```
#include "GameMainScene.h"
#include "../Object/RankingData.h"
#include "DxLib.h"
#include <math.h>

GameMainScene::GameMainScene() : high_score(0), back_ground(NULL),
barrier_image(NULL),
                                          mileage(0), player(nullptr),
enemy(nullptr)
{
      for (int i = 0; i < 3; i++)
      {
            enemy_image[i] = NULL;
            enemy_count[i] = NULL;
      }
}

GameMainScene::~GameMainScene()
{

}

// 初期化処理
void GameMainScene::Initialize()
{
      // 高得点値を読み込む
      ReadHighScore();

      // 画像の読み込み
      back_ground = LoadGraph("Resource/images/back.bmp");
      barrier_image = LoadGraph("Resource/images/barrier.png");
      int result = LoadDivGraph("Resource/images/car.bmp", 3, 3, 1, 63, 120,
enemy_image);

      // エラーチェック
      if (back_ground == -1)
      {
            throw ("Resource/images/back.bmpがありません\n");
      }
```

```
	if (result == -1)
	{
		throw ("Resource/images/car.bmpがありません\n");
	}
	if (barrier_image == -1)
	{
		throw ("Resource/images/barrier.pngがありません\n");
	}

	// オブジェクトの生成
	player = new Player;
	enemy = new Enemy* [10];

	// オブジェクトの初期化
	player->Initialize();

	for (int i = 0; i < 10; i++)
	{
		enemy[i] = nullptr;
	}
}

// 更新処理
eSceneType GameMainScene::Update()
{
	// プレイヤーの更新
	player->Update();

	// 移動距離の更新
	mileage += (int)player->GetSpeed
	() + 5;

	// 敵生成処理
	if (mileage / 20 % 100 == 0)
	{
		for (int i = 0; i < 10; i++)
		{
			if (enemy[i] == nullptr)
			{
```

```
                int type = GetRand(3) % 3;
                enemy[i] = new Enemy(type, enemy_image[type]);
                enemy[i]->Initialize();
                break;
            }
        }
}

// 敵の更新と当たり判定チェック
for (int i = 0; i < 10; i++)
{
        if (enemy[i] != nullptr)
        {
            enemy[i]->Update(player->GetSpeed());

            // 画面外に行ったら、敵を削除してスコア加算
            if (enemy[i]->GetLocation().y >= 640.0f)
            {
                enemy_count[enemy[i]->GetType()]++;
                enemy[i]->Finalize();
                delete enemy[i];
                enemy[i] = nullptr;
            }

            // 当たり判定の確認
            if (IsHitCheck(player, enemy[i]))
            {
                player->SetActive(false);
                player->DecreaseHp(-50.0f);
                enemy[i]->Finalize();
                delete enemy[i];
                enemy[i] = nullptr;
            }
        }
}

// プレイヤーの燃料か体力が0未満なら、リザルトに遷移する
if (player->GetFuel() < 0.0f || player->GetHp() < 0.0f)
{
```

```
			return eSceneType::E_RESULT;
		}

		return GetNowScene();
}

// 描画処理
void GameMainScene::Draw() const
{
		// 背景画像の描画
		DrawGraph(0, mileage % 480 - 480, back_ground, TRUE);
		DrawGraph(0, mileage % 480, back_ground, TRUE);

		// 敵の描画
		for (int i = 0; i < 10; i++)
		{
			if (enemy[i] != nullptr)
			{
				enemy[i]->Draw();
			}
		}

		// プレイヤーの描画
		player->Draw();

		// UIの描画
		DrawBox(500, 0, 640, 480, GetColor(0, 153, 0), TRUE);
		SetFontSize(16);
		DrawFormatString(510, 20, GetColor(0, 0, 0), "ハイスコア");
		DrawFormatString(560, 40, GetColor(255, 255, 255), "%08d", high_score);
		DrawFormatString(510, 80, GetColor(0, 0, 0), "避けた数");
		for (int i = 0; i < 3; i++)
		{
			DrawRotaGraph(523 + (i * 50), 120, 0.3, 0, enemy_image[i], TRUE,
FALSE);
			DrawFormatString(510 + (i * 50), 140, GetColor(255,255,255), "%03d",
enemy_count[i]);
		}
		DrawFormatString(510, 200, GetColor(0, 0, 0), "走行距離");
```

```
	DrawFormatString(555, 220, GetColor(255, 255, 255), "%08d", mileage / 10);
	DrawFormatString(510, 240, GetColor(0, 0, 0), "スピード");
	DrawFormatString(555, 260, GetColor(255, 255, 255), "%08.1f",
player->GetSpeed());

	// バリア枚数の描画
	for (int i = 0; i < player->GetBarriarCount(); i++) {
		DrawRotaGraph(520 + i * 25, 340, 0.2f, 0, barrier_image, TRUE, FALSE);
	}

	// 燃料ゲージの描画
	float fx = 510.0f;
	float fy = 390.0f;
	DrawFormatStringF(fx, fy, GetColor(0, 0, 0), "FUEL METER");
	DrawBoxAA(fx, fy + 20.0f, fx + (player->GetFuel() * 100 / 20000), fy +
40.0f, GetColor(0, 102, 204), TRUE);
	DrawBoxAA(fx, fy + 20.0f, fx + 100.0f, fy + 40.0f, GetColor(0, 0, 0),
FALSE);

	// 体力ゲージの描画
	fx = 510.0f;
	fy = 430.0f;
	DrawFormatStringF(fx, fy, GetColor(0, 0, 0), "PLAYER HP");
	DrawBoxAA(fx, fy + 20.0f, fx + (player->GetHp() * 100 / 1000), fy + 40.0f,
GetColor(255, 0, 0), TRUE);
	DrawBoxAA(fx, fy + 20.0f, fx + 100.0f, fy + 40.0f, GetColor(0, 0, 0),
FALSE);
}

// 終了時処理
void GameMainScene::Finalize()
{
	// スコアを計算する
	int score = (mileage / 10 * 10);
	for (int i = 0; i < 3; i++)
	{
		score += (i + 1) * 50 * enemy_count[i];
	}
```

```
// リザルトデータの書き込み
FILE* fp = nullptr;
// ファイルオープン
errno_t result = fopen_s(&fp, "Resource/dat/result_data.csv", "w");

// エラーチェック
if (result != 0)
{
	throw ("Resource/dat/result_data.csvが開けません\n");
}

// スコアを保存
fprintf(fp, "%d,\n", score);

// 避けた数と得点を保存
for (int i = 0; i < 3; i++)
{
	fprintf(fp, "%d,\n", enemy_count[i]);
}

// ファイルクローズ
fclose(fp);

// 動的確保したオブジェクトを削除する
player->Finalize();
delete player;

for (int i = 0; i < 10; i++)
{
	if (enemy[i] != nullptr)
	{
		enemy[i]->Finalize();
		delete enemy[i];
		enemy[i] = nullptr;
	}
}

delete[] enemy;
```

```
}

// 現在のシーン情報を取得
eSceneType GameMainScene::GetNowScene() const
{
	return eSceneType::E_MAIN;
}

// ハイスコアの読み込み
void GameMainScene::ReadHighScore()
{
	RankingData data;
	data.Initialize();

	high_score = data.GetScore(0);

	data.Finalize();
}

// 当たり判定処理（プレイヤーと敵）
bool GameMainScene::IsHitCheck(Player* p, Enemy* e)
{
	// プレイヤーがバリアを貼っていたら、当たり判定を無視する
	if (p->IsBarrier())
	{
		return false;
	}

	// 敵情報が無ければ、当たり判定を無視する
	if (e == nullptr)
	{
		return false;
	}

	// 位置情報の差分を取得
	Vector2D diff_location = p->GetLocation() - e->GetLocation();

	// 当たり判定サイズの大きさを取得
	Vector2D box_ex = p->GetBoxSize() + e->GetBoxSize();
```

```
	// コリジョンデータより位置情報の差分が小さいなら、ヒット判定とする
	return ((fabsf(diff_location.x) < box_ex.x)&&(fabsf(diff_location.y) < box_ex.y));
}
```