```
#include "DxLib.h"
#include "SceneMgr.h"
#include "Input.h"
#include "Enemy.h"
#include "Player.h"
// *** +1line
#include <math.h>


/*************************
** 変数 **
*************************/
// エネミー
struct ENEMY mEnemy[ENEMY_MAX];
int mImageEnemyZako[ENEMY_IMAGE_MAX];
int mImageEnemyMiko[ENEMY_IMAGE_MAX];

// エネミー弾
struct ENEMY_SHOT mEnemyShot[ENEMY_SHOT_MAX];
int mImageEnemyShot[ENEMY_SHOT_IMAGE_MAX];


/*********************************************
 * 敵 初期化
 * 引　数：なし
 * 戻り値：なし
 *********************************************/
void Enemy_Initialize(){
    LoadDivGraph("images/enemyZako.png", 2, 2, 1, 30, 30, mImageEnemyZako); //エネミー画像（雑魚：ザコ）
    LoadDivGraph("images/enemyMiko.png", 2, 2, 1, 40, 45, mImageEnemyMiko); //エネミー画像（巫女：ミコ）
    LoadDivGraph("images/EnemyDan.png", 4, 4, 1, 16, 16, mImageEnemyShot);  //エネミー弾画像

    // エネミーの初期設定
    for (int i = 0; i < ENEMY_MAX; i++) {
        mEnemy[i].flg = FALSE;
    }

    // エネミー弾の初期設定
    for (int i = 0; i < ENEMY_SHOT_MAX; i++) {
        mEnemyShot[i].flg = FALSE;
    }
}


/*********************************************
 * 敵 終了更新
 * 引　数：なし
 * 戻り値：なし
 *********************************************/
void Enemy_Finalize(){
    for (int i = 0; i < ENEMY_IMAGE_MAX; i++) {
        DeleteGraph(mImageEnemyZako[i]);    //画像の解放
        DeleteGraph(mImageEnemyMiko[i]);    //画像の解放
    }
    for (int i = 0; i < ENEMY_SHOT_IMAGE_MAX; i++) {
        DeleteGraph(mImageEnemyShot[i]);    //画像の解放
    }
```

```
 58  }
 59
 60  /*********************************************
 61   * 敵 処理更新
 62   * 引　数：なし
 63   * 戻り値：なし
 64   *********************************************/
 65  void Enemy_Update(){
 66      //エネミー処理
 67      for (int i = 0; i < ENEMY_MAX; i++) {
 68          if (mEnemy[i].flg == TRUE) {
 69
 70              // エネミー移動
 71              mEnemy[i].x += mEnemy[i].mx;
 72              mEnemy[i].y += mEnemy[i].my;
 73
 74              mEnemy[i].cnt++;
 75
 76              // パターン別　敵の動き
 77              switch (mEnemy[i].type) {
 78                  case 1: EnemyType01(&mEnemy[i]); break;
 79                  case 2: EnemyType02(&mEnemy[i]); break;
 80  // *** +2lines
 81                  case 3: EnemyType03(&mEnemy[i]); break;
 82                  case 4: EnemyType04(&mEnemy[i]); break;
 83              }
 84
 85              // 画面をはみ出したら消去
 86              if (mEnemy[i].y > SCREEN_HEIGHT + mEnemy[i].h)
 87                  mEnemy[i].flg = FALSE;
 88
 89          }
 90      }
 91
 92      //エネミー弾処理
 93      for (int i = 0; i < ENEMY_SHOT_MAX; i++) {
 94          if (mEnemyShot[i].flg == TRUE) {
 95
 96              // エネミー弾移動
 97              mEnemyShot[i].x += mEnemyShot[i].mx;
 98              mEnemyShot[i].y += mEnemyShot[i].my;
 99
100              // 画面をはみ出したら消去
101              if (mEnemyShot[i].y > SCREEN_HEIGHT + mEnemyShot[i].h)
102                  mEnemyShot[i].flg = FALSE;
103
104          }
105      }
106  }
107
108
109  /*********************************************
110   * 敵 描画処理
111   * 引　数：なし
112   * 戻り値：なし
113   *********************************************/
114  void Enemy_Draw(){
115      //エネミー処理
116      for (int i = 0; i < ENEMY_MAX; i++) {
117          if (mEnemy[i].flg == TRUE) {
```

```
118             DrawRotaGraph(mEnemy[i].x, mEnemy[i].y, 1.0f, 0, mEnemy[i].img, TRUE, FALSE);
119         }
120     }
121     //エネミー弾処理
122     for (int i = 0; i < ENEMY_SHOT_MAX; i++) {
123         if (mEnemyShot[i].flg == TRUE) {
124                 DrawRotaGraph(mEnemyShot[i].x, mEnemyShot[i].y, 1.0f, 0, mEnemyShot
                      [i].img, TRUE, FALSE);
125         }
126     }
127 }
128
129
130 /**********************************
131  * エネミーの生成
132  * 引　数：type:タイプ１－４はザコ、５－７は巫女
133  *         img：１はザコ青、２はザコ赤、1は巫女青。２は巫女赤
134  *
135  * 戻り値：TRUE:成功 FALSE:失敗　（戻り値は使用していない）
136  **********************************/
137 int CreateEnemy(int type, int img, int x, int y, int w, int h, int point, int mx, int my)
138 {
139     for (int i = 0; i < ENEMY_MAX; i++) {
140         if (mEnemy[i].flg == FALSE) {
141             mEnemy[i].flg = TRUE;
142             mEnemy[i].type = type;
143             if (type <= 4) {
144                 mEnemy[i].img = mImageEnemyZako[img];
145                 mEnemy[i].hp = ENEMY_ZAKO_LIFE;
146                 mEnemy[i].r = ENEMY_ZAKO_HIT_R;
147             }
148             else {
149                 mEnemy[i].img = mImageEnemyMiko[img];
150                 mEnemy[i].hp = ENEMY_MIKO_RED_LIFE;
151                 mEnemy[i].r = ENEMY_MIKO_HIT_R;
152             }
153
154             mEnemy[i].x = x;
155             mEnemy[i].y = y;
156             mEnemy[i].w = w;
157             mEnemy[i].h = h;
158             mEnemy[i].point = point;
159             mEnemy[i].mx = mx;
160             mEnemy[i].my = my;
161             mEnemy[i].cnt = 0;
162
163             // 成功
164             return TRUE;
165         }
166     }
167
168     // 失敗
169     return FALSE;
170 }
171
172
173 /**********************************************
174  * 敵 タイプ１の移動と攻撃
175  * 引　数：*mEnemy
176  * 戻り値：なし
```

```
177  ************************************************/
178 void EnemyType01(ENEMY* mEnemy)
179 {
180     if (mEnemy->cnt == 80) {
181         mEnemy->mx = -1;
182     }
183 }
184
185 /************************************************
186  * 敵 タイプ２の移動と攻撃
187  * 引　数：*mEnemy
188  * 戻り値：なし
189  ************************************************/
190 void EnemyType02(ENEMY* mEnemy)
191 {
192     if (mEnemy->cnt == 80) {
193         mEnemy->mx = 1;
194     }
195 }
196
197
198 // *** +15lines(1-function)
199 /************************************************
200  * 敵 タイプ３の移動と攻撃
201  * 引　数：*mEnemy
202  * 戻り値：なし
203  ************************************************/
204 void EnemyType03(ENEMY* mEnemy)
205 {
206     if (mEnemy->cnt == 80) {
207         mEnemy->mx = 1;
208     }
209     //狙い撃ち弾
210     if (mEnemy->cnt % 80 == 0 && mEnemy->y < 400) {
211         CreateTargetShot(mEnemy);
212     }
213 }
214
215
216 // *** +15lines(1-function)
217 /************************************************
218  * 敵 タイプ４の移動と攻撃
219  * 引　数：*mEnemy
220  * 戻り値：なし
221  ************************************************/
222 void EnemyType04(ENEMY* mEnemy)
223 {
224     if (mEnemy->cnt == 80) {
225         mEnemy->mx = -1;
226     }
227     //狙い撃ち弾
228     if (mEnemy->cnt % 80 == 0 && mEnemy->y < 400) {
229         CreateTargetShot(mEnemy);
230     }
231 }
232
233 // *** +28lines(1-function)
234 /**********************************
235  * エネミー弾の生成
236  * 引　数：type:タイプ１－４はザコ、５－７は巫女
```

```
237  *          img : １はザコ青、２はザコ赤、1は巫女青。２は巫女赤
238  *
239  * 戻り値 : TRUE:成功 FALSE:失敗
240  ********************************/
241 int CreateEnemyShot(int img, double x, double y, double mx, double my)
242 {
243     for (int i = 0; i < ENEMY_SHOT_MAX; i++) {
244         if (mEnemyShot[i].flg == FALSE) {
245             mEnemyShot[i].flg = TRUE;
246             mEnemyShot[i].w = 16;
247             mEnemyShot[i].h = 16;
248 
249             mEnemyShot[i].img = mImageEnemyShot[img];
250             mEnemyShot[i].x = x;
251             mEnemyShot[i].y = y;
252             mEnemyShot[i].mx = mx;
253             mEnemyShot[i].my = my;
254 
255             // 成功
256             return TRUE;
257         }
258     }
259 
260     // 失敗
261     return FALSE;
262 }
263 
264 
265 // *** +16lines(1-function)
266 /************************************************
267  * 敵弾　狙い撃ち
268  * 引　数 : *mEnemy
269  * 戻り値 : なし
270  ************************************************/
271 void CreateTargetShot(ENEMY* mEnemy)
272 {
273     // 敵機と自機のX,Y方向の角度を求める
274     double Angle = atan2((double)(mPlayer.y - (int)mEnemy->y), (double)(mPlayer.x - (int) ➥
      mEnemy->x));
275 
276     // 飛んでいく方向とスピードから、X軸方向への移動速度とY方向への移動速度を得る
277     double mx = cos(Angle) * ENEMY_SHOT_SPEED;
278     double my = sin(Angle) * ENEMY_SHOT_SPEED;
279 
280     CreateEnemyShot(3, mEnemy->x, mEnemy->y, mx, my);
281 }
282 
```