Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Лабораторная работа 7

Выполнил: студент группы ИП-814

Якунин А.В.

Проверил: доцент кафедры ПМ и К

Мерзлякова Е. Ю.

Новосибирск 2020г.

**Содержание**

1. Задание к лабораторной работе 3

2.Листинг программы 4

3.Скриншоты работы программы 10

**Задание к лабораторной работе**

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 мышь. Она должна перемещаться с заданной скоростью, сталкиваться со «стенами» и домом на сцене. Используйте таймер и функцию обнаружения столкновений.

**Листинг программы**

**animate7\_yakunin.h**

#ifndef ANIMATE7\_YAKUNIN\_H

#define ANIMATE7\_YAKUNIN\_H

#include "endlesscycle.h"

#include <QWidget>

QT\_BEGIN\_NAMESPACE

namespace Ui { class animate7\_yakunin; }

QT\_END\_NAMESPACE

class animate7\_yakunin : public QWidget

{

Q\_OBJECT

public:

animate7\_yakunin();

private:

endlesscycle elc;

};

#endif // ANIMATE7\_YAKUNIN\_H

**endlesscycle.h**

#ifndef ENDLESSCYCLE\_H

#define ENDLESSCYCLE\_H

#include <QBrush>

#include <QFont>

#include <QPen>

#include <QWidget>

#include <QString>

#include <QImage>

#include <QMouseEvent>

#include <QCursor>

class endlesscycle

{

public:

endlesscycle();

int speed\_x = 10, speed\_y = 10, bufferNumberOBJ = 0;

void setNewPos(int x, int y);

bool collision(int x, int y, int wOBJ, int hOBJ, int n);

int x\_mouse = 100, y\_mouse = 100, wMouse = 30, hMouse = 30, ObjectMinX, ObjectMaxX, ObjectMinY, ObjectMaxY, W, H;

QRect rect[6] = {{0,0,500,20},

{0,0,20,500},

{480,0,20,500},

{0,480,500,20},

{250,250,100,100},

{x\_mouse, y\_mouse, wMouse, hMouse}};

public:

void paint(QPainter \*painter, QPaintEvent \*event);

private:

QString temp = "C:/Andrey/prog/prog/kurs3/Visualnoe programirovanie/laby/lab7/dom.png";

QString temp2 = "C:/Andrey/prog/prog/kurs3/Visualnoe programirovanie/laby/lab7/mysh.png";

QImage mouse;

QImage home;

QBrush background;

QBrush circleBrush;

QFont textFont;

QPen circlePen;

QPen textPen;

void drug();

void move();

};

#endif // ENDLESSCYCLE\_H

**widget.h**

#ifndef WIDGET\_H

#define WIDGET\_H

#include <QWidget>

#include <endlesscycle.h>

class endlesscycle;

class widget : public QWidget

{

Q\_OBJECT

public:

widget(endlesscycle \*helper, QWidget \*parent);

public slots:

void mousePressEvent(QMouseEvent \*event) override;

void animate();

private:

void paintEvent(QPaintEvent \*event) override;

endlesscycle \*elc;

};

#endif // WIDGET\_H

**animate7\_yakunin.cpp**

#include "widget.h"

#include "animate7\_yakunin.h"

#include <QGridLayout>

#include <QLabel>

#include <QTimer>

animate7\_yakunin::animate7\_yakunin()

{

setWindowTitle(tr("Yaku"));

widget \*native = new widget(&elc, this);

QGridLayout \*layout = new QGridLayout;

layout->addWidget(native, 0, 0);

setLayout(layout);

QTimer \*timer = new QTimer(this);

connect(timer, &QTimer::timeout, native, &widget::animate);

timer->start(50);

}

**endlesscycle.cpp**

#include "endlesscycle.h"

#include <QPainter>

#include <QPaintEvent>

#include <QWidget>

#include <QDebug>

endlesscycle::endlesscycle()

{

QLinearGradient gradient(QPointF(50, -20), QPointF(80, 20));

gradient.setColorAt(0.0, Qt::white);

gradient.setColorAt(1.0, QColor(0xa6, 0xce, 0x39));

background = QBrush(QColor(64, 32, 64));

circlePen = QPen(Qt::black);

circlePen.setWidth(1);

textPen = QPen(Qt::white);

textFont.setPixelSize(12);

home.load(temp);

mouse.load(temp2);

}

void endlesscycle::setNewPos(int x, int y)

{

if(bufferNumberOBJ == 5)

{

x\_mouse = x;

y\_mouse = y;

rect[bufferNumberOBJ].setX(x);

rect[bufferNumberOBJ].setY(y);

}

else

{

rect[bufferNumberOBJ].setX(x);

rect[bufferNumberOBJ].setY(y);

rect[bufferNumberOBJ].setWidth(W);

rect[bufferNumberOBJ].setHeight(H);

}

}

bool endlesscycle::collision(int x, int y, int wOBJ, int hOBJ, int n)

{

int CheckMinX, CheckMaxX, CheckMinY, CheckMaxY, interLeft, interTop, interRight, interBottom;

ObjectMinX = x;

ObjectMaxX = x + wOBJ;

ObjectMinY = y;

ObjectMaxY = y + hOBJ;

for(int i = 0; i < n; i++)

{

CheckMinX = rect[i].x();

CheckMaxX = rect[i].x() + rect[i].width();

CheckMinY = rect[i].y();

CheckMaxY = rect[i].y() + rect[i].height();

interLeft = std::max(CheckMinX, ObjectMinX);

interBottom = std::max(CheckMinY, ObjectMinY);

interRight = std::min(CheckMaxX, ObjectMaxX);

interTop = std::min(CheckMaxY, ObjectMaxY);

if ((interLeft < interRight) && (interTop > interBottom))

{

bufferNumberOBJ = i;

return true;

}

}

return false;

}

void endlesscycle::move()

{

x\_mouse += speed\_x;

rect[5].setX(x\_mouse);

if(collision(x\_mouse, y\_mouse, wMouse, hMouse, 5))

{

x\_mouse -= speed\_x;

speed\_x \*= -1;

x\_mouse += speed\_x;

rect[5].setX(x\_mouse);

}

y\_mouse += speed\_y;

rect[5].setY(y\_mouse);

if(collision(x\_mouse, y\_mouse, wMouse, hMouse, 5))

{

y\_mouse -= speed\_y;

speed\_y \*= -1;

y\_mouse += speed\_y;

rect[5].setY(y\_mouse);

}

}

void endlesscycle::paint(QPainter \*painter, QPaintEvent \*event)

{

qDebug() << rect[4].width() << " " << rect[4].height();

move();

painter->fillRect(event->rect(), background);

painter->save();

painter->setBrush(circleBrush);

painter->setPen(circlePen);

painter->drawRect(rect[0].x(), rect[0].y(), rect[0].width(), rect[0].height());

painter->drawRect(rect[1].x(), rect[1].y(), rect[1].width(), rect[1].height());

painter->drawRect(rect[2].x(), rect[2].y(), rect[2].width(), rect[2].height());

painter->drawRect(rect[3].x(), rect[3].y(), rect[3].width(), rect[3].height());

painter->drawPixmap(rect[4].x(), rect[4].y(), rect[4].width(), rect[4].height(),QPixmap::fromImage(home));

painter->drawPixmap(x\_mouse, y\_mouse, wMouse, hMouse, QPixmap::fromImage(mouse));

painter->restore();

painter->setPen(textPen);

painter->setFont(textFont);

painter->drawText(QRect(x\_mouse, y\_mouse, 30, -10), Qt::AlignCenter, QStringLiteral("Мышь"));

}

**main.cpp**

#include "animate7\_yakunin.h"

#include <QApplication>

#include <QSurfaceFormat>

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

{

QApplication a(argc, argv);

QSurfaceFormat fmt;

fmt.setSamples(4);

QSurfaceFormat::setDefaultFormat(fmt);

animate7\_yakunin w;

w.show();

return a.exec();

}

**widget.cpp**

#include "widget.h"

#include "endlesscycle.h"

#include <QPainter>

#include <QTimer>

widget::widget(endlesscycle \*elc, QWidget \*parent)

: QWidget(parent), elc(elc)

{

setFixedSize(500, 500);

}

void widget::animate()

{

update();

}

void widget::mousePressEvent(QMouseEvent\* event)

{

if( (event->button() == Qt::LeftButton) and

elc->collision(event->x(), event->y(), 10, 10, 6) )

{

elc->speed\_x = 0;

elc->speed\_y = 0;

}

else if((event->button() == Qt::RightButton))

{

elc->W = elc->rect[elc->bufferNumberOBJ].width();

elc->H = elc->rect[elc->bufferNumberOBJ].height();

elc->setNewPos(event->x(), event->y());

elc->speed\_x = 10;

elc->speed\_y = 10;

}

}

void widget::paintEvent(QPaintEvent \*event)

{

QPainter painter;

painter.begin(this);

painter.setRenderHint(QPainter::Antialiasing);

elc->paint(&painter, event);

painter.end();

}

**Скриншоты работы программы**

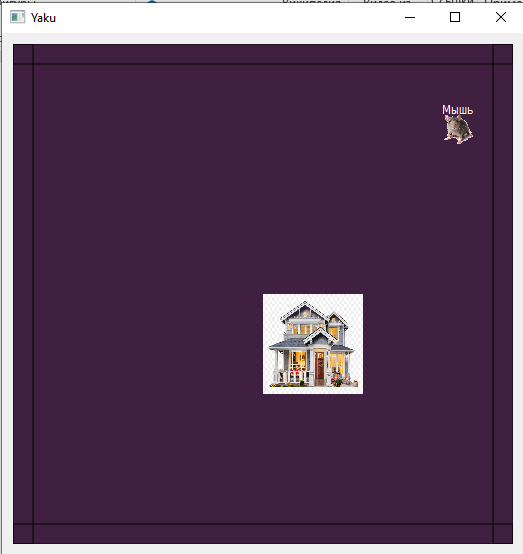


Рис. 1 Дом и движущаяся мышь

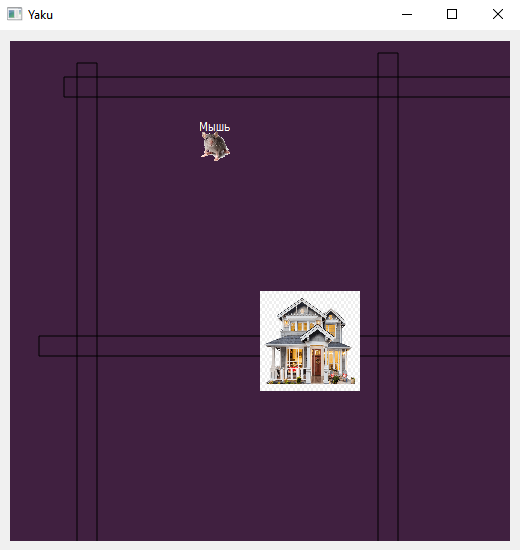


Рис. 2 Drag and drop стен

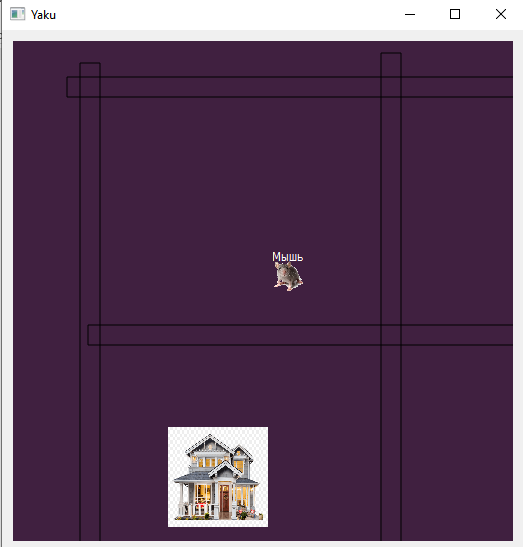


Рис. 3 Drag and drop дома

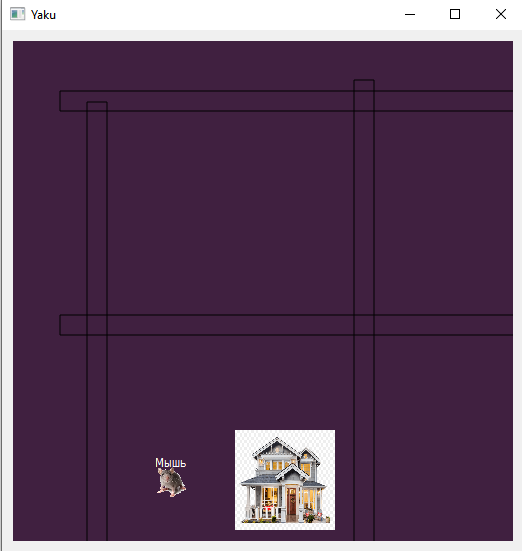


Рис. 4 Drag and drop мыши