    if (shotFlg == true) {
      DrawGraph(bulletPosX, bulletPosY, bulletImage, true);
    }

    // 爆発画像を表示する
    if (blastAnimFlg == true) {
      DrawGraph(blastPosX, blastPosY, blastImageArray[blastAnimCounter], true);
    }

    // 自機の現在の位置座標を表示する
    DrawFormatString(0, 0, 0xffffff, "playerPos = (%d, %d)", playerPosX, playerPosY);

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

  //-------------------------
  // システム終了処理
  //-------------------------
  DxLib_End();          // DXライブラリの終了処理
  return 0;             // ゲームの終了
}
```