```
#include "DxLib.h"
#include "SceneMgr.h"
#include "Input.h"
#include "Player.h"
#include "Enemy.h"
#include "Effect.h"
#include <math.h>


/*************************
** プロトタイプ宣言 **
*************************/
int HitCirclePlayerShot(ENEMY*, PLAYER_SHOT*);  //エネミー当たり判定（円）


/*************************
** 変数 **
*************************/
// 自機
struct PLAYER mPlayer;
struct PLAYER_SHOT mPlayerShot[PLAYER_SHOT_MAX];

static int mImagePlayer;    //画像ハンドル格納用変数
static int mImageHitPoint[2];  //画像ハンドル格納用変数
static int mImageHodai[3];  //画像ハンドル格納用変数
static int mImageDan[3];    //画像ハンドル格納用変数

static int mSoundsShot;    //音ファイル格納用変数


int MPlayerHodaiPosion[2][7][3] = {
    {
        {-24, -12, 1},
        {-16, -24, 1},
        { -8, -36, 0},
        {  0, -48, 2},
        {  8, -36, 0},
        { 16, -24, 1},
        { 24, -12, 1}
    },
    {
        {-48, -12, 1},
        {-32, -24, 1},
        {-16, -36, 0},
        {  0, -48, 2},
        { 16, -36, 0},
        { 32, -24, 1},
        { 48, -12, 1}
    }
};

int MPlayerShotMove[2][7][2] = {
    {
        { 0, -14},
        { 0, -14},
        { 0, -14},
        { 0, -14},
        { 0, -14},
        { 0, -14},
        { 0, -14}
```

```
61
62      },
63      {
64          {-3, -11},
65          {-2, -12},
66          {-1, -13},
67          { 0, -14},
68          { 1, -13},
69          { 2, -12},
70          { 3, -11}
71      }
72  };
73
74
75  /*********************************************
76   * プレイヤー　初期化
77   * 引　数：なし
78   * 戻り値：なし
79   *********************************************/
80  void Player_Initialize(){
81      mImagePlayer = LoadGraph("images/player.png");    //プレイヤー画像
82      LoadDivGraph("images/playerhantei.png", 2, 2, 1, 60, 60, mImageHitPoint);   //プレイ
          ヤー当たり判定
83      LoadDivGraph("images/playerHodai1.png", 3, 3, 1, 12, 16, mImageHodai);  //プレイヤー
          砲台画像
84      LoadDivGraph("images/playerDan1.png", 3, 3, 1, 12, 24, mImageDan);  //プレイヤー弾画
          像
85
86      mSoundsShot = LoadSoundMem("sounds/shoot.wav");    //弾発射音のロード
87
88      // プレイヤーの初期設定
89      mPlayer.flg = true;
90      mPlayer.x = PLAYER_POS_X;
91      mPlayer.y = PLAYER_POS_Y;
92      mPlayer.w = PLAYER_WIDTH;
93      mPlayer.h = PLAYER_HEIGHT;
94      mPlayer.angle = 0.0;
95      mPlayer.count = 0;
96      mPlayer.hp = PLAYER_HP;
97      mPlayer.score = 0;
98      // *** +1line
99      mPlayer.r = PLAYER_HIT_R;
100
101     // プレイヤー弾の初期設定
102     for (int x = 0; x < PLAYER_SHOT_MAX; x++) {
103         mPlayerShot[x].flg = FALSE;
104     }
105 }
106
107
108 /*********************************************
109  * プレイヤー　終了処理
110  * 引　数：なし
111  * 戻り値：なし
112  *********************************************/
113 void Player_Finalize(){
114     DeleteGraph(mImagePlayer);    //画像の解放
115 }
116
117
```

```
118 /*********************************************
119  * プレイヤー　更新
120  * 引　数：なし
121  * 戻り値：なし
122  *********************************************/
123 void Player_Update(){
124 
125     // 上下左右移動
126     if (iNowKey & PAD_INPUT_UP)     mPlayer.y -= PLAYER_SPEED;
127     if (iNowKey & PAD_INPUT_DOWN)   mPlayer.y += PLAYER_SPEED;
128     if (iNowKey & PAD_INPUT_LEFT)   mPlayer.x -= PLAYER_SPEED;
129     if (iNowKey & PAD_INPUT_RIGHT)  mPlayer.x += PLAYER_SPEED;
130 
131     // 画面をはみ出さないようにする
132     if (mPlayer.x < SCREEN_WIDTH_L + mPlayer.w/2) mPlayer.x = SCREEN_WIDTH_L +
          mPlayer.w/2;
133     if (mPlayer.x > SCREEN_WIDTH_R - mPlayer.w/2) mPlayer.x = SCREEN_WIDTH_R -
          mPlayer.w/2;
134     if (mPlayer.y < 32)                  mPlayer.y = 32;
135     if (mPlayer.y > SCREEN_HEIGHT - 32) mPlayer.y = SCREEN_HEIGHT - 32;
136 
137     // *** +2lines
138     // プレイヤーの状態がTRUEの時
139     if (mPlayer.flg) {
140         //Zキーが押されていたらプレイヤー弾生成
141         mPlayer.shotCount++;
142         if (mPlayer.shotCount > PLAYER_SHOT_INTERVAL) {
143             if (iNowKey & PAD_INPUT_1) {
144                 mPlayer.shotCount = 0;
145 
146                 if (MakePlayerShot()) {
147                     PlaySoundMem(mSoundsShot, DX_PLAYTYPE_BACK);
148 
149                 }
150             }
151         }
152     // *** +1line
153     }
154 
155     //プレイヤー弾移動；
156     MovePlayerShot();
157 
158     //経過時間　（背景スクロール＆エネミー生成時に使用する）
159     mPlayer.time++;
160 
161 }
162 
163 
164 /*********************************************
165  * プレイヤー　描画処理
166  * 引　数：なし
167  * 戻り値：なし
168  *********************************************/
169 void Player_Draw(){
170 
171     // プレイヤーの表示
172     if (mPlayer.flg) {
173         DrawRotaGraph(mPlayer.x, mPlayer.y, 1.0f, 0, mImagePlayer, TRUE, FALSE);
174         // *** +2lines
175         // プレイヤー当たり判定ポイント
```

```
176         DrawRotaGraph(mPlayer.x, mPlayer.y, 1.0f, 0, mImageHitPoint[1], TRUE, FALSE);
177
178         // ストレート弾（収束ショット）
179         if (iNowKey & PAD_INPUT_B) {
180             for (int i = 0; i < 7; i++) {
181                 DrawRotaGraph(mPlayer.x + MPlayerHodaiPosion[0][i][0],
182                               mPlayer.y + MPlayerHodaiPosion[0][i][1],
183                               1.0f, 0, mImageHodai[MPlayerHodaiPosion[0][i][2]], TRUE,
                          FALSE);
184             }
185             DrawRotaGraph(mPlayer.x, mPlayer.y, 1.0f, 0, mImageHitPoint[1], TRUE, FALSE);
186         }
187         else {//  ｎ－ｗａｙ弾
188             for (int i = 0; i < 7; i++) {
189                 DrawRotaGraph(mPlayer.x + MPlayerHodaiPosion[1][i][0],
190                               mPlayer.y + MPlayerHodaiPosion[1][i][1],
191                               1.0f, 0, mImageHodai[MPlayerHodaiPosion[1][i][2]], TRUE,
                          FALSE);
192             }
193         }
194     }
195     else {
196         // プレイヤー復活までの時間3秒（180フレーム）
197         if (++mPlayer.count >= 80)  mPlayer.flg = true;
198         if (mPlayer.count / 5 % 2 == 0) {
199             DrawRotaGraph(mPlayer.x, mPlayer.y, 1.0f, 0, mImagePlayer, TRUE, FALSE);
200         }
201     }
202
203     DrawPlayerShot();
204
205 }
206
207
208 /***********************************************
209  * プレイヤー弾生成
210  * 引　数：なし
211  * 戻り値：なし
212  ***********************************************/
213 int MakePlayerShot() {
214
215     bool flg = FALSE;
216
217     // ストレート弾（収束ショット）
218     if (iNowKey & PAD_INPUT_B) {
219         for (int i = 0; i < 7; i++) {
220             // プレイヤー弾の生成
221             for (int x = 0; x < PLAYER_SHOT_MAX; x++) {
222                 if (mPlayerShot[x].flg == FALSE) {
223                     flg = TRUE;
224
225                     mPlayerShot[x].flg = TRUE;
226                     mPlayerShot[x].w = 12;
227                     mPlayerShot[x].h = 14;
228                     mPlayerShot[x].x = mPlayer.x + MPlayerHodaiPosion[0][i][0];
229                     mPlayerShot[x].y = mPlayer.y + MPlayerHodaiPosion[0][i][1];
230                     mPlayerShot[x].image = mImageDan[MPlayerHodaiPosion[0][i][2]];
231                     mPlayerShot[x].mx = MPlayerShotMove[0][i][0];
232                     mPlayerShot[x].my = MPlayerShotMove[0][i][1];
233                     break;
```

```
234                 }
235             }
236         }
237     }
238     else{//  ｎ－ｗａｙ弾
239         for (int i = 0; i < 7; i++) {
240             // プレイヤー弾の生成
241             for (int x = 0; x < PLAYER_SHOT_MAX; x++) {
242                 if (mPlayerShot[x].flg == FALSE) {
243                     flg = TRUE;
244 
245                     mPlayerShot[x].flg = TRUE;
246                     mPlayerShot[x].w = 12;
247                     mPlayerShot[x].h = 14;
248                     mPlayerShot[x].x = mPlayer.x + MPlayerHodaiPosion[1][i][0];
249                     mPlayerShot[x].y = mPlayer.y + MPlayerHodaiPosion[1][i][1];
250                     mPlayerShot[x].image = mImageDan[MPlayerHodaiPosion[1][i][2]];
251                     mPlayerShot[x].mx = MPlayerShotMove[1][i][0];
252                     mPlayerShot[x].my = MPlayerShotMove[1][i][1];
253                     break;
254                 }
255             }
256         }
257     }
258 
259     return flg;
260 }
261 
262 
263 /**********************************************
264  * プレイヤー弾移動
265  * 引　数：なし
266  * 戻り値：なし
267  **********************************************/
268 void MovePlayerShot() {
269     // プレイヤー弾の初期設定
270     for (int x = 0; x < PLAYER_SHOT_MAX; x++) {
271         if (mPlayerShot[x].flg == TRUE) {
272             mPlayerShot[x].x += mPlayerShot[x].mx;
273             mPlayerShot[x].y += mPlayerShot[x].my;
274 
275             if (mPlayerShot[x].y < -14) {
276                 mPlayerShot[x].flg = FALSE;
277             }
278 
279             // 当たり判定
280             for (int y = 0; y < ENEMY_MAX; y++) {
281                 if (mEnemy[y].flg == false) continue;
282                 if (HitCirclePlayerShot(&mEnemy[y], &mPlayerShot[x]) == TRUE) {
283                     if (--mEnemy[y].hp < 0) {
284                         mEnemy[y].flg = FALSE;
285                         mPlayer.score += mEnemy[y].point;
286                         CreateEffect(mEnemy[y].x, mEnemy[y].y);
287 
288                     }
289                     mPlayerShot[x].flg = FALSE;
290                     break;
291                 }
292             }
293         }
```

```
294     }
295 }
296
297
298 /***********************************************
299  * プレイヤー弾描画
300  * 引　数：なし
301  * 戻り値：なし
302  ***********************************************/
303 void DrawPlayerShot() {
304     // プレイヤー弾の初期設定
305     for (int x = 0; x < PLAYER_SHOT_MAX; x++) {
306         if (mPlayerShot[x].flg == TRUE) {
307             DrawRotaGraph(mPlayerShot[x].x, mPlayerShot[x].y, 1.0f, 0, mPlayerShot
                  [x].image, TRUE, FALSE);
308         }
309     }
310
311 }
312
313
314 /*********************************
315  * プレイヤー弾当たり判定（円）
316  * 引　数：PLAYERポインタ, ENEMY_SHOTポインタ
317  * 戻り値：TRUE:当たり, FALSE:なし
318  *********************************/
319 int HitCirclePlayerShot(ENEMY* e, PLAYER_SHOT* p)
320
321 {
322     double x = (double)p->x - (double)e->x;
323     double y = (double)p->y - (double)e->y;
324
325     int r = (int)sqrt(x * x + y * y);
326
327     if (r <= p->r + e->r) return TRUE;
328
329     return FALSE;
330
331 }
332
```