



МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



**НГТУ
НЭТИ** | **Факультет прикладной
математики и информатики**

Кафедра прикладной математики

Практическая работа №1
по дисциплине «Компьютерная графика»

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ OpenGL

Студенты	ГРОСС АЛЕКСЕЙ ШИШКИН НИКИТА
Вариант	3
Группа	ПМ-92
Преподаватель	ЗАДОРОВ Ж. А. Г.

Новосибирск, 2022

Цель работы

Ознакомиться с основами использования библиотеки OpenGL и работе с примитивами.

Вариант: тип примитивов GL_LINE_STRIP.

Практическая часть

1. Отобразить в окне множество примитивов (вершины которых задаются кликами мыши) в соответствии с вариантом задания.
2. Для завершения текущего (активного) набора (множества) примитивов и начала нового зарезервировать специальную клавишу (пробел или правый клик).
3. Для текущего набора примитивов предоставить возможность изменения цвета и координат его вершин.
4. Предусмотреть возможность удаления последнего примитива и последнего набора примитивов.

Дополнительные задания:

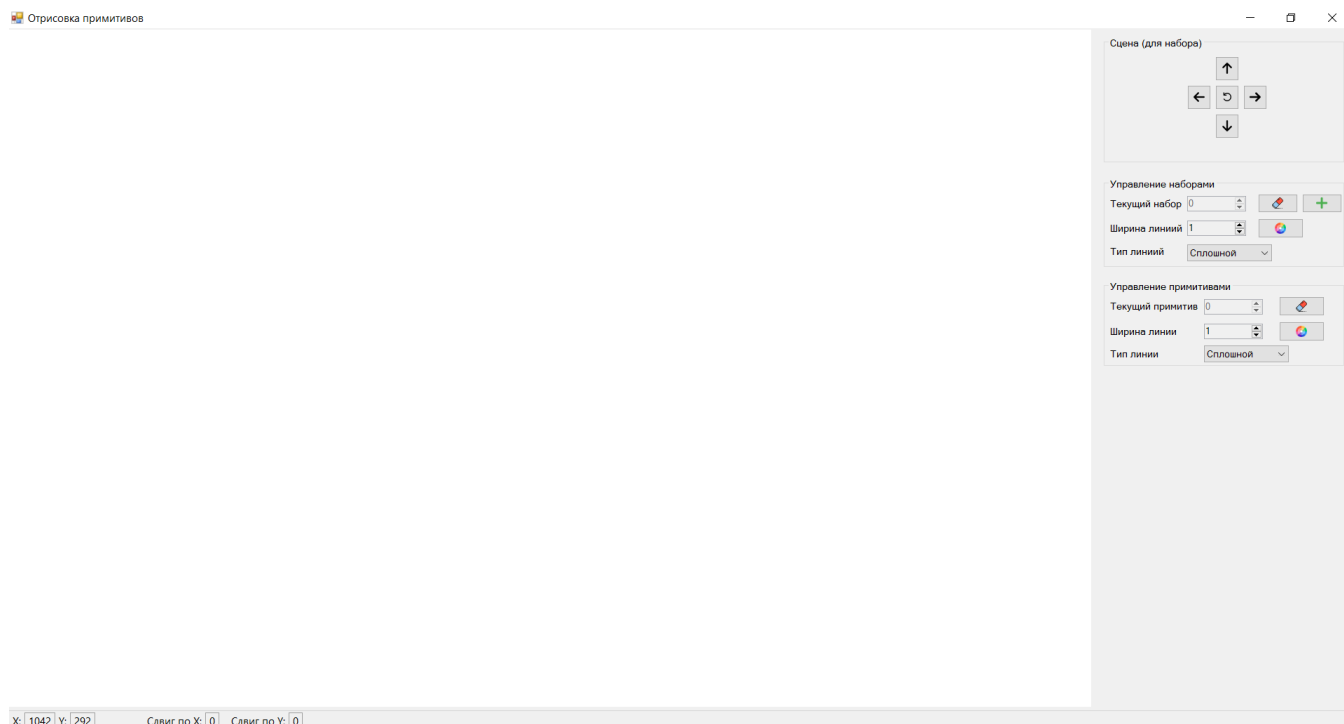
- изменение не только координат и цвета вершин примитивов, но и шаблона закрашивания и т.д.;
- изменение параметров (в том числе и удаление) не только текущего набора, но произвольного;
- изменение параметров произвольного примитива в наборе.

Руководство пользователя

Введение

Окно приложения разделено на две части: левая часть – **рабочая область**, правая – **панель управления**. Рабочая область предназначена для отрисовки наборов примитивов.

Общий вид окна:



Панель управления состоит из нескольких элементов:

- мини-панель управления сцены, руководясь которой пользователь может передвигать конкретный набор примитивов;
- мини-панель управления наборами примитивов;
- мини-панель управления примитивами;

В нижней части окна приложения отображаются текущие координаты курсора мыши и текущий сдвиг системы координат для выбранного набора примитивов.

Работа с мышью в рабочей области приложения


Пользователь в рабочей области левым кликом мыши может задавать вершины, которые, в свою очередь, будут преобразовываться в примитив. Правый клик предназначен для того, чтобы прекратить отрисовку текущего примитива.

Основные элементы панели управления и команды

Управление сценой

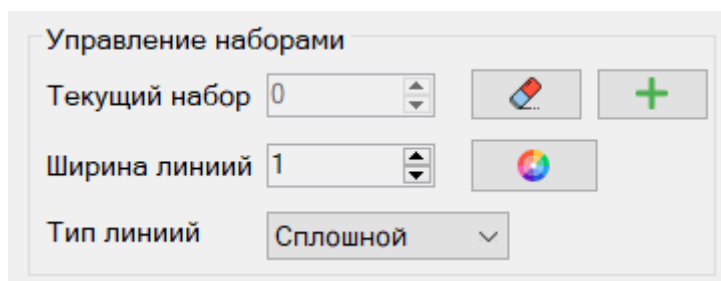
Вид мини-панели:



Данный элемент панели управления представляет собой набор из 5 кнопок, а именно стрелок, при нажатии которых смещается начало системы координат в выбранном направлении и кнопки возврата , при нажатии которой система координат вернется в исходное положение.

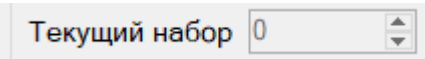
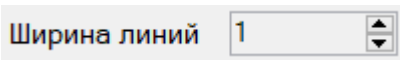
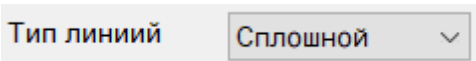
Управление наборами




Вид мини-панели:



Данный элемент панели управления представляет собой управление наборами в рабочей области приложения.

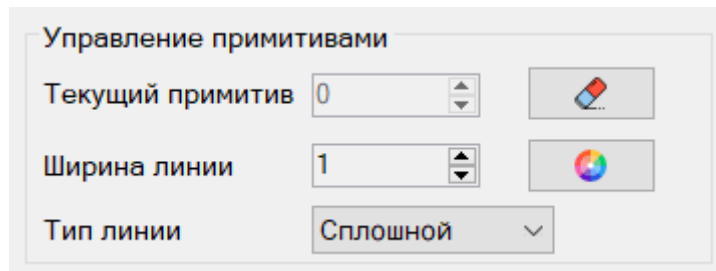
Элементы управления:

- числовое поле , с помощью которого можно переключаться между наборами примитивов;
- числовое поле , с помощью которого можно изменять ширину линий набора примитивов (от 1 до 5);
- выпадающий список , с помощью которого можно выбрать тип линий набора примитивов (сплошной, точечный, штриховой, штрихпунктирный);

- кнопка  для удаления текущего набора примитивов;
- кнопка  для добавления нового набора примитивов;
- кнопка  для открытия диалогового окна выбора цвета набора примитивов.

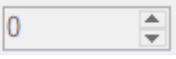
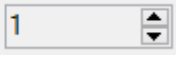
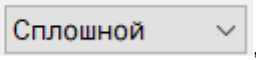


Управление примитивами

Вид мини-панели:



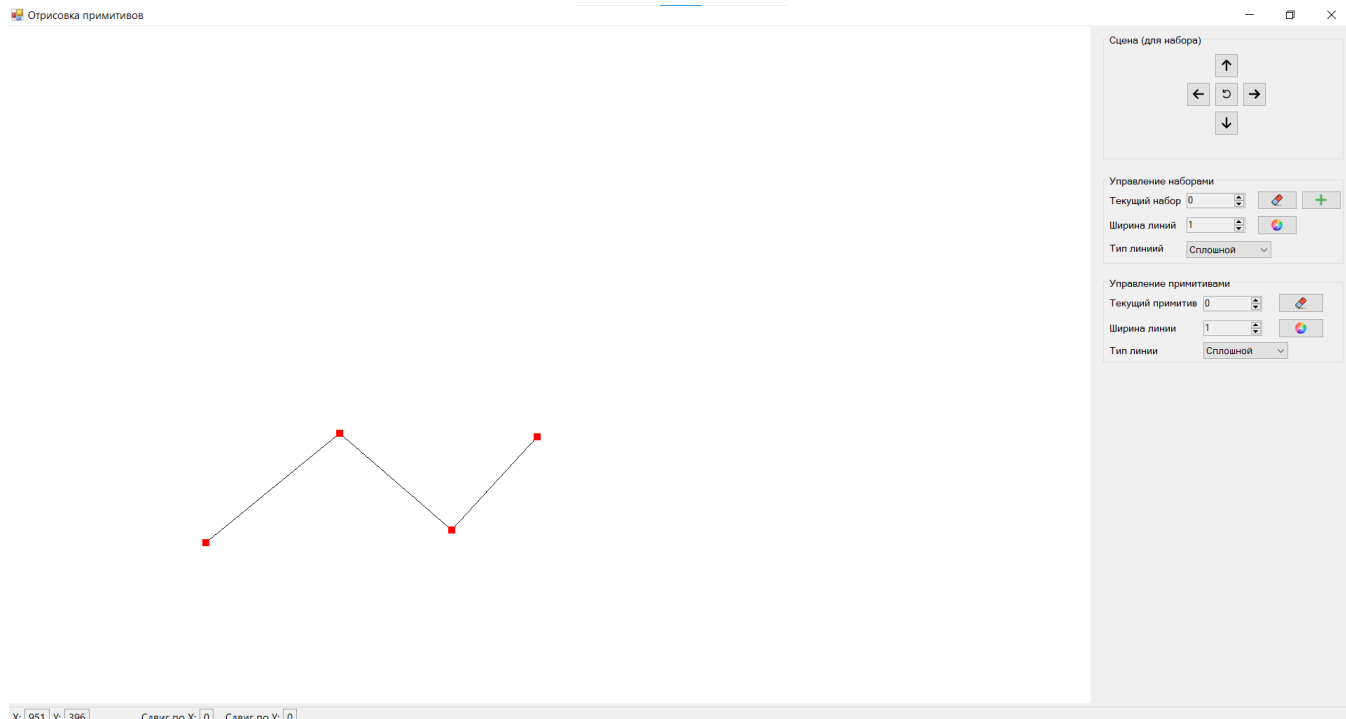
Данный элемент панели управления представляет собой управление примитивами в наборе в рабочей области приложения.

Элементы управления:

- числовое поле **Текущий примитив** , с помощью которого можно переключаться между примитивами в наборе;
- числовое поле **Ширина линии** , с помощью которого можно изменять ширину линий выбранного примитива (от 1 до 5);
- выпадающий список **Тип линии** , с помощью которого можно выбрать тип линий выбранного примитива (сплошной, точечный, штриховой, штрихпунктирный);
- кнопка  для удаления текущего примитива из набора;
- кнопка  для открытия диалогового окна выбора цвета текущего примитива.

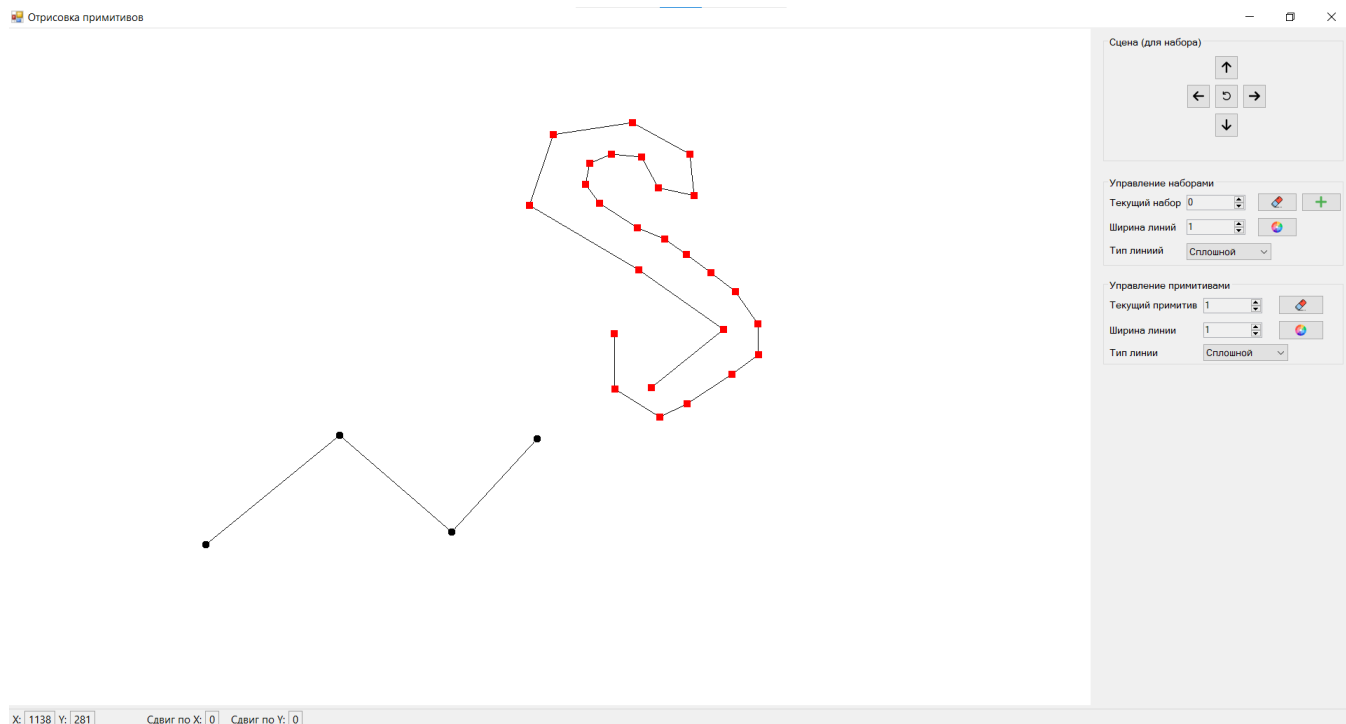
Тестирование программы

Отрисовка простого примитива

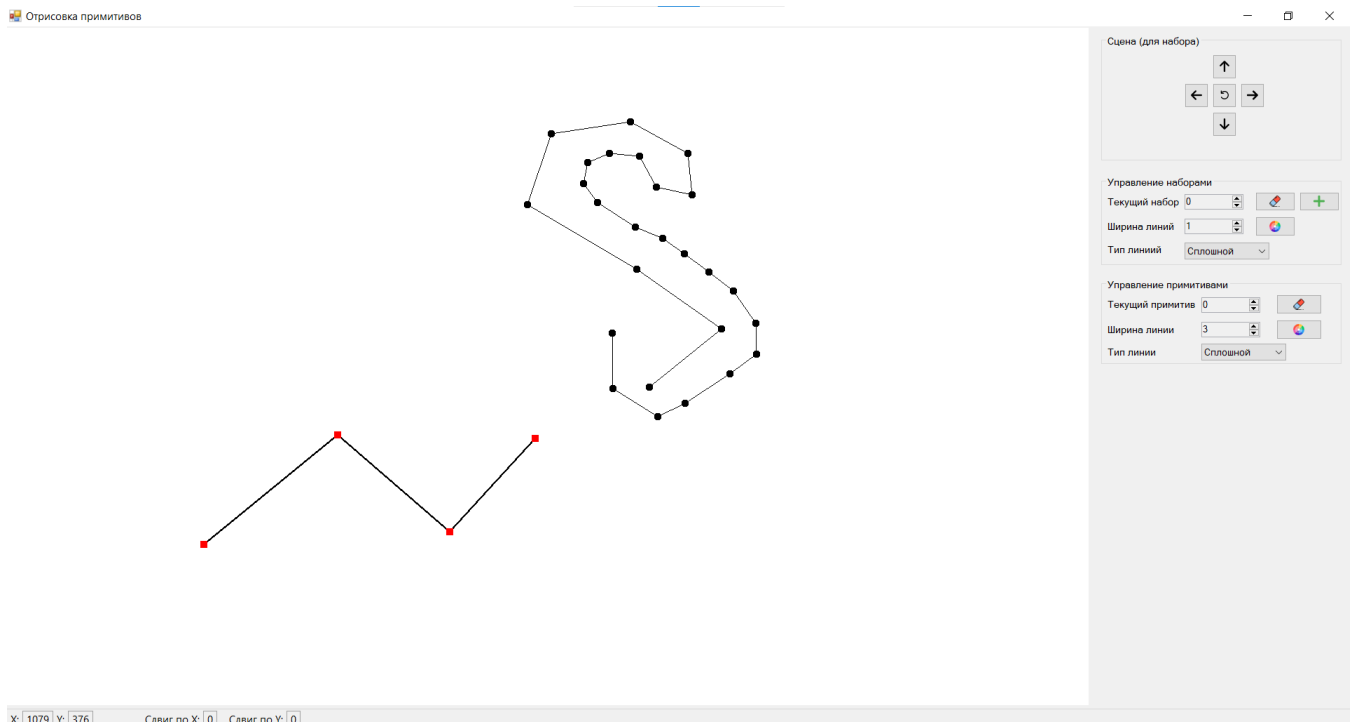


Добавление нового примитива

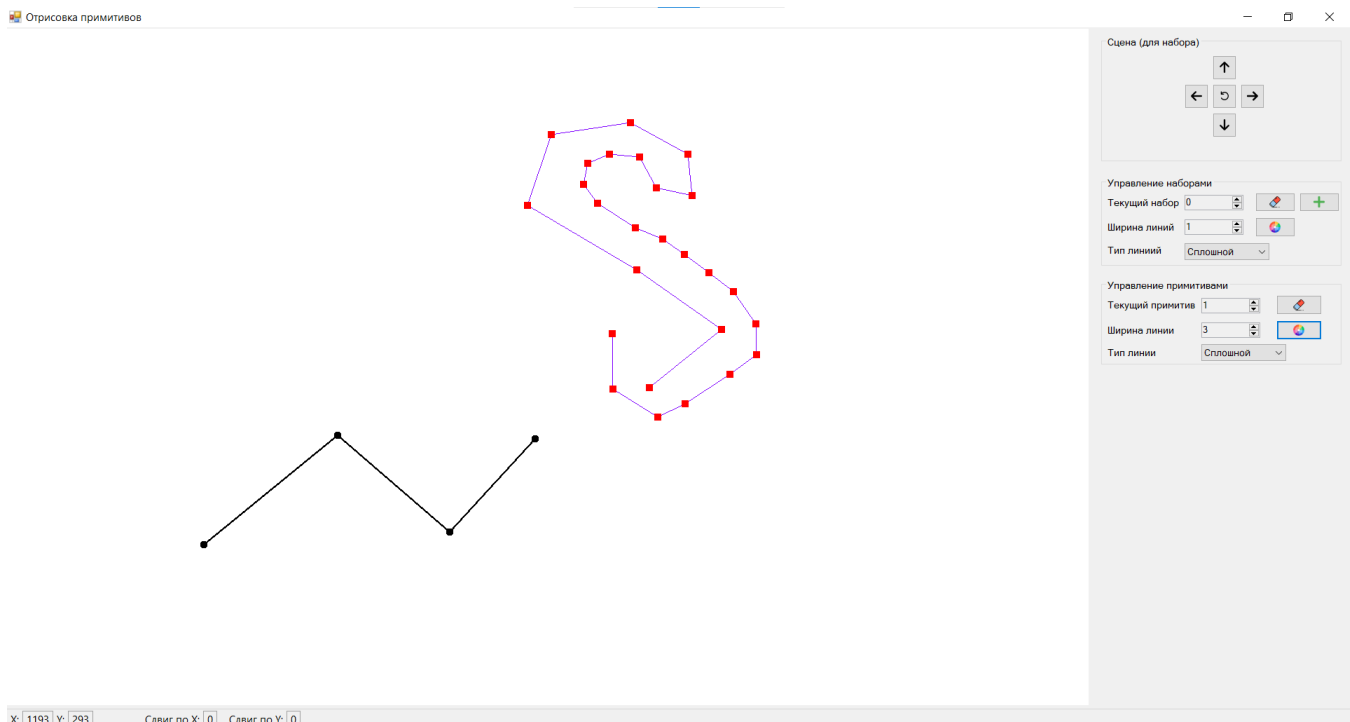
Текущий (новый) примитив выделен красными точками.



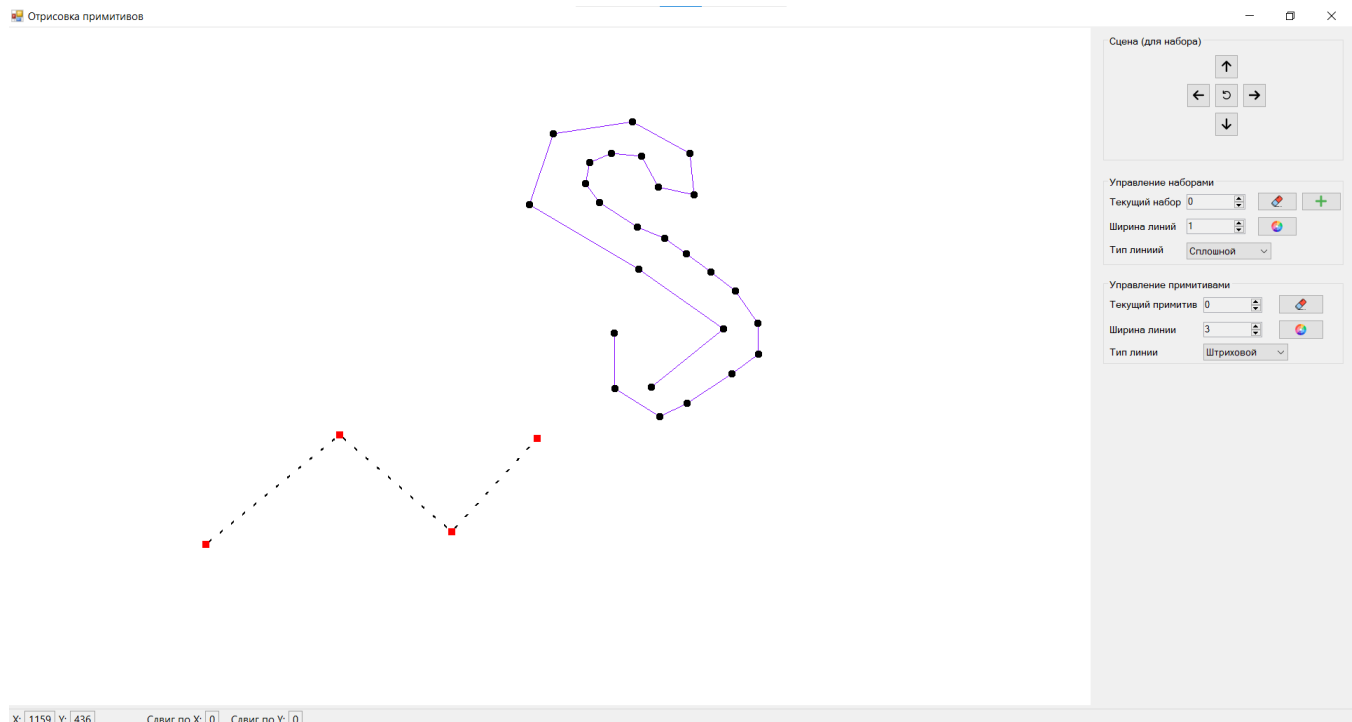
Изменение ширины линии выбранного примитива



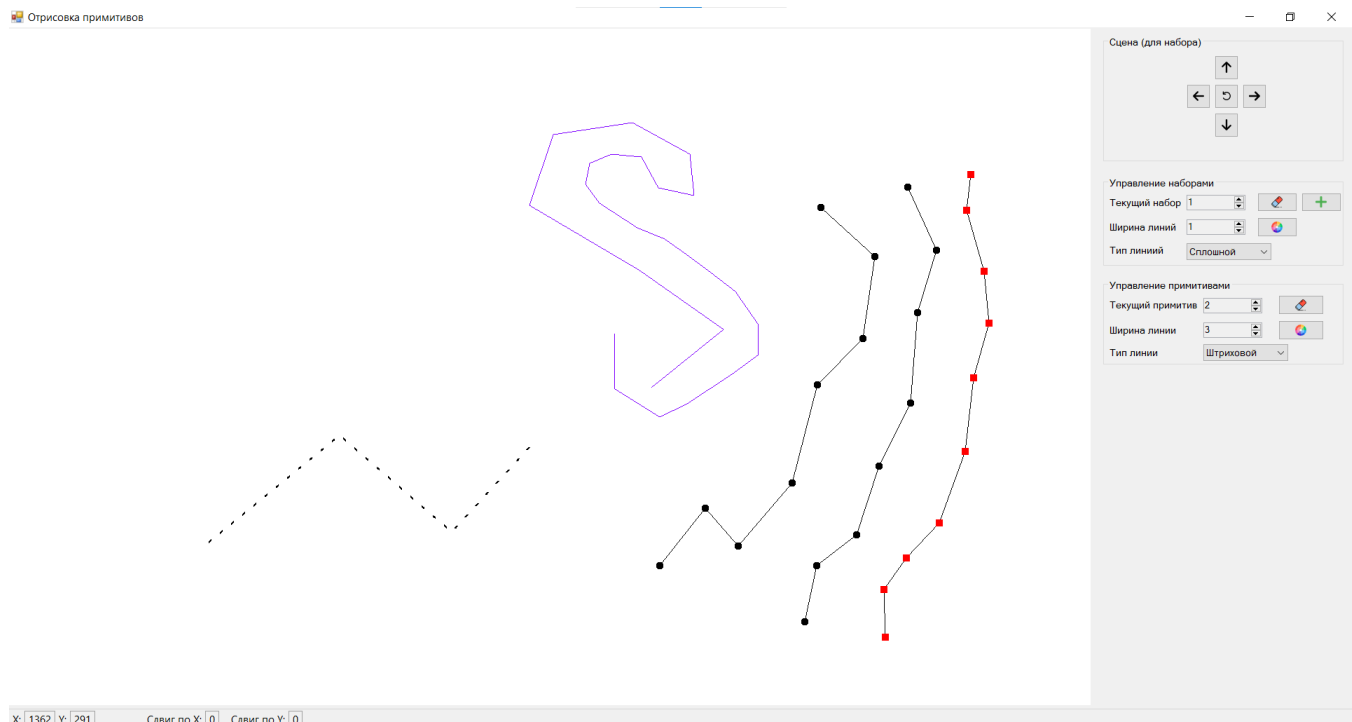
Изменение цвета выбранного примитива



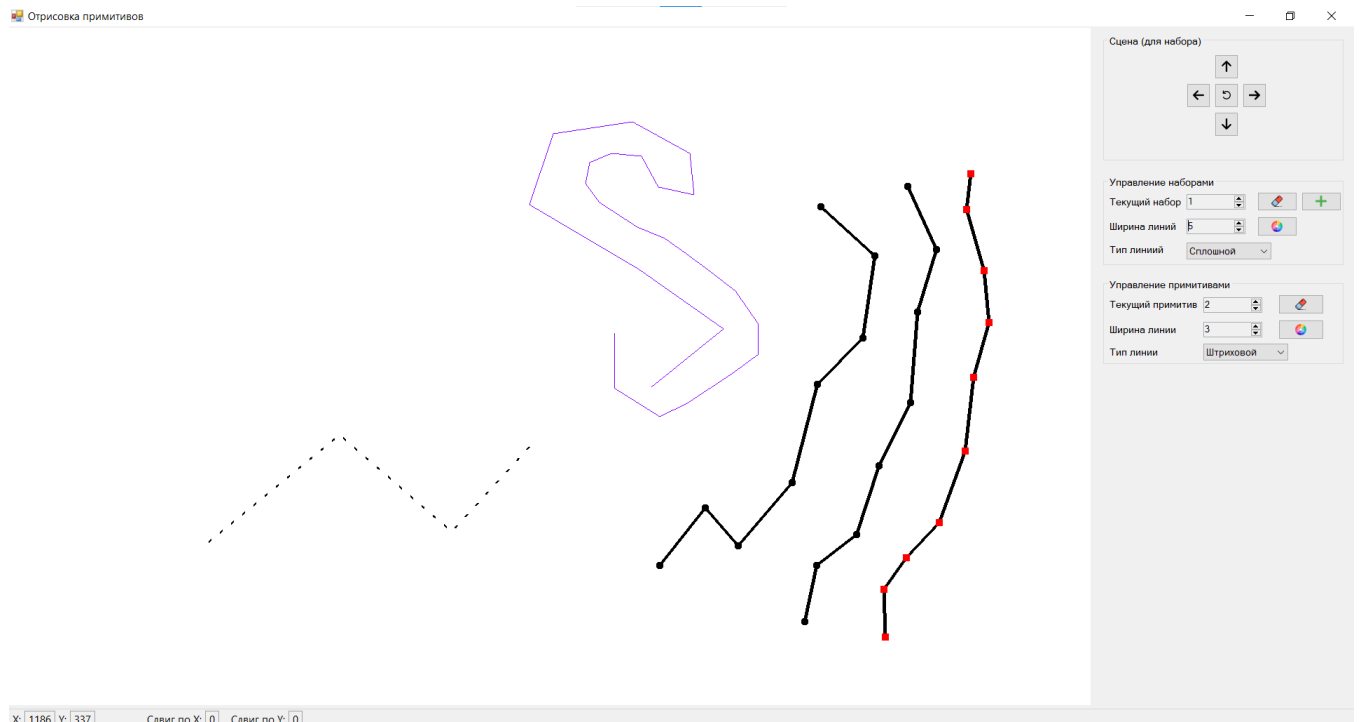
Изменение типа линии выбранного примитива



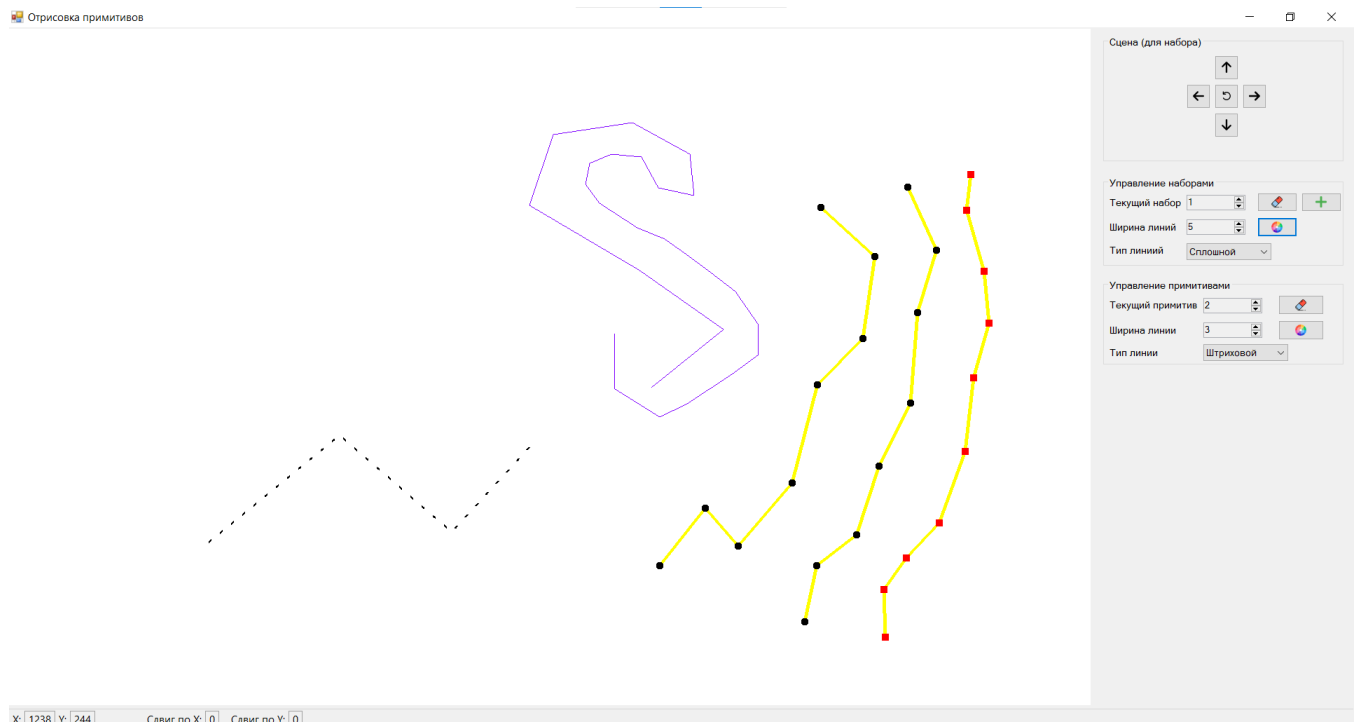
Создание нового набора примитивов



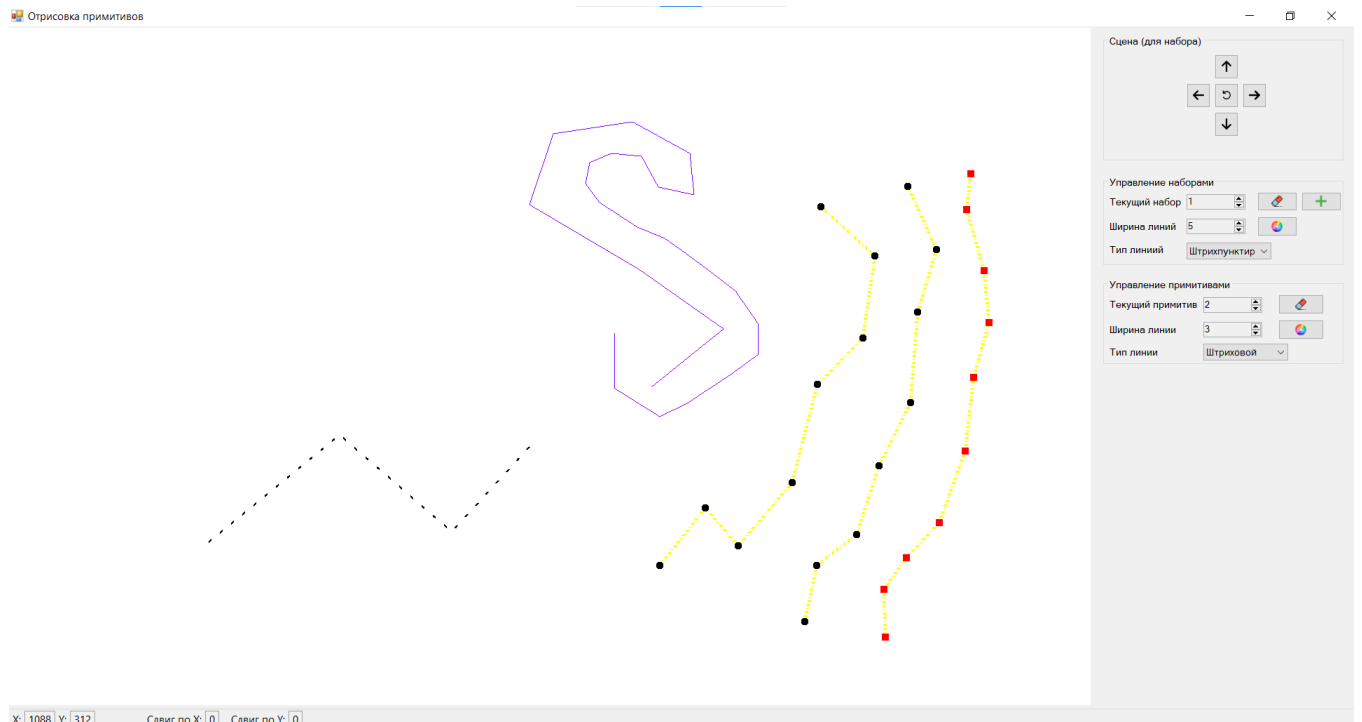
Изменение ширины линий выбранного набора примитивов



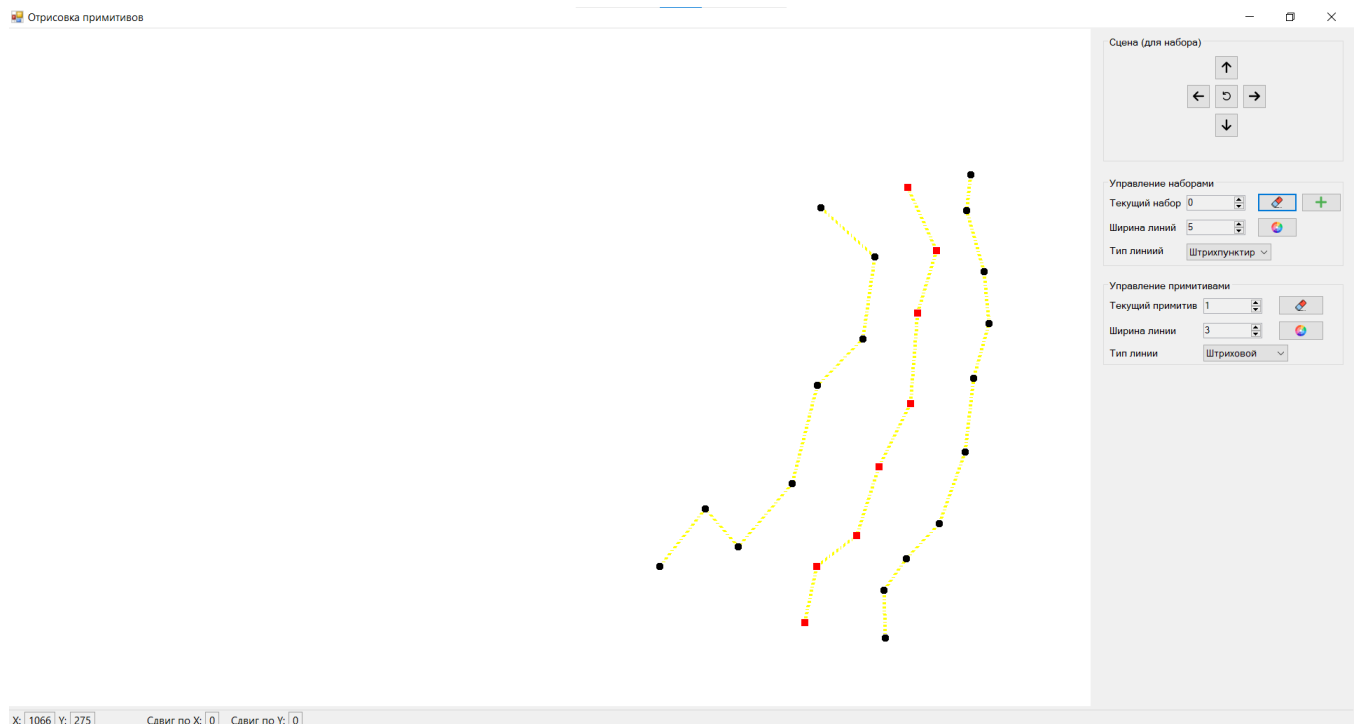
Изменение цвета выбранного набора примитивов



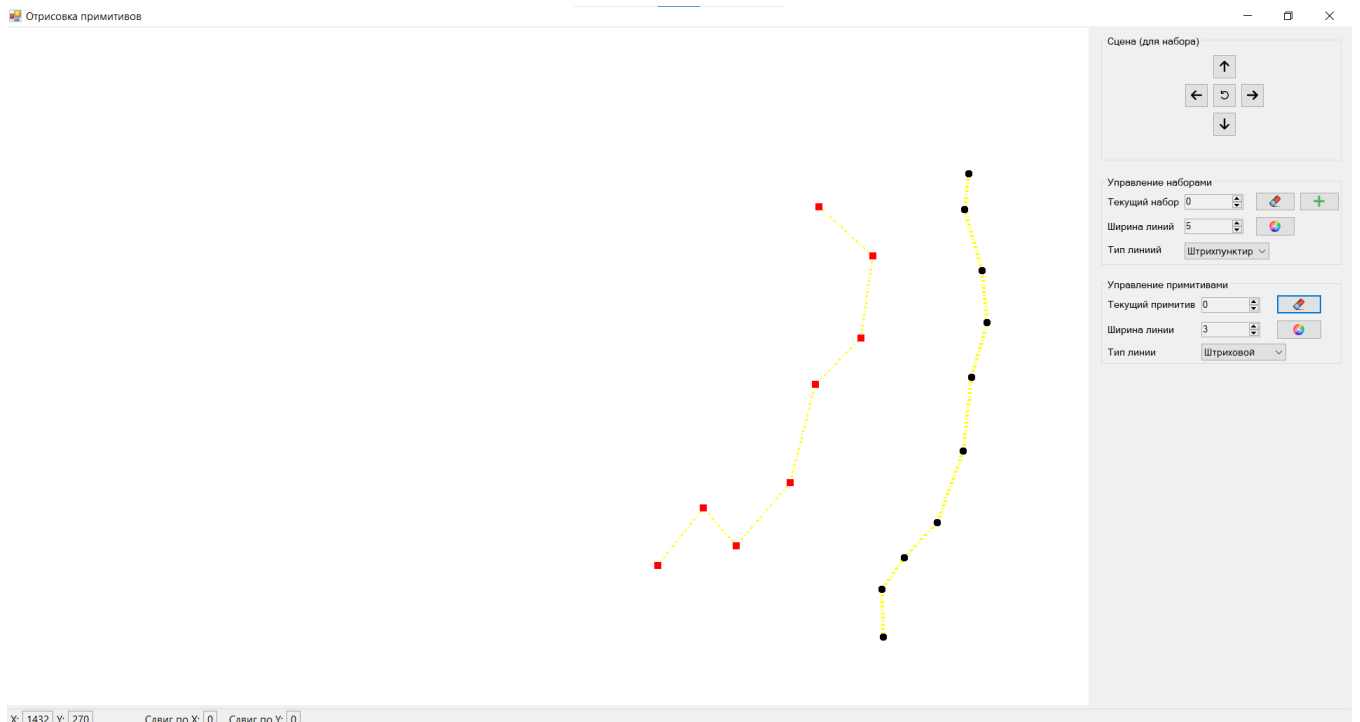
Изменение типа линий выбранного набора



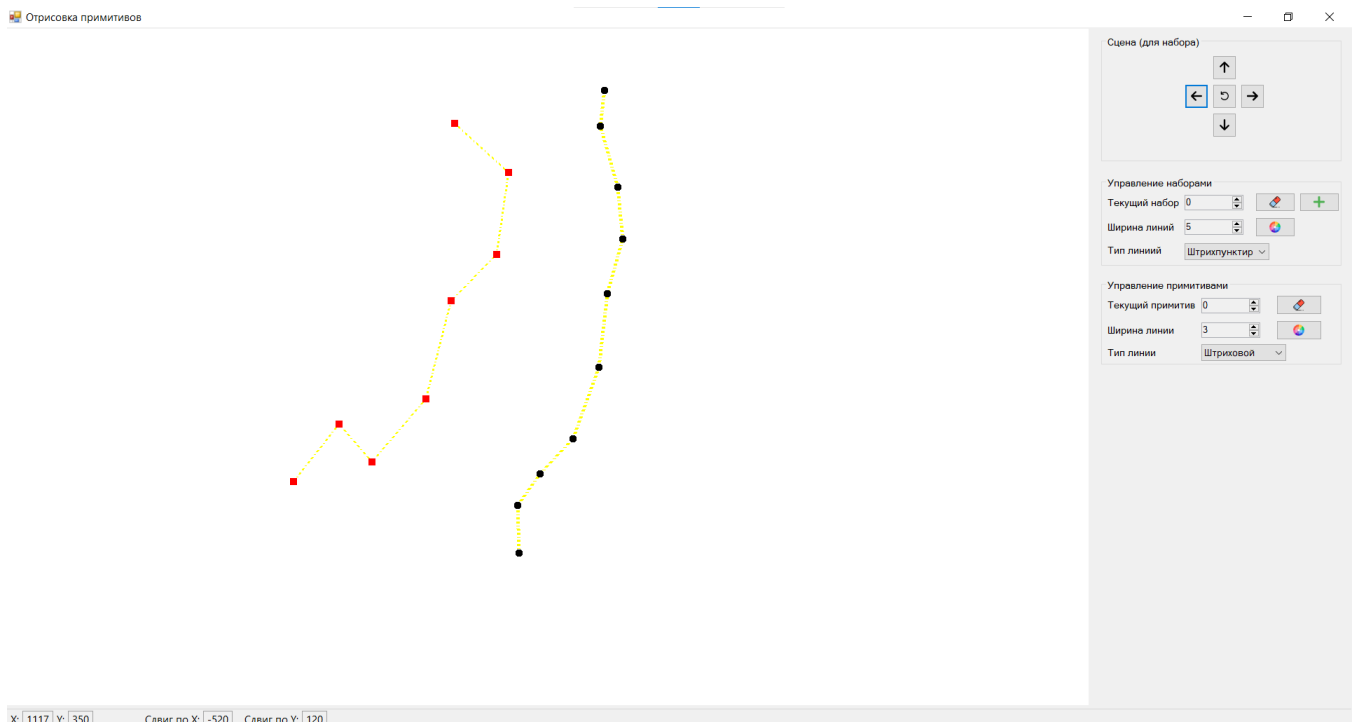
Удаление текущего набора



Удаление выбранного примитива в наборе



Изменение положения выбранного набора примитивов

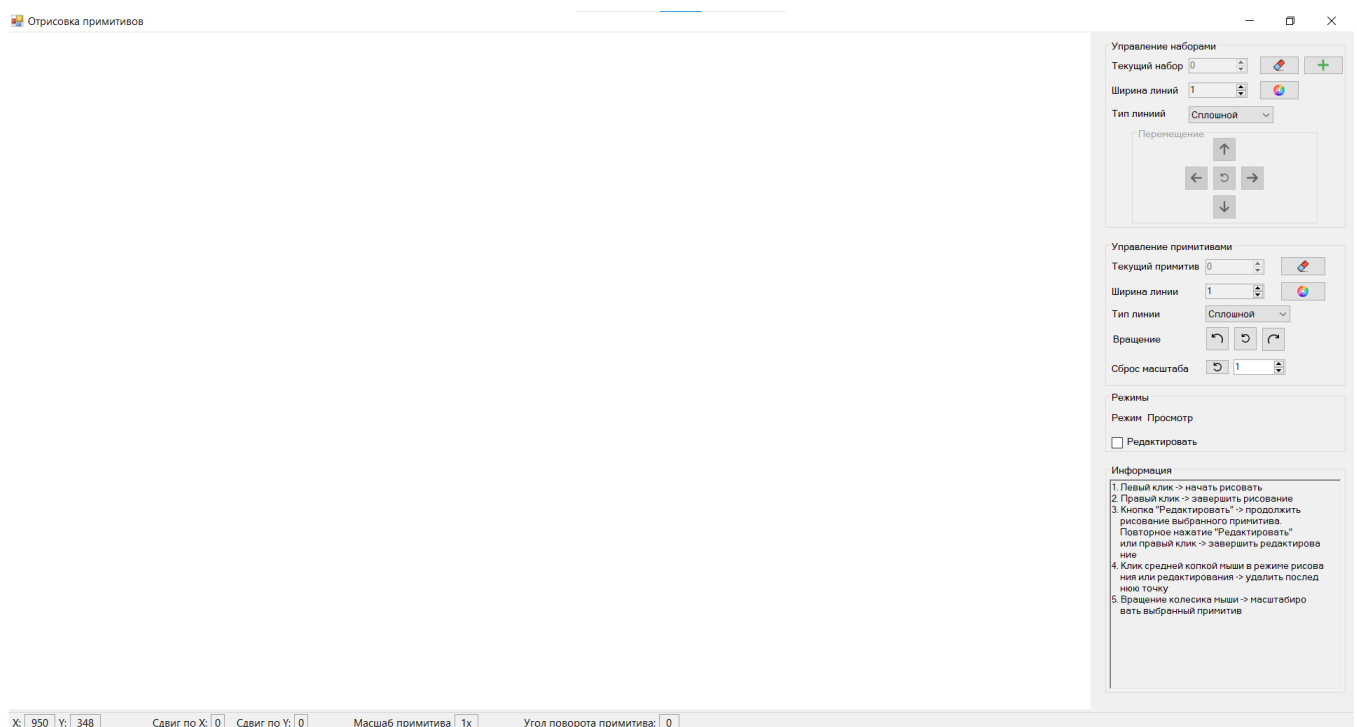


Нововведения в программе

Нововведения:

1. Обновление панели управления приложения.
2. Добавлены режимы "Просмотр", "Рисование" и "Редактирование".
3. Добавлена кнопка редактирования выбранного примитива.
4. Добавлено отображение угла поворота и масштаба текущего примитива.
5. Добавлена возможность вращать и масштабировать примитив.
6. Добавлена возможность удалить последнюю точку при редактировании.
7. Добавлено информационное поле с инструкцией использования программы.

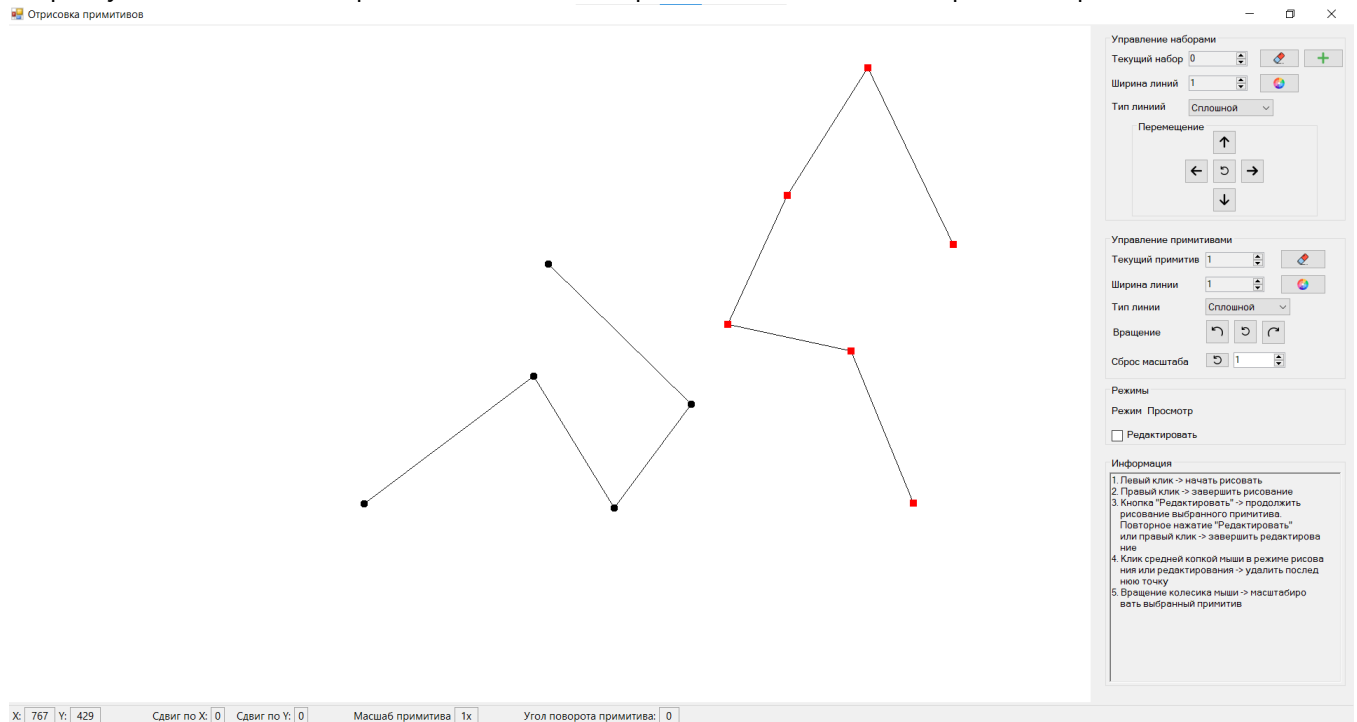
Обновленный вид окна:



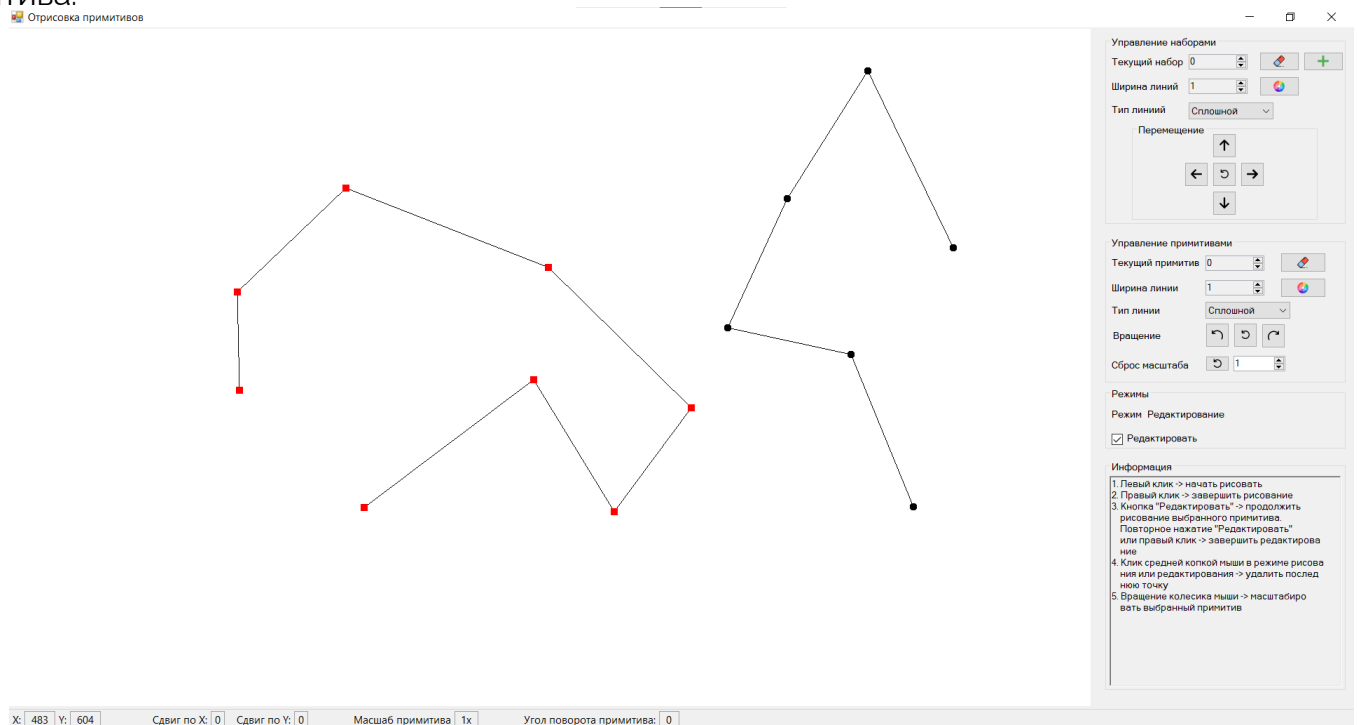
Тестирование нововведений

Редактирование выбранного примитива

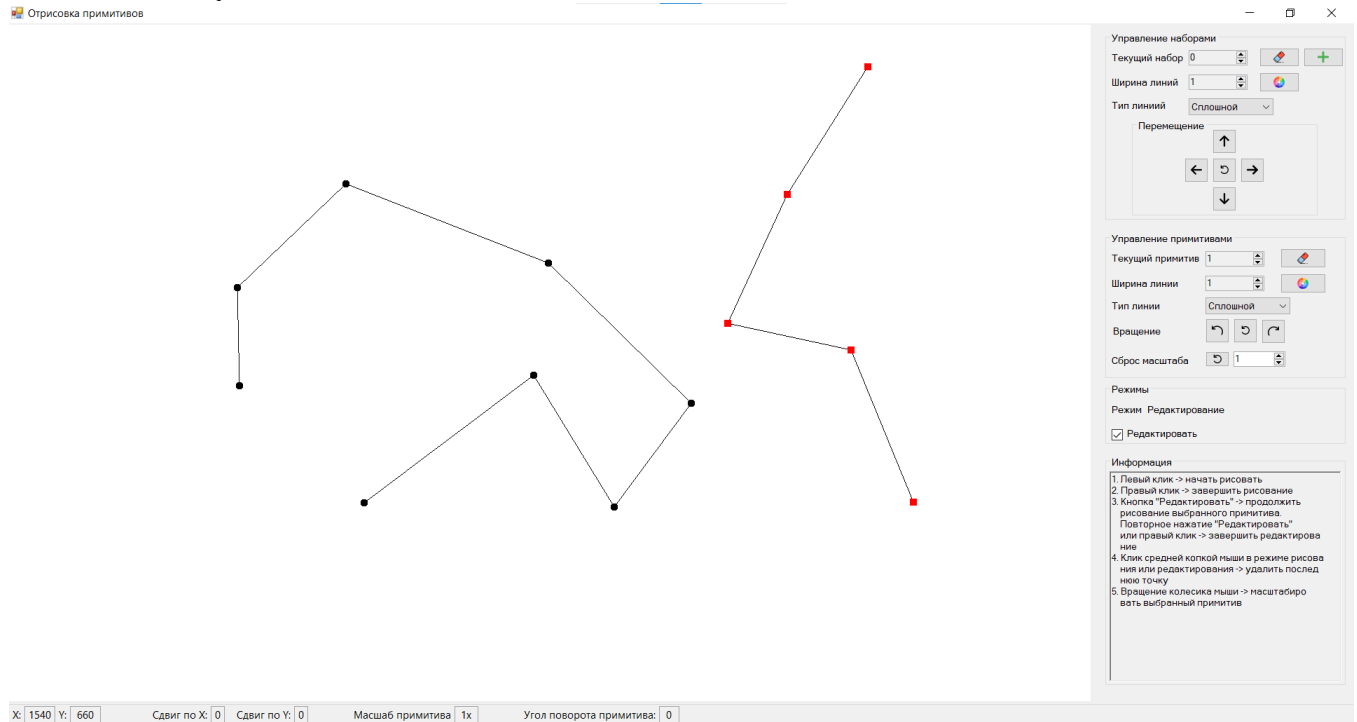
Нарисуем несколько примитивов и выберем один из них для редактирования.



Далее, нажимая кнопку "Редактировать", продолжаем рисование текущего примитива.

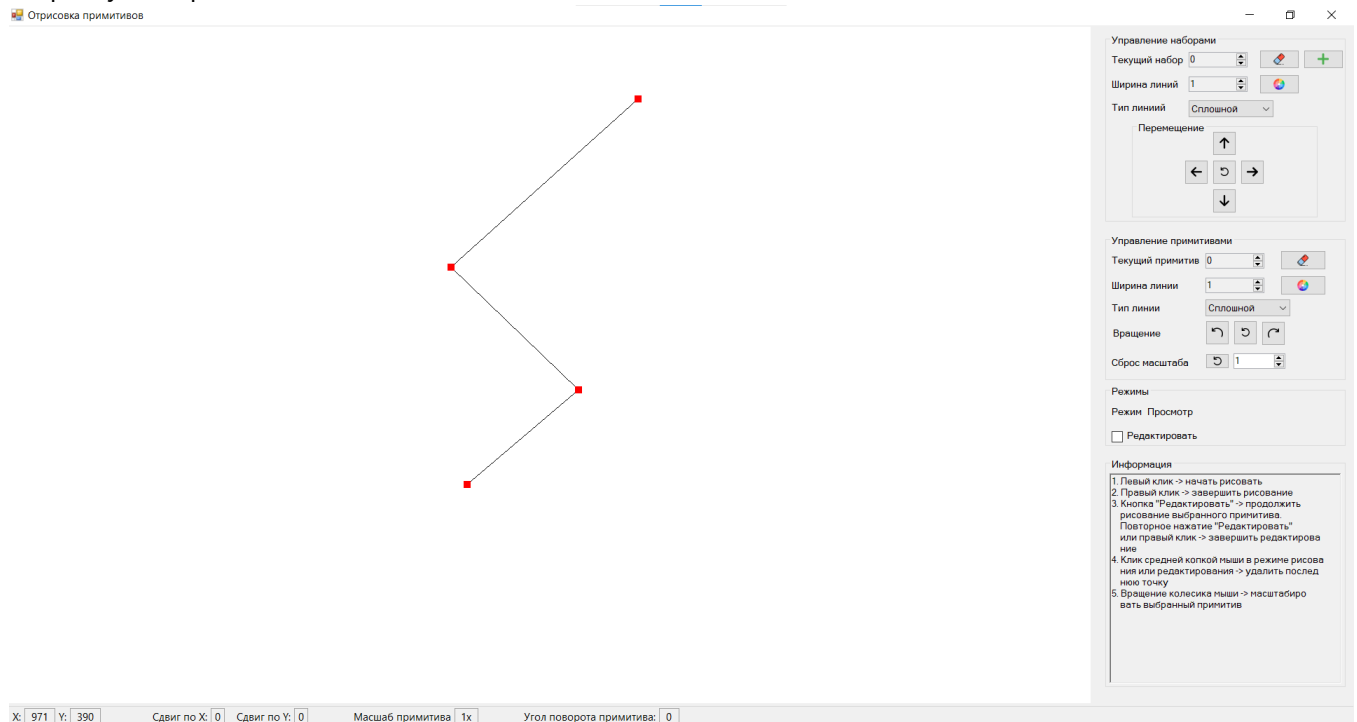


Теперь выберем второй примитив и с помощью режима редактирования удалим последнюю точку.

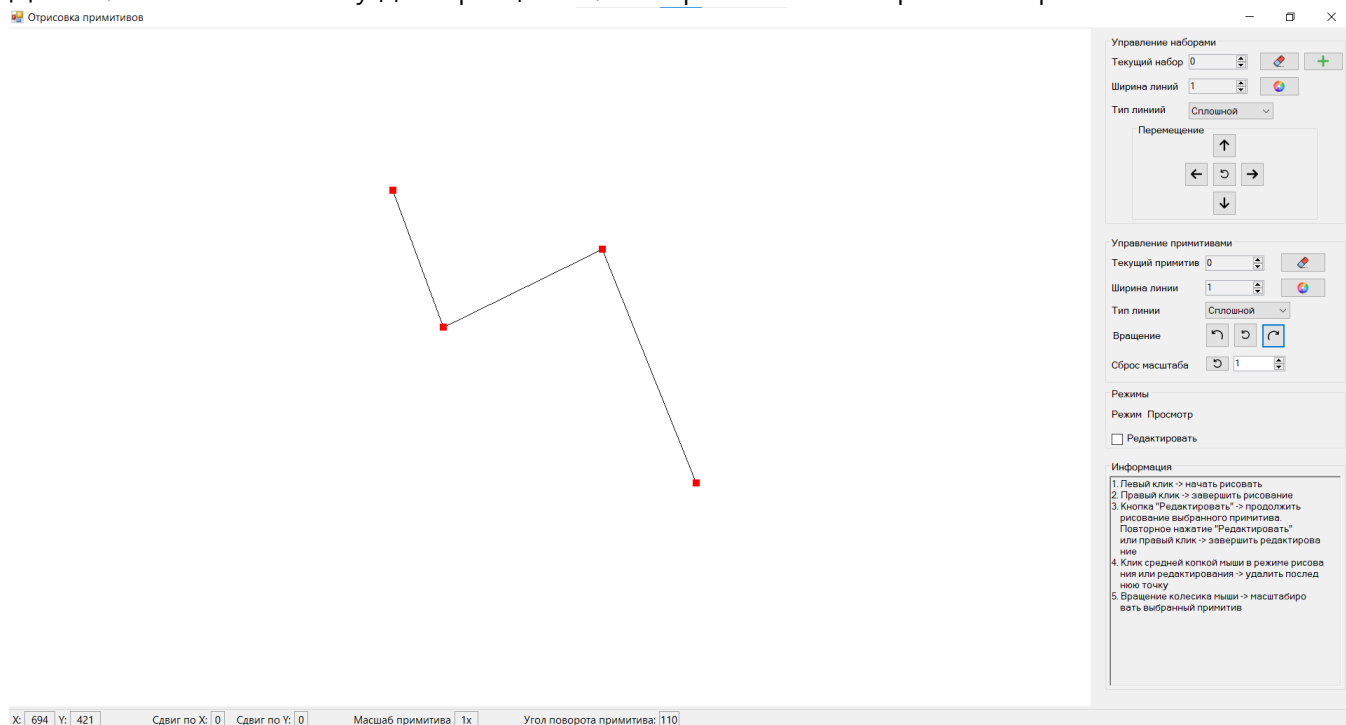


Вращение выбранного примитива

Нарисуем примитив.

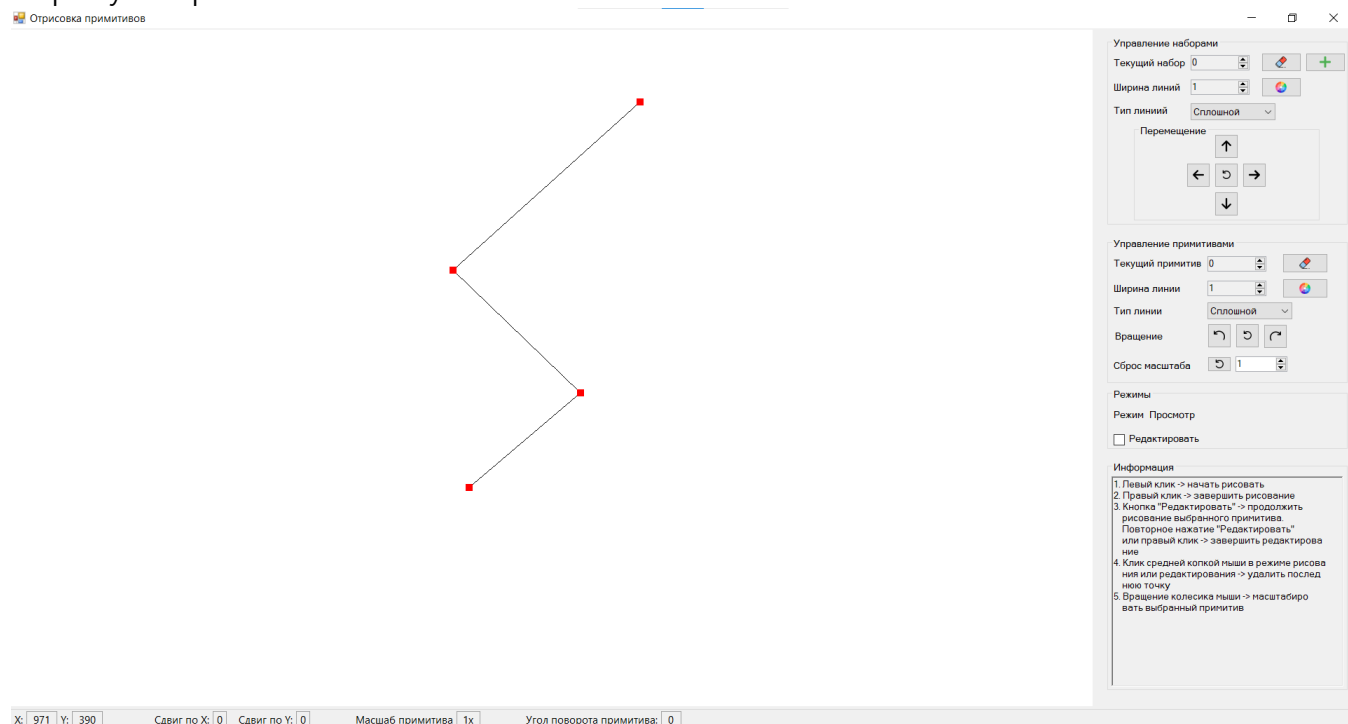


Далее, нажимая кнопку для вращения, поворачиваем выбранный примитив.

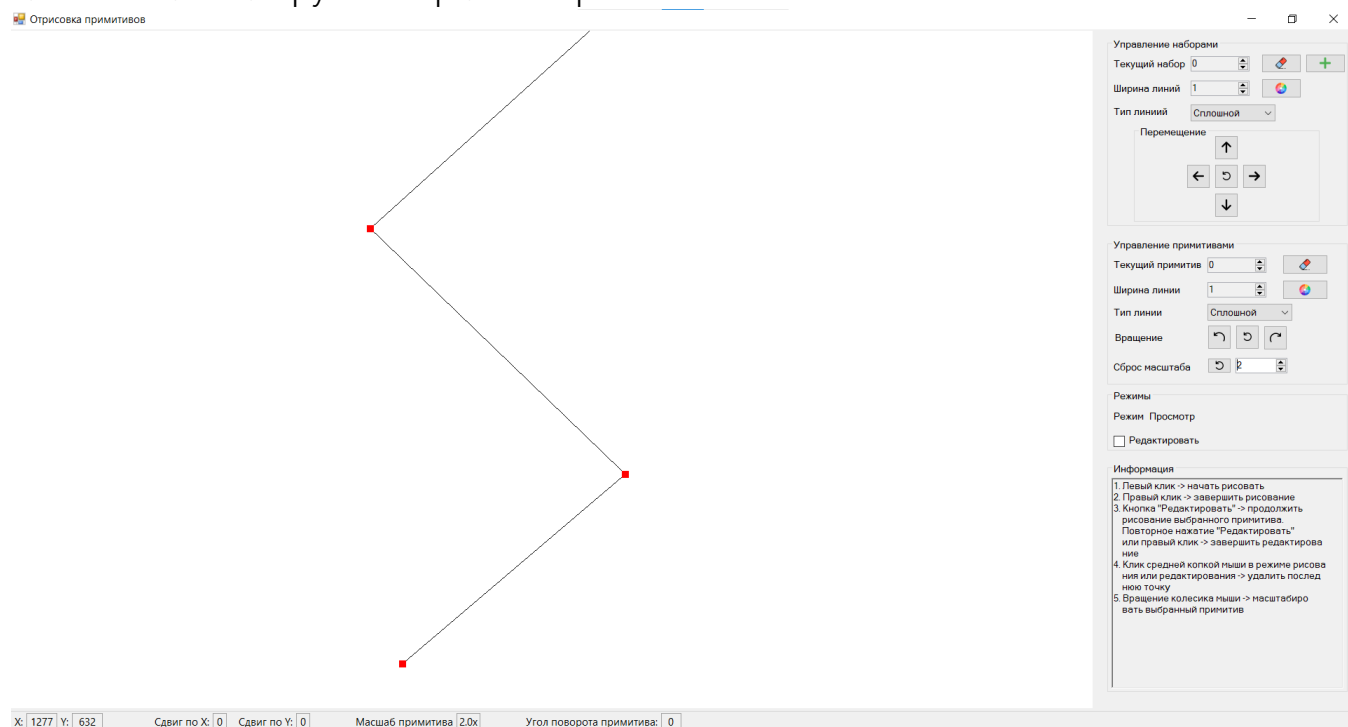


Масштабирование выбранного примитива

Нарисуем примитив.



Далее, с помощью выбора значения коэффициента изменения масштаба и колесика мыши масштабируем выбранный примитив.



ЛИСТИНГ

MainForm.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Globalization;
4 using System.Linq;
5 using System.Windows.Forms;
6 using ComputerGraphics.Source;
7 using SharpGL;
8
9 namespace ComputerGraphics
10 {
11     public partial class MainForm : Form
12     {
13         private readonly List<List<StripLine>> _lines = new List<List<StripLine>>();
14         private readonly List<Point2D> _shifts = new List<Point2D>();
15         private readonly List<ushort> _stipples = new List<ushort>();
16         private StripLine _line = new StripLine();
17         private byte _currentSet;
18         private byte _currentLine;
19         private bool _isDrawingCurrent;
20         private bool _isEdit;
21
22         public MainForm()
23         {
24             InitializeComponent();
25             comboBoxLine.SelectedIndex = 0;
26             comboBoxSet.SelectedIndex = 0;
27         }
28
29         private void GL_OpenGLInitialized(object sender, EventArgs e)
30         {
31             OpenGL gl = GL.OpenGL;
32
33             gl.Disable(OpenGL.GL_DEPTH_TEST);
34             gl.ClearColor(1f, 1f, 1f, 1f);
35             gl.Clear(OpenGL.GL_COLOR_BUFFER_BIT);
36
37             gl.MatrixMode(OpenGL.GL_PROJECTION);
38             gl.LoadIdentity();
39             gl.Ortho2D(0, GL.Width, 0, GL.Height);
40             gl.MatrixMode(OpenGL.GL_MODELVIEW);
41             gl.LoadIdentity();
42         }
43
44         private void GL_Resized(object sender, EventArgs e)
45         {
46             GL_OpenGLInitialized(sender, e);
47         }
48
49         private void GL_OpenGLDraw(object sender, RenderEventArgs args)
50         {
51             OpenGL gl = GL.OpenGL;
52
53             gl.Clear(OpenGL.GL_COLOR_BUFFER_BIT);
54
55             for (int iset = 0; iset < _lines.Count; iset++)
56             {
```

```

57 gl.PushMatrix();
58 gl.Translate(_shifts[iset].X, _shifts[iset].Y, 0);
59
60 for (int iline = 0; iline < _lines[iset].Count; iline++)
61 {
62     var line = _lines[iset][iline];
63
64     gl.Color(line.Color.R, line.Color.G, line.Color.B);
65     gl.Enable(OpenGL.GL_LINE_STIPPLE);
66     gl.LineStipple(1, line.Stipple);
67     gl.LineWidth(line.Thickness);
68     gl.Begin(OpenGL.GL_LINE_STRIP);
69
70     foreach (var p in line.Points)
71     {
72         gl.Vertex(p.X, p.Y);
73     }
74
75     gl.Disable(OpenGL.GL_LINE_STIPPLE);
76     gl.End();
77
78     // Текущий набор выделяем точками
79     if (iset == _currentSet)
80     {
81         gl.PointSize(10);
82
83         // Выделяем "активную" линию
84         if (iline == _currentLine)
85         {
86             gl.Color(1.0f, 0.0f, 0.0f);
87         }
88         else
89         {
90             gl.Color(0.0f, 0.0f, 0.0f);
91             gl.Enable(OpenGL.GL_POINT_SMOOTH);
92         }
93
94         gl.Begin(OpenGL.GL_POINTS);
95
96         foreach (var p in line.Points)
97         {
98             gl.Vertex(p.X, p.Y);
99         }
100
101         gl.End();
102
103         if (iline != _currentLine)
104         {
105             gl.Disable(OpenGL.GL_POINT_SMOOTH);
106         }
107     }
108 }
109
110 if (_isDrawingCurrent && !_isEdit)
111 {
112     gl.Color(_line.Color.R, _line.Color.G, _line.Color.B);
113     gl.LineWidth(_line.Thickness);
114     gl.Enable(OpenGL.GL_LINE_STIPPLE);
115     gl.LineStipple(1, _line.Stipple);
116     gl.Begin(OpenGL.GL_LINE_STRIP);

```

```

117         foreach (var p in _line.Points)
118         {
119             gl.Vertex(p.X, p.Y);
120         }
121
122         gl.Disable(OpenGL.GL_LINE_STIPPLE);
123         gl.End();
124
125         // Сразу выделяем линию точками
126         gl.PointSize(10);
127         gl.Color(1.0f, 0.0f, 0.0f);
128         gl.Begin(OpenGL.GL_POINTS);
129
130         foreach (var p in _line.Points)
131         {
132             gl.Vertex(p.X, p.Y);
133         }
134
135         gl.End();
136     }
137     else
138     {
139         ChangeSet.Enabled = true;
140         ChangePrimitive.Enabled = true;
141     }
142
143     gl.PopMatrix();
144 }
145
146 gl.Finish();
147 }
148
149 private void GL_MouseClick(object sender, MouseEventArgs e)
150 {
151     if (e.Button == MouseButton.Left)
152     {
153         // Если наборов еще нет -> добавляем
154         if (_lines.IsEmpty())
155         {
156             AddSet_Click(sender, e);
157         }
158
159         _isDrawingCurrent = true;
160
161         reg.Text = "Рисование";
162
163         if (_isEdit) reg.Text = "Редактирование";
164
165         float mouseX = e.X - _shifts[_currentSet].X;
166         float mouseY = GL.Height - e.Y - _shifts[_currentSet].Y;
167
168         _line.Points.Add(new Point2D(mouseX, mouseY));
169
170         if (_line.Points.Count == 1)
171         {
172             if (_lines[_currentSet].IsEmpty())
173             {
174                 _currentLine = 0;
175             }
176         }
177     }
178 }

```

```

177         else
178         {
179             ChangePrimitive.Maximum = _lines[_currentSet].Count;
180             ChangePrimitive.Value = ChangePrimitive.Maximum;
181             _currentLine = (byte)ChangePrimitive.Maximum;
182         }
183
184         ChangePrimitive.Enabled = true;
185     }
186 }
187
188 if (e.Button == MouseButton.Right)
189 {
190     if (_line.Points.Count == 0) return;
191
192     if (!_isEdit)
193     {
194         _lines[_currentSet].Add(_line.Clone() as StripLine);
195         _line.Points.Clear();
196     }
197     else
198     {
199         _line = new StripLine();
200     }
201
202     _isDrawingCurrent = false;
203     SetMove.Enabled = true;
204     reg.Text = "Просмотр";
205     checkBox1.Checked = false;
206     _isEdit = false;
207 }
208
209 if (e.Button == MouseButton.Middle)
210 {
211     if (_line.Points.Count == 0) return;
212
213     _line.Points.RemoveAt(_line.Points.Count - 1);
214
215     if (_line.Points.Count == 0)
216     {
217         _isDrawingCurrent = false;
218         _isEdit = false;
219         reg.Text = "Просмотр";
220         checkBox1.Checked = false;
221         DeletePrimitive_Click(sender, e);
222     }
223 }
224
225
226 private void GL_MouseMove(object sender, MouseEventArgs e)
227 {
228     short xPos = (short)e.X;
229     short yPos = (short)e.Y;
230
231     statusXPosValue.Text = xPos.ToString();
232     statusYPosValue.Text = yPos.ToString();
233 }
234
235 private void GL_MouseScroll(object sender, MouseEventArgs e)
236 {

```

```

237         if (_lines.IsEmpty()) return;
238
239         if (!_isDrawingCurrent || !_isEdit) return;
240
241         var center = _lines[_currentSet][_currentLine].MassCenter();
242
243         _lines[_currentSet][_currentLine].Scale(center, Math.Sign(e.Delta) == 1 ?
↪ 1.1f : 0.9f);
244
245         PrimitiveScale.Text =
↪ _lines[_currentSet][_currentLine].ScaleXY.ToString("F1") + "x";
246     }
247
248     // Панель управления *****
249     // Управление наборами *****
250     private void UpBtn_Click(object sender, EventArgs e)
251     {
252         if (!_isDrawingCurrent && !_lines[_currentSet].IsEmpty())
253         {
254             _shifts[_currentSet] = new Point2D(_shifts[_currentSet].X,
↪ (_shifts[_currentSet].Y + 40));
255             statusXShiftValue.Text =
↪ _shifts[_currentSet].X.ToString(CultureInfo.InvariantCulture);
256             statusYShiftValue.Text =
↪ _shifts[_currentSet].Y.ToString(CultureInfo.InvariantCulture);
257         }
258     }
259
260     private void RightBtn_Click(object sender, EventArgs e)
261     {
262         if (!_isDrawingCurrent && !_lines[_currentSet].IsEmpty())
263         {
264             _shifts[_currentSet] = new Point2D(_shifts[_currentSet].X + 40,
↪ _shifts[_currentSet].Y);
265             statusXShiftValue.Text =
↪ _shifts[_currentSet].X.ToString(CultureInfo.InvariantCulture);
266             statusYShiftValue.Text =
↪ _shifts[_currentSet].Y.ToString(CultureInfo.InvariantCulture);
267         }
268     }
269
270     private void LeftBtn_Click(object sender, EventArgs e)
271     {
272         if (!_isDrawingCurrent && !_lines[_currentSet].IsEmpty())
273         {
274             _shifts[_currentSet] = new Point2D(_shifts[_currentSet].X - 40,
↪ _shifts[_currentSet].Y);
275             statusXShiftValue.Text =
↪ _shifts[_currentSet].X.ToString(CultureInfo.InvariantCulture);
276             statusYShiftValue.Text =
↪ _shifts[_currentSet].Y.ToString(CultureInfo.InvariantCulture);
277         }
278     }
279
280     private void DownBtn_Click(object sender, EventArgs e)
281     {
282         if (!_isDrawingCurrent && !_lines[_currentSet].IsEmpty())
283         {
284             _shifts[_currentSet] = new Point2D(_shifts[_currentSet].X,
↪ _shifts[_currentSet].Y - 40);

```

```

285         statusXShiftValue.Text =
↪     _shifts[_currentSet].X.ToString(CultureInfo.InvariantCulture);
286         statusYShiftValue.Text =
↪     _shifts[_currentSet].Y.ToString(CultureInfo.InvariantCulture);
287     }
288 }
289
290 private void ResetBtn_Click(object sender, EventArgs e)
291 {
292     _shifts[_currentSet] = new Point2D();
293     statusXShiftValue.Text =
↪     _shifts[_currentSet].X.ToString(CultureInfo.InvariantCulture);
294     statusYShiftValue.Text =
↪     _shifts[_currentSet].Y.ToString(CultureInfo.InvariantCulture);
295 }
296
297 private void ChangeSet_ValueChanged(object sender, EventArgs e)
298 {
299     if (ChangeSet.Value != 0 && ChangeSet.Value == _lines.Count)
300     {
301         ChangeSet.Value--;
302     }
303
304     _currentSet = (byte)ChangeSet.Value;
305
306     if (!_lines.IsEmpty())
307     {
308         if (_lines[_currentSet].IsEmpty())
309         {
310             ChangePrimitive.Maximum = 0;
311         }
312         else
313         {
314             ChangePrimitive.Maximum = _lines[_currentSet].Count - 1;
315         }
316     }
317 }
318
319 private void AddSet_Click(object sender, EventArgs e)
320 {
321     // Если кнопка "Создать новый набор" нажата до завершения рисования
322     // примитива, то принудительно завершаем его рисование
323     if (_isDrawingCurrent)
324     {
325         _lines[_currentSet].Add(_line.Clone() as StriLine);
326         _line.Reset();
327         _isDrawingCurrent = false;
328         SetMove.Enabled = true;
329     }
330
331     // Если нет еще ни одного набора -> создаем его
332     if (_lines.IsEmpty())
333     {
334         ChangeSet.Enabled = true;
335         _lines.Add(new List<StriLine>());
336         _shifts.Add(new Point2D());
337         _stipples.Add(0xFFFF);
338         return;
339     }
340

```

```

341 // Создать новый набор можно только в том случае, если предшествующий
342 // ему набор не пуст
343 if (!_lines[_currentSet].IsEmpty())
344 {
345     _line.Reset();
346
347     _lines.Add(new List<StripLine>());
348     _shifts.Add(new Point2D());
349     _stipples.Add(0xFFFF);
350
351     ChangeSet.Maximum = _lines.Count - 1;
352     ChangeSet.Value = ChangeSet.Maximum;
353
354     ChangeWidthS.Value = 1;
355     ChangePrimitive.Value = 0;
356     ChangePrimitive.Maximum = 0;
357 }
358 }
359
360 private void DeleteSet_Click(object sender, EventArgs e)
361 {
362     // Удалять можно только если не рисуется примитив
363     // либо если есть хотя бы один набор
364     if (!_isDrawingCurrent && !_lines.IsEmpty())
365     {
366         _currentSet = (byte)ChangeSet.Value;
367
368         _lines.RemoveAt(_currentSet);
369         _shifts.RemoveAt(_currentSet);
370         _stipples.RemoveAt(_currentSet);
371
372         ChangeSet_ValueChanged(sender, e);
373
374         ChangeSet.Maximum = _lines.Count == 0 ? 0 : _lines.Count - 1;
375         ChangeSet.Value = ChangeSet.Maximum;
376     }
377
378     // Не отображаем "Текущий набор", если их нет
379     if (_lines.IsEmpty())
380     {
381         SetMove.Enabled = false;
382     }
383 }
384
385 private void ChangeWidthS_ValueChanged(object sender, EventArgs e)
386 {
387     _line.Thickness = (float)ChangeWidthS.Value;
388
389     if (!_lines.IsEmpty())
390     {
391         foreach (var line in _lines[_currentSet])
392         {
393             line.Thickness = _line.Thickness;
394         }
395     }
396 }
397
398 private void ChangeColorS_Click(object sender, EventArgs e)
399 {
400     colorDialog1.ShowDialog();

```

```

401         _line.Color = colorDialog1.Color;
402
403         if (!_lines.IsEmpty())
404         {
405             foreach (var line in _lines[_currentSet])
406             {
407                 line.Color = _line.Color;
408             }
409         }
410     }
411 }
412
413 private void comboBoxSet_SelectedIndexChanged(object sender, EventArgs e)
414 {
415     switch (comboBoxSet.SelectedIndex)
416     {
417         case 0:
418             _line.Stipple = 0xFFFF;
419             break;
420         case 1:
421             _line.Stipple = 0x0101;
422             break;
423         case 2:
424             _line.Stipple = 0x00F0;
425             break;
426         case 3:
427             _line.Stipple = 0x1C47;
428             break;
429     }
430
431     if (!_lines.IsEmpty())
432     {
433         foreach (var line in _lines[_currentSet])
434         {
435             line.Stipple = _line.Stipple;
436         }
437     }
438 }
439
440
441 // Управление примитивами *****
442 private void ChangeColorP_Click(object sender, EventArgs e)
443 {
444     colorDialog1.ShowDialog();
445
446     _line.Color = colorDialog1.Color;
447
448     if (!_lines.IsEmpty() && !_lines[_currentSet].IsEmpty())
449     {
450         _lines[_currentSet][_currentLine].Color = colorDialog1.Color;
451     }
452 }
453
454 private void DeletePrimitive_Click(object sender, EventArgs e)
455 {
456     if (!_isDrawingCurrent && !_lines.IsEmpty())
457     {
458         if (!_lines[_currentSet].IsEmpty())
459             ↪ _lines[_currentSet].RemoveAt(_currentLine);

```



```

460         ChangePrimitive.Value = ChangePrimitive.Value == 0 ? 0 :
↪ --ChangePrimitive.Value;
461         ChangePrimitive.Maximum = ChangePrimitive.Maximum == 0 ? 0 :
↪ --ChangePrimitive.Maximum;
462     }
463     else return;
464
465     if (_lines[_currentSet].IsEmpty()) DeleteSet_Click(sender, e);
466 }
467
468 private void ChangePrimitive_ValueChanged(object sender, EventArgs e)
469 {
470     if (!_isDrawingCurrent) _currentLine = (byte)ChangePrimitive.Value;
471 }
472
473 private void ChangeWidthP_ValueChanged(object sender, EventArgs e)
474 {
475     _line.Thickness = (float)ChangeWidthP.Value;
476
477     if (!_isDrawingCurrent && !_lines.IsEmpty() &&
↪ !_lines[_currentSet].IsEmpty())
478     {
479         _lines[_currentSet][_currentLine].Thickness =
↪ (float)ChangeWidthP.Value;
480     }
481 }
482
483 private void comboBoxLine_SelectedIndexChanged(object sender, EventArgs e)
484 {
485     switch (comboBoxLine.SelectedIndex)
486     {
487         case 0:
488             _line.Stipple = 0xFFFF;
489             break;
490         case 1:
491             _line.Stipple = 0x0101;
492             break;
493         case 2:
494             _line.Stipple = 0x00F0;
495             break;
496         case 3:
497             _line.Stipple = 0x1C47;
498             break;
499     }
500
501     if (!_lines.IsEmpty() && !_lines[_currentSet].IsEmpty())
502     {
503         _lines[_currentSet][_currentLine].Stipple = _line.Stipple;
504     }
505 }
506
507 private void RotateLeft_Click(object sender, EventArgs e)
508 {
509     if (_lines.IsEmpty()) return;
510
511     if (_lines[_currentSet].IsEmpty()) return;
512
513     if (_isDrawingCurrent || _isEdit) return;
514
515     var center = _lines[_currentSet][_currentLine].MassCenter();

```

```

516         _lines[_currentSet][_currentLine].Rotate(center, -10);
517
518         PrimitiveAngle.Text = _lines[_currentSet][_currentLine].Angle.ToString();
519     }
520
521     private void RotateRight_Click(object sender, EventArgs e)
522     {
523         if (_lines.IsEmpty()) return;
524
525         if (_lines[_currentSet].IsEmpty()) return;
526
527         if (_isDrawingCurrent || _isEdit) return;
528
529         var center = _lines[_currentSet][_currentLine].MassCenter();
530
531         _lines[_currentSet][_currentLine].Rotate(center, 10);
532
533         PrimitiveAngle.Text =
534     ↪ _lines[_currentSet][_currentLine].Angle.ToString(CultureInfo.InvariantCulture);
535     }
536
537     private void checkBox1_CheckedChanged(object sender, EventArgs e)
538     {
539         _isEdit = checkBox1.Checked;
540
541         if (!_isDrawingCurrent && !_lines.IsEmpty() &&
542     ↪ !_lines[_currentSet].IsEmpty() && _isEdit)
543         {
544             _line = _lines[_currentSet][_currentLine];
545             _isDrawingCurrent = true;
546             reg.Text = "Редактирование";
547         }
548         else if (!_isEdit)
549         {
550             _isDrawingCurrent = false;
551             _isEdit = false;
552             _line = new StripLine();
553             SetMove.Enabled = true;
554             checkBox1.Checked = false;
555             reg.Text = "Просмотр";
556         }
557     }
558
559     private void RotateResetBtn_Click(object sender, EventArgs e)
560     {
561         if (_lines.IsEmpty()) return;
562
563         if (_lines[_currentSet].IsEmpty()) return;
564
565         if (_isDrawingCurrent || _isEdit) return;
566
567         var angle = _lines[_currentSet][_currentLine].Angle;
568         var center = _lines[_currentSet][_currentLine].MassCenter();
569
570         _lines[_currentSet][_currentLine].Rotate(center, -angle);
571
572         PrimitiveAngle.Text = "0";
573     }

```

```

574     private void ScaleResetBtn_Click(object sender, EventArgs e)
575     {
576         if (_lines.IsEmpty()) return;
577
578         if (_lines[_currentSet].IsEmpty()) return;
579
580         if (_isDrawingCurrent || _isEdit) return;
581
582         PrimitiveScale.Text = "1";
583
584         var scaleXY = _lines[_currentSet][_currentLine].ScaleXY;
585         var center = _lines[_currentSet][_currentLine].MassCenter();
586
587         _lines[_currentSet][_currentLine].Scale(center, (float)(1.0 / scaleXY));
588     }
589 }
590

```

Point2D.cs

```

1  using System;
2
3  namespace ComputerGraphics.Source
4  {
5      public struct Point2D
6      {
7          public float X { get; set; }
8          public float Y { get; set; }
9
10         public Point2D(float x, float y) => (X, Y) = (x, y);
11
12         public static Point2D operator +(Point2D first, Point2D second) =>
13             new Point2D(first.X + second.X, first.Y + second.Y);
14
15         public static Point2D operator -(Point2D first, Point2D second) =>
16             new Point2D(first.X - second.X, first.Y - second.Y);
17     }
18 }

```

Primitive.cs

```
1 using System;
2 using System.Drawing;
3 using System.Collections.Generic;
4
5 namespace ComputerGraphics.Source
6 {
7     public class StripLine : ICloneable
8     {
9         public List<Point2D> Points { get; private set; }
10        public Color Color { get; set; }
11        public float Thickness { get; set; }
12        public ushort Stipple { get; set; }
13        public float ScaleXY { get; set; }
14        public float Angle { get; set; }
15
16        public Point2D MassCenter()
17        {
18            float x = 0, y = 0;
19
20            foreach (var p in Points)
21            {
22                x += p.X;
23                y += p.Y;
24            }
25
26            x = x / Points.Count;
27            y = y / Points.Count;
28
29            return new Point2D(x, y);
30        }
31
32        public StripLine()
33        {
34            Points = new List<Point2D>();
35            Color = new Color();
36            Thickness = 1.0f;
37            Stipple = 0xFFFF;
38            ScaleXY = 1.0f;
39        }
40
41        public object Clone() => new StripLine
42        {
43            Points = new List<Point2D>(Points),
44            Color = Color,
45            Thickness = Thickness,
46            Stipple = Stipple,
47            ScaleXY = ScaleXY
48        };
49
50        public void Reset()
51        {
52            Points.Clear();
53            Color = new Color();
54            Thickness = 1.0f;
55            Stipple = 0xFFFF;
56            ScaleXY = 1.0f;
57        }
58    }
```

```

59     public void Scale(Point2D pivot, float scaling)
60     {
61         ScaleXY *= scaling;
62
63         float xStep = pivot.X * scaling - pivot.X;
64         float yStep = pivot.Y * scaling - pivot.Y;
65
66         for (int i = 0; i < Points.Count; i++)
67         {
68             var x = Points[i].X * scaling - xStep;
69             var y = Points[i].Y * scaling - yStep;
70
71             Points[i] = new Point2D(x, y);
72         }
73     }
74
75     public void Rotate(Point2D pivot, float angle)
76     {
77         Angle += angle;
78
79         float radAngle = (float)(Math.PI * angle / 180.0);
80
81         for (int i = 0; i < Points.Count; i++)
82         {
83             var x = Points[i].X - pivot.X;
84             var y = Points[i].Y - pivot.Y;
85
86             var newX = (float)(x * Math.Cos(radAngle) + y * Math.Sin(radAngle));
87             var newY = (float)(-x * Math.Sin(radAngle) + y * Math.Cos(radAngle));
88
89             newX += pivot.X;
90             newY += pivot.Y;
91
92             Points[i] = new Point2D(newX, newY);
93         }
94     }
95 }
96

```

Extensions.cs

```

1  using System.Collections.Generic;
2  using System.Linq;
3
4  namespace ComputerGraphics.Source
5  {
6      public static class EnumerableExtensions
7      {
8          public static bool IsEmpty<T>(this IEnumerable<T> collection) =>
9          ↪ !collection.Any();
10     }
11 }

```