ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Департамент прикладной математики

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ 1
по дисциплине ««Компьютерный практикум»

Работу выполнила
студентка группы БПМ 173 _____ М.В. Самоделкина
дата, подпись

Работу проверил _____ С.А. Булгаков
дата, подпись

Москва 2019

Содержание

Постановка задачи

Разработать набор классов, позволяющих реализовать игру *Twin Bee*, согласно варианту 6. Априори известно, что пространство игры двумерное. Необходимо реализовать класс, позволяющий записывать и считывать информацию из файла формата INI. Необходимо реализовать минимальную логику игры, позволяющую выполнить проверку данных считанных из INI файла и выполнить начальную конфигурацию объектов. Разработать минимум два уровня для игры. Задействовать минимум два паттерна проектирования.

1 Основная часть

1.1 Общая идея решения задачи

Для решения задачи были созданы классы:

1. *Game* - класс игры, управляющий всеми объектами игры
2. *Initialization* - класс инициализации, выполняющий начальную конфигурацию объектов
3. Класс *GameItem* - класс, являющийся базовым для всех персонажей игры
4. Классы *Enemy*, *Bee*, *Cloud* и *FlyingObj*, которые являются производными от класса GameItem

К классам *Game* и *Initialization* применен паттерн проектирования *Singleton*, который гарантирует наличие единственного экземпляра класса. Реализован паттерн *Abstract Factory*, который позволит работать с разными видами *Enemy*, не завися от конкретно типа *Enemy*.

1.2 Структура и принципы действия

Класс *Game* - класс одиночка, который позволяет использовать единственный экземпляр игры, доступный всем частям программы. Для его реализации использован приватный конструктор умолчания, приватный деструктор, конструкторы копирования и присваивания также приватные, что делает их недоступными. С помощью статического метода *Instance* можно получить единственный экземпляр этого класса. Класс включает в себя поля *height*, *width*, *score*, *play*, *level*, асессоры и геттеры для этих полей. Класс содержит в себе класс абстрактной фабрики *Abstract Factory*, он используется в методе *Draw* для отображения врагов разного типа. Метод *Move* будет отвечать за передвижение всех элементов игры.

Чисто виртуальный класс *GameItem* содежит общие для всех элементов игры поля: *x, y* (координаты объекта), *play* (определяет существование объекта). В классе есть конструктор умолчания, чисто виртуальный деструктор и чисто виртуальные функции *Move*, *Draw*. Функция *isIn* проверят, находится ли элемент внутри области игры.

Класс-потомок *Bee* - описывает главного персонажа игры, содержит конструктор умолчания, в котором инициализируется стартовыми координатами, и деструктор.

Класс-наследник *Cloud* - представляет реализацию облака, летающего по полю. Класс содержит поле speed, конструктор умолчания, в котором инциализируется speed, деструктор.

Производный класс *FlyingObj* отображает шарик, которым будет стрелять пчела, и колокольчик, выпадающий из облака. Класс включает в себя поле *speed*, которое инициизируется в конструкторе с параметрами. Этот конструктор принимает координаты объекта, из которого он вылетает, и инициализирует ими свои координаты.

Подкласс *Enemy* - описывает общий тип врагов, содержит конструктор умолчания, в котором инициализируется стартовыми координатами, и деструктор. От этого класса наследуются *RedEnemy* и *BlueEnemy* - разные виды врагов. Они содержат поля *score* и *speed*, методы *Move* (изменяет координаты врага) и *Draw* (выводит в консоль координаты появления врага).

Паттерн Абстрактная фабрика позволяет объявить интерфейсы конкретных продуктов (*RedEnemy* и *BlueEnemy*), относящихся к абстрактному продукту *Enemy*. Класс *EnemyFactory* объявляет чисто виртуальный метод создания абстрактного продукта *Enemy CreateEnemy*. Конкретные фабрики *RedEnemyFactory* и *BlueEnemyFactory* реализуют метод абстрактной фабрики *EnemyFactory CreateEnemy*, который создает экземпляр класса *RedEnemy* или *BlueEnemy* и возвращает ссылку на класс *Enemy*. Класс *Client* позволяет обращаться к произвольным конкретным продуктам через абстрактные классы. Этот класс содержит метод *Draw*, который вызывает метод *Draw* для конкретного врага .

Классы *Game*, *Enemy*, *Bee*, *Cloud* и *FlyingObj* инициализируются с помощь класса *Initialization*, выполняющего начальную конфигурацию объектов. При каждой инициализации внутри класса производится проверка считанных данных. Это класс-одиночка, для реализации которого использован приватный конструктор умолчания, приватный деструктор, конструкторы копирования и присваивания также приватные, что делает их недоступными. С помощью статического метода *Instance* можно получить единственный экземпляр этого класса. Класс включает в себя поле *path*, которое хранит путь до файла с настройками, словарь *map*, который хранит в себе значения переменных из определенной в секции *ini* файла. В классе есть метод *Sett*, с помощью которого можно получить значение нужной переменной, метод *Save*, позволяющий записать необходимую переменную в *ini* файл, метод *Init* нужен для

записи значений в словарь и использует вспомогательную функцию *Parse*, которая из текста *ini* файла отделяет переменную в секции от ее значения.

Также были разработаны 2 уровня игры: уровни будут отличаться скоростью движения врагов и их количеством.

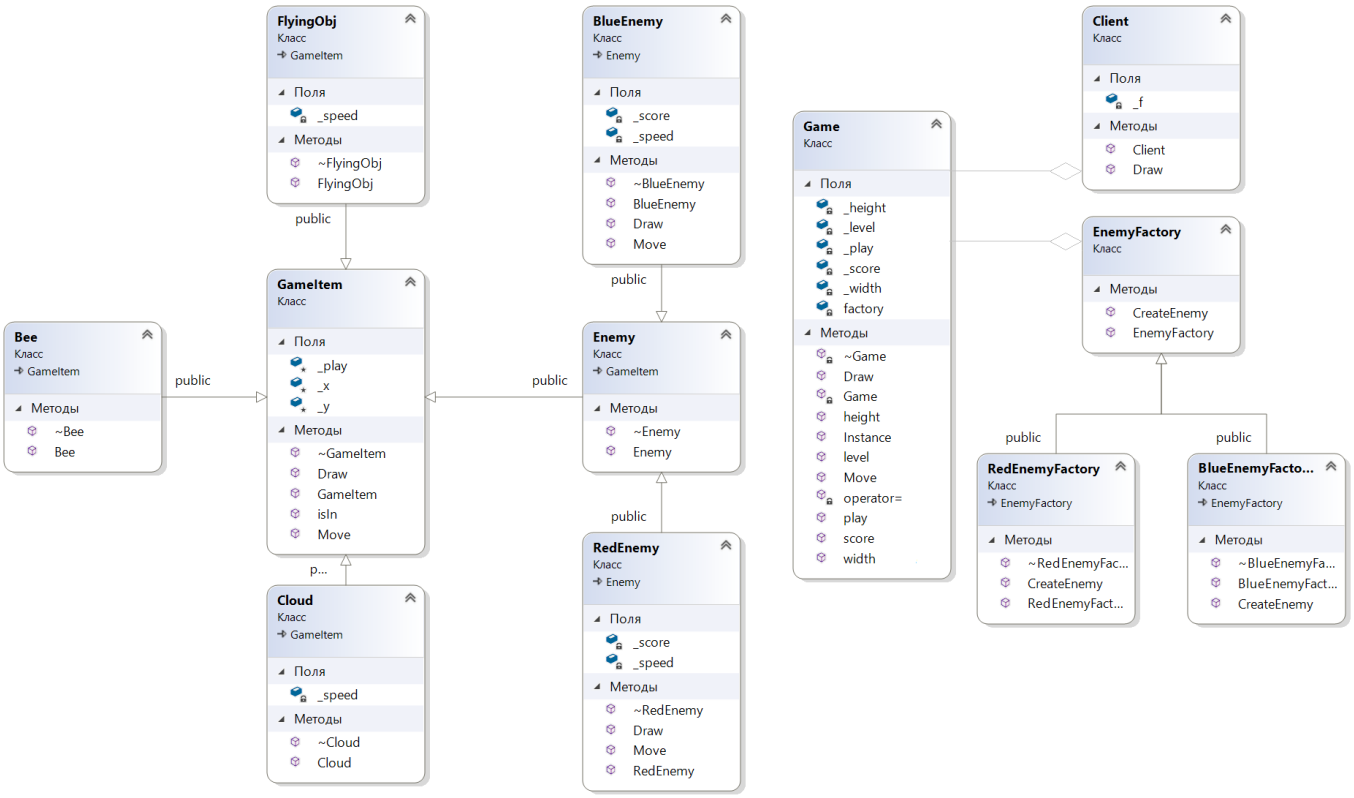

Рис. 1: Диаграмма классов

1.3 Процедура получения исполняемых программных модулей

Программный код был скомпилирован с среде *Qt Creator*. Компиляция раздельная: исходный код программы разделён на несколько файлов. Никаких дополнительных ключей не добавлялось, использовались ключи, которые добавляются по умолчанию. Параметры сборки: компилятор C++ MinGW 4.8.2, профиль Qt: Qt 4.8.7, отладчик GDB.

1.4 Результаты тестирования

Тестирование программы представлено в файле *"main.cpp"* в функции *Main()*. Ожидаемый вывод функции:

RedEnemy appeared (450,0)

900700

INI файл:

```
;here comes the game config
[setgame]
width=900
height=700
score=100
level=2

;see the score for game items
[setscore]
RedEnemy=100
BlueEnemy=200
Cloud=20

[setspeed]
RedEnemy=10
BlueEnemy=20
FlyingObj=10

[setcoord]
bee_x=450
bee_y=350
enemy_x=450
enemy_y=0

[logs]
no logs=1
```

Приложение А

полный код программы

A.1 - Game.h

```
#pragma once
#include "Bee.h"
#include "Enemy.h"
#include "Cloud.h"
#include "initialization.h"
class Game
{
    Game(){
        Initialization& ini = Initialization::Instance();
        _level = ini.Sett("setgame/level", 1);
        if (_level < 0){
            _level = 1;
        }
        _play = true;
        _width = ini.Sett("setgame/width", 900);
        if (_width < 0){
            _width = 900;
        }
        _height = ini.Sett("setgame/height", 700);
        if (_width < 0){
            _width = 700;
        }
        _score = ini.Sett("setgame/score");
        if (_score < 0){
            _score = 0;
        }
    }
    ~Game(){}
        Game(Game const&) = delete;
        Game& operator= (Game const&) = delete;
```

```
        EnemyFactory *factory;
    int _width, _height;
    int _score;
    bool _play;
    int _level;
public:
        static Game& Instance()
    {
                static Game g;
                return g;
    }
    void Draw()
    {
        factory = new RedEnemyFactory;
        Client *c = new Client(factory);
        c->Draw();
        delete factory;
        delete c;
    }
    void Move(){}
        int width() const { return _width; }
    int height() const { return _height; }
    int score() const { return _score; }
        bool play() const { return _play; }
    int level() const { return _level; }
    void height(int height) { _height = height; }
    void width(int width) { _width = width; }
    void score(int score) { _score = score; }
    void play(bool play) { _play = play; }
    void level(int level) { _level = level; }
};
```

A.2 - GameItem.h

```
#pragma once
#include <iostream>
```

```
using namespace std;
class GameItem
{
protected:
        int _x, _y;
        bool _play;
public:
    GameItem(){
        _play = true;
    }
    virtual ~GameItem() = 0;
        virtual void Draw() = 0;
        virtual void Move() = 0;
    bool isIn();
};
```

A.3 - Bee.h

```
#pragma once
#include "GameItem.h"
#include "initialization.h"

class Bee :
        public GameItem
{
public:
    Bee(){
        Initialization& ini = Initialization::Instance();
        _x = ini.Sett("setcoord/bee_x");
        if (_x < 0){
            _x = 0;
        }
        _y = ini.Sett("setcoord/bee_y");
        if (_y < 0){
            _y = 0;
```

```
        }
    }
    ~Bee(){}
};
```

A.4 - Cloud.h

```
#pragma once
#include "GameItem.h"
#include "initialization.h"

class Cloud :
        public GameItem
{
    int _speed;
public:
    Cloud(){
        Initialization& ini = Initialization::Instance();
        _speed = ini.Sett("setspeed/Cloud");
        if (_speed < 0){
            _speed = 0;
        }
    }
    ~Cloud(){}
};
```

A.5 - FlyingObj.h

```
#pragma once
#include "GameItem.h"
#include "initialization.h"

class FlyingObj :
    public GameItem
{
    int _speed;
public:
```

```
    FlyingObj(int x, int y){
        Initialization& ini = Initialization::Instance();
        _speed = ini.Sett("setspeed/FlyingObj");
        if (_speed < 0){
            _speed = 0;
        }
        _x = x;
        _y = y;
    }
    ~FlyingObj(){}
};
```

A.6 - Enemy.h

```
#pragma once
#include "GameItem.h"
#include "initialization.h"

//Abstract base product
class Enemy :
        public GameItem
{
public:
    Enemy() {
        Initialization& ini = Initialization::Instance();
        _x = ini.Sett("setcoord/enemy_x");
        if (_x < 0){
            _x = 0;
        }
        _y = ini.Sett("setcoord/enemy_y");
        if (_y < 0){
            _y = 0;
        }
    }
    ~Enemy(){}
};
```

```
//Concrete product type Red
class RedEnemy :
        public Enemy
{
    int _score;
    int _speed;
public:
    RedEnemy(){
        Initialization& ini = Initialization::Instance();
        _score = ini.Sett("setscore/RedEnemy");
        if (_score < 0){
            _score = 0;
        }
        _speed = ini.Sett("setspeed/RedEnemy");
        if (_speed < 0){
            _speed = 0;
        }
    }
    void Draw() { cout << "RedEnemy␣appeared␣"
                          "(" <<_x<<"," <<_y<<")"<< endl; }
    void Move() {
        _x += _speed;
        _y += _speed;
    }

    ~RedEnemy() {}
};
//Concrete product type Blue
class BlueEnemy :
        public Enemy
{
    int _score;
    int _speed;
public:
```

```
    BlueEnemy(){
        Initialization& ini = Initialization::Instance();
        _score = ini.Sett("setscore/BlueEnemy");
        if (_score < 0){
            _score = 0;
        }
        _speed = ini.Sett("setspeed/BlueEnemy");
        if (_speed < 0){
            _speed = 0;
        }
    }
    void Draw() { cout << "BlueEnemy␣appeared␣"
                          "(" <<_x<<"," <<_y<<")"<< endl; }
    void Move() {
        _x -= _speed;
        _y += _speed;
    }
    ~BlueEnemy(){}
};
//Abstract factory
class EnemyFactory {
public:
    EnemyFactory(){}
    virtual Enemy* CreateEnemy() = 0;
    //virtual ~EnemyFactory() = 0;
};
//Conctete factory type Red
class RedEnemyFactory :
        public EnemyFactory {
public:
    RedEnemyFactory() {}
    Enemy* CreateEnemy() override{
        return new RedEnemy;
    }
```

```
    ~RedEnemyFactory() {}
};
//Conctete factory type Blue
class BlueEnemyFactory :
        public EnemyFactory {
public:
    BlueEnemyFactory() {}
    Enemy* CreateEnemy() override {
                return new BlueEnemy;
        }
    ~BlueEnemyFactory() {}
};
//Working through abstract interface
class Client {
        EnemyFactory *_f;
public:
        Client(EnemyFactory *f): _f(f) {}
        void Draw()
        {
                Enemy *e = _f->CreateEnemy();
                e->Draw();
        }
};
```

A.6 - Initialization.h

```
#pragma once
#include <QCoreApplication>
#include <QSettings>
#include <QString>
#include <QTextStream>
#include <QMap>
#include <QFile>
using namespace std;

class Initialization
```

```
{
    QString _path;
    QMap<QString, QString> map;
    Initialization(){}
    ~Initialization(){}
public:
    static Initialization& Instance()
    {
        static Initialization i;
        return i;
    }
    int Sett(QString key, const int & val = 0){
        if (map.find(key) == map.end()){
            return  val;
        }
        return map.value(key).toInt();
    }
    void Save(const QString &section,
              const QString &variable, int value){
        QSettings sett(_path, QSettings::IniFormat);
        QFile file(_path);
        if ((file.open(QIODevice::ReadWrite)))
        {
            QString line;
            QString comment;
            bool flag = false;
            QTextStream stream( &file );
            do {
                line = file.readLine();
                if (line.contains(section, Qt::CaseSensitive)
                        && !line.isEmpty() && line[0] != ';')
                {
                    stream << line;
                    line = file.readLine();
```

```
                while(!line.isEmpty() && line != "\n"){
                    if (line.contains(variable,
                                      Qt::CaseSensitive)
                            && !line.isEmpty()
                            && line[0] != ';')
                    {
                        flag = true;
                        line = variable + "="
                                + QString::number(value)
                                + '\n';
                    }
                    stream << line;
                    line = file.readLine();
                }
                if (!flag){
                    flag = true;
                    line = variable + "="
                            + QString::number(value)
                            + "\n\n";
                }
            }
            stream << line;
        } while (!line.isEmpty());
        if (!flag){
            stream << "\n\n[" + section + "]" + '\n';
            stream << variable + "="
                      + QString::number(value) + "\n";
        }
        file.resize(0);
        file.close();
    }
}
QString Parse(QString &str, QString &sec){
    //ignore comments
```

```
if (str[0] == ';')
    return "";
//init section
if (str[0] == '['){
    int i = 1;
    sec="";
    while (str[i] != ']'){
        sec.append(str[i]);
        i++;
    }
    sec.append('/');
    return sec;
}
//init variable
int i = 0;
QString key="";
while(str[i] != 0 && str[i] != '='){
    if (str[i] == '␣')
        i++;
    else{
        key.append(str[i]);
        i++;
    }
}
QString val="";
i++;
while(str[i] != 0 && str[i] != ';'){
    if (str[i] == '␣')
        i++;
    else{
        val.append(str[i]);
        i++;
    }
}
```

```
        if(val.length())
            map.insert(sec + key, val);
        return sec;
    }
    void Init(QString p)
    {
        _path = QCoreApplication:: applicationDirPath();
        _path.append("/" + p);
        QSettings sett(_path, QSettings::IniFormat);
        QFile file(_path);
        if ((file.open(QIODevice::ReadOnly)))
        {
            QString str, sec;
            while(!file.atEnd())
            {
                str = file.readLine();
                sec = Parse(str, sec);
            }
            file.close();
        }
    }
};
```

A.7 - GameItem.cpp

```
#pragma once
#include "Game.h"
#include "GameItem.h"

bool GameItem::isIn(){
    Game& game = Game::Instance();
    return (0 < _x && _x < game.width()
            && 0 <_y && _y <game.height() );
}

GameItem::~GameItem()
```

```
{
}
```

A.8 - main.cpp

```
#include <QCoreApplication>
#include <iostream>
#include <QSettings>
#include <QVariant>
#include <QString>
#include "Game.h"
#include "Initialization.h"

using namespace std;


int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Initialization& ini = Initialization::Instance();
    ini.Init("settings.ini");
    Game& game = Game::Instance();
    game.Draw();
    cout << game.width() << game.height();
    ini.Save("setgame", "level", 2);
    ini.Save("setgame", "score", 100);
    ini.Save("logs", "no logs", 1);
    return a.exec();
}
```