**Тема 4: Работа с графикой**

**Цель работы:** Получить практические навыки разработки проектов с использованием команд графики.

**Вариант 5.**

Движение эллипса по периметру формы, не проходя верхнюю границу формы, в прямом и обратном направлении, в прямом направлении движения у эллипса красный цвет, в обратном направлении – синий цвет. Окончание работы – нажатие клавиши Esc.

Настройки должны изменять два цвета фигуры (для прямого и для обратного направления движения), направление движения фигуры (эллипс не проходит либо верхнюю границу формы) и скорость движения фигуры.

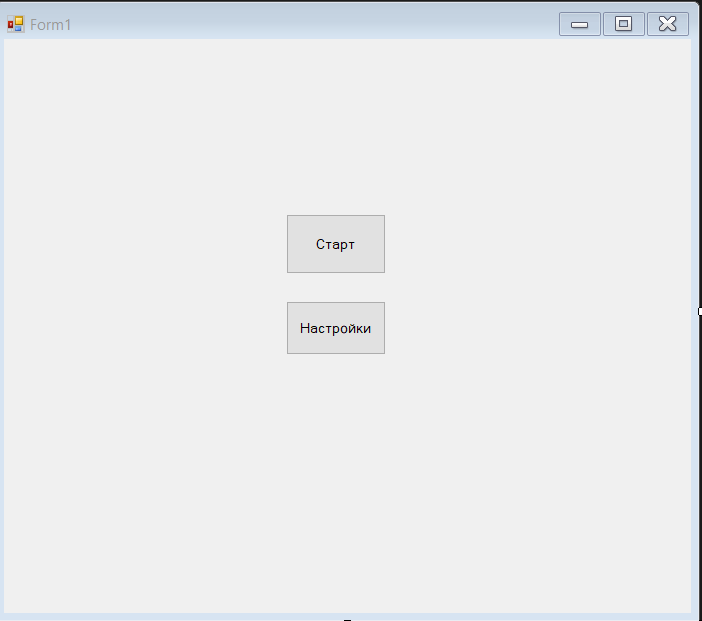
**Задание 1.** Основной проект

Разработать программу движения фигуры по поверхности формы при помощи таймера. Регулируя интервал работы таймера, можно управлять скоростью движения фигуры. Направление движения фигуры задается с помощью изменения координат по горизонтали и (или) по вертикали. Учесть, что при изменении размеров формы фигура все равно должна двигаться в заданных направлениях в пределах формы.

При нажатии на кнопку «Стоп» фигура останавливается, надпись на кнопке меняется на «Старт». При повторном нажатии кнопки фигура возобновляет движение. Программа должна завершаться по нажатию клавиши Esc.

1. **Создание пользовательского интерфейса**
   1. **Проектирование пользовательского интерфейса**

Внешний вид проекта представлен на рис. 1.1, значения измененных свойств компонентов представлены в таблице 1.1.



button1

Form1

button1

**Рис. 1.1 – Form1**

* 1. **Свойства компонентов формы**

*Таблица 1.1.* Значения свойств, установленные на этапе конструирования интерфейса

|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| form1 | Text | «Графика» |
| BackColor | GradientInactiveCaption |
|  | Размер: 10,2 |
| Начертание: обычный |
| button1 | Text | «Старт» |
| Button2 | Text | «Настройки» |

* 1. **Сценарий использования программы**
     1. Нажатие кнопки «Старт».
     2. Фигура движется.
     3. Нажатие на кнопку «Настройки».
     4. Поменять цвет фигуры.
     5. Возможность вернуться к пунктам 1.1.1, 1.1.2, 1.1.3 или 1.1.4.
     6. Завершение работы программы.

**Код класса Form1:**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Timer timer;

private bool isRunning = false;

private Color forwardColor = Color.Red;

private Color backwardColor = Color.Blue;

int sec = 0;

int w = 50, h = 50;

int x = 0, y =0;

int dx = 5;

bool da = false;

enum STATUS { Left, Right, Up, Down };

STATUS flag;

public SolidBrush brush = new SolidBrush(Color.Red);

Rectangle rc;

public Form1()

{

InitializeComponent();

timer = new Timer();

timer.Interval = 10;

timer.Tick += timer1\_Tick;

flag = STATUS.Down;

this.KeyPreview = true;

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

e.Graphics.FillEllipse(brush, rc);

}

public void setcolor (Color color)

{

backwardColor = color;

brush = new SolidBrush(color);

}

public void setcolor\_for(Color color)

{

forwardColor = color;

brush = new SolidBrush(color);

}

private void timer1\_Tick(object sender, EventArgs e)

{

sec++;

rc = new Rectangle(x, y, w, h);

this.Invalidate(rc, true);

if (flag == STATUS.Left)

{

x -= dx;

if (x < 1 && y > 0)

{

if (da)

{

flag = STATUS.Up;

da = false;

}

else

{

y += dx;

flag = STATUS.Down;

}

}

else if (x <= 1 && y <= 0)

{

flag = STATUS.Right;

}

else if (x <= 1 && y >= (this.ClientSize.Height))

{

flag = STATUS.Up;

}

}

else if (flag == STATUS.Right)

{

x += dx;

if (x >= (this.ClientSize.Width - w))

{

da = true;

flag = STATUS.Up;

}

}

else if (flag == STATUS.Up)

{

y -= dx;

if (y <= 1 && x >= 0)

{

flag = STATUS.Down;

if (da) {

brush = new SolidBrush(backwardColor);

}

else

{

brush = new SolidBrush(forwardColor);

}

}

}

else if (flag == STATUS.Down)

{

y += dx;

if (y >= (this.ClientSize.Height - h))

{

flag = STATUS.Right;

}

if (y >= (this.ClientSize.Height - h) && x >= 0)

{

flag = STATUS.Left;

}

if (y > (this.ClientSize.Height ))

{

flag = STATUS.Up;

}

}

rc = new Rectangle(x, y, w, h);

this.Invalidate(rc, true);

}

private void button1\_Click(object sender, EventArgs e)

{

if (isRunning)

{

timer.Stop();

button1.Text = "Старт";

}

else

{

timer.Start();

button1.Text = "Стоп";

}

isRunning = !isRunning;

}

private void Form1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Escape)

this.Close();

}

protected override void OnKeyDown(KeyEventArgs e)

{

base.OnKeyDown(e);

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 frm = new Form2(this);

frm.Show();

}

}

public Color MyColor

{

get

{

return brush.Color;

}

set

{

brush.Color = value;

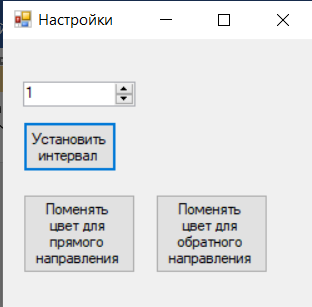
}

}

**Задание 2.** Настройка дополнительных параметров

Добавьте в проект настройку дополнительных параметров: выбор цвета фигуры с помощью диалогового окна выбора цвета, настройку скорости движения фигуры, вида фигуры или направления движения (в зависимости от варианта). Настройка параметров должна происходить с помощью второй формы в немодальном режиме и применяться интерактивно (в процессе движения).

Внешний вид проекта представлен на рис. 2.1, значения измененных свойств компонентов представлены в таблице 2.1.



Numericupdown1

button1

Button3

Form2

Button2

**Рис. 2.1 – Form2**

* 1. **Свойства компонентов формы**

*Таблица 2.1.* Значения свойств, установленные на этапе конструирования интерфейса

|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| Form2 | Text | «Настройки» |
| button1 | Text | «Установить интервал» |
| Button2 | Text | «Поменять цвет для прямого направления» |
| Button3 | Text | «Поменять цвет для обратного направления» |

* 1. **Сценарий использования программы**
     1. Поменять значение интервала в numericupdown.
     2. Нажать на кнопку «Установить интервал»
     3. Нажатие на кнопку «Поменять цвет для прямого направления»
     4. Установить новый цвет.
     5. Нажатие на кнопку «Поменять цвет для обратного направления»
     6. Установить новый цвет.
     7. Возможность вернуться к предыдущим пунктам.
     8. Завершение работы программы.

**Код класса Form2:**

using System;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form2 : Form

{

Form1 frm;

public Form2(Form1 f)

{

InitializeComponent();

frm = f;

Text = "Настройки";

}

private void button1\_Click\_1(object sender, EventArgs e)

{

frm.timer.Interval = (int)numericUpDown1.Value;

}

private void Form2\_Load(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

colorDialog1 = new ColorDialog();

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

frm.setcolor(colorDialog1.Color);

}

}

private void button3\_Click(object sender, EventArgs e)

{

colorDialog1 = new ColorDialog();

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

frm.setcolor\_for(colorDialog1.Color);

}

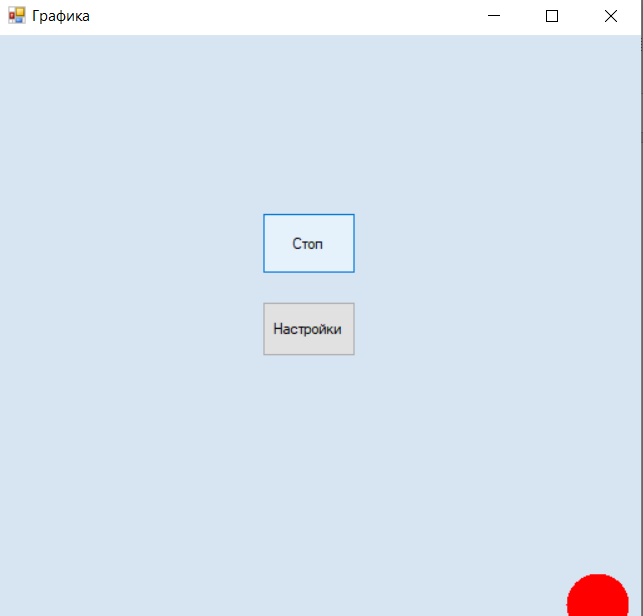
}

**3. Протокол испытаний:**

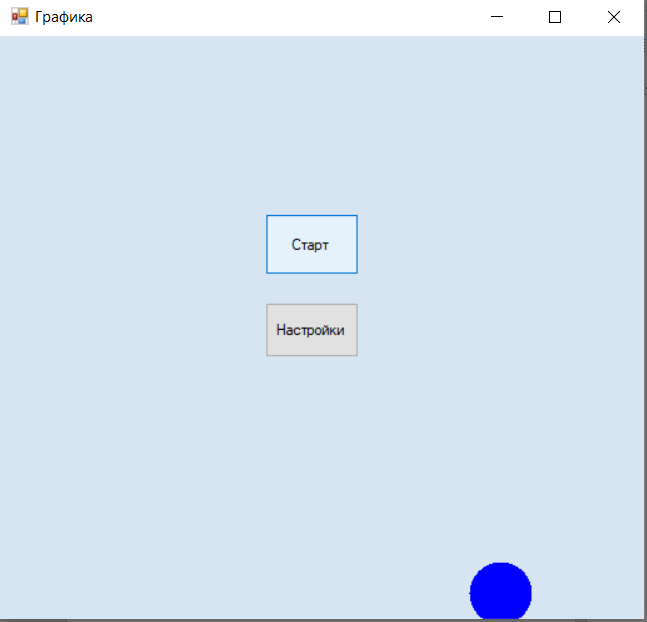
*Таблица 3.1.* Протокол испытаний

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Вариант входных данных** | **Ожидаемые результаты** | **Фактические результаты** |
| Изменение названия кнопки | Фигура не движется, и фигура движется | Когда фигура в движении кнопка имеет названия «Стоп», при остановке имеет название «Старт» | Соответствуют ожидаемым результатам  (См. Рисунки 3.1.1-3.1.2) |
| Изменение скорости движения | Число вводится в numericupdown от 1 до 10000 | Чем больше число, тем медленнее движение эллипса | Соответствуют ожидаемым результатам  (См. Рисунок 3.1.3) |
| Установка нового цвета фигуры для прямого движения | Выбор цвета в colordialog | Фигура окрашивается в выбранный цвет и поддерживает его при прямом движении | Соответствуют ожидаемым результатам  (См. Рисунки 3.1.4 – 3.1.5) |
| Установка нового цвета фигуры для обратного движения | Выбор цвета в colordialog | Фигура окрашивается в выбранный цвет при обратном движении | Соответствуют ожидаемым результатам  (См. Рисунки 3.1.6 – 3.1.7) |
| Способность фигуры двигаться по периметру формы, не учитывая верхней границы | Наблюдение за работой | Фигура движется по трем сторонам | Соответствует ожидаемым результатам  (См. Рисунок 3.1.8) |
| Способность закрытия формы по клавише Esc | Нажать на кнопку Esc при работе формы | Приложение прекращает работу | Соответствует ожидаемым результатам |

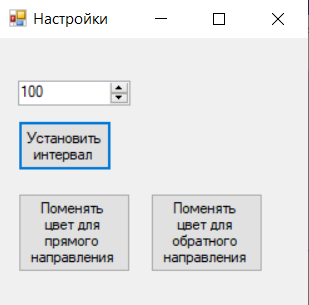
* 1. **Результаты тестирования проекта:**

****

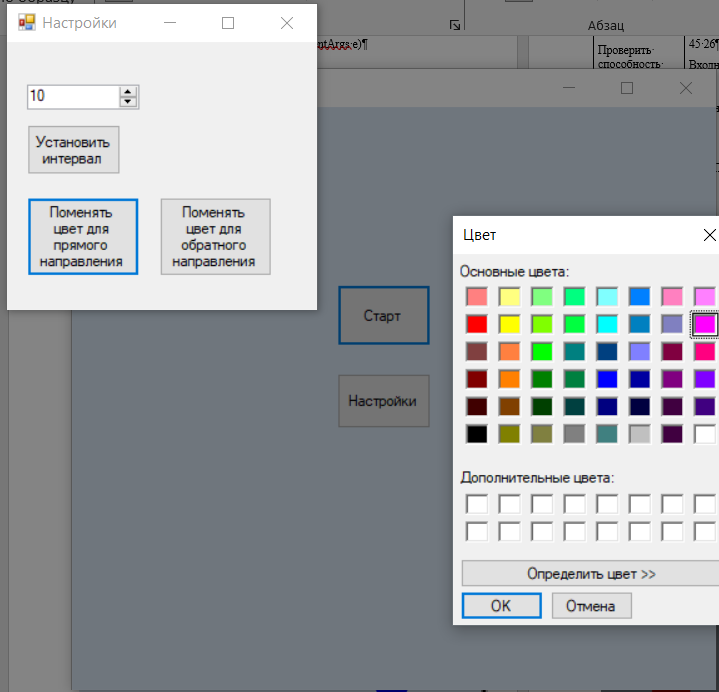
**Рис. 3.1.1**

****

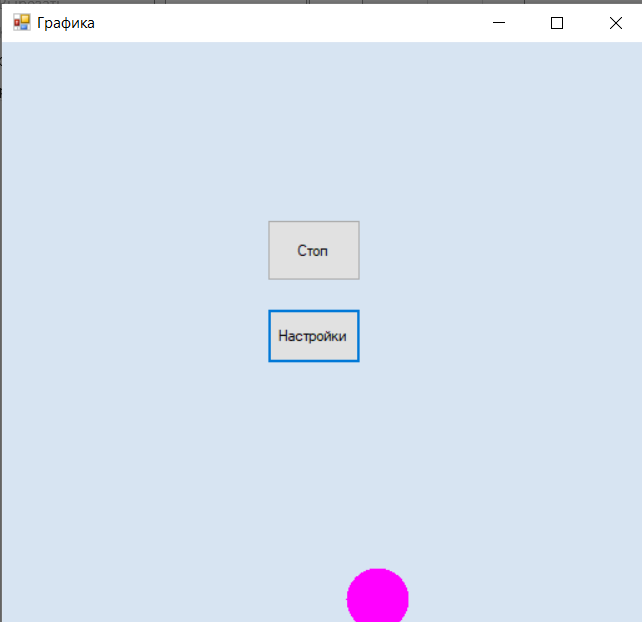
**Рис. 3.1.2**

****

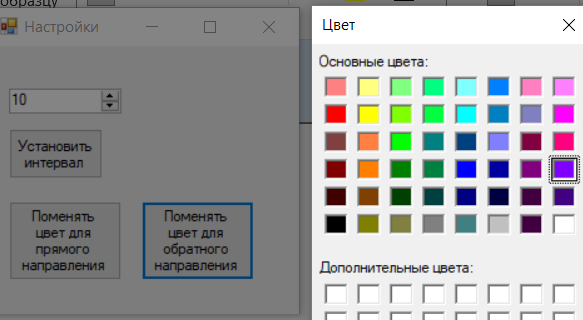
**Рис. 3.1.3**

****

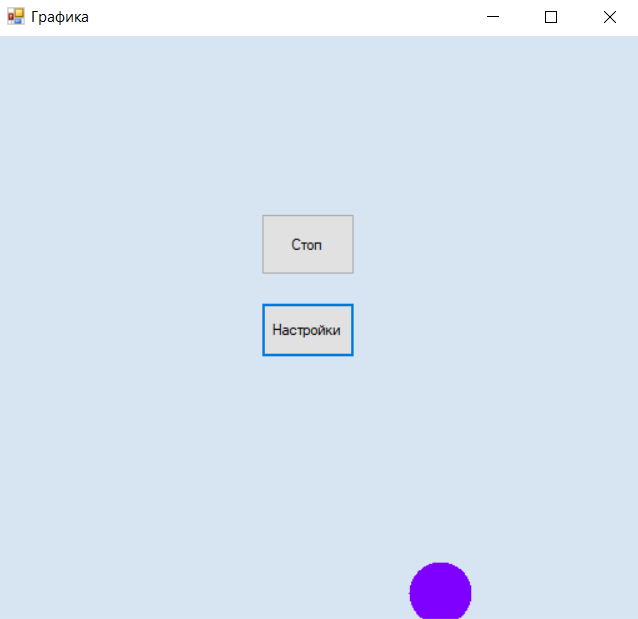
**Рис. 3.1.4**

****

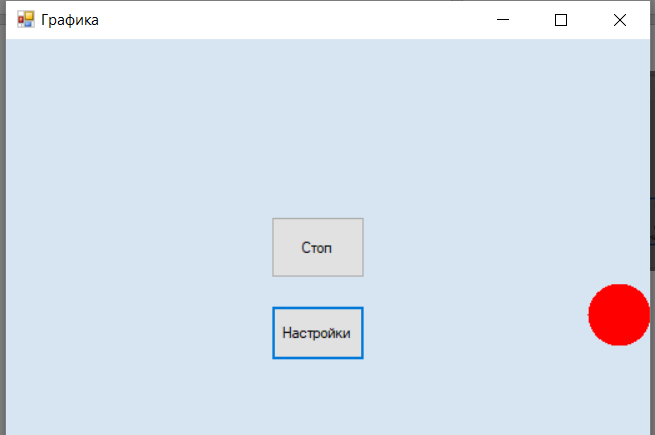
**Рис. 3.1.5**

****

**Рис. 3.1.6**

****

**Рис. 3.1.7**

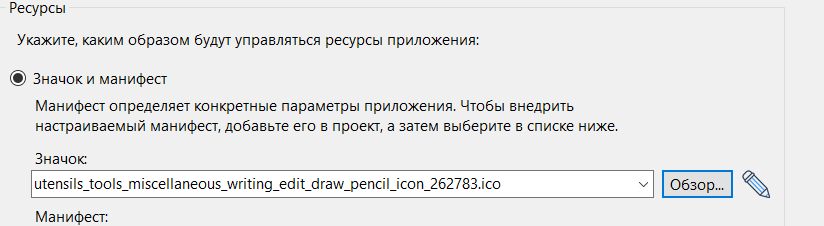
****

**Рис. 3.1.8**

**Задание 3.** Инсталляция приложения

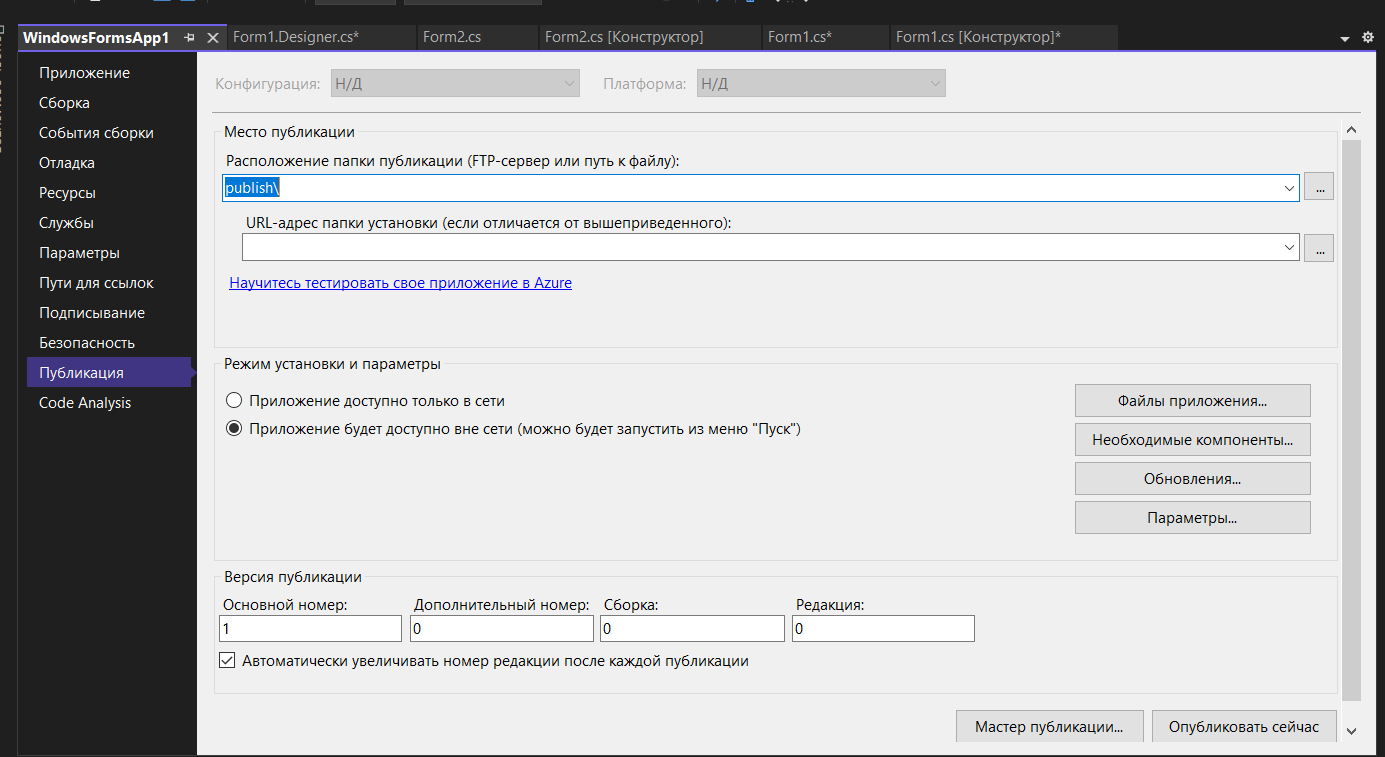
Добавить иконку в проект и название проекта. Для созданного вами приложения провести инсталляцию на свой компьютер. Инсталляция происходит при помощи мастера публикации (находится в свойствах проекта).

Установка иконки приложения (см. Рисунок 3.2.1)

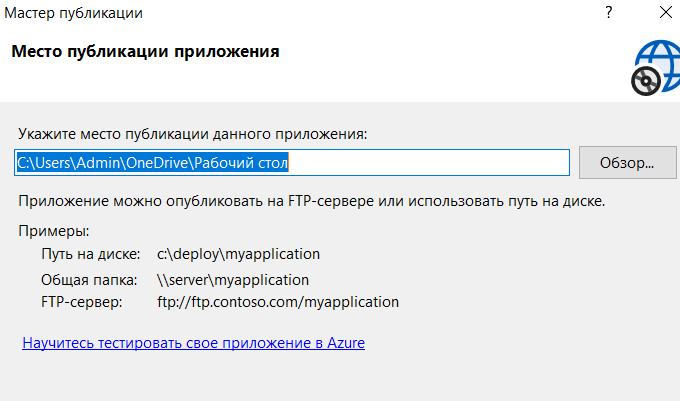
****

**Рис. 3.2.1**

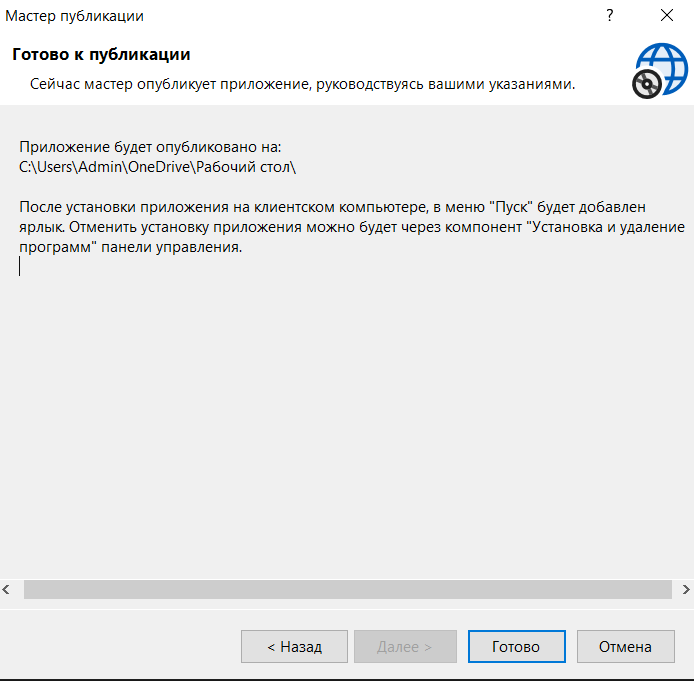
Работа с мастером публикаций (см. Рисунки 3.2.2 – 3.2.6)

****

**Рис 3.2.2 – начало работы мастера публикаций**



**Рис 3.2.3 – выбор места установки**



**Рис 3.2.4 – готовность инсталлятора**



**Рис 3.2.5– результат**