Московский технологический университет
(МИРЭА/МГУПИ)

Физико-технологический институт

Кафедра аппаратного, программного и математического обеспечения вычислительных систем

Курсовая работа по дисциплине “Вычислительная геометрия”

Анимация движения круго по правильному многоугольнику

Группа: ТМБО-01-15
Студент: И. И. Аметов
Преподаватель: А. А. Кожевников

Москва, 2017

Содержание

Введение

Целью данной курсовой работы является отработка навыков самостоятельного проведения исследований некоторой задачи, выявления закономерностей, формирование на базе этих закономерностей математической модели и реализация этой модели в виде программы для ЭВМ.

В качестве задачи курсовой работы была выбрана проблема моделирования вращения круга вокруг правильного n угольника. Ставилась цель написать программу, способную визуально отобразить этот процесс на экране компьютера.

Такая постановка задачи позволяет в достаточно полной мере отразить полученные навыки работы с виджетами фреймворка Qt в объектно-ориентированном программировании. Помимо работы с виджетами в данной курсовой работе используется библиотека QPainter. Эта библиотека используется для рисования на некоторой области виджета.

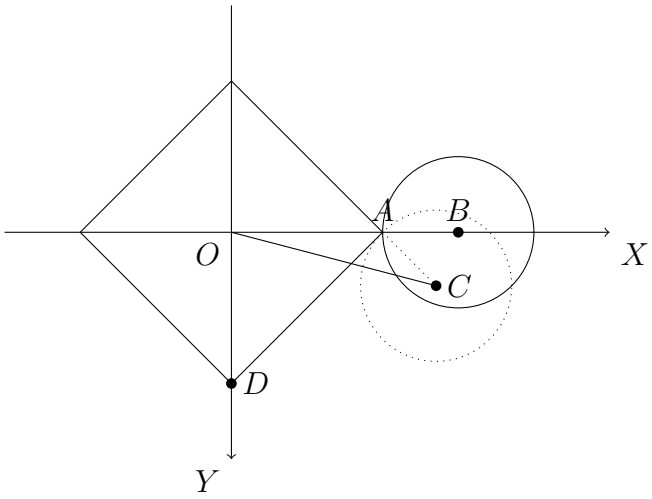

Рис. 1: Чертёж задачи

Постановка задачи

Требуется реализовать вращение круга с окружностью r вокруг правильного n угольника с радиусом описанной окружности R. Круг должен вращаться постоянно касаясь одной точкой правильного n угольника.

На входе программа запрашивает число сторон правильного многоугольника, значение радиуса описанной вокруг многоугольника окружности и радиус круга.

В основном окне отображается процесс работы модели.

Основная задача заключается в определении координат центра окружности, производящей движение. В качестве аргумента функции вычисления координат центра окружности был взят угол α между направлением оси OX и отрезка соединяющего точку начала координат O с центром вращающейся окружности.

В этом случае движение окружности можно разбить на две составные части: движение вокруг угла (на рисунке точка A) и движение по прямой линии вдоль ребра многоугольника. Поскольку многоугольник правильный, то его возможно представить в виде множества отрезков, повёрнутых относительно начала координат на угол $\frac{360k}{n}$, где $k = 0, 1, 2, ...$ — целое число, n — количество сторон или углов многоугольника. Тогда становится возможным определять положение центра окружности на отрезке AD и в дальнейшем осуществлять поворот этого центра вокруг начала координат. Так можно описать движение центра окружности по всей фигуре.

Движение вокруг угла многоугольника возникает при угле

$$\alpha \in \left(\frac{360k}{n} - \text{arctg}\left(\frac{r \sin(\frac{180}{n})}{R + r\cos(\frac{180}{n})} \right), \frac{360k}{n} + \text{arctg}\left(\frac{r \sin(\frac{180}{n})}{R + r\cos(\frac{180}{n})} \right) \right)$$

В этом случае координаты x' и y' возле угла многоугольника можно вычислить по формулам:

$$x' = R + r\cos\left(\alpha - \frac{360k}{n} + \arcsin\left(\frac{R}{r}\sin\left(\alpha - \frac{360k}{n}\right)\right)\right)$$
$$y' = wr\sin\left(\alpha - \frac{360k}{n} + \arcsin\left(\frac{R}{r}\sin\left(\alpha - \frac{360k}{n}\right)\right)\right)$$

где k — порядковый номер угла, возле которого осуществляется движение, а w — принимает значение -1, если угол α меньше $\frac{360k}{n}$ или равен 1, если угол α больше $\frac{360k}{n}$.

После этого осуществляется поворот на угол $\frac{360k}{n}$:

$$x = x'\cos(\frac{360k}{n}) - y'\sin(\frac{360k}{n}) + P_x$$
$$y = y'\cos(\frac{360k}{n}) + x'\sin(\frac{360k}{n}) + P_y$$

здесь $P_x = \frac{\text{ширина экрана}}{2}$, а $P_y = \frac{\text{длина экрана}}{2}$.

При движении по прямой вычисления центра движущейся окружности будут такими:

$$x' = \left(R + r\cos^{-1}\left(\frac{180}{n}\right)\right)\cos\left(\alpha \mod \frac{360}{n}\right) / \cos\left(\frac{180}{n} - \alpha \mod \frac{360}{n}\right)$$
$$y' = \left(R + r\cos^{-1}\left(\frac{180}{n}\right)\right)\sin\left(\alpha \mod \frac{360}{n}\right) / \cos\left(\frac{180}{n} - \alpha \mod \frac{360}{n}\right)$$

После чего нужно осуществить поворот:

$$x = x'\cos(\beta) + y'\sin(\beta) + P_x$$
$$y = y'\cos(\beta) - x'\sin(\beta) + P_y$$

здесь $\beta = -((\alpha n) \div 360)(\frac{180}{n})_{\text{radian}}$, где "$\div$" — операция получения неполного частного, $()_{\text{radian}}$ — перевод в радианы.

Алгоритм программы

Программа реализована в виде двух виджетов. Один виджет представляет основную канву на которой отображается анимация, другой виджет служит в качестве панели управления.

В основном виджете введены следующие функции и процедуры:

- `calcPolygon()` — формирует n-угольник. В цикле вычисляются координаты вершин многоугольника, полученные точки заносятся в указатель `polygon` для последующей отрисовки.
- `Widget(QWidget *parent)` — конструктор класса. Задаёт следующие переменные: радиус окружности описанной вокруг многоугольника окружности (`R = 50`), радиус окружности, вращающейся вокруг многоугольника (`r = 50`), количество углов многоугольника (`n = 3`), начальный угол (`angle = 0`) и шаг приращения угла (`angleStep = 1`). Здесь же создаётся виджет управления. Виджет управления позволяет менять размеры радиусов и количество углов. Для анимации вращения используется таймер класса `QTimer`.
- `drawScene(int x, int y)` — процедура отрисовки многоугольника и круга, вращающегося по этому многоугольнику.
- `paintEvent(QPaintEvent *)` — слот для отрисовки.
- `onTimer()` — слот реагирующий на событие от таймера. Предназначен для приращения угла поворота и для отрисовки движения окружности либо по прямой, либо вокруг угла многоугольника
- `around()` — процедура вычисления координат x и y центра окружности, осуществляющей вращение вокруг угла многоугольника.
- `straightLine()` — процедура вычисления координат x и y центра окружности, осуществляющей движение вдоль прямой.
- `DegreeToRadian(int degree)` — функция перевода градусов в радианы.
- `RadianToDegree(double radian)` — функция перевода радиан в градусы.
- `isRotateArea(int angle)` — функция, определяющая, находится ли угол в участке поворота.

Текст программы

Listing 1: widget.h

```
#ifndef WIDGET_H
#define WIDGET_H

#include <QWidget>
#include <QPainter>
#include <cmath>
#include "Sleeper.h"
#include <QTimer>
#include <QLabel>
#include <QVBoxLayout>
#include <QLineEdit>
#include <QPushButton>

class Widget : public QWidget
{
    Q_OBJECT
private:
    int R, r, n, x, y;
    QPolygon* polygon;
    int angle;
    int angleStep; // Шаг приращения угла шатуна
    int circleAngle; // Угол поворота колеса
    QLineEdit* REdit;
    QLineEdit* NEdit;
    QLineEdit* rEdit;
    double DegreeToRadian(int);
    double RadianToDegree(double);
    bool isRotateArea(int);
    void around();
    void straightLine();
    int getRotateArea(int);
    int getTreshold(); // Пороговое значение поворота вокруг угла
    int circleRotate(int);
public:
    Widget(QWidget *parent = 0);
    ~Widget();
    void calcPolygon();
    void drawScene(int x, int y);
protected:
    void paintEvent(QPaintEvent *);
public slots:
    void onTimer();
    void newValues(); // Смена радиусов и углов
};

#endif // WIDGET_H
```

Listing 2: widget.cpp

```
#include "widget.h"

void Widget::calcPolygon()
{
    polygon->clear();
    for (int i = 0; i < n; i++)
    {
        polygon->append(QPoint(cos(6.28*i/n)*R+width()/2, sin(6.28*i/n)*R+height()/2));
    }
}

Widget::Widget(QWidget *parent)
    : QWidget(parent)
{
    R = 50;
    r = 50;
    n = 3;
    polygon = new QPolygon();
    x = 0; y=0;
```

```
    angle = 0;
    circleAngle = 180;
    angleStep = 1;

    QTimer * myTimer = new QTimer(this);
    connect(myTimer, SIGNAL(timeout()), this, SLOT(onTimer()));
    myTimer->start(62);
    QWidget* controlWindow;
    controlWindow = new QWidget;
    QVBoxLayout* verticalLayout = new QVBoxLayout;

    // Радиус многоугольника
    QHBoxLayout* RHorLayout = new QHBoxLayout;
    QLabel* RLabel = new QLabel("Polygon radius:");
    RHorLayout->addWidget(RLabel);

    REdit = new QLineEdit(QString::number(R));
    RHorLayout->addWidget(REdit);
    verticalLayout->addLayout(RHorLayout);

    // Количество углов многоугольника
    QHBoxLayout* NHorLayout = new QHBoxLayout;
    QLabel* NLabel = new QLabel("Polygon angle count:");
    NHorLayout->addWidget(NLabel);

    NEdit = new QLineEdit(QString::number(n));
    NHorLayout->addWidget(NEdit);
    verticalLayout->addLayout(NHorLayout);

    // Радиус катящегося круга
    QHBoxLayout* rHorLayout = new QHBoxLayout;
    QLabel* rLabel = new QLabel("Polygon angle count:");
    rHorLayout->addWidget(rLabel);

    rEdit = new QLineEdit(QString::number(r));
    rHorLayout->addWidget(rEdit);
    verticalLayout->addLayout(rHorLayout);

    // Кнопка отправки значений
    QPushButton* ChangeValueButton = new QPushButton("Change values");
    verticalLayout->addWidget(ChangeValueButton);
    connect(ChangeValueButton, SIGNAL(clicked()), this, SLOT(newValues()));
    controlWindow->setLayout(verticalLayout);
    //RLabel.show();
    controlWindow->show();
}

Widget::~Widget()
{

}

void Widget::drawScene(int x, int y)
{
    QPainter painter(this);
    calcPolygon();
    painter.drawPolygon(*polygon);
    painter.setPen(Qt::blue);
    painter.drawText(width()/2, 50, "Degree: "+QString::number(angle));
    if (isRotateArea(angle))
    {
        painter.drawText(width()/2, 60, "Rotate area");
        painter.drawLine(x, y,
                         r*cos(DegreeToRadian(circleAngle - circleRotate(angle))) + x,
                         -r*sin(DegreeToRadian(circleAngle - circleRotate(angle)))+ y);

    }
    painter.drawEllipse(QPoint(x, y), r, r);
    painter.drawLine(width()/2, height()/2, x, y);
}

void Widget::paintEvent(QPaintEvent *)
{
    drawScene(x, y);
}

void Widget::onTimer()
```

```
{
    if (isRotateArea(angle))
        around();
    else straightLine();
    this->repaint();
    angle += angleStep;
    angle %= 360;
}

void Widget::around()
{
    int k = getRotateArea(angle);
    int rotateAngle = k*360/n;
    int sA = angle - rotateAngle;
    double drA = -DegreeToRadian(rotateAngle);
    double mA = DegreeToRadian(abs(sA));
    double asn = asin(R*sin(mA)/r);
    double oX = R + r*cos(mA + asn);
    double oY = r*sin(mA + asn);
    if (sA < 0) oY *= -1;
    x = oX*cos(drA) + oY*sin(drA) + width()/2;
    y = oY*cos(drA) - oX*sin(drA) + height()/2;
}

void Widget::straightLine()
{
    double partAngle = DegreeToRadian(180.0/n);
    double Rm = R + r/cos(partAngle);
    double remainderAngle = DegreeToRadian(angle%(360/n));
    int fullAngle = angle*n/360;
    double circlePart = 2*partAngle;
    double oldX = Rm*cos(partAngle)*cos(remainderAngle)/cos(partAngle - remainderAngle);
    double oldY = Rm*cos(partAngle)*sin(remainderAngle)/cos(partAngle - remainderAngle);
    double rotateConst = -fullAngle*circlePart;
    x = oldX*cos(rotateConst) + oldY*sin(rotateConst) + width()/2;
    y = oldY*cos(rotateConst) - oldX*sin(rotateConst) + height()/2;
}

double Widget::DegreeToRadian(int degree)
{
    return (3.141592653589793*degree)/180;
}

double Widget::RadianToDegree(double radian)
{
    return 180*radian/3.141592653589793;
}

bool Widget::isRotateArea(int angle)
{
    if (getRotateArea(angle) <= n) return true;
    return false;
}

int Widget::getRotateArea(int angle)
{
    int threshold = getTreshold();
    bool inRotateArea = false;
    int areaNumber = n+1;
    for (int k = 0; k <= n && !inRotateArea; k++)
        if (angle < (k*360/n + threshold) && angle > (k*360/n - threshold))
        {
            inRotateArea = true;
            areaNumber = k;
        }
    return areaNumber;
}

int Widget::getTreshold()
{
    double halfCirclePart = DegreeToRadian(180/n);
    return RadianToDegree(atan(r*sin(halfCirclePart)/(R + r*cos(halfCirclePart))));
}

int Widget::circleRotate(int phi)
{
    return phi + RadianToDegree(asin(R*sin(DegreeToRadian(phi))/r));
```

```
}

void Widget::newValues()
{
    R = REdit->text().toInt();
    r = rEdit->text().toInt();
    n = NEdit->text().toInt();
}
```

Listing 3: main.cpp

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

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

    return a.exec();
}
```

Литература

В процессе написания курсовой работы использовалась следующая литература:

1. Qt 4.8 Профессиональное программирование на C++. Макс Шлее. “БХВ-Петербург”. 2012.
2. Лекции по аналитической геометрии, пополненные необходимыми сведениями из алгебры. П.С. Александров. Издательство “Наука”. 1968.