МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2

по дисциплине «ООП»

ТЕМА: ДОБАВЛЕНИЕ ИГРОКА И ЭЛЕМЕНТОВ ПОЛЯ

Студент гр. 9304 Прокофьев М.Д.

Преподаватель Размочаева Н.В.

Санкт-Петербург
2020

Цель работы.

Обучение работе с классами, конструкторами на языке в C++.

Задание.

Создан класс игрока, которым управляет пользователь. Объект класса игрока может перемещаться по полю, а также взаимодействовать с элементами поля. Для элементов поля должен быть создан общий интерфейс и должны быть реализованы 3 разных класса элементов, которые по разному взаимодействуют с игроком. Для взаимодействия игрока с элементом должен использоваться перегруженный оператор (Например, оператор +). Элементы поля могут добавлять очки игроку/замедлять передвижения/и.т.д.

Обязательные требования:

-Реализован класс игрока

-Реализованы три класса элементов поля

-Объект класса игрока появляется на клетке со входом

-Уровень считается пройденным, когда объект класса игрока оказывается на клетке с выходом (и при определенных условиях: например, набрано необходимое кол-во очков)

-Взаимодействие с элементами происходит через общий интерфейс

-Взаимодействие игрока с элементами происходит через перегруженный оператор

Дополнительные требования:

-Для создания элементов используется паттерн Фабричный метод/Абстрактная фабрика

-Реализовано динамическое изменение взаимодействия игрока с элементами через паттерн Стратегия. Например, при взаимодействии с определенным количеством элементов, игрок не может больше с ними взаимодействовать

Основные теоретические положения.

Класс - это пользовательский тип данных.

Конструктор копирования используется для инициализации класса путем создания копии необходимого объекта.

Оператор присваивания копированием (или «копирующее присваивание») используется для копирования одного класса в другой (существующий) класс.

Конструкторы перемещения принимают ссылку на значение объекта класса и используются для реализации передачи владения ресурсами объекта параметра.

Синглтон — порождающий шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр некоторого класса, и предоставляющий глобальную точку доступа к этому экземпляру.

Метод в объектно-ориентированном программировании — это функция или процедура, принадлежащая какому-то классу или объекту. Как и процедура в процедурном программировании, метод состоит из некоторого количества операторов для выполнения какого-то действия и имеет набор входных аргументов.

Интерфейс — программная/синтаксическая структура, определяющая отношение между объектами, которые разделяют определённое множество и не связаны никак иначе. При проектировании классов, разработка интерфейса тождественна разработке спецификации (множества методов, которые должен реализовывать каждый класс, использующий интерфейс).

Перегрузка операторов позволяет определить действия, которые будет выполнять оператор. Перегрузка подразумевает создание функции, название которой содержит слово operator и символ перегружаемого оператора. Функция оператора может быть определена как член класса, либо вне класса.

Фабричный метод — порождающий шаблон проектирования, предоставляющий подклассам интерфейс для создания экземпляров некоторого класса.

Выполнение работы.

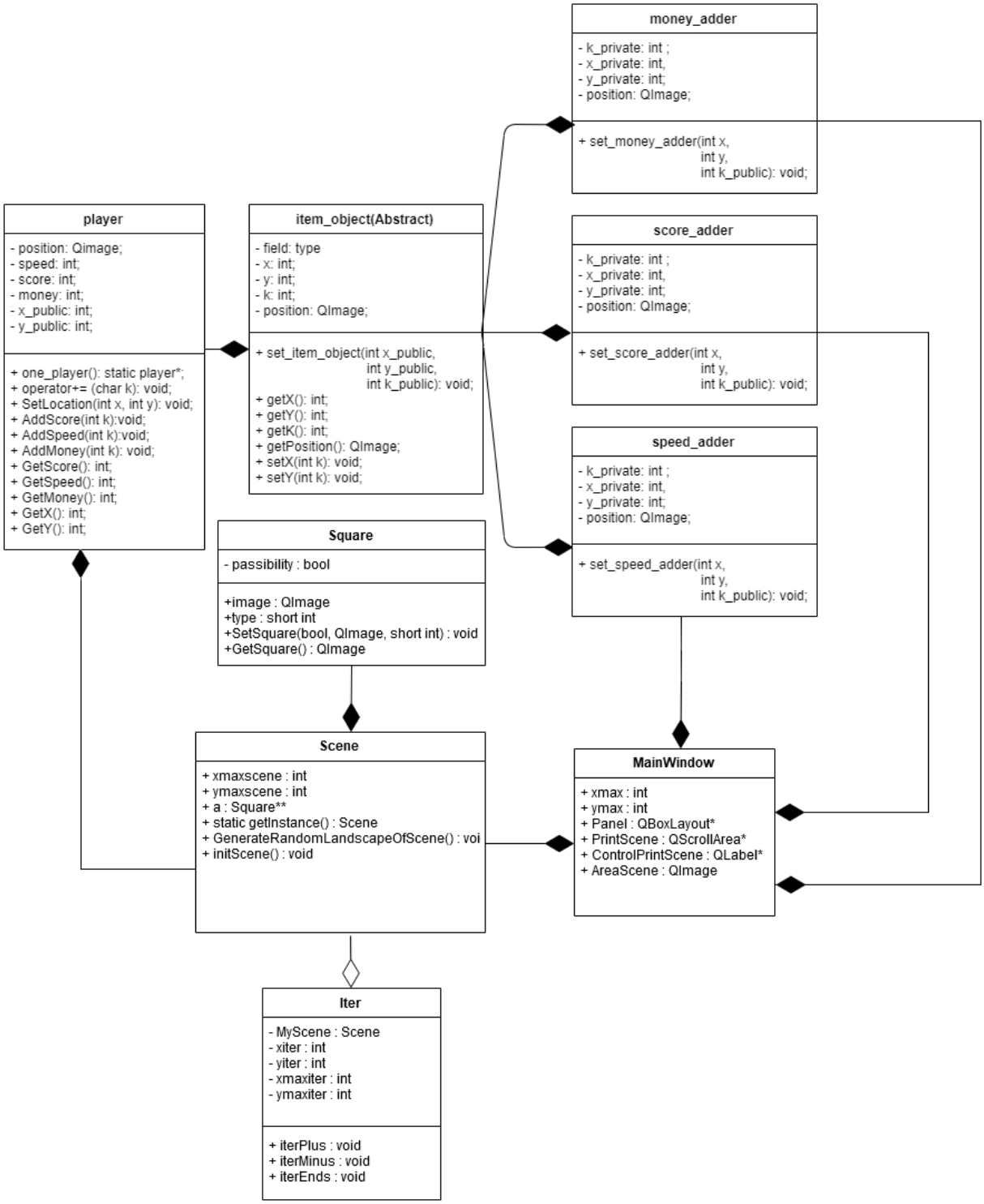

Рисунок 1 – UML-схема

Был создан класс player, который отвечает за игрока. Этот класс был создан синглтоном, так как в игре может быть один игрок. В этом классе играют роль 6 параметров: координаты X(x_public) и Y(y_public) игрока, количество очков(score), количество монет(money), скорость(speed) и изображение игрока(position). Создано 3 метода, с помощью которых можно добавлять какое то значение одного из параметров к текущему(для параметров score, money, speed). Кроме того создана функция SetLocation() позволяющая перемещать объект игрока. При перемещении игрока, проверяется в какую точку он встал (что в этой точке лежит) и с помощью перегруженного оператора += у игрока увеличивается значение одного из параметров, в зависимости от того, какой предмет лежит в клетке, куда он переместился.

Взаимодействие предметов происходит через общий интерфейс.

Для предметов был создан абстрактный класс, именуемый item_object. Этот класс, подобно игроку имеет параметры координат: X(x), Y(y), изображение(position), а также параметр значения увеличения конкретного параметра(k). Функция set_item_object() используется для изменения параметров какого либо предмета.

Также создано три класса для самих предметов: score_adder(очки), money_adder(монеты), speed_adder(скорость), которые являются наследниками класса item_object. Каждый из классов содержит функцию set(т.е. set_score_adder, set_money_adder и set_speed_adder). В этой функции у трех классов вызывается функция set_item_object.

В MainWindow реализация движения сделана через функцию keyPressEvent(). При нажатии одной из клавиш(W,A,S,D), происходит движение игрока. Если игрок дошел до финиша, и при этом он собрал 1 очко, он выигрывает, о чем приходит оповещение

Тестирование.

1) Запуск программы:

Рисунок 2 – Запуск

2) Игрок дошел до финиша:

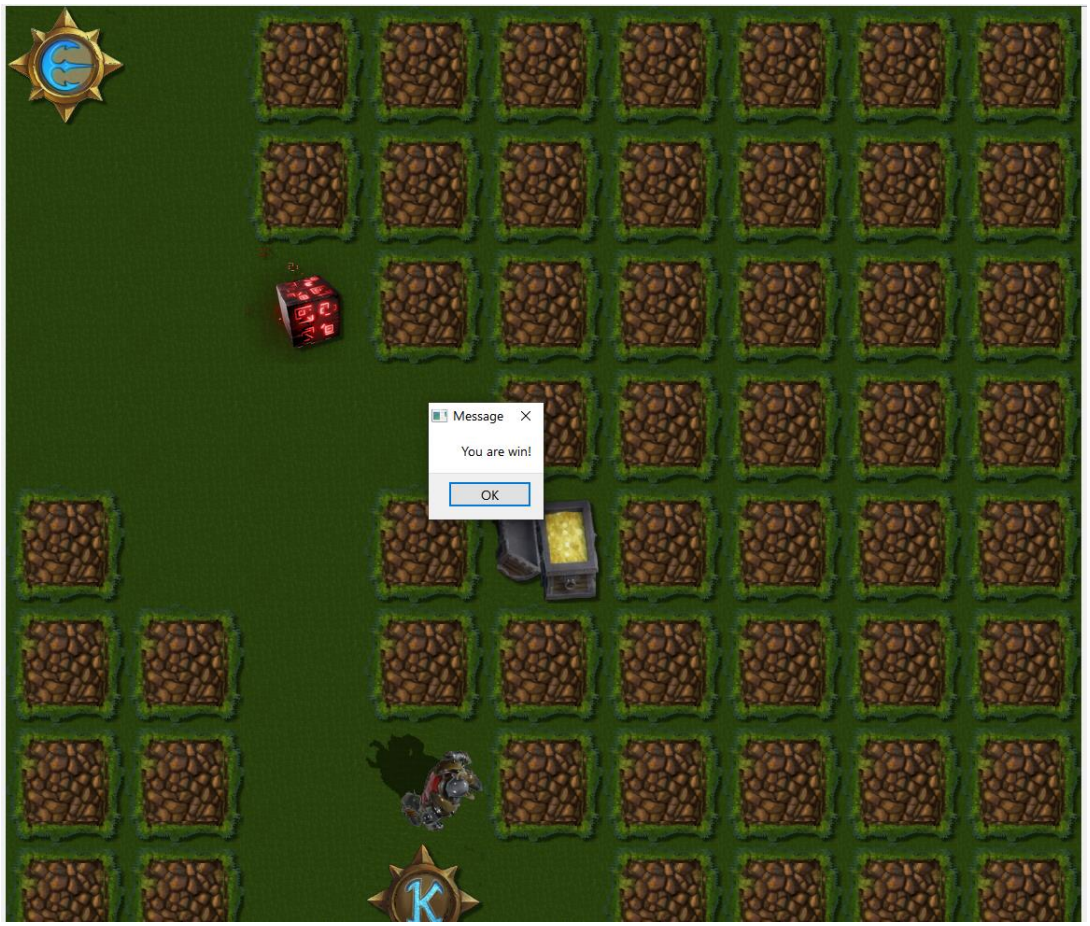

Рисунок 3 – Игрок на финише

Выводы

Получены знания о классах, создан игрок, реализовано для него управление, создан перегруженный оператор для взаимодействия с предметами. создан интерфейс для предметов, где абстрактным классом является предмет, а наследниками являются сами типы предметов(очки, монеты и скорость).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include "mainwindow.h"

#include <QApplication>



int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Название файла: libs.h

```
#ifndef LIBS_H
#define LIBS_H

#include <QMainWindow>
#include <QAction>
#include <QBoxLayout>
#include <QColor>
#include <QDesktopWidget>
#include <QDialog>
#include <QEvent>
#include <QHBoxLayout>
#include <QLabel>
```

```
#include <QLayout>
#include <QMainWindow>
#include <QMenu>
#include <QMenuBar>
#include <QMouseEvent>
#include <QMoveEvent>
#include <QPushButton>
#include <QStatusBar>
#include <QStyle>
#include <QToolBar>
#include <QWidget>
#include <QTextEdit>
#include <QLineEdit>
#include <QFileDialog>
#include <QRect>
#include <QIcon>
#include <QFile>
#include <QColor>
#include <QColorDialog>
#include <QScrollArea>
#include <QCommandLinkButton>
#include <QComboBox>
#include <QInputDialog>
#include <QSpinBox>
#include <QGroupBox>
#include <iterator>
#include <memory>

#include <cstring>
#include <fstream>
```

```
#include <iostream>
#include <math.h>

#include "QMessageBox"
#include <tchar.h>



#endif // LIBS_H
```

Название файла: Square.h

```
#ifndef POINT_H
#define POINT_H
#include "libs.h"
using namespace std;

const QImage standard("D:/MyLargeProjects/game/block01.png");
const QImage rock("D:/MyLargeProjects/game/block02.png");
const QImage entry("D:/MyLargeProjects/game/block03.png");
const QImage exite("D:/MyLargeProjects/game/block04.png");
const QImage main_player(QDir::currentPath() + "/block05.png");
const QImage scoreup(QDir::currentPath() + "/block06.png");
const QImage speedup(QDir::currentPath() + "/block07.png");
const QImage teleporter1(QDir::currentPath() + "/block08.png");
const QImage teleporter2(QDir::currentPath() + "/block09.png");
const QImage moneyup(QDir::currentPath() + "/block10.png");

class Square
{
   friend class mainwindow;
```

```
private:
    bool passibility;

public:

    QImage image;
    short int type; // 0 - свободная, 1 - стена, 2 - вход, 3 - выход
    void SetSquare(bool pass, QImage img, short int tp);
    QImage GetSquare();
    short int GetTP();
    bool GetPass();
    Square(bool pass=false, QImage img=rock, short int tp=1);
};

#endif // POINT_H
```

Название файла: Square.cpp

```
#include "Square.h"



Square::Square(bool pass, QImage img, short int tp)
{
    this->passibility=pass;
    this->image=img;
    this->type=tp;



}
void Square::SetSquare(bool pass, QImage img, short int tp)
{
    this->passibility=pass;
```

```
    this->image=img;
    this->type=tp;

}


QImage Square::GetSquare()
{
    return(this->image);
}

short int Square::GetTP()
{
    return(this->type);
}

bool Square::GetPass()
{
    return(this->passibility);
}
```

Название файла: Scene.h

```
#ifndef SCENE_H
#define SCENE_H
#include "Square.h"
//class Square;

class Scene
{
    Scene();
```

```
    Scene(const Scene& TimeScene);
    Scene& operator=(const Scene& TimeScene);

    Scene (Scene&& TimeScene);
    Scene& operator=(Scene&& TimeScene);

public:
    int xmaxscene=5, ymaxscene=5;
    Square** a;
   static Scene* getInstance();
    ~Scene();
    void GenerateRandomLandscapeOfScene();
    void initScene();

};

#endif // SCENE_H
```

Название файла: Scene.cpp

```
#include "scene.h"

Scene::Scene()
{

}

void Scene::initScene()
{
    this->xmaxscene;
```

```
    this->ymaxscene;


    Scene::a=new Square*[ymaxscene];
    for(int x=0; x<xmaxscene; x++)
        Scene::a[x] = new Square[xmaxscene];

}

Scene::Scene(const Scene &TimeScene){
    xmaxscene = TimeScene.xmaxscene;
    ymaxscene = TimeScene.ymaxscene;
    a = new Square* [xmaxscene];
    for (int x = 0; x < xmaxscene; x++) {
        a[x] = new Square[ymaxscene];
    }
    int x=-1,y=-1;
    while(++x<xmaxscene)
        while(++y<ymaxscene)
        {
            if(TimeScene.a[x][y].type == 2) a[x][y].type=2;
            if(TimeScene.a[x][y].type == 3) a[x][y].type=3;
            a[x][y] = TimeScene.a[x][y];
        }

}

Scene& Scene::operator= (const Scene &TimeScene){
    if(this != &TimeScene) {
        Square** an = new Square *[xmaxscene];
```

```
        for (int i = 0; i < TimeScene.xmaxscene; i++) {
            an[i] = new Square[ymaxscene];
        }
        int x=-1,y=-1;
        while(++x<xmaxscene) delete a[x];
        delete a;
        a = an;
        ymaxscene = TimeScene.ymaxscene; xmaxscene = TimeScene.xmaxscene;
        x=-1;
        while(++x<xmaxscene)
            while(++y<ymaxscene)
            {
                if(TimeScene.a[x][y].type == 2) a[x][y].type=2;
                if(TimeScene.a[x][y].type == 3) a[x][y].type=3;
                a[x][y] = TimeScene.a[x][y];
            }
    }
    return *this;
}

Scene::Scene (Scene&& TimeScene){
    ymaxscene = TimeScene.ymaxscene; xmaxscene = TimeScene.xmaxscene;
    a = TimeScene.a;
    TimeScene.a = nullptr;
}

Scene& Scene::operator=(Scene&& TimeScene){
    if (&TimeScene == this)
        return *this;
    int x=-1;
```

```
    while(++x<xmaxscene) delete a[x];
    delete a;
    xmaxscene = TimeScene.xmaxscene; ymaxscene = TimeScene.ymaxscene;
    a = TimeScene.a;
    TimeScene.a = nullptr;
    return *this;
}

void Scene::GenerateRandomLandscapeOfScene()
{
    int x=0,y=0, random;
    srand( time(0) );

   for(int i=0; i<2; i++)
   {
    x=0,y=0;
    while((x>=0)&&(y>=0)&&(x<xmaxscene)&&(y<ymaxscene))
    {
        random=rand()%2+0;
        switch(random)
        {
        case 0:
           if(x<xmaxscene) { Scene::a[x][y].SetSquare(1, standard, 0); x++; }
        case 1:
           if(y<ymaxscene) { Scene::a[x][y].SetSquare(1, standard, 0); y++; }
        }
    }
    }
```

```
    Scene::a[0][0].SetSquare(1, entry, 2);
    Scene::a[x-1][y-1].SetSquare(1, exite, 3);



}



Scene* Scene::getInstance()
{
    static Scene instance;
            return &instance;

}
Scene::~Scene(){
    for(int x = 0;x<xmaxscene;x++) {
                delete a[x];
            }
            delete[] a;
}
```

Название файла: iter.h

```
#ifndef ITER_H
#define ITER_H
#include "scene.h"
```

```
class iter
{
private:
    const Scene &MyScene;
    int xiter;
    int yiter;
    int xmaxiter=Scene::getInstance()->xmaxscene;
    int ymaxiter=Scene::getInstance()->ymaxscene;

public:
    iter(const Scene &MyScene);
    void iterPlus();
    void iterMinus();
    bool iterEnds();
    const Square& gc();
    void operator++();
    void operator--();
};

#endif // ITER_H
```

Название файла: iter.cpp

```
#include "iter.h"

iter::iter(const Scene& MyScene) : MyScene(MyScene)
{
    this->xiter=0;
    this->yiter=0;
}
```

```
void iter::iterPlus()
{
   if(yiter != ymaxiter)
   {
      if(xiter==xmaxiter) { xiter=0; yiter++;}
      else xiter++;
   }
}

void iter::iterMinus()
{
   if(yiter != -1)
   {
      if(xiter==xmaxiter) { xiter=0; yiter--;}
      else xiter--;
   }
}

bool iter::iterEnds()
{
   return(yiter == ymaxiter);
}

const Square& iter::gc()
{
   return Scene::getInstance()->a[xiter][yiter];
}
void iter::operator++()
{
   iterPlus();
```

```
}

void iter::operator--()
{
   iterMinus();
}
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include "scene.h"
#include "iter.h"

class MainWindow : public QMainWindow
{
   Q_OBJECT
public:
   int xmax, ymax;
   MainWindow(QWidget *parent = nullptr);
   ~MainWindow();
   QBoxLayout *Panel;
   QScrollArea *PrintScene;
   QLabel *ControlPrintScene;
   QImage *AreaScene;
```

```
};
#endif // MAINWINDOW_H
```

Название файла: mainwindow.cpp

```
#include "mainwindow.h"

int const const1=1;
int const const2=1;
int const const3=1;

void MainWindow::printImage(int x,int y, QImage my_image)
{
   for(int i = 0; i < 128; i++){
         for(int j = 0; j < 128; j++ ){
            QRgb pixColor;
            pixColor = my_image.pixel(j, i);

if((qRed(pixColor)!=0)&&(qGreen(pixColor)!=0)&&(qBlue(pixColor)!=0))
AreaScene->setPixel( i+128*x, j+128*y, pixColor);

                              }
                        }
}

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent)
{
   bool bOk;
   xmax = QInputDialog::getInt(0,"Параметры поля","Размеры поля(x):", 2, 2,
10, 1, &bOk); if (!bOk) {}
```

```
        ymax = QInputDialog::getInt(0,"Параметры поля","Размеры поля(y):", 2, 2,
10, 1, &bOk); if (!bOk) {}
        // // // // Настройка поля
        Scene::getInstance()->xmaxscene=xmax;
        Scene::getInstance()->ymaxscene=ymax;
        Scene::getInstance()->initScene();
        Scene::getInstance()->GenerateRandomLandscapeOfScene();
        // // // //Проверка итератора, например, найдём выход
           iter MyIter(*Scene::getInstance());
           qDebug() << "Координата выхода по X:";
           qDebug() << MyIter.iterSearchSceneX(3);
           qDebug() << "Координата выхода по Y:";
           qDebug() << MyIter.iterSearchSceneY(3);
        // // // // Работа с окном
        winxmax=xmax*128;
        winymax=ymax*128;
        QWidget *cw = new QWidget(this);
        Panel = new QVBoxLayout(cw);
        PrintScene = new QScrollArea();

        ControlPrintScene = new QLabel(PrintScene);
        ControlPrintScene->setMouseTracking(true);
        ControlPrintScene->setSizePolicy(QSizePolicy::Ignored,
QSizePolicy::Ignored);

        Panel->addWidget(PrintScene);

        setMinimumSize(winxmax, winymax);
        setCentralWidget(cw);
        AreaScene = new QImage(winxmax, winymax,  QImage::Format_RGB888 );
```

```
    for(int i = 0; i < winxmax; i++){
        for(int j = 0; j < winymax; j++ ){
            AreaScene->setPixel( i, j, qRgb(0, 0, 0 ) );
            }
        }
    //speed_adder a(0,1,1);
    //a.give_it();
    player::one_player();
    MainWindow::obj1=new speed_adder[const1];
    obj1[0].set_speed_adder(2,2,1);
    MainWindow::obj2=new score_adder[const2];
    obj2[0].set_score_adder(3,3,1);
    MainWindow::obj3=new money_adder[const3];
    obj3[0].set_money_adder(4,4,1);


    // // // // Рисуем само поле
    for(int x=0; x<xmax; x++)
        for(int y=0;y<ymax;y++)
            printImage(x,y, Scene::getInstance()->a[x][y].image);

    ControlPrintScene->resize(winxmax, winymax);
    ControlPrintScene->setPixmap(QPixmap::fromImage(*AreaScene));



}
   /*
*/
```

```
void MainWindow::keyPressEvent(QKeyEvent *event) {


    int key=event->key();
    int x=player::one_player()->GetX(), move_x=0;
    int y=player::one_player()->GetY(), move_y=0;
    qDebug() << "X = " << x << " Y = " << y;
    qDebug() << "speed:";
    qDebug() << player::one_player()->GetSpeed();
    if (key==Qt::Key_W)   move_y=player::one_player()->GetSpeed()*(-1);
    if (key==Qt::Key_D)   move_x=player::one_player()->GetSpeed();
    if (key==Qt::Key_A)   move_x=player::one_player()->GetSpeed()*(-1);
    if (key==Qt::Key_S)   move_y=player::one_player()->GetSpeed();
    qDebug() << "scores:";
    qDebug() << player::one_player()->GetScore();
    qDebug() << "money:";
    qDebug() << player::one_player()->GetMoney();

    player::one_player()->SetLocation(move_x,move_y);
    printImage(x,y, Scene::getInstance()->a[x][y].image);
    printImage(player::one_player()->GetX(),y=player::one_player()->GetY(),
Scene::getInstance()->a[player::one_player()->GetX()][player::one_player()->GetY()
].image);

if((Scene::getInstance()->a[player::one_player()->GetX()][player::one_player()->Get
Y()].GetTP()==3)
            &&(player::one_player()->GetScore()==1))
        QMessageBox::about(this, "Message", "You are win!");

    ControlPrintScene->resize(winxmax, winymax);
```

```
   ControlPrintScene->setPixmap(QPixmap::fromImage(*AreaScene));
}
/**/
MainWindow::~MainWindow()
{
   free(MainWindow::obj3);
   free(MainWindow::obj2);
   free(MainWindow::obj1);
}
```

Название файла: player.h

```
#ifndef PLAYER_H
#define PLAYER_H
#include "scene.h"
class player
{
private:
   QImage position;
   int speed;
   int score;
   int money;
   int x_public;
   int y_public;
public:
   player(int x_public=0, int y_public=1, int speed_public=1, QImage position=main_player);
   static player* one_player();
   void operator+= (char k);
   void SetLocation(int x, int y);
   void AddScore(int k);
```

```
    void AddSpeed(int k);
    void AddMoney(int k);
    int GetScore();
    int GetSpeed();
    int GetMoney();
    int GetX();
    int GetY();

};

#endif // PLAYER_H
```

Название файла: player.cpp

```
#include "player.h"

player::player(int x_public, int y_public, int speed_public, QImage position)
{
    this->x_public=x_public;
    this->y_public=y_public;
    this->position=position;

    this->score=0;
    this->money=0;
    this->speed=speed_public;
    Scene::getInstance()->a[x_public][y_public].SetImage(position);
}

player* player::one_player()
{
    static player instance;
        return &instance;
```

```
}
void player::SetLocation(int x, int y)
{
   if((this->x_public+x>=0)
    &&(this->y_public+y>=0)
    &&(this->y_public+y<=Scene::getInstance()->ymaxscene)

&&(this->x_public+x<=Scene::getInstance()->xmaxscene)&&(Scene::getInstance()->a[this->x_public+x][this->y_public+y].GetSquare()!=rock))
      {
         player& my_player= *player::one_player();

if(Scene::getInstance()->a[this->x_public+x][this->y_public+y].GetSquare()==scoreup) my_player+='c';

if(Scene::getInstance()->a[this->x_public+x][this->y_public+y].GetSquare()==speedup) my_player+='s';

if(Scene::getInstance()->a[this->x_public+x][this->y_public+y].GetSquare()==moneyup) my_player+='m';


      if(Scene::getInstance()->a[this->x_public][this->y_public].GetTP()==2)
Scene::getInstance()->a[this->x_public][this->y_public].SetImage(entry);
      else if(Scene::getInstance()->a[this->x_public][this->y_public].GetTP()==3)
Scene::getInstance()->a[this->x_public][this->y_public].SetImage(exite);
      else if(Scene::getInstance()->a[this->x_public][this->y_public].GetTP()==6)
Scene::getInstance()->a[this->x_public][this->y_public].SetImage(teleporter1);
```

```
        else if(Scene::getInstance()->a[this->x_public][this->y_public].GetTP()==7)
Scene::getInstance()->a[this->x_public][this->y_public].SetImage(teleporter2);
        else
Scene::getInstance()->a[this->x_public][this->y_public].SetImage(standard);




        this->x_public+=x;
        this->y_public+=y;

Scene::getInstance()->a[this->x_public][this->y_public].SetImage(this->position);

        }

    }

    void player::operator+=(char k)
    {
        if(k=='c') player::one_player()->AddScore(1);
        if(k=='s') player::one_player()->AddSpeed(1);
        if(k=='m') player::one_player()->AddMoney(1);
    }


    void player::AddScore(int k)
    {
        this->score+=k;
    }
```

```
void player::AddMoney(int k)
{
   this->money+=k;
}

void player::AddSpeed(int k)
{
   this->speed+=k;
}

int player::GetScore()
{
   return this->score;
}

int player::GetSpeed()
{
   return this->speed;
}
int player::GetMoney()
{
   return this->money;
}

int player::GetX()
{
   return this->x_public;
}

int player::GetY()
```

```
{
    return this->y_public;
}
```

Название файла: item_object.h

```
#ifndef ITEM_OBJECT_H
#define ITEM_OBJECT_H
#include "player.h"

class item_object
{
private:
    int x,y,k;
    QImage position;
public:
    item_object(int x_public=1, int y_public=1, int k_public=0, QImage position=scoreup);
    void set_item_object(int x_public, int y_public, int k_public);
    int getX();
    int getY();
    int getK();
    QImage getPosition();
    void setX(int k);
    void setY(int k);
    ~item_object();
};

#endif // ITEM_OBJECT_H
```

Название файла: item_object.cpp

```
#include "item_object.h"
```

```
item_object::item_object(int x, int y, int k, QImage position)
{

    this->k=k;
    this->x=x;
    this->y=y;
    this->position=position;
    Scene::getInstance()->a[this->x][this->y].SetSquare(1, this->position, 4);

}


void item_object::set_item_object(int x, int y, int k)
{
    Scene::getInstance()->a[this->x][this->y].SetSquare(0, standard, 1);
    this->k=k;
    this->x=x;
    this->y=y;
    Scene::getInstance()->a[this->x][this->y].SetSquare(1, this->position, 4);

}


int item_object::getX()
{
    return this->x;
}

int item_object::getY()
```

```
{
    return this->y;
}

int item_object::getK()
{
    return this->k;
}

void item_object::setX(int k)
{
    this->x=k;
}

void item_object::setY(int k)
{
    this->y=k;
}

QImage item_object::getPosition()
{
    return this->position;
}

item_object::~item_object(){

}
```

Название файла: money_adder.h

```
#ifndef MONEY_ADDER_H
#define MONEY_ADDER_H
#include "item_object.h"

class money_adder: public item_object
{
private:
    int k_private;
    int x_private, y_private;
    QImage position;
public:
    money_adder(int x=1, int y=1, int k_public=1, QImage position=moneyup);
    void set_money_adder(int x, int y, int k_public);
};

#endif // money_ADDER_H
```

Название файла: money_adder.cpp

```
#include "money_adder.h"


money_adder::money_adder(int x, int y, int k_public, QImage position): item_object(x,y,getK(),position)
{
    this->k_private=k_public;
    this->x_private=x;
    this->y_private=y;
    this->position=position;
}
void money_adder::set_money_adder(int x,int y, int k_public)
{
```

```
    item_object::set_item_object(this->x_private=x, this->y_private=y,
this->k_private=k_public);
    }
```

Название файла: score_adder.h

```
#ifndef SCORE_ADDER_H
#define SCORE_ADDER_H
#include "item_object.h"

class score_adder: public item_object
{
private:
    int k_private;
    int x_private, y_private;
    QImage position;
public:
    score_adder(int x=1, int y=1, int k_public=1, QImage position=scoreup);
    void set_score_adder(int x, int y, int k_public);
};
#endif // SCORE_ADDER_H
```

Название файла: score_adder.cpp

```
#include "score_adder.h"


score_adder::score_adder(int x, int y, int k_public, QImage position):
item_object(x,y,getK(),position)
{
    this->k_private=k_public;
    this->x_private=x;
    this->y_private=y;
```

```
    this->position=position;
}
void score_adder::set_score_adder(int x,int y, int k_public)
{
    item_object::set_item_object(this->x_private=x, this->y_private=y, this->k_private=k_public);
}
```

Название файла: speed_adder.h

```
#ifndef SPEED_ADDER_H
#define SPEED_ADDER_H
#include "item_object.h"

class speed_adder: public item_object
{
private:
    int k_private;
    int x_private, y_private;
    QImage position;
public:
    speed_adder(int x=1, int y=1, int k_public=1, QImage position=speedup);
    void set_speed_adder(int x,int y, int k_public);
};
#endif // SPEED_ADDER_H
```

Название файла: speed_adder.cpp

```
#include "speed_adder.h"


speed_adder::speed_adder(int x, int y, int k_public, QImage position): item_object(x,y,getK(),position)
```

```
{
    this->k_private=k_public;
    this->x_private=x;
    this->y_private=y;
    this->position=position;
}
void speed_adder::set_speed_adder(int x,int y, int k_public)
{
    item_object::set_item_object(this->x_private=x, this->y_private=y,
this->k_private=k_public);
}
```