Плюсы:

1. Использование классов
2. Код выдержан в едином стиле

```
public:
    humen(int x, int y, QGraphicsEllipseItem* i);
    QPointF getNowPoint();
    QPointF getEndPoint();
    QPointF getAddStep();
    QPoint getOld();
    QGraphicsEllipseItem* getItem();
    int getHP();
    void setAddPoint(int x, int y);
    void setEndPoint(int x,int y);
    void addPoint();
    void setWeyExit();
    void setOld(int x,int y);
    bool getWeyExit();
    bool deliteHP();

private:
    int hp;
    int nowX;
    int nowY;
    int endX;
    int endY;
    int addX;
    int addY;
    int oldX;
    int oldY;
    bool toExit;
    QGraphicsEllipseItem* item;
```

3. Отсутствие глобальных переменных
4. Использование стандартных структур данных

```
QList< QList<int> > roomMatrica;
QList<humen*> bodys;
QList<fire*> burn;
QList<fire*> tempBurn;
```

Минусы:

1. Используются генерируемые методы

```
private slots:
    void on_pushButton_clicked();
```

2. Применяются непонятные константы

```
hp-=10;
if(hp>0)
    return true;
else
    return false;
```

3. Присутствуют закомментированные участки кода

```
void MainWindow::createMenu()
{
    QMenu *menuFile=new QMenu("File");
    QAction *newAction = new QAction("New",menuFile);
    //QAction *loadAction = new QAction("Load",menuFile);
    //QAction *saveAction = new QAction("Save",menuFile);
    QAction *exitAction = new QAction("Exit",menuFile);
    menuFile->addAction(newAction);
    //menuFile->addAction(loadAction);
    //menuFile->addAction(saveAction);
    menuFile->addAction(exitAction);
    ui->menuBar->addMenu(menuFile);
    connect(newAction, SIGNAL(triggered()), this, SLOT(newRoom()));
    //connect(loadAction, SIGNAL(triggered()), this, SLOT(loadRoom()));
    //connect(saveAction, SIGNAL(triggered()), this, SLOT(saveRoom()));
    connect(exitAction, SIGNAL(triggered()), this, SLOT(Exit()));
}
```

4. Отсутствуют переносы строк в некоторых местах, приходится использовать прокрутку

```
if(roomMatrica[stepY][stepX]==1)
{
    sc->addRect(stepX*house->getS()+1,stepY*house->getS()+1,house->getS()-1,house->getS()-1,QPen(Qt::NoPen),QBrush(Qt::darkGreer
    roomMatrica[stepY][stepX]=4;
}
```

5. Повторяющиеся участки кода

```
if(ui->addWall->isChecked()==true)//Если выбрано добавление стены
{
    int x=mouseEvent->x()-4;
    int y=mouseEvent->y()-37;
    int stepX=x/house->getS();
    int stepY=y/house->getS();
    if(roomMatrica[stepY][stepX]==0)
    {
        sc->addRect(stepX*house->getS()+1,stepY*house->getS()+1,house->getS()-1,house->getS()-1,QPen(Qt::NoPen),QBrush(Qt::gray));
        roomMatrica[stepY][stepX]=2;
    }
}
if(ui->addExit->isChecked()==true)//Если выбрано добавление выхода
{
    int x=mouseEvent->x()-4;
    int y=mouseEvent->y()-37;
    int stepX=x/house->getS();
    int stepY=y/house->getS();
    if(roomMatrica[stepY][stepX]==1)
    {
        sc->addRect(stepX*house->getS()+1,stepY*house->getS()+1,house->getS()-1,house->getS()-1,QPen(Qt::NoPen),QBrush(Qt::darkGreer
        roomMatrica[stepY][stepX]=4;
    }
```

6. Методы большого размера
7. Недостаточное количество комментариев, а имеющиеся недостаточно информативны

8. Некорректные названия переменных

```
QTimer *timerHumen;
QTimer *timerFire;
QGraphicsEllipseItem *Humen;
```

9. Код неудобочитаем вследствие таких факторов, как отсутствие обособления знаков отношений и операций пробелами

```
    for(int i=0;i<roomMatrica[i].size();i++)
        roomMatrica[i].clear();
    roomMatrica.clear();
    bodys.clear();
    burn.clear();
}
Dialog dialog;
dialog.show();
this->hide();
dialog.exec(); // показываем диалог, здесь стоим, пока диалог не закроют
this->show();
if(dialog.getUsed())
{
    use=true;
    house=new room(dialog.getHeight(),dialog.getWidht(),dialog.getStep());
```

10. Пустые методы в классах

```
void MainWindow::loadRoom()
{

}

void MainWindow::saveRoom()
{

}
```

11. Логика не отделена от графических компонентов

Итоговая оценка 3 из 10