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

Департамент прикладной математики

ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ 2
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 и реализовать набор классов с использованием библиотеки *Qt 4.8*, позволяющей выполнить отрисовку объектов из лабораторной работы 1 согласно варианту 6 (*TwinBee*). Для отрисовки использовать средства класса *QPainter*, объекты представить в виде разноцветных плоских фигур. Применить паттерн посетитель.

1 Основная часть

1.1 Общая идея решения задачи

Проект был разработан с главным окном *MainWindow* на основе базового класса *QMainWindow*. Для отрискови объектов был использован класс *QPainer*, методы *drawRect* и *drawEllipse*, то есть все объекты были представлены в виде разноцветных овалов или прямоугольников. Был применен паттерн посетитель: поведение (функции *Draw* и *Move* классов *Bee*, *Cloud*, *FlyingObj*, *RedEnemy* и *BlueEnemy*), было реализовано в отдельном классе *Visitor*.

1.2 Структура и принципы действия

Класс *MainWindow*, который является наследником базового класса *QMainWindow*, содержит макрос *Q_OBJECT*. Для проверки корректной работы программы был реализован слот *keyReleaseEvent*, который при нажатии на кнопки *лево* и *право* перемещает объект пчелы по экрану. В классе *MainWindow* переопределена функция базового класса *paintEvent*, которая вызывает функцию *Draw* в классе *Game*.

Для удобства реализации функции передвижения и добавления в классы новых однотипных методов был применен паттерн посетитель. Класс *Visitor* содержит методы *visit* для каждого класса, отличающиеся типом входного параметра, чтобы запустить метод для конкретного класса. Класс *DrawGameItems* является посетителем, реализующим конкретное поведение классов при передвижении. В реализованном ранее родительском классе *GameItem* описывается чисто виртуальный метод доступа посетителя *access* с типом входного параметра *Visitor*. В каждом конкретном классе (*Bee*, *Cloud*, *FlyingObj*, *RedEnemy* и *BlueEnemy*) переопределяются методы доступа посетителя *access*: вызывается функция *visit* для данного класса. Также для проверки корректной работы паттерна был добавлен еще один класс конкретного посетителя *MoveGameItems*, который должен будет описывать перемещение объектов.

В классе *DrawGameItems* содержится закрытое поле *QMainWindow*, которое определяет событие изменения формы, и в котором переопределены функции *visit* для всех классов. Общий принцип работы функций одинаковый для всех классов. В функции создается экземпляр класса *QPainter*, который будет рисовать в *MainWindow*, настраиваются цвет и стиль кисти, рисуется геометрическая фигура с помощью функций *drawEllipse* и *drawRect*, и сохраняется событие.

Функция *Draw* в классе *Game* создает экземпляр класса *DrawGameItems* и для каждого объекта игры вызывает функцию *access* с созданным ранее посетителем.

Функция *setLevel* производит создание экземпляра класса *Bee*, заполнение вектора врагов *enemes* экземплярами классов *RedEnemy* или *BlueEnemy* (в зависимости от созданной конкретной фабрики *EnemyFactory*), заполнение вектора *items* экземплярами классов *Cloud* и *FlyingObj*.

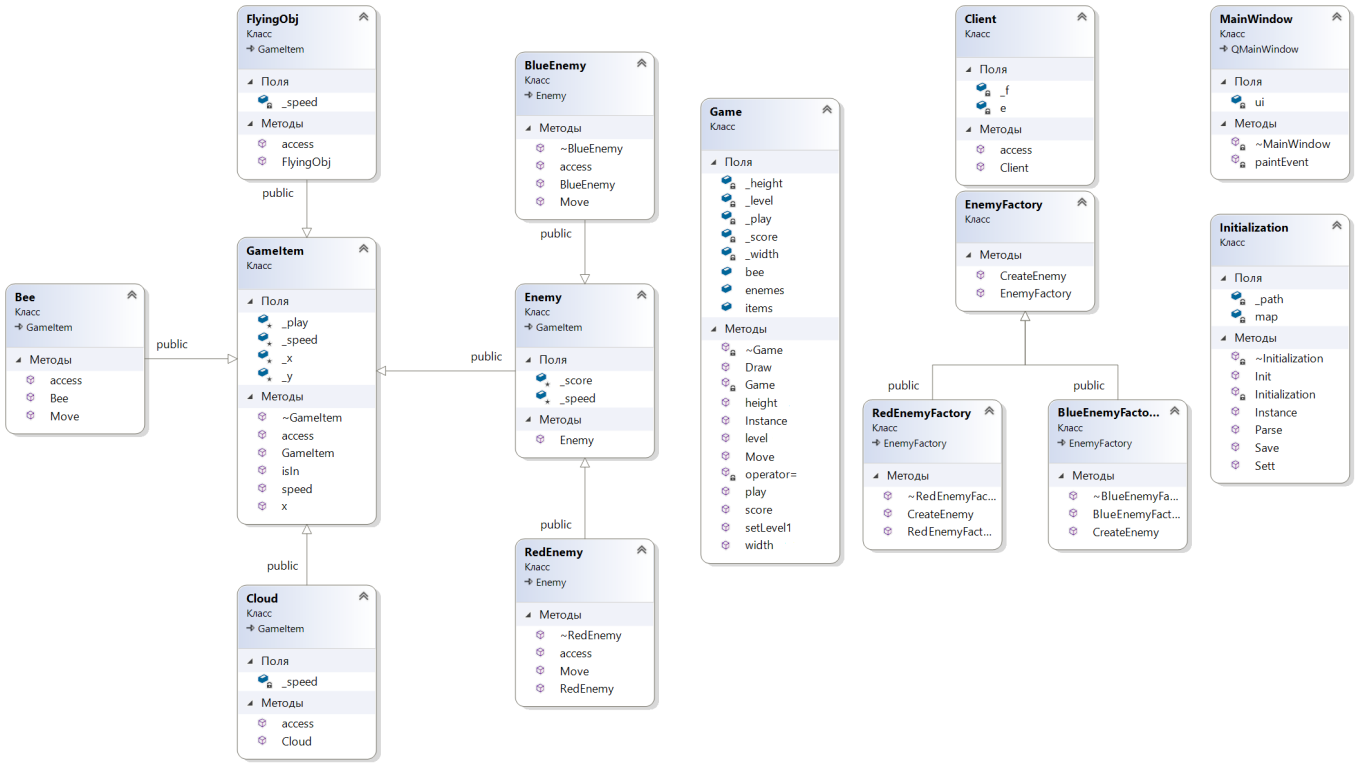

Рис. 1: Диграма классов

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()*. Ожидаемая отрисовка объектов в окне *MainWindow*:

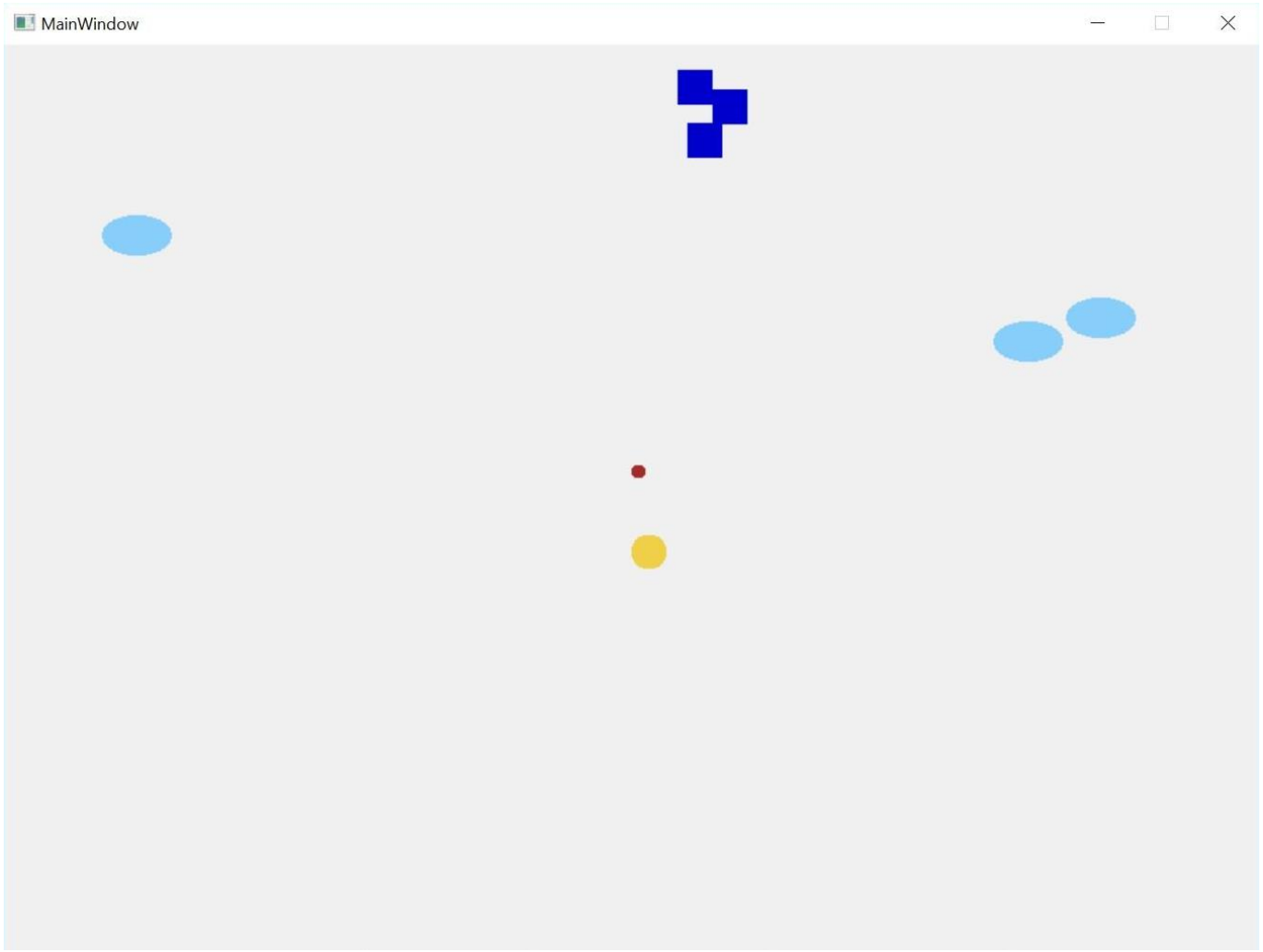

Рис. 2: Результат тестирования

Приложение А

полный код программы

A.1 - Game.h

```
1  #pragma once
2  #include "Bee.h"
3  #include "Enemy.h"
4  #include "Cloud.h"
5  #include "flyingobj.h"
6  #include "initialization.h"
7
8
9  class Game
10 {
11   Game(){
12     Initialization& ini = Initialization::Instance();
13     _level = ini.Sett("setgame/level", 1);
14     if (_level < 0){
15       _level = 1;
16     }
17     _play = true;
18     _width = ini.Sett("setgame/width", 900);
19     if (_width < 0){
20       _width = 900;
21     }
22     _height = ini.Sett("setgame/height", 700);
23     if (_width < 0){
24       _width = 700;
25     }
26     _score = ini.Sett("setgame/score");
27     if (_score < 0){
28       _score = 0;
29     }
30     setLevel1();
```

```
31    }
32    ~Game() {
33      delete bee;
34      for (auto it = items.begin(); it != items.end(); it++)
          {
35        delete *it;
36        items.erase(it);
37        it--;
38      }
39      for (auto it = enemes.begin(); it != enemes.end(); it
          ++) {
40        delete *it;
41        enemes.erase(it);
42        it--;
43      }
44    }
45    Game(Game const&) = delete;
46    Game& operator= (Game const&) = delete;
47    int _width, _height;
48    int _score;
49    bool _play;
50    int _level;
51  public:
52    Bee *bee;
53    QVector<GameItem*> items;
54    QVector<Client*> enemes;
55    void setLevel1(){
56      bee = new Bee();
57      EnemyFactory *factory = new BlueEnemyFactory;
58      for (int i = 0; i < 3; i++) {
59        Client* enemy = static_cast<Client*>(new Client(
      factory));
60        enemes.push_back(enemy);
61      }
```

```
62      for (int i = 0; i < 3; i++) {
63        GameItem* item = static_cast<GameItem*>(new Cloud);
64        items.push_back(item);
65      }
66      GameItem* item = static_cast<GameItem*>(new FlyingObj(
    bee->x(), bee->y() - 50));
67      items.push_back(item);
68      delete factory;
69    }
70    static Game& Instance()
71    {
72      static Game g;
73      return g;
74    }
75    void Draw(QMainWindow *e)
76    {
77      DrawGameItems visitor(e);
78      for(auto c: enemes){
79        c->access(visitor);
80      }
81      for(auto c: items){
82        c->access(visitor);
83      }
84      bee->access(visitor);
85    }
86    void Move(){}
87    int width() const { return _width; }
88    int height() const { return _height; }
89    int score() const { return _score; }
90    bool play() const { return _play; }
91    int level() const { return _level; }
92    void height(int height) { _height = height; }
93    void width(int width) { _width = width; }
94    void score(int score) { _score = score; }
```

```
95    void play(bool play) { _play = play; }
96    void level(int level) { _level = level; }
97 };
```

A.2 - GameItem.h

```
 1 #pragma once
 2 #include <iostream>
 3 #include <QPainter>
 4 #include <QMainWindow>
 5
 6 class Bee;
 7 class Cloud;
 8 class FlyingObj;
 9 class RedEnemy;
10 class BlueEnemy;
11
12 using namespace std;
13
14 class Visitor{
15 public:
16    virtual void visit(Bee &b) = 0;
17    virtual void visit(Cloud &c) = 0;
18    virtual void visit(FlyingObj &f) = 0;
19    virtual void visit(RedEnemy &r) = 0;
20    virtual void visit(BlueEnemy &b) = 0;
21    virtual ~Visitor() = default;
22 };
23
24 //element
25 class GameItem
26 {
27 protected:
28    int _x, _y, _speed;
29    bool _play;
30 public:
```

```
31    GameItem(){
32      _play = true;
33      _x = 0;
34      _y = 0;
35      _speed = 0;
36    }
37    virtual void access(Visitor &v) = 0;
38    virtual ~GameItem() = 0;
39    //virtual void Move() = 0;
40    bool isIn();
41    int x(){ return _x;}
42    int y(){ return _y;}
43    int speed() {return _speed;}
44    void x( int x ){ _x = x;}
45    void y( int y ){ _y = y;}
46  };
47
48  //concrete visitor
49  class DrawGameItems:
50      public Visitor{
51  private:
52    QMainWindow *e;
53  public:
54    DrawGameItems(QMainWindow *event): e(event){}
55    void visit(Bee &b)override ;
56    void visit(Cloud &c) override;
57    void visit(FlyingObj &f) override;
58    void visit(RedEnemy &r) override;
59    void visit(BlueEnemy &b) override;
60  };
61
62  //concrete visitor
63  class MoveGameItems:
64      public Visitor{
```

```
65 public:
66   void visit(Bee &b)override;
67   void visit(Cloud &c) override{}
68   void visit(FlyingObj &f) override{}
69   void visit(RedEnemy &r) override{}
70   void visit(BlueEnemy &b) override{}
71 };
```

A.3 - Bee.h

```
 1 #pragma once
 2 #include "GameItem.h"
 3 #include "initialization.h"
 4
 5 class Bee :
 6   public GameItem
 7 {
 8 public:
 9   Bee(){
10     Initialization& ini = Initialization::Instance();
11     _x = ini.Sett("setcoord/bee_x");
12     if (_x < 0){
13       _x = 0;
14     }
15     _y = ini.Sett("setcoord/bee_y");
16     if (_y < 0){
17       _y = 0;
18     }
19     _speed = ini.Sett("setspeed/Bee");
20     if (_speed < 0){
21       _speed = 0;
22     }
23   }
24   void access(Visitor &v) override {
25     v.visit(*this);
26    }
```

```
27    void Move(int x){
28      _x += x;
29    }
30  };
```

A.4 - Cloud.h

```
 1  #pragma once
 2  #include ”GameItem.h”
 3  #include ”initialization.h”
 4
 5  class Cloud :
 6    public GameItem
 7  {
 8    int _speed;
 9  public:
10    Cloud(){
11      Initialization& ini = Initialization::Instance();
12      _speed = ini.Sett(”setspeed/Cloud”);
13      if (_speed < 0){
14        _speed = 0;
15      }
16      int x = ini.Sett(”setgame/width”, 900);
17      if (x < 0){
18        x = 900;
19      }
20      x -= 50;
21      int y = ini.Sett(”setgame/height”, 700);
22      if (y < 0){
23        y = 700;
24      }
25      y /= 2;
26       _x = rand()%x;
27       _y = rand()%y;
28    }
29    void access(Visitor &v) override {
```

```
30      v.visit(*this);
31     }
32 };
```

A.5 - FlyingObj.h

```
 1 #pragma once
 2 #include "GameItem.h"
 3 #include "initialization.h"
 4
 5 //class visitor
 6 class FlyingObj :
 7   public GameItem
 8 {
 9   int _speed;
10 public:
11   FlyingObj(int x, int y){
12     Initialization& ini = Initialization::Instance();
13     _speed = ini.Sett("setspeed/FlyingObj");
14     if (_speed < 0){
15       _speed = 0;
16     }
17     _x = x;
18     _y = y;
19   }
20   void access(Visitor &v) override {
21     v.visit(*this);
22   }
23 };
```

A.6 - Enemy.h

```
 1 #pragma once
 2 #include "GameItem.h"
 3 #include "initialization.h"
 4
 5 //Abstract base product
```

```
6  class Enemy :
7    public GameItem
8  {
9  protected:
10   int _score;
11   int _speed;
12 public:
13   Enemy() {
14     Initialization& ini = Initialization::Instance();
15     _x = ini.Sett(”setcoord/enemy_x”) + + rand()%70;
16     if (_x < 0){
17       _x = 0;
18     }
19     _y = ini.Sett(”setcoord/enemy_y”) + rand()%70;
20     if (_y < 0){
21       _y = 0;
22     }
23     _score = 0;
24     _speed = 0;
25   }
26 };
27 //Concrete product type Red
28 class RedEnemy :
29   public Enemy
30 {
31 public:
32   RedEnemy(){
33     Initialization& ini = Initialization::Instance();
34     _score = ini.Sett(”setscore/RedEnemy”);
35     if (_score < 0){
36       _score = 0;
37     }
38     _speed = ini.Sett(”setspeed/RedEnemy”);
39     if (_speed < 0){
```

```
40        _speed = 0;
41      }
42    }
43    void access(Visitor &v) override {
44      v.visit(*this);
45     }
46    void Move() {
47      _x += _speed;
48      _y += _speed;
49    }
50    ~RedEnemy() {}
51  };
52  //Concrete product type Blue
53  class BlueEnemy :
54    public Enemy
55  {
56  public:
57    BlueEnemy(){
58      Initialization& ini = Initialization::Instance();
59      _score = ini.Sett("setscore/BlueEnemy");
60      if (_score < 0){
61        _score = 0;
62      }
63      _speed = ini.Sett("setspeed/BlueEnemy");
64      if (_speed < 0){
65        _speed = 0;
66      }
67    }
68    void access(Visitor &v) override {
69      v.visit(*this);
70     }
71    void Move() {
72      _x -= _speed;
73      _y += _speed;
```

```
74    }
75    ~BlueEnemy(){}
76  };
77  //Abstract factory
78  class EnemyFactory {
79  public:
80    EnemyFactory(){}
81    virtual Enemy* CreateEnemy() = 0;
82    //virtual ~EnemyFactory() = 0;
83  };
84  //Conctete factory type Red
85  class RedEnemyFactory :
86    public EnemyFactory {
87  public:
88    RedEnemyFactory() {}
89    Enemy* CreateEnemy() override{
90      return new RedEnemy;
91    }
92    ~RedEnemyFactory() {}
93  };
94  //Conctete factory type Blue
95  class BlueEnemyFactory :
96    public EnemyFactory {
97  public:
98    BlueEnemyFactory() {}
99    Enemy* CreateEnemy() override {
100     return new BlueEnemy;
101   }
102   ~BlueEnemyFactory() {}
103 };
104 //Working through abstract interface
105 class Client {
106   EnemyFactory *_f;
107   Enemy *e;
```

```
public:
  Client(EnemyFactory *f): _f(f) {
    e = _f->CreateEnemy();
  }
  void access(Visitor &v)
  {
    e->access(v);
  }
};
```

A.7 - 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;
```

```
25      }
26      return map.value(key).toInt();
27    }
28    void Save(const QString &section,
29          const QString &variable, int value){
30      QSettings sett(_path, QSettings::IniFormat);
31      QFile file(_path);
32      if ((file.open(QIODevice::ReadWrite)))
33      {
34        QString line;
35        QString comment;
36        bool flag = false;
37        QTextStream stream( &file );
38        do {
39          line = file.readLine();
40          if (line.contains(section, Qt::CaseSensitive) && !
    line.isEmpty() && line[0] != ';')
41          {
42            stream << line;
43            line = file.readLine();
44            while(!line.isEmpty() && line != "\n" && line[0]
    != ';'){
45              if (line.contains(variable, Qt::CaseSensitive)
    && !line.isEmpty() && line[0] != ';')
46              {
47                flag = true;
48                line = variable + "=" + QString::number(value
    ) + '\n';
49              }
50              stream << line;
51              line = file.readLine();
52            }
53            if (!flag){
54              flag = true;
```

```
55              line = variable + "=" + QString::number(value)
    + "\n\n";
56            }
57          }
58          stream << line;
59        } while (!line.isEmpty());
60        if (!flag){
61          stream << "\n\n[" + section + "]" + '\n';
62          stream << variable + "=" + QString::number(value) +
     "\n";
63        }
64        file.resize(0);
65        file.close();
66      }
67    }
68    QString Parse(QString &str, QString &sec){
69      //ignore comments
70      if (str[0] == ';')
71        return "";
72      //init section
73      if (str[0] == '['){
74        int i = 1;
75        sec="";
76        while (str[i] != ']'){
77          sec.append(str[i]);
78          i++;
79        }
80        sec.append('/');
81        return sec;
82      }
83      //init variable
84      int i = 0;
85      QString key="";
86      while(str[i] != 0 && str[i] != '='){
```

```
87        if (str[i] == '␣')
88          i++;
89        else{
90          key.append(str[i]);
91          i++;
92        }
93      }
94      QString val="";
95      i++;
96      while(str[i] != 0 && str[i] != ';'){
97        if (str[i] == '␣')
98          i++;
99        else{
100         val.append(str[i]);
101         i++;
102       }
103     }
104     if(val.length())
105       map.insert(sec + key, val);
106     return sec;
107   }
108   void Init(QString p)
109   {
110     _path = QCoreApplication:: applicationDirPath();
111     _path.append("/" + p);
112     QSettings sett(_path, QSettings::IniFormat);
113     QFile file(_path);
114     if ((file.open(QIODevice::ReadOnly)))
115     {
116       QString str, sec;
117       while(!file.atEnd())
118       {
119         str = file.readLine();
120         sec = Parse(str, sec);
```

```
121         }
122         file.close();
123       }
124     }
125 };
```

A.8 - MainWindow.h

```
 1 #pragma once
 2 #include <QMainWindow>
 3 #include <QKeyEvent>
 4
 5 namespace Ui {
 6 class MainWindow;
 7 }
 8
 9 class MainWindow : public QMainWindow
10 {
11   Q_OBJECT
12
13 public:
14   explicit MainWindow(QWidget *parent = nullptr);
15   ~MainWindow();
16   void paintEvent(QPaintEvent *event);
17 private:
18   Ui::MainWindow *ui;
19 public slots:
20   void keyReleaseEvent(QKeyEvent*);
21 };
```

A.9 - GameItem.cpp

```
 1 #pragma once
 2 #include "Game.h"
 3
 4 bool GameItem::isIn(){
 5   Game& game = Game::Instance();
```

```
6     return (0 < _x && _x < game.width() && 0 <_y && _y <game.
      height() );
7  }
8
9  GameItem::~GameItem(){}
10
11 void DrawGameItems::visit(Bee &b){
12    QPainter painter(e);
13    QColor yellow("#f0d048");
14    Qt::BrushStyle style = Qt::SolidPattern;
15    QBrush brush(yellow, style);
16    painter.setBrush(brush);
17    painter.setPen(Qt::NoPen);
18    painter.drawEllipse(b.x(),b.y(), 25,25);
19    cout << b.x() << endl;
20    painter.save();
21 }
22
23 void DrawGameItems::visit(Cloud &c){
24    QPainter painter(e);
25    QColor gray("#87CEFA");
26    Qt::BrushStyle style = Qt::SolidPattern;
27    QBrush brush(gray, style);
28    painter.setBrush(brush);
29    painter.setPen(Qt::NoPen);
30    painter.drawEllipse(c.x(),c.y(), 50,30);
31    painter.save();
32 }
33
34 void DrawGameItems::visit(FlyingObj &f){
35    QPainter painter(e);
36    QColor gray("#A52A2A");
37    Qt::BrushStyle style = Qt::SolidPattern;
38    QBrush brush(gray, style);
```

```
39    painter.setBrush(brush);
40    painter.setPen(Qt::NoPen);
41    painter.drawEllipse(f.x(),f.y(), 10,10);
42    painter.save();
43  }
44
45  void MoveGameItems::visit(Bee &b) {
46    b.x(b.x() + b.speed());
47  }
48
49  void DrawGameItems::visit(RedEnemy &r){
50    QPainter painter(e);
51    QColor red(”#ff00ff”);
52    Qt::BrushStyle style = Qt::SolidPattern;
53    QBrush brush(red, style);
54    painter.setBrush(brush);
55    painter.setPen(Qt::NoPen);
56    painter.drawRect(r.x(),r.y(), 25,25);
57    painter.save();
58  }
59
60  void DrawGameItems::visit(BlueEnemy &b){
61    QPainter painter(e);
62    QColor blue(”#0000CD”);
63    Qt::BrushStyle style = Qt::SolidPattern;
64    QBrush brush(blue, style);
65    painter.setBrush(brush);
66    painter.setPen(Qt::NoPen);
67    painter.drawRect(b.x(),b.y(), 25,25);
68    painter.save();
69  }
```

A.10 - main.cpp

```
1  #include <QCoreApplication>
2  #include <iostream>
```

```
3  #include <QSettings>
4  #include <QVariant>
5  #include <QString>
6  #include "Game.h"
7  #include "mainwindow.h"
8  #include <QApplication>
9  #include "Initialization.h"
10 #include <ctime>
11
12 class QPaintEvent;
13 using namespace std;
14
15 int main(int argc, char *argv[])
16 {
17   QApplication a(argc, argv);
18   srand((unsigned int)time(NULL));//helps to generate
     random coords of objects
19   Initialization& ini = Initialization::Instance();
20   ini.Init("settings.ini");
21   Game& game = Game::Instance();
22   MainWindow w;
23   w.show();
24   ini.Save("setgame", "level", 1);
25   ini.Save("setgame", "score", 100);
26   ini.Save("logs", "no␣logs", 1);
27   return a.exec();
28 }
```

A.11 - MainWindow.cpp

```
1  #include "mainwindow.h"
2  #include "ui_mainwindow.h"
3  #include "Game.h"
4
5  MainWindow::MainWindow(QWidget *parent) :
6    QMainWindow(parent),
```

```
7    ui(new Ui::MainWindow)
8  {
9    ui->setupUi(this);
10   Game& game = Game::Instance();
11   this->setFixedSize(game.width(),game.height());
12 }
13
14 MainWindow::~MainWindow()
15 {
16   delete ui;
17 }
18
19 void MainWindow::paintEvent(QPaintEvent *event)
20 {
21   Game& game = Game::Instance();
22   game.Draw(this);
23 }
24
25 void MainWindow::keyReleaseEvent(QKeyEvent * event)
26 {
27   if( event->key() == Qt::Key_Left )
28   {
29     Game& game = Game::Instance();
30     //MoveGameItems visitor;
31     //game.bee->access(visitor);
32     game.bee->Move(-5);
33     update();
34   }
35   if( event->key() == Qt::Key_Right )
36   {
37     Game& game = Game::Instance();
38     game.bee->Move(5);
39     update();
40   }
```

```
}
```

A.12 - Settings.ini

```
;here comes the game config
[setgame]
width=900
height=700
score=100
level=1

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

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

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

[logs]
no logs=1
```