Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики» (СибГУТИ)

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств
вычислительной техники и автоматизированных
систем"

ОТЧЕТ

по дисциплине «Визуальное программирование и
человеко-машинное взаимодействие»

Лабораторная работа 7

Выполнил:
студент группы ИП-811 Разумов Д.Б.

Проверил:
преподаватель Мерзлякова Е.Ю.

Новосибирск 2020

Оглавление

Задание

Цель: Научиться работать с графикой в Qt.
Требование: Задание по вариантам. Выполняется по подгруппам 1-2 человека. Можно придумать свой вариант, предварительно согласовав с преподавателем.
Задание:
1. Создать графическую сцену.
2. Поместить на сцену различные элементы для составления картинки по варианту. Обязательно использовать и геометрические фигуры, и картинки. Они должны перемещаться с помощью мыши.
3. Ограничить края сцены «стенами» в виде каких-либо элементов.
4. Поместить на сцену движущийся элемент по заданию. Он должен перемещаться с заданной скоростью, сталкиваться со «стенами» и фигурами на сцене. Используйте таймер и функцию обнаружения столкновений.

Варианты по формуле (n mod 10)+1, где n – номер студента из подгруппы.
1) Башня и движущийся рыцарь
2) Дом и движущаяся мышь
3) Яблоня и движущаяся птица
4) Машина и движущийся мяч
5) Грядки и движущаяся ворона
6) Пальма и движущаяся обезьяна
7) Поезд и движущийся лист
8) Обед и движущаяся муха
9) Новогодняя ель и движущаяся снежинка
10) Человек и движущийся телефон

Мой номер по списку 14, поэтому я выбрал 5 вариант.

Описание программы

В моей программе есть следующие ключевые файлы:

- razumov_moveitem.h
- razumov_widget.h
- main.cpp
- razumov_moveitem.cpp
- razumov_widget.cpp
- razumov_widget.ui
- и также несколько изображений

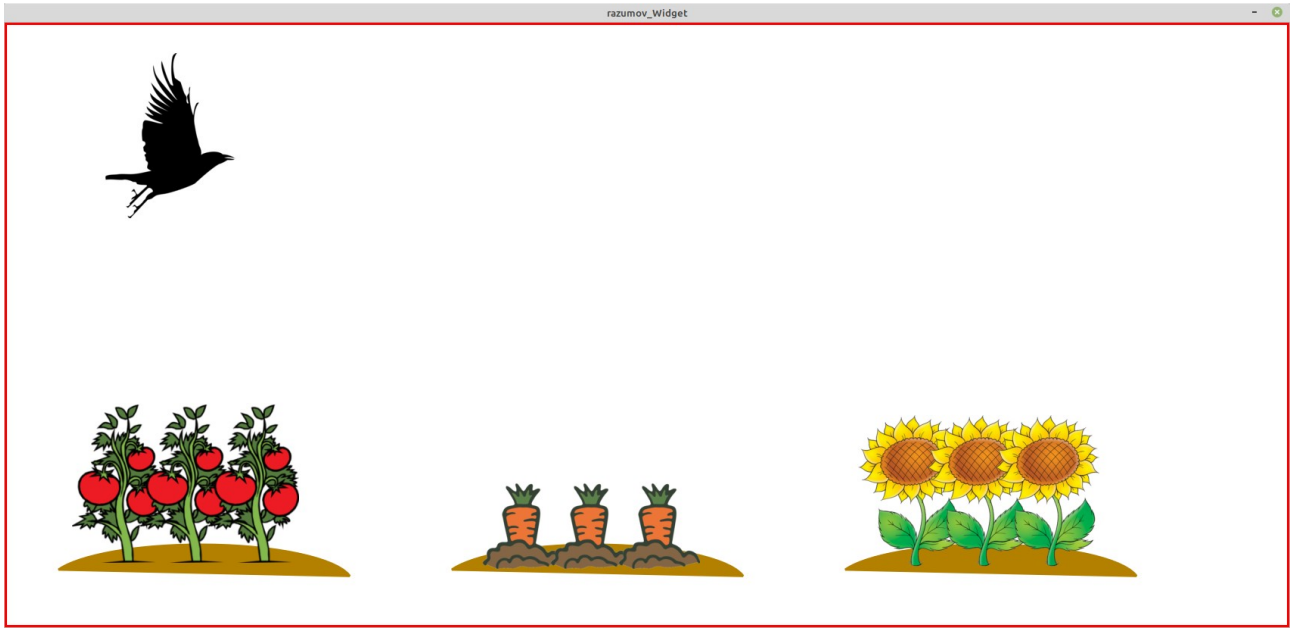
В **razumov_moveitem.h** и **razumov_moveitem.cpp** описаны несколько классов для изображений. Главный класс среди них **MoveGardenBed**, который наследуется от QObject и QGraphicsItem. В нем описаны методы: **boundingRect()** для обозначения границ изображения, **paint()** для отображения грядки, и еще три метода для перетаскивания изображения. Все остальные классы в этом файле наследуются от **MoveGardenBed**, в этих классах переопределяются методы boundingRect() и paint().

MoveCrow — класс для изображения вороны. В есть поля **valueX, valueY, imageDirection**, которые нужны для автоматического передвижения изображения. Ворона «двигается» только горизонтально в двух направлениях. Если ворона «сталкивается» с других изображением, то ее направление меняется на противоположное. Изображение вороны можно перетаскивать, как и другие изображения.

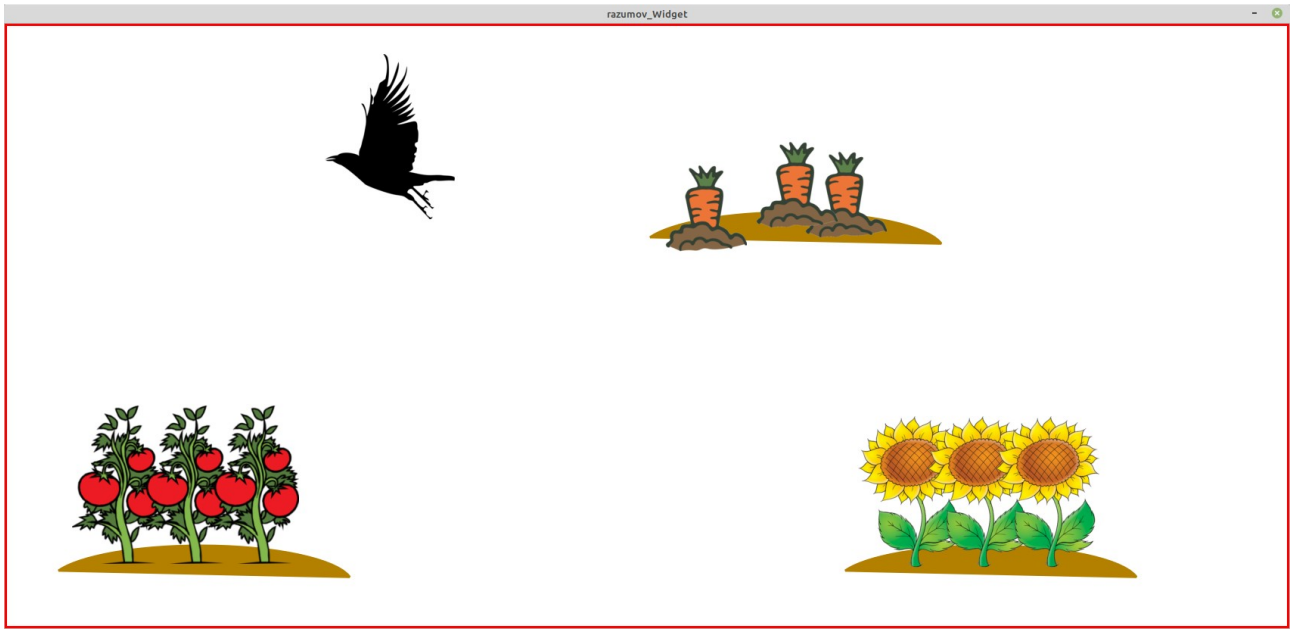
В **razumov_widget.cpp** создается окно и задаются его размеры. Помещается QGraphicsScene в QGpaphicsView. В QgraphicsScene добавляются линии по краям, объекты вышеописанных классов (т.е. изображения). Объект, в котором отрисовывается ворона, добавляется в отдельный поток, чтобы изображение вороны перемещалось одновременно с перетаскиванием других изображений.

Грядки и границы окна нарисованы только с помощью средств Qt, остальное — с помощью загрузки изображений на жестком диске.

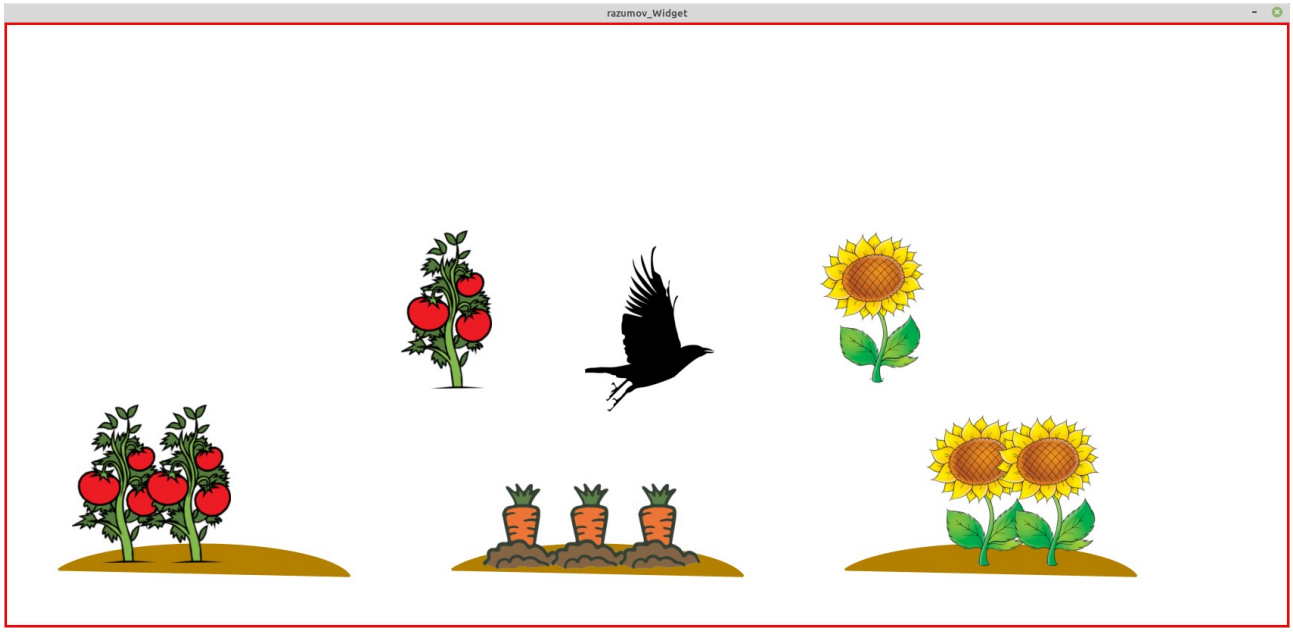
Выполнение программы, скриншоты



Скриншот 1. Начальный вид программы.



Скриншот 2. Перетаскивание грядки с морковками. Ворона, столкнувшись с ней, полетела в другую сторону.



Скриншот 3. Перетаскивание вороны.

Листинг

Файл razumov_moveitem.h

```
#ifndef RAZUMOV_MOVEITEM_H
#define RAZUMOV_MOVEITEM_H

#include <QObject>
#include <QGraphicsItem>
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QDebug>
#include <QCursor>
#include <QThread>
#include <QList>

///--- MoveGardenBed ---/// класс для грядок
class MoveGardenBed : public QObject, public QGraphicsItem
{
    Q_OBJECT
public:
    explicit MoveGardenBed(QObject *parent = nullptr);
    ~MoveGardenBed();

public:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
    void mouseMoveEvent(QGraphicsSceneMouseEvent *event);
    void mousePressEvent(QGraphicsSceneMouseEvent *event);
    void mouseReleaseEvent(QGraphicsSceneMouseEvent *event);
};

///--- MoveTomato ---/// класс для помидоров
class MoveTomato : public MoveGardenBed
{
    Q_OBJECT
public:
    explicit MoveTomato(QObject *parent = nullptr);
    ~MoveTomato();

private:
    //переопределяемые методы:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
};

///--- MoveCarrot ---/// класс для морковок
class MoveCarrot : public MoveGardenBed
{
    Q_OBJECT
public:
    explicit MoveCarrot(QObject *parent = nullptr);
    ~MoveCarrot();
```

```
private:
    //переопределяемые методы:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
};

///--- MoveSunflower ---/// класс для подсолнухов
class MoveSunflower : public MoveGardenBed
{
    Q_OBJECT
public:
    explicit MoveSunflower(QObject *parent = nullptr);
    ~MoveSunflower();

private:
    //переопределяемые методы:
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
};

///--- MoveCrow ---/// класс для вороны
class MoveCrow : public MoveGardenBed
{
    Q_OBJECT
public:
    explicit MoveCrow(QObject *parent = nullptr);
    ~MoveCrow();

public slots:
    void flying_crow(); //слот для автопередвижения вороны

private:
    qreal valueX; //координаты изображения вороны
    qreal valueY; //qreal для setPos(mapToScene())
    char imgDirection;
    //переопределяемые методы:
    void mouseMoveEvent(QGraphicsSceneMouseEvent *event);
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
};

#endif // RAZUMOV_MOVEITEM_H
```

Файл razumov_widget.h

```
#ifndef RAZUMOV_WIDGET_H
#define RAZUMOV_WIDGET_H

#include <QWidget>
#include <QGraphicsScene>
```

```
#include "razumov_moveitem.h"

namespace Ui {
class razumov_Widget;
}

class razumov_Widget : public QWidget
{
    Q_OBJECT

public:
    explicit razumov_Widget(QWidget *parent = nullptr);
    ~razumov_Widget();

private:
    Ui::razumov_Widget *ui;
    QGraphicsScene *scene;
};

#endif // RAZUMOV_WIDGET_H
```

Файл main.cpp

```
#include "razumov_widget.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    razumov_Widget w;
    w.show();

    return a.exec();
}
```

Файл razumov_moveitem.cpp

```
#include "razumov_moveitem.h"

///--- MoveGardenBed ---/// класс для грядок
MoveGardenBed::MoveGardenBed(QObject *parent) :
    QObject(parent), QGraphicsItem() {}

MoveGardenBed::~MoveGardenBed() {}

QRectF MoveGardenBed::boundingRect() const
{
    //создание прямоугольника, который ограничивает изображение,
    //координаты от центра изображения:
    return QRectF (-200, -10, 400, 100);
```

```
}

void MoveGardenBed::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    //границы:
    painter->setPen(QPen(QColor::fromRgbF(0.7, 0.5, 0, 1), 3));
    //заливка:
    painter->setBrush(QBrush(QColor::fromRgbF(0.7, 0.5, 0, 1),
Qt::SolidPattern));
    //нарисовать хорду (часть окружности):
    painter->drawChord(-200, -10, 400, 100, 10*16, 150*16);
    //без этого класс будет считаться абстрактным:
    Q_UNUSED(option);
    Q_UNUSED(widget);
}

void MoveGardenBed::mouseMoveEvent(QGraphicsSceneMouseEvent *event)
{
    //изменить координаты изображения на координаты курсора:
    this->setPos(mapToScene(event->pos()));
}

void MoveGardenBed::mousePressEvent(QGraphicsSceneMouseEvent *event)
{
    //изменение вида курсора на руку:
    this->setCursor(QCursor(Qt::ClosedHandCursor));
    Q_UNUSED(event);
}

void MoveGardenBed::mouseReleaseEvent(QGraphicsSceneMouseEvent *event)
{
    //поменять вид курсора на обычную стрелку:
    this->setCursor(QCursor(Qt::ArrowCursor));
    Q_UNUSED(event);
}

///--- MoveTomato ---/// класс для помидоров
MoveTomato::MoveTomato(QObject *parent) :
    MoveGardenBed(parent) {}

MoveTomato::~MoveTomato() {}

QRectF MoveTomato::boundingRect() const
{
    //создание прямоугольника, который ограничивает изображение,
    //координаты от центра изображения:
    return QRectF (-60, -105, 130, 229);
}

void MoveTomato::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    QImage img_tomato("../razumov_lab7/tomato.png");
    painter->drawImage(-60, -105, img_tomato);
    //без этого класс будет считаться абстрактным:
    Q_UNUSED(option);
```

```
    Q_UNUSED(widget);
}

///--- MoveCarrot ---/// класс для морковок
MoveCarrot::MoveCarrot(QObject *parent) :
    MoveGardenBed(parent) {}

MoveCarrot::~MoveCarrot() {}

QRectF MoveCarrot::boundingRect() const
{
    //создание прямоугольника, который ограничивает изображение,
    //координаты от центра изображения:
    return QRectF (-60, -63, 120, 127);
}

void MoveCarrot::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    QImage img_carrot("../razumov_lab7/carrot.png");
    painter->drawImage(-60, -63, img_carrot);
    //без этого класс будет считаться абстрактным:
    Q_UNUSED(option);
    Q_UNUSED(widget);
}

///--- MoveSunflower ---/// класс для подсолнухов
MoveSunflower::MoveSunflower(QObject *parent) :
    MoveGardenBed(parent) {}

MoveSunflower::~MoveSunflower() {}

QRectF MoveSunflower::boundingRect() const
{
    //создание прямоугольника, который ограничивает изображение,
    //координаты от центра изображения:
    return QRectF (-71, -101, 142, 202);
}

void MoveSunflower::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    QImage img_sunflower("../razumov_lab7/sunflower.png");
    painter->drawImage(-71, -101, img_sunflower);
    //без этого класс будет считаться абстрактным:
    Q_UNUSED(option);
    Q_UNUSED(widget);
}

///--- MoveCrow ---/// класс для вороны
MoveCrow::MoveCrow(QObject *parent) :
    MoveGardenBed(parent)
{
    valueX = valueY = 150; //начальные координаты
    imgDirection = 'r'; //направление движения
}
```

```
MoveCrow::~MoveCrow() {}

void MoveCrow::flying_crow() {
    //слот для автопередвижения изобржения вороны
    int summandX = 10; //на сколько пикселей перемещение
    while(1) {
        valueX += summandX; //изменение координаты по оси X
        this->setX(valueX); //отрисовка с новыми координатами
        //получить список QGraphicsItem, с которыми ворона пересекается:
        QList<QGraphicsItem*> *list = new QList<QGraphicsItem*>;
        *list = this->collidingItems();
        //если ворона с чем-то столкнулась, то поменять направление:
        if (list->count() > 0) {
            if (imgDirection == 'r')
                imgDirection = 'l';
            else
                imgDirection = 'r';
            summandX *= -1;
        }
        QThread::msleep(300); //задержка
    }
}

void MoveCrow::mouseMoveEvent(QGraphicsSceneMouseEvent *event)
{
    //переопределение mouseMoveEvent:
    valueX = event->pos().x();
    valueY = event->pos().y();
    this->setPos(mapToScene(valueX, valueY));
    QThread::msleep(300); //без задержки критический баг
}

QRectF MoveCrow::boundingRect() const
{
    return QRectF (-100, -109, 200, 218);
    //размере по ширине немного больше, засчет этого картинка вороны
    //не перекрывает другие картинки при "столкновении"
}

void MoveCrow::paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget)
{
    QImage img_crow;
    if (imgDirection == 'r') //если ворона летит вправо
        img_crow.load("../razumov_lab7/crow_toright.png");
    else //иначе влево
        img_crow.load("../razumov_lab7/crow_toleft.png");
    painter->drawImage(-86, -109, img_crow);
    //без этого класс будет считаться абстрактным:
    Q_UNUSED(option);
    Q_UNUSED(widget);
}
```

Файл razumov_widget.cpp

```
#include "razumov_widget.h"
#include "ui_razumov_widget.h"

razumov_Widget::razumov_Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::razumov_Widget)
{
    ui->setupUi(this);
    //размеры окна:
    int winWidth = 1700, winHeight = 800;
    this->resize(winWidth, winHeight);
    this->setFixedSize(winWidth, winHeight);
    //создание графической сцены:
    scene = new QGraphicsScene(this);
    scene->setItemIndexMethod(QGraphicsScene::NoIndex); //индексация элементов
    //настройка graphicsView:
    ui->graphicsView->setGeometry(0, 0, winWidth, winHeight); //размер
graphicsView
    ui->graphicsView->setScene(scene); //установить графическую сцену в
graphicsView
    ui->graphicsView->setRenderHint(QPainter::Antialiasing); //рендер
    ui->graphicsView->setCacheMode(QGraphicsView::CacheBackground); //кэш фона
    ui->graphicsView-
>setViewportUpdateMode(QGraphicsView::BoundingRectViewportUpdate);
    //размер сцены:
    scene->setSceneRect(10, 10, winWidth-20, winHeight-20);
    //рамки, чтобы ворона "не улетела" за края окна:
    scene->addLine(0, 2, winWidth, 2, QPen(Qt::red, 4));
    scene->addLine(winWidth-2, 0, winWidth-2, winHeight, QPen(Qt::red, 4));
    scene->addLine(0, winHeight-2, winWidth, winHeight-2, QPen(Qt::red, 4));
    scene->addLine(2, 0, 2, winHeight, QPen(Qt::red, 4));
    //создание изображений:
    MoveGardenBed *bed[3];
    MoveTomato *tomato[3];
    MoveCarrot *carrot[3];
    MoveSunflower *sunflower[3];
    for (int i = 0; i < 3; i++) {
        bed[i] = new MoveGardenBed();
        bed[i]->setPos(260 + i * 520, 700);
        scene->addItem(bed[i]);
    }
    for (int i = 0; i < 3; i++) {
        tomato[i] = new MoveTomato();
        tomato[i]->setPos(150 + i * 90, 610);
        scene->addItem(tomato[i]);
        carrot[i] = new MoveCarrot();
        carrot[i]->setPos(680 + i * 90, 660);
        scene->addItem(carrot[i]);
        sunflower[i] = new MoveSunflower();
        sunflower[i]->setPos(1200 + i * 90, 620);
        scene->addItem(sunflower[i]);
    }
    //ворона:
    MoveCrow *crow = new MoveCrow();
    crow->setPos(150, 150);
    scene->addItem(crow);
```

```
    //добавления слота для автопередвижения вороны в отдельный поток,
    //чтобы ворона двигалась одновременно с действиями пользователя:
    QThread *thread= new QThread;
    crow->moveToThread(thread);
    connect(thread, SIGNAL(started()), crow, SLOT(flying_crow()));
    thread->start();
}

razumov_Widget::~razumov_Widget()
{
    delete ui;
}
```