後期課題１

３日で分かるC++言語　課題

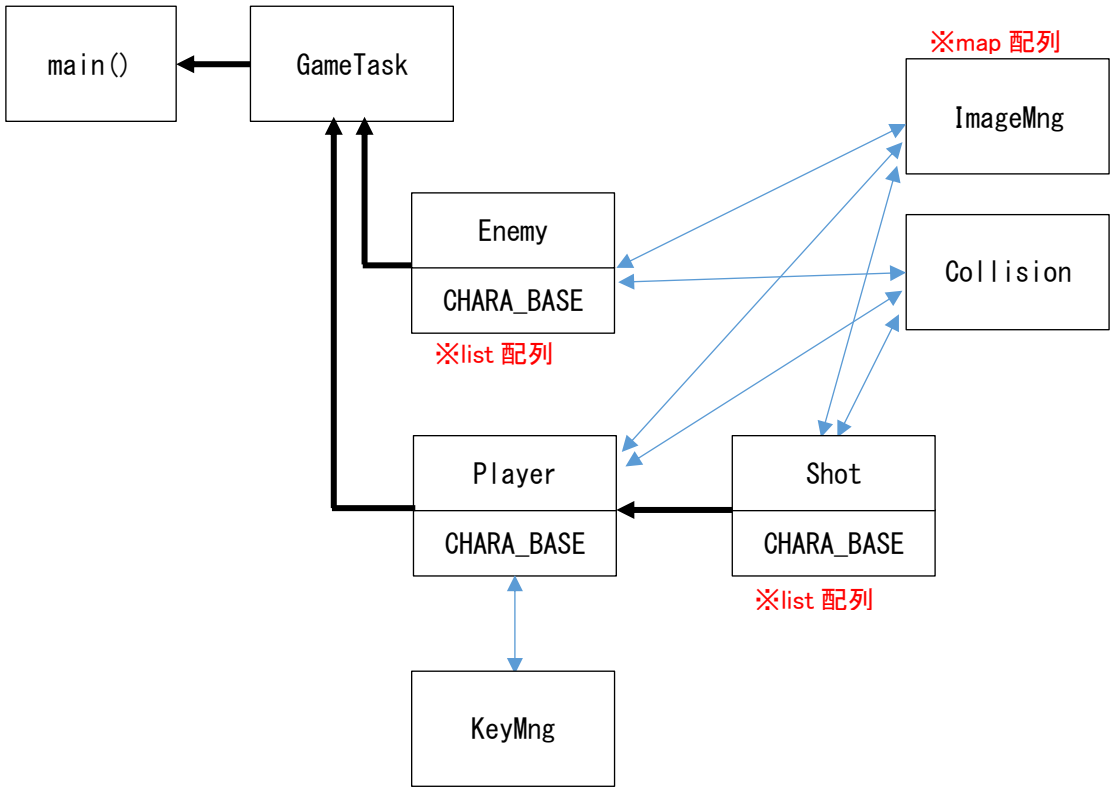
■仕様

１）ファイル及びクラス構成は、下記の構成を基本とする。

・main.cpp【エントリーポイント。ゲームタスククラスのインスタンスを実行】
・ゲームタスククラス【シングルトン】
・イメージマネージャークラス【シングルトン】
・キー入力チェッククラス【シングルトン】
・キャラクター基底クラス
・プレイヤークラス【キャラクタ基底クラスを継承】
・敵クラス【キャラクタ基底クラスを継承】
・ショットクラス【キャラクタ基底クラスを継承】
・コリジョンクラス

２）クラス構成

クラス図に従い、各クラスの関係性を維持する様にします。



■実装する内容

1）スペースキーなどのトリガ押下によって画面遷移ができている事。
　又、どの画面にいるか分かる様に表記をする事（文字列で OK）。
　Init → Title → GameMain → Over → Clear → Init ・・・

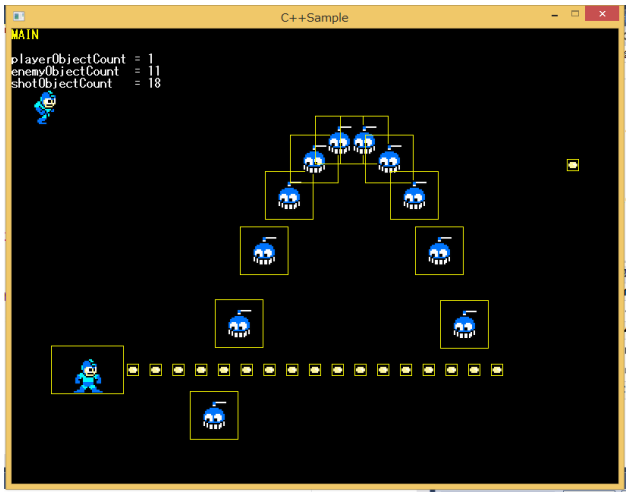
2）GameMain ではプレイヤー、ショット、敵のオブジェクトが登場する。
　プレイヤー・・・キー入力で左右などの移動ができる。
　　特定のキーでショットが発射できる。

　ショット・・・・・プレイヤーの左右の向きに合わせて発射される。
　　画面の外に出ると消える。
　　敵に当たると消える。
　　連射する事ができる。 ※配列か list 配列を使用して実装

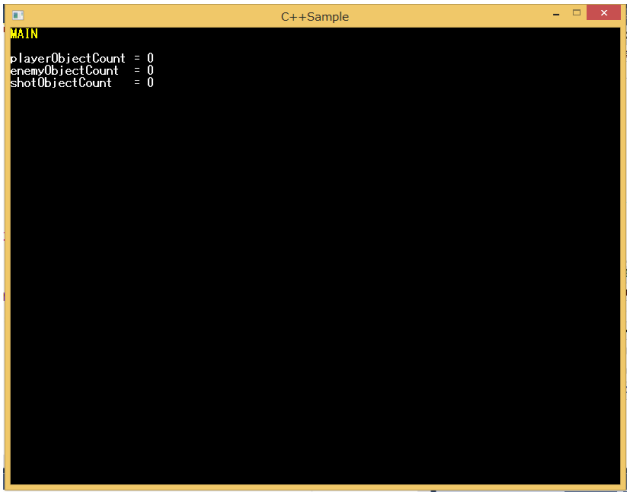
　敵・・・・・・・・・動きに制限はないが、画面の外に消えて行くような移動を行う。
　　画面の外に出ると消える。
　　ショットに当たると消える。
　　一定時間で複数の敵が出現するようにする。 ※配列か list 配列を使用して実装

　その他・・・・・・特定のキーを押すと、プレイヤーとプレイヤーショットが一斉に消える。
　　再度、特定のキーを押すと、プレイヤーが出現してショットが撃てるようになる。
　　特定のキーを押すと敵が一斉に消える（プレイヤーと同様のキーで可）。
　　再度、特定のキーを押すと、敵が出現するようになる（プレイヤーと同様で可）。

　目標・・・・・・・メモリーリークが起きない様に、確実なメモリ解放ができている。



GameMain 実行中



特定のキーを押してクリアした瞬間

■メモリーリークチェックの仕込み

メモリーリークチェックとは、確保したメモリを確実にクリアしているかどうかを確認する事です。
オブジェクトを new してそのまま終了すると、そのオブジェクト分のメモリがどこかに確保されたままになって使用できるメモリの領域を圧迫していきます。

確保したメモリは責任を持ってクリアしなければなりません。この課題では、手動でメモリの解放を行う事でどのようにプログラムを動かせばいいかを学んでいきます。

1)どのように確認するのか？

実際には「new したものは必ず delete する」を徹底する事で実現できますので、new した数と delete した数をカウントして 0 になっていればできている事が確認できます。

カウント確認の仕込み

```
class GameTask
{
public:
    int playerObjectCnt;    // ﾌﾟﾚｲﾔｰｶｳﾝﾄ
    int shotObjectCnt;      // ｼｮｯﾄｶｳﾝﾄ
    int enemyObjectCnt;     // 敵ｶｳﾝﾄ

    void addPlayerObjectCnt();     // ﾌﾟﾚｲﾔｰｶｳﾝﾄ追加
    void removePlayerObjectCnt();  // ﾌﾟﾚｲﾔｰｶｳﾝﾄ削除
    void addEnemyObjectCnt();      // 敵ｶｳﾝﾄ追加
    void removeEnemyObjectCnt();   // 敵ｶｳﾝﾄ削除
    void addShotObjectCnt();       // ｼｮｯﾄｶｳﾝﾄ追加
    void removeShotObjectCnt();    // ｼｮｯﾄｶｳﾝﾄ削除
};
```

```
void GameTask::addPlayerObjectCnt()
{
    playerObjectCnt++;
}
void GameTask::removePlayerObjectCnt()
{
    playerObjectCnt--;
}
※敵、ｼｮｯﾄは同様
```

状態の表示(どの画面遷移でも描画を行う)

```
int GameTask::Update()
{
    // 省略

    DrawFormatString(0, 32, 0xffffff, "playerObjectCount = %d", playerObjectCnt);
    DrawFormatString(0, 48, 0xffffff, "enemyObjectCount  = %d", enemyObjectCnt);
    DrawFormatString(0, 64, 0xffffff, "shotObjectCount   = %d", shotObjectCnt);

    return 0;
}
```

2）実際の作業

コンストラクタ　→　カウント追加を実行

例）プレイヤー

```
Player::Player()
{
    Init();
    GameTask::GetInstance().addPlayerObjectCnt();  // ﾌﾟﾚｲﾔｰｶｳﾝﾄ+1
}
```

デストラクタ　→　カウント削除を実行

```
Player::~Player()
{
    GameTask::GetInstance().removePlayerObjectCnt();  // ﾌﾟﾚｲﾔｰｶｳﾝﾄ+1
}
```

■提出先

¥¥stfs¥APC_ABCC クリエイティブ¥gakuseigame¥2 年生 classC¥課題 1_9 月提出

期限：平成 30 年 9 月 28 日(金) 18:00

提出方法：氏名のフォルダを作成し、その中にプロジェクトフォルダを入れておく。

※「.vs」「Debug」フォルダは削除しておくこと。