```
#pragma once
#include <vector>
#include <string>
#include <stack>
#include <map>
#include "GameCommon.h"
#include "Vector2.h"
#include "SceneBase.h"
class SceneManager;
class Stage;
class Unit;
class Box;
class Storage;
class Fader;
class TimeLimit;

class GameScene : public SceneBase
{

public:

    // 最大ステージ数
    static constexpr int MAX_STAGE_NO = 5;

    // ステージクリアの表示時間
    static constexpr float TIME_CLEAR_MESSAGE = 3.0f;


    // 状態
    enum class STATE
    {
        GAME,
        CLEAR,
        CHANGE_STAGE
    };

    // 巻き戻し機能用構造体
    struct History
    {
        DIR dir;
        Vector2 unitPos;
        Box* box;
        Vector2 boxPos;
    };

    GameScene(SceneManager* manager);
    void Init(void) override;
```

```
    void Update(void) override;
    void UpdateGame(void);
    void UpdateClear(void);
    void UpdateChangeStage(void);

    void Draw(void) override;
    void DrawGame(void);
    void DrawClear(void);
    void DrawChangeStage(void);

    void DrawScore(void);

    void Release(void) override;

    Stage* GetStage(void);

    // 荷物との衝突判定
    Box* GetCollsionBox(Vector2 pos);

    // 荷物置き場との衝突判定
    Storage* GetCollsionStorage(Vector2 pos);

    // 演出用のフェードクラス
    Fader* mFader;

    // ステージ構成ファイルパスを取得する
    std::string GetCsvPathGround(int stageNo);

    // ギミック構成ファイルパスを取得する
    std::string GetCsvPathGimmick(int stageNo);

    // Bestスコアファイルパスを取得する
    std::string GetCsvPathScore(void);

    // 操作履歴に登録
    void RegistHistory(DIR dir, Vector2 pos, Box* box);

    // 移動歩数をプラス
    void PlusCntMove(void);

    // 移動歩数をマイナス
    void MinusCntMove(void);

private:

    Stage* mStage;
    Unit* mUnit;
```

```
    TimeLimit* mTimeLimit;

    // ステージクリア画像
    int mImageClear;

    // 荷物
    std::vector<Box*> mBoxes;

    // 荷物置き場
    std::vector<Storage*> mStorages;

    // ステージ番号
    int mStageNo;

    // 状態管理
    STATE mState;

    // 巻き戻し機能
    std::stack<History> mHistoryBack;

    // ステージクリアの演出時間
    float mStepClear;

    // 移動歩数
    int mCntMove;

    // 現在のステージのBestスコア
    int mBestScore;

    // ステージごとのBestスコア
    std::map<int, int> mBestScores;

    // ステージ遷移
    void ChangeStage(void);

    // コードでステージ設定
    void SetStage(void);

    // 外部ファイルを使用してステージ設定
    void LoadGimmickData(void);

    // 外部ファイルを使用してBestスコアを読み取る
    void LoadScore(void);

    // 外部ファイルを使用してBestスコアを書き込む
    void SaveScore(void);

    // 現在のステージのBestスコアを取得
```

```
    int GetBestScore(void);

    // 状態遷移
    void ChangeState(STATE state);

};
```