Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Лабораторная работа 7

Выполнил: студент группы ИП-814

Краснов И.В.

Проверил: доцент кафедры ПМ и К

Мерзлякова Е. Ю.

Оглавление

[Постановка задачи 3](#_Toc62073070)

[Листинг программы 4](#_Toc62073071)

[Скриншоты работы программы 10](#_Toc62073072)

# **Постановка задачи**

**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.

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.

**Вариант:** 9)Новогодняя ель и движущаяся снежинка

# **Листинг программы**

main.cpp:

#include "widget\_krasnov.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

widget\_krasnov window;

window.setWindowTitle("Krasnov");

window.show();

*return* a.exec();

}

widget\_krasnov.h:

#ifndef WIDGET\_KRASNOV\_H

#define WIDGET\_KRASNOV\_H

#include <QWidget>

#include <QGraphicsScene>

#include <QGraphicsItem>

#include "windows.h"

#include <QTimer>

#include <snowflake.h>

#include <tree.h>

*namespace* **Ui** {

*class* **widget\_krasnov**;

}

*class* **widget\_krasnov** : *public* QWidget

{

Q\_OBJECT

*public*:

*explicit* **widget\_krasnov**(QWidget \*parent = *nullptr*);

~***widget\_krasnov***();

*private*:

Ui::widget\_krasnov \*ui;

QGraphicsScene \*scene;

QTimer timer;

Snowflake \*sf;

Tree \*ch;

};

#endif *//* *WIDGET\_KRASNOV\_H*

widget\_krasnov.cpp:

#include "widget\_krasnov.h"

#include "ui\_widget\_krasnov.h"

widget\_krasnov::**widget\_krasnov**(QWidget \*parent) :

QWidget(*parent*),

ui(*new* Ui::widget\_krasnov)

{

ui->setupUi(*this*);

*this*->resize(700, 700);

scene = *new* QGraphicsScene;

ui->graphicsView->setScene(*scene*);

ui->graphicsView->setVerticalScrollBarPolicy(Qt::*ScrollBarAlwaysOff*);

ui->graphicsView->setHorizontalScrollBarPolicy(Qt::*ScrollBarAlwaysOff*);

scene->setSceneRect(0, 0, 700, 700);

scene->addRect(30,30,58,20,QPen(Qt::*NoPen*),Qt::*green*);

scene->addRect(300,500,120,80,QPen(Qt::*NoPen*),Qt::*blue*);

scene->addRect(39,300,98,157,QPen(Qt::*NoPen*),Qt::*red*);

scene->addLine(10, 10, 10, 690);

scene->addLine(10, 10, 690, 10);

scene->addLine(690, 10, 690, 690);

scene->addLine(690, 690, 10, 690);

sf = *new* Snowflake;

sf->setPos(300, 300);

scene->addItem(*sf*);

ch = *new* Tree;

ch->setPos(200, 200);

scene->addItem(*ch*);

}

widget\_krasnov::~***widget\_krasnov***()

{

*delete* ui;

}

snowflake.h:

#ifndef SNOWFLAKE\_H

#define SNOWFLAKE\_H

#include <QObject>

#include <QGraphicsItem>

#include <QGraphicsScene>

#include <QPainter>

#include <QPolygon>

#include <QTimer>

#include <QDebug>

#include <QCursor>

#include <cstdlib>

#include <time.h>

#include <windows.h>

*class* **Snowflake** : *public* QObject, *public* QGraphicsItem

{

Q\_OBJECT

Q\_INTERFACES(**QGraphicsItem**)

*public*:

*explicit* **Snowflake**(QObject \*parent = *nullptr*);

~***Snowflake***();

*private*:

QRectF ***boundingRect***() *const*;

void ***paint***(QPainter \*painter, *const* QStyleOptionGraphicsItem \*option, QWidget \*widget);

QPainterPath ***shape***() *const*;

*private* slots:

void **slotGameTimer**();

*private*:

bool shot;

bool right = *true*;

bool down = *true*;

bool check = *true*;

QPixmap \*spriteImage;

QTimer \*gameTimer;

int prevDir = 1;

int currDir = 1;

};

#endif *//* *SNOWFLAKE\_H*

snowflake.cpp:

#include "snowflake.h"

#include <math.h>

Snowflake::**Snowflake**(QObject \*parent):

QObject(*parent*), QGraphicsItem()

{

spriteImage = *new* QPixmap("C:\\Study\\Visual Programming\\Lab7\\snowflake.png");

gameTimer = *new* QTimer();

connect(gameTimer, &QTimer::timeout, *this*, &Snowflake::slotGameTimer);

gameTimer->start(5);

srand(time(NULL));

}

Snowflake::~***Snowflake***()

{

}

QRectF Snowflake::***boundingRect***() *const*

{

*return* QRectF(-12,-15,70,80);

}

QPainterPath Snowflake::***shape***() *const*

{

QPainterPath path;

path.addEllipse(*boundingRect*());

*return* path;

}

void Snowflake::***paint***(QPainter \*painter, *const* QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

painter->drawPixmap(0, 0, \*spriteImage, 0, 0, 50, 50);

Q\_UNUSED(option);

Q\_UNUSED(widget);

}

void Snowflake::**slotGameTimer**()

{

*if*(!scene()->collidingItems(*this*).isEmpty()){

*do* {

currDir = rand() % 8;

} *while*(currDir == prevDir);

prevDir = currDir;

}

*if* (currDir == 0){

*this*->setY(*this*->y() - 1);

}

*if* (currDir == 1){

*this*->setY(*this*->y() - 1);

*this*->setX(*this*->x() + 1);

}

*if* (currDir == 2){

*this*->setX(*this*->x() + 1);

}

*if* (currDir == 3){

*this*->setX(*this*->x() + 1);

*this*->setY(*this*->y() + 1);

}

*if* (currDir == 4){

*this*->setY(*this*->y() + 1);

}

*if* (currDir == 5){

*this*->setY(*this*->y() + 1);

*this*->setX(*this*->x() - 1);

}

*if* (currDir == 6){

*this*->setX(*this*->x() - 1);

}

*if* (currDir == 7){

*this*->setX(*this*->x() - 1);

*this*->setY(*this*->y() - 1);

}

}

tree.h:

#ifndef TREE\_H

#define TREE\_H

#include <QObject>

#include <QGraphicsItem>

#include <QPainter>

#include <QGraphicsSceneMouseEvent>

#include <QDebug>

#include <QCursor>

*class* **Tree** : *public* QObject, *public* QGraphicsItem

{

Q\_OBJECT

*public*:

*explicit* **Tree**(QObject \*parent = 0);

~***Tree***();

signals:

*private*:

QRectF ***boundingRect***() *const*;

QPixmap \*spriteImage;

void ***paint***(QPainter \*painter, *const* QStyleOptionGraphicsItem \*option, QWidget \*widget);

void ***mouseMoveEvent***(QGraphicsSceneMouseEvent \*event);

void ***mousePressEvent***(QGraphicsSceneMouseEvent \*event);

void ***mouseReleaseEvent***(QGraphicsSceneMouseEvent \*event);

};

#endif *//* *TREE\_H*

tree.cpp:

#include "tree.h"

Tree::**Tree**(QObject \*parent) :

QObject(*parent*), QGraphicsItem()

{

}

Tree::~***Tree***()

{

}

QRectF Tree::***boundingRect***() *const*

{

*return* QRectF (-12,-15,70,100);

}

void Tree::***paint***(QPainter \*painter, *const* QStyleOptionGraphicsItem \*option, QWidget \*widget)

{

spriteImage = *new* QPixmap("C:\\Study\\Visual Programming\\Lab7\\christmas\_tree.png");

painter->drawPixmap(-10, -10, \*spriteImage, 0, 0, 60, 90);

Q\_UNUSED(option);

Q\_UNUSED(widget);

}

void Tree::***mouseMoveEvent***(QGraphicsSceneMouseEvent \*event)

{

*this*->setPos(mapToScene(event->pos()));

}

void Tree::***mousePressEvent***(QGraphicsSceneMouseEvent \*event)

{

*this*->setCursor(QCursor(Qt::*ClosedHandCursor*));

Q\_UNUSED(event);

}

void Tree::***mouseReleaseEvent***(QGraphicsSceneMouseEvent \*event)

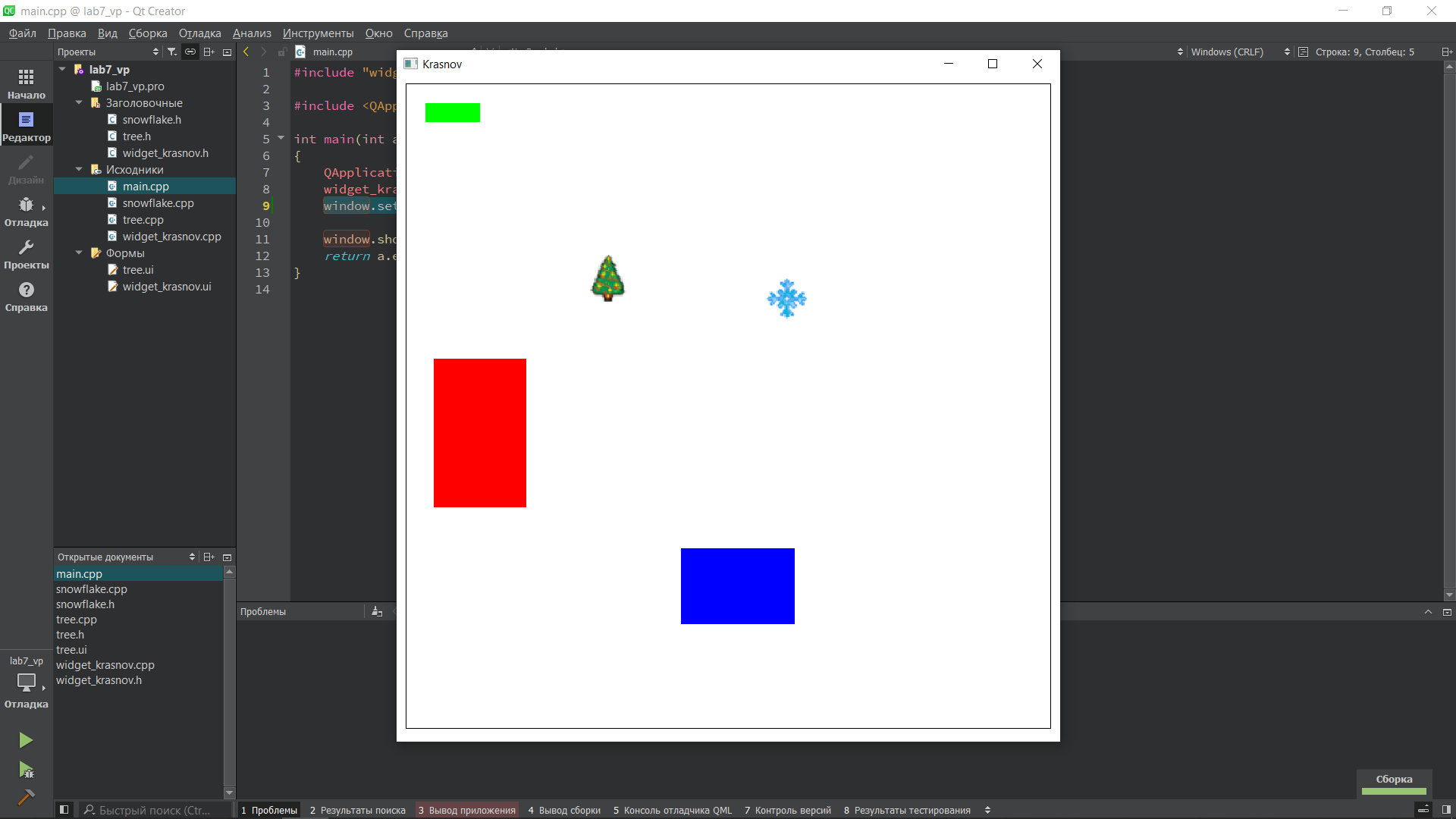
{

*this*->setCursor(QCursor(Qt::*ArrowCursor*));

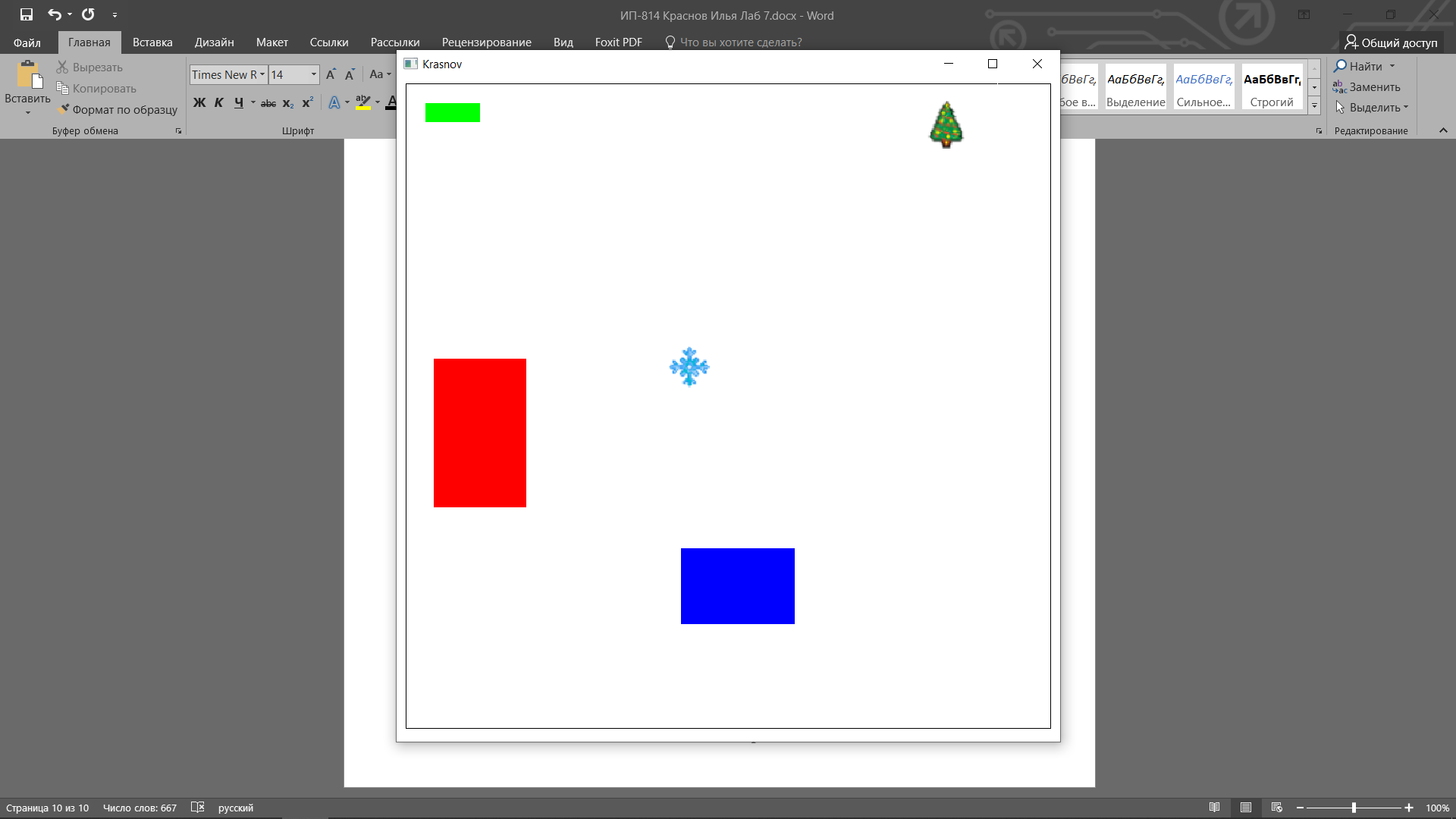
Q\_UNUSED(event);

}

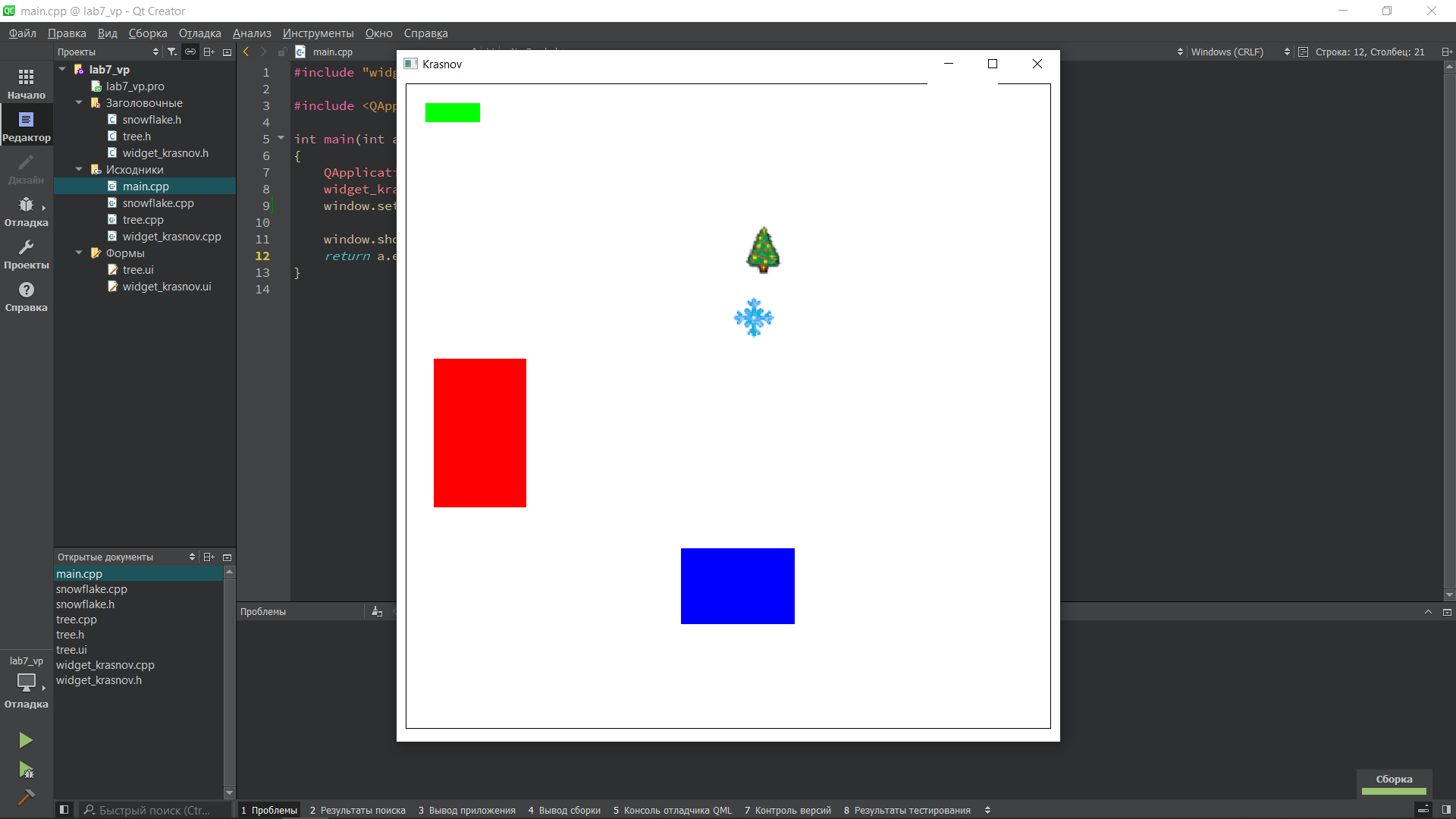
# **Скриншоты работы программы**



(рис.1) Программа после запуска

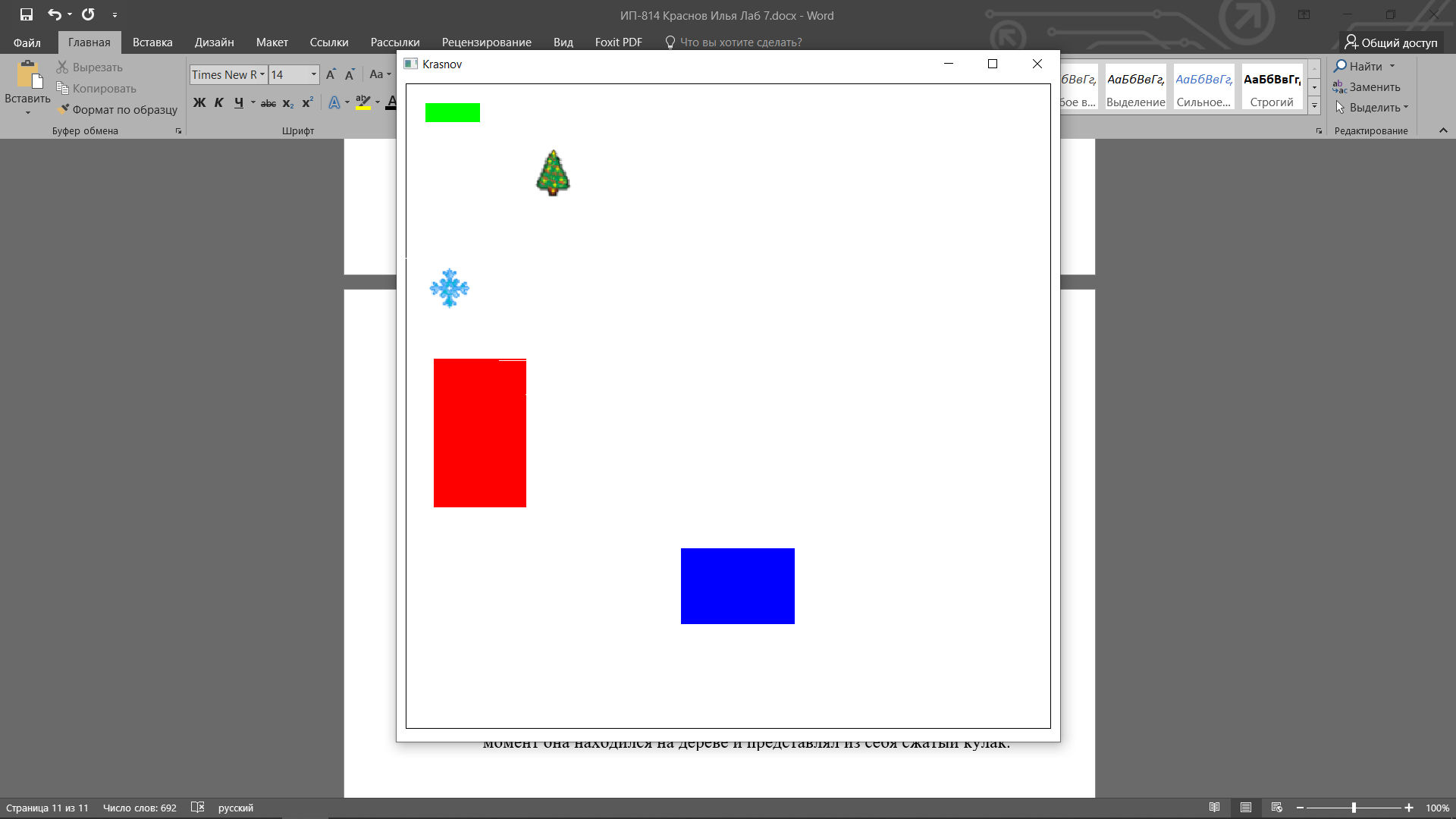


(рис.2) Снежинка переместилась



(рис.3) Перемещение ёлки

К сожалению, при сохранении скриншота не отображается курсор, но в тот момент она находился на дереве и представлял из себя сжатый кулак.



(рис.4) Снежинка загнана в угол

С помощью ёлки удалось изменить направление движения снежинки. Снежинка сталкивалась с объектами на экране и выбирала новое направление движения