**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

отчет

**по лабораторной работе №1**

**по дисциплине «Компьютерная графика»**

Тема: **Исследование математических методов представления и преобразования графических объектов на плоскости и в пространстве**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Степовик В.С.  Соболев М.С. |
| Преподаватель |  | Матвеева И.В. |

Санкт-Петербург

2022

Оглавление

[Цель работы 3](#_Toc98686003)

[Задание 3](#_Toc98686004)

[Используемые ресурсы 3](#_Toc98686005)

[Основные теоретические положения 4](#_Toc98686006)

[Пример работы программы 10](#_Toc98686007)

[Вывод 11](#_Toc98686008)

### **Цель работы**

Исследовать математические методы представления и преобразования графических объектов на плоскости и в пространстве.

### **Задание**

Сформировать отрезок, проведенный из произвольно расположенной  
точки на плоскости к заданной окружности, определив предварительно координаты точки касания. Необходимо предусмотреть возможность редактирования положения точки и параметры окружности.

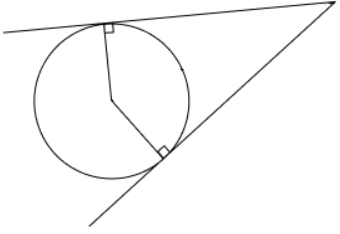
### Используемые ресурсы

Для выполнения лабораторной работы использовался язык C++ c использованием фреймворка Qt.

### Основные теоретические положения

Данную работу поделим на две теоретические составляющие: отрисовка графических объектов и формулы, задающие положение объектов.

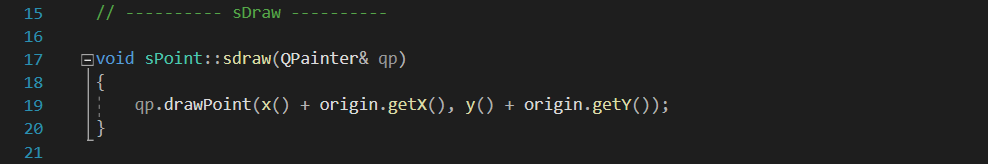
Для изображения окружности и касательных к ней из любой точки на плоскости, нам потребуются, соответственно, изображение окружности, точки и отрезков (касательных и радиусов, перпендикулярным им):



Ввиду того, что объекты строятся в плоской системе координат (а, значит, каждая точка определяется двумя переменными – x,y), задающие объекты уравнения и алгоритмы отрисовки выглядят следующим образом:

Точка:

Точка на плоскости рисуется очень просто: при помощи библиотечной функции drawPoint() фреймворка Qt, с передача в неё необходимых координат как параметров (на скрине так же присутствует смещение)

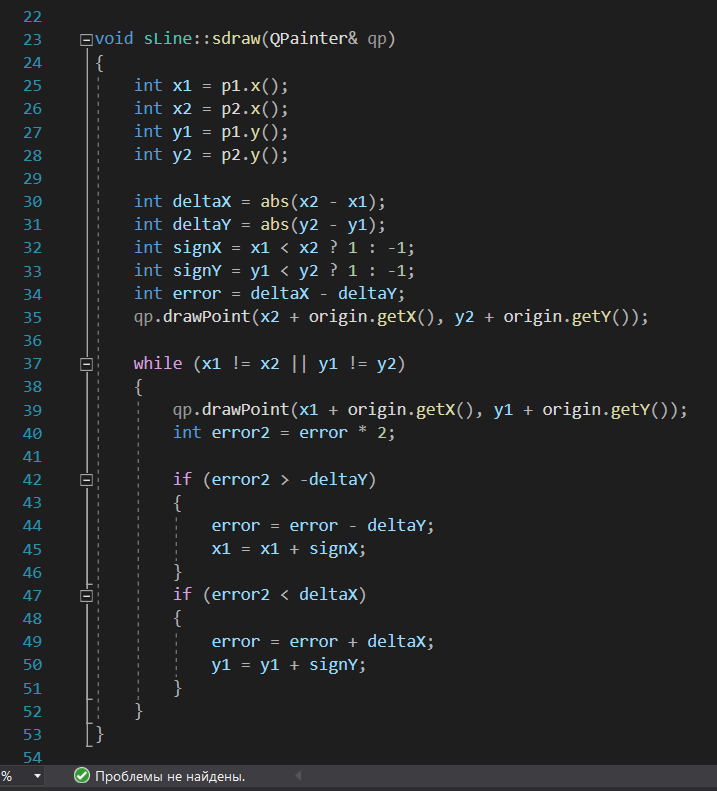


Отрезок:

Длинна задаётся уравнением на плоскости: , где – концы отрезка.

Алгоритм отрисовки отрезка заключается в определении положения концов отрезка относительно друг друга и поточечного смещения от одного конца к другому.

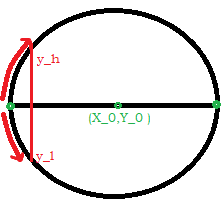
Реализация вышеописанного алгоритма на языке C++ с использованием библиотек фреймворка Qt:

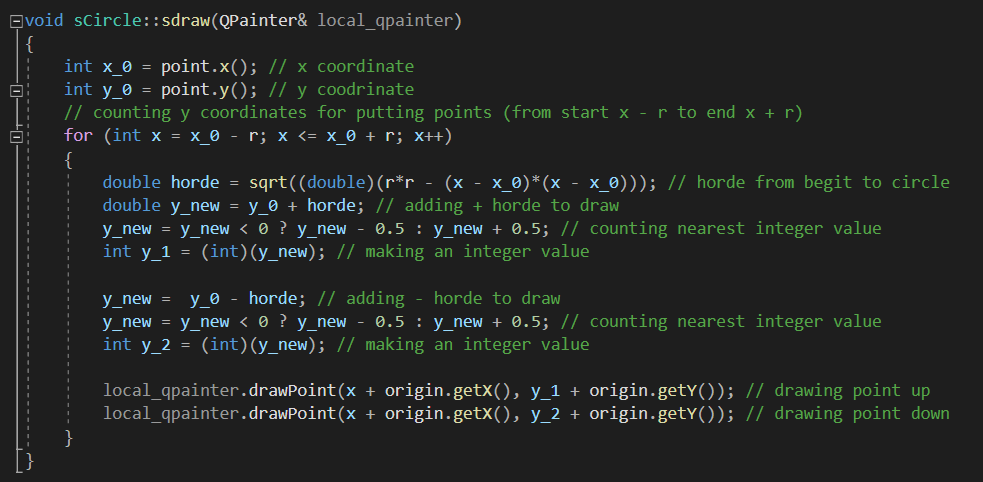


Окружность:

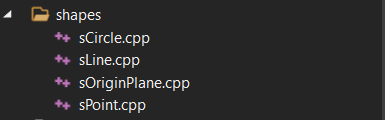
Задаётся уравнением на плоскости , где R – радиус, а () – координаты центра окружности.

По сути, алгоритм вывода окружности на экран представляет собой пошаговую отрисовку её верхней и нижней полуокружностей от до , при том на каждом шаге вычисляются y\_h и y\_l, после чего получаем две точки верхней и нижней полуокружностей соответственно: (x; y\_h), (x; y\_l):

**



Представленные выше алгоритмы являются методами соответствующих классов:



Теперь выведем формулу, по которой будут определятся точки касания.

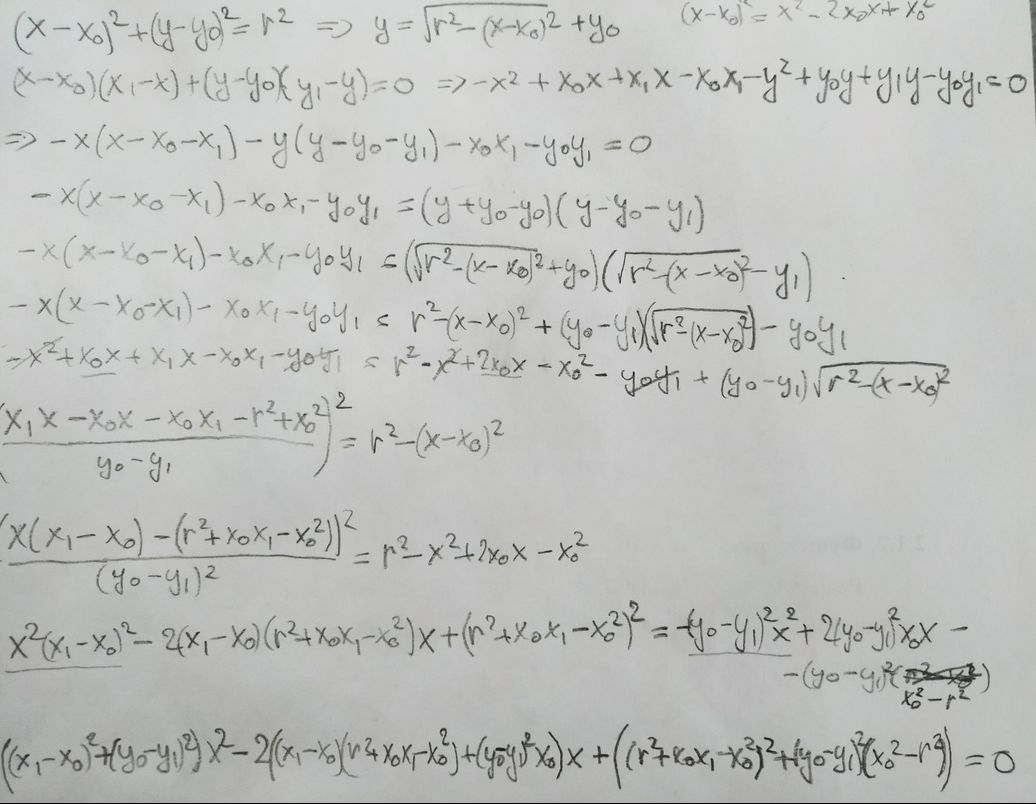
Для этого воспользуемся следующими данными: т.к. точки касания лежат на окружности, а сами касательные перпендикулярны радиусам, то справедлива система уравнений, состоящая из уравнения окружности и скалярного произведения векторов, определяющих радиус и касательную.

,

Где

*.*

Ниже представленно решение системы уравнений сведённой к квадратному уравнению относительно x:



Таким образом, алгоритм нахождения точек касания сводится к подсчёту коэфицентов и решении квадратного уравнения с полседующей подстановкой результата в формулу .

Реализация :

sOriginPlane cb(0, 0);

sPoint xo1 = sPoint(x01, y01, cb);

sPoint xo2 = sPoint(x02, y02, cb);

sPoint line11 = sPoint(0, 0, cb);

sPoint line12 = sPoint(0, 0, cb);

sPoint line21 = sPoint(0, 0, cb);

sPoint line22 = sPoint(0, 0, cb);

sPoint line31 = sPoint(0, 0, cb);

sPoint line32 = sPoint(0, 0, cb);

sPoint line41 = sPoint(0, 0, cb);

sPoint line42 = sPoint(0, 0, cb);

float r = r1;

float x\_0 = x01; // center of circle

float y\_0 = y01; // center of circle

float x\_1 = x02;

float y\_1 = y02;

float point\_1[] = { 0,0 };

float point\_2[] = { 0,0 };

float a = (x\_1 - x\_0) \* (x\_1 - x\_0) + (y\_0 - y\_1) \* (y\_0 - y\_1);

float b = -2 \* ((x\_1 - x\_0) \* (r \* r + x\_0 \* x\_1 - x\_0 \* x\_0) + x\_0 \* (y\_0 - y\_1) \* (y\_0 - y\_1));

float c = (r \* r + x\_0 \* x\_1 - x\_0 \* x\_0) \* (r \* r + x\_0 \* x\_1 - x\_0 \* x\_0) + (y\_0 - y\_1) \* (y\_0 - y\_1) \* (x\_0 \* x\_0 - r \* r);

float D = b \* b - 4 \* a \* c;

if (D > 0)

{

// x0 -- circle

// x1 -- dot

float y\_h, y\_l;

point\_1[0] = (sqrt(D) - b)/(2\*a);

y\_h = y\_0 + sqrt( r\*r - (point\_1[0] - x\_0)\*(point\_1[0] - x\_0) ); // y high -- "+" square root

y\_l = y\_0 - sqrt( r\*r - (point\_1[0] - x\_0)\*(point\_1[0] - x\_0) ); // y low -- "-"" square root

if(((point\_1[0] - x\_0)\*(x\_1 - point\_1[0]) + (y\_l - y\_0)\*(y\_1 - y\_l))==0)

{

point\_1[1] = y\_l;

}

else if (((point\_1[0] - x\_0)\*(x\_1 - point\_1[0]) + (y\_h - y\_0)\*(y\_1 - y\_h))==0)

{

point\_1[1] = y\_h;

}

else // abs (number module) is necessary

{

if (abs(((point\_1[0] - x\_0)\*(x\_1 - point\_1[0]) + (y\_l - y\_0)\*(y\_1 - y\_l)))

>= abs(((point\_1[0] - x\_0)\*(x\_1 - point\_1[0]) + (y\_h - y\_0)\*(y\_1 - y\_h)))) // abs -- absolute

{

point\_1[1] = y\_h; // if y\_l >= y\_h => setting y\_h (measurement error is less)

}

else // final else

{

point\_1[1] = y\_l;

}

}

point\_2[0] = (-sqrt(D) - b)/(2\*a);

y\_h = y\_0 + sqrt( r\*r - (point\_2[0] - x\_0)\*(point\_2[0] - x\_0) );

y\_l = y\_0 - sqrt( r\*r - (point\_2[0] - x\_0)\*(point\_2[0] - x\_0) );

if(((point\_2[0] - x\_0)\*(x\_1 - point\_2[0]) + (y\_l - y\_0)\*(y\_1 - y\_l))==0)

{

point\_2[1] = y\_l;

}

else if (((point\_2[0] - x\_0)\*(x\_1 - point\_2[0]) + (y\_h - y\_0)\*(y\_1 - y\_h))==0)

{

point\_2[1] = y\_h;

}

else // abs (number module) is necessary

{

if (abs(((point\_2[0] - x\_0)\*(x\_1 - point\_2[0]) + (y\_l - y\_0)\*(y\_1 - y\_l)))

>= abs(((point\_2[0] - x\_0)\*(x\_1 - point\_2[0]) + (y\_h - y\_0)\*(y\_1 - y\_h))))

{

point\_2[1] = y\_h; // if y\_l >= y\_h => setting y\_h (measurement error is less)

}

else // final else

{

point\_2[1] = y\_l;

}

}

}

else if (D == 0)

{

point\_1[0] = -b / (2 \* a);

point\_1[1] = y\_0 + sqrt(r \* r - (point\_1[0] - x\_0) \* (point\_1[0] - x\_0));

point\_2[0] = point\_1[0];

point\_2[1] = y\_0 - sqrt(r \* r - (point\_1[0] - x\_0) \* (point\_1[0] - x\_0));

}

// setting values for sending to other function

float res[4] = { point\_1[0],point\_1[1],point\_2[0],point\_2[1] };

line21.setX(point\_1[0]);

line21.setY(point\_1[1]);

line22.setX(x02);

line22.setY(y02);

line31.setX(point\_2[0]);

line31.setY(point\_2[1]);

line32.setX(x02);

line32.setY(y02);

### Пример работы программы

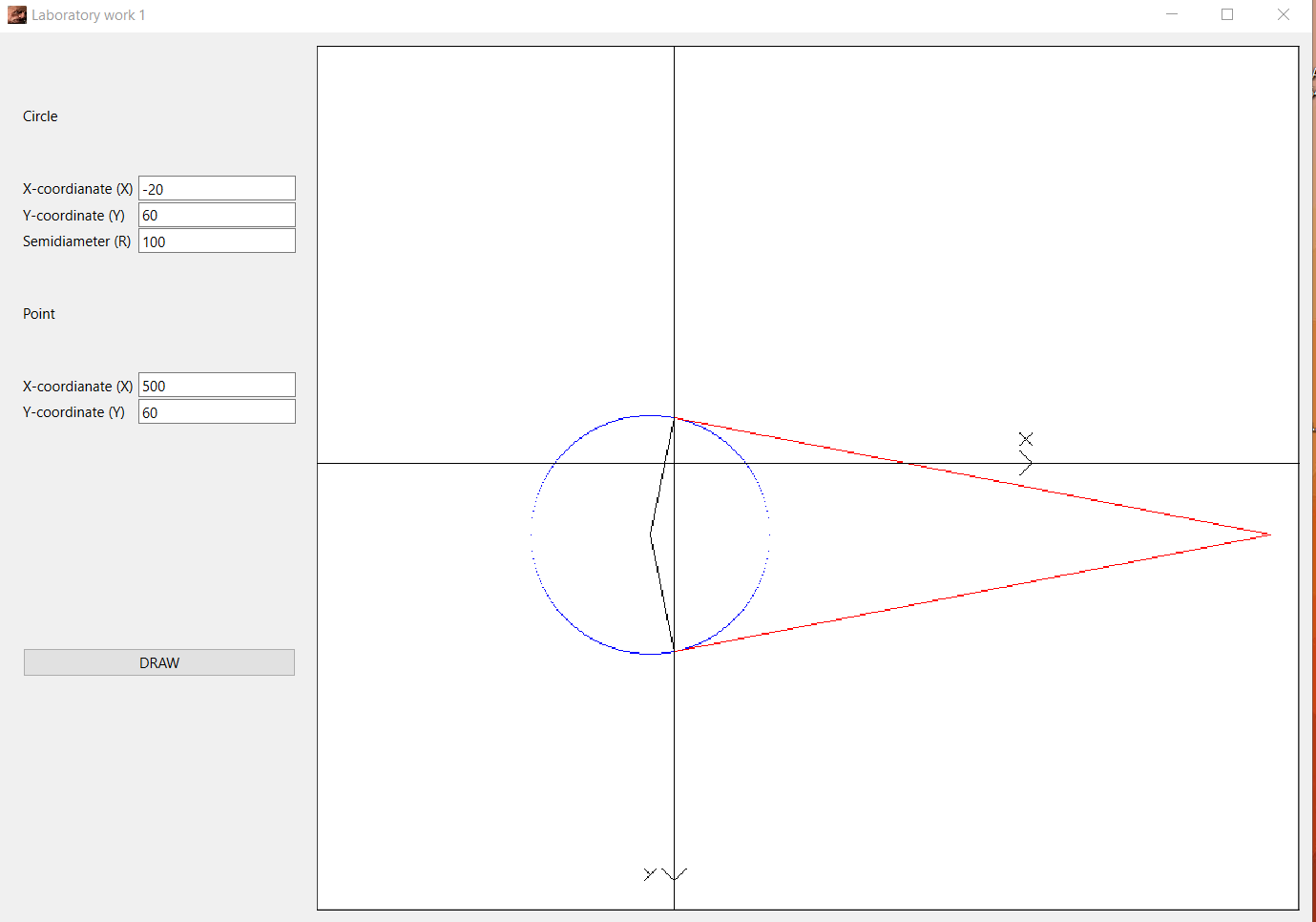
Пример работы программы представлен на рисунках ниже: 

Рисунок 1. Начальное окно

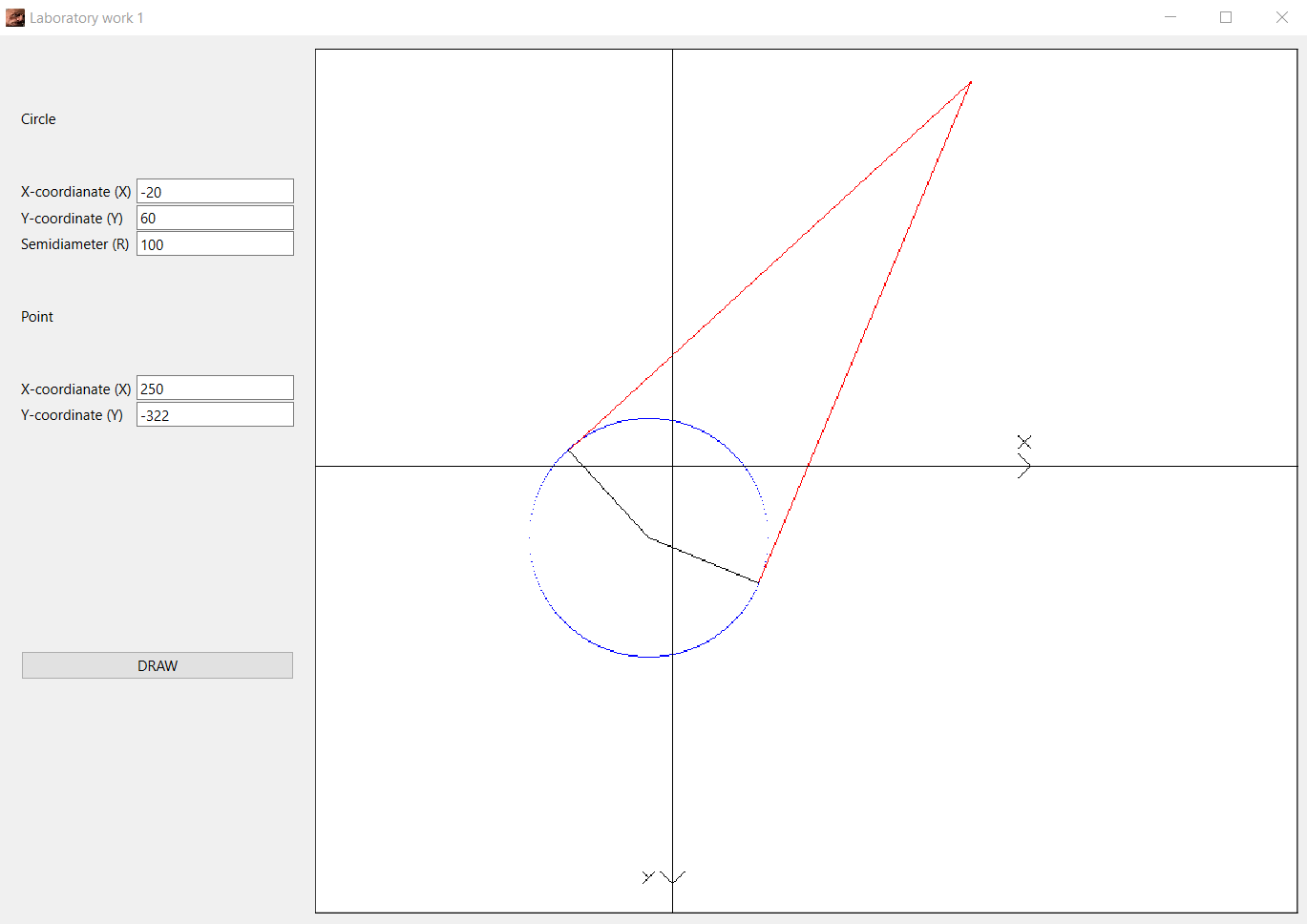


Рисунок 2. Результат изменения координат точки

### Вывод

При выполнении лабораторной работы изучены базовые преобразования графических объектов на плоскости. В частности, реализован механизм отрисовки касательной из любой точки плоскости к окружности различного диаметра.