|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | ПМИИ |

|  |  |
| --- | --- |
|  |  |
| **Предмет: «Системное программирование»**  **Отчет по лабораторной работе №4**  **Выполнил: студент группы А-13а-19**  **Рамазанов Н. М.**  **Преподаватель: Меньшикова К. Г.**  **Москва, 2021** | |

**1. Краткое описание функций, параметров и используемых событий.**

1) Свойство Canvas компонента TForm.

Вывод графических примитивов происходит на поверхность формы. Эту поверхность принято называть холстом, т. е. Canvas.

Canvas предоставляет битовую карту поверхности окна приложения, компоненты, принтера и т.п., которая может быть использована для вывода графики. Canvas - не самостоятельный объект, оно всегда является свойством какого-то другого графического объекта.

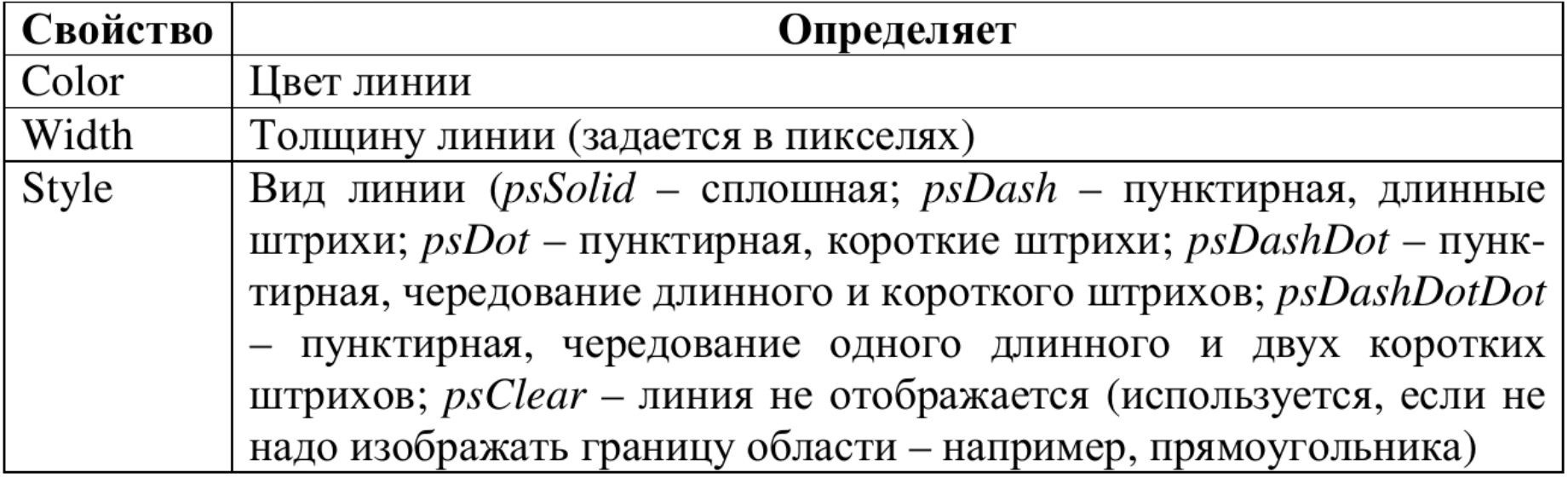
Доступ к канве любого объекта происходит следующим образом:

имя\_объекта -> Canvas -> Свойство/Метод;

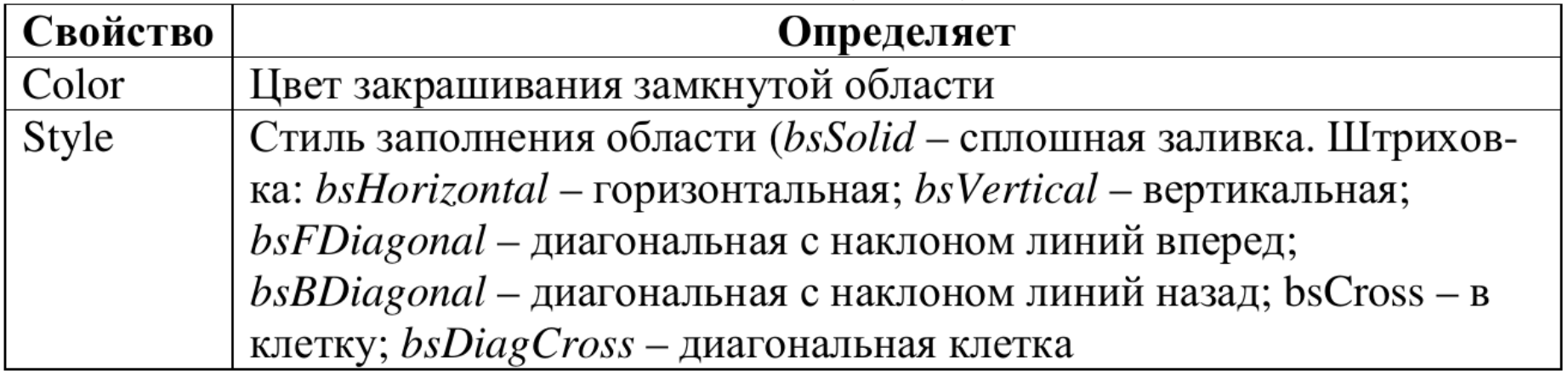
2) Инструменты рисования и их свойства.

Методы вычерчивания графических примитивов обеспечивают только вычерчивание. Вид графического элемента определяют свойства Pen (карандаш) и Brush (кисть) той поверхности (Canvas), на которой рисует метод.

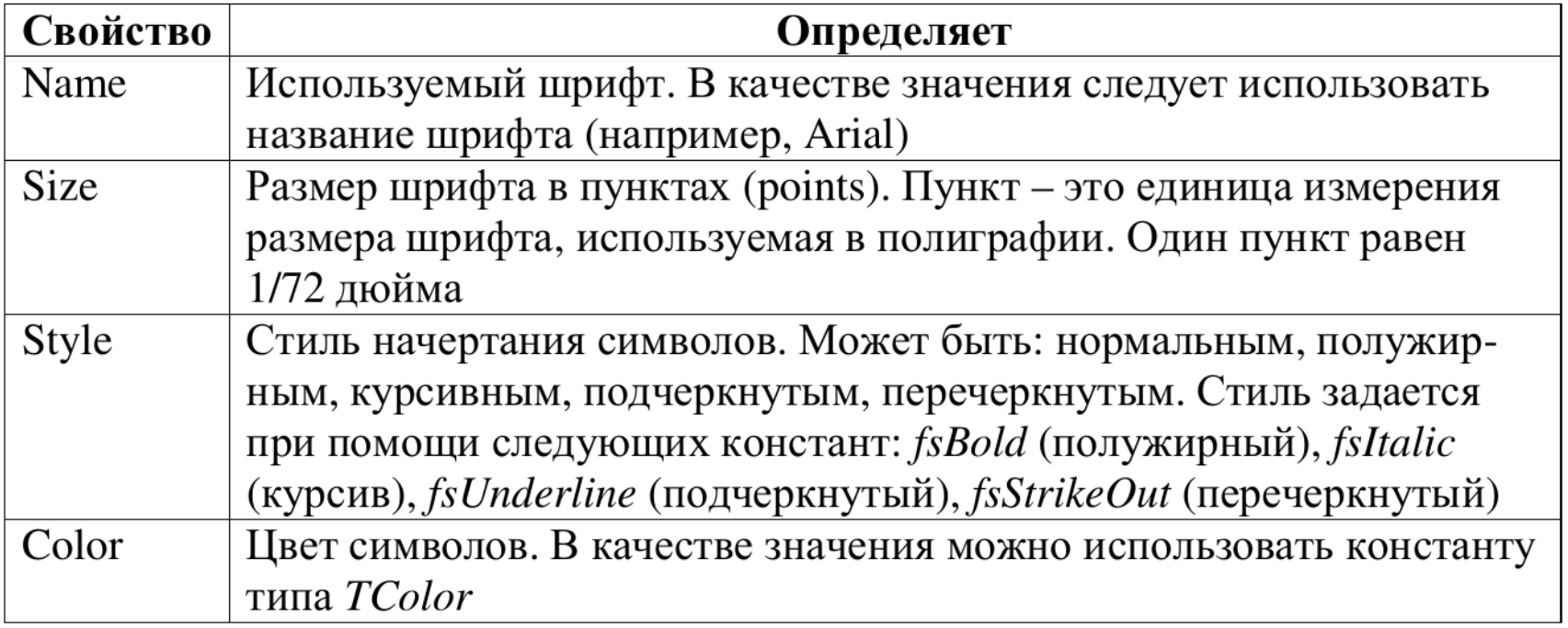
Карандаш и кисть, являясь свойствами объекта Canvas, в свою очередь представляют собой объекты Pen и Brush. Свойства объекта Pen задают цвет, толщину и тип линии или границы геометрической фигуры.



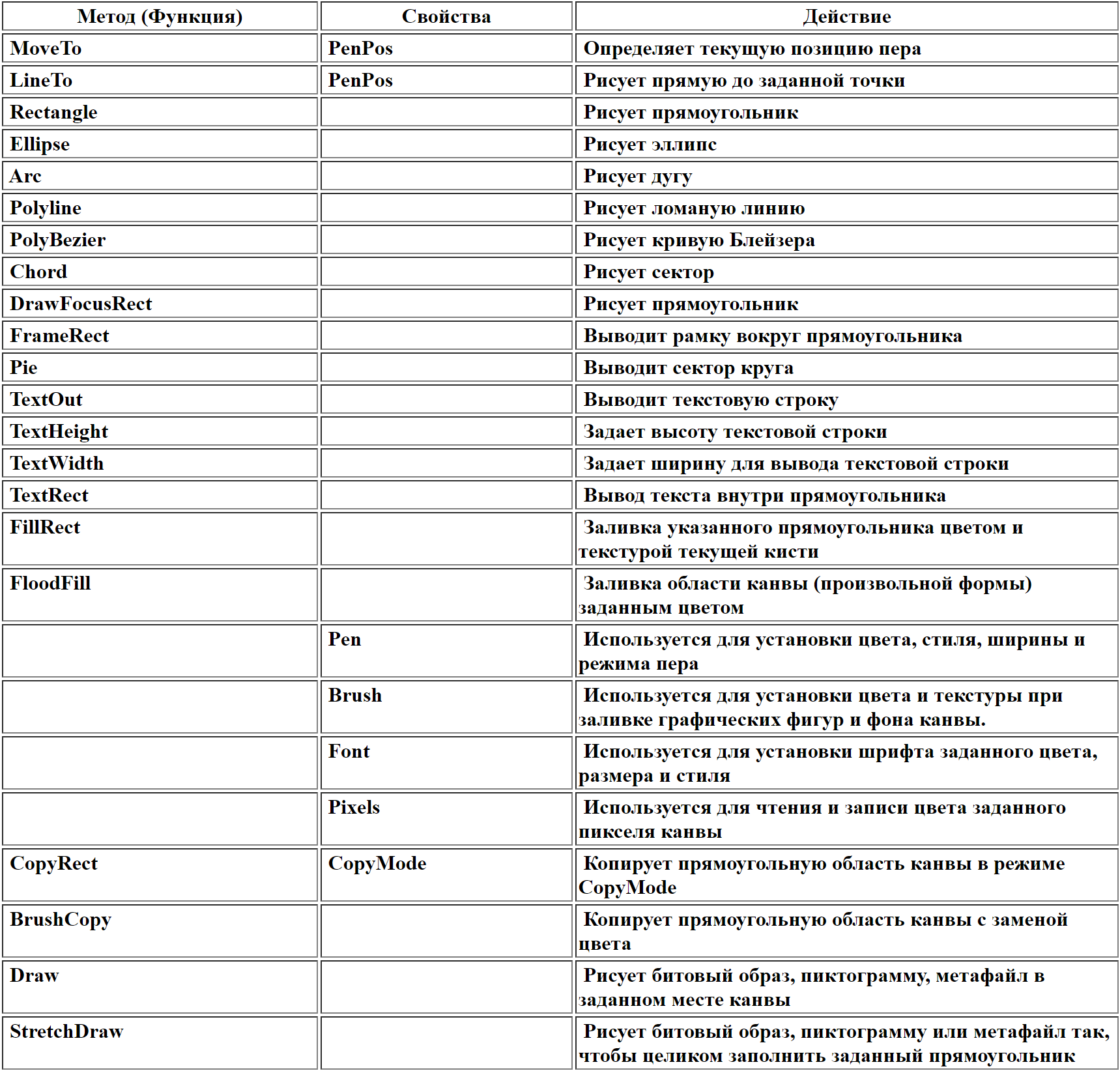
Свойства объекта Brush задают цвет и способ закраски области внутри прямоугольника, круга, сектора или замкнутого контура.



Шрифт, который используется для вывода текста, определяется значением свойства Font соответствующего объекта Canvas. Свойство Font представляет собой объект типа TFont. В таблице перечислены свойства объекта TFont, определяющие характеристики шрифта.



3 и 4) Функции канвы.



В том числе:

* функции рисования на канве: LineTo, Rectangle, Ellipse, Arc, Polyline, PolyBezier, Chord, DrawFocusRect, FrameRect, Pie, Draw, StretchDraw;
* функции заливки: FillRect, FloodFill.

5) События.

* OnClick

Обычно событие OnClick наступает, если пользователь нажал и отпустил левую кнопку мыши в то время, когда указатель мыши находился на компоненте. Кроме того, это событие происходит в следующих случаях:

- пользователь нажал клавишу пробела, когда кнопка или индикатор были в фокусе;

- пользователь нажал клавишу <Enter>, а активная форма имеет кнопку по умолчанию, указанную свойством Default;

- пользователь нажал клавишу <Esc>, а активная форма имеет кнопку прерывания, указанную свойством Cancel;

- пользователь нажал клавиши быстрого доступа к кнопке или индикатору;

- пользователь выбрал элемент в сетке, дереве, списке или выпадающем списке, нажав клавишу со стрелкой;

- приложение установило в True свойство Checked радиокнопки RadioButton:

- приложение изменило свойство Checked индикатора CheckBox;

- вызван метод Click элемента меню.

* OnPaint

Событие OnPaint наступает, когда возникает сообщение Windows о необходимости перерисовать испорченное изображение. Изображение может испортиться из-за временного перекрытия данного окна другим окном того же или постороннего приложения. Обработчик данного события должен перерисовать изображение.

* OnResize

Событие OnResize возникает при изменении размеров формы во время работы приложения. Обработчик этого события необходим только в том случае, если планируется выполнять какие-либо действия при изменении размеров формы.

* OnMouseDown и OnMouseUp

Событие OnMouseDown наступает в момент нажатия пользователем кнопки мыши над компонентом. Имеется также парное к нему событие OnMouseUp, наступающее при отпускании нажатой кнопки мыши над объектом.

Обработка событий OnMouseDown и OnMouseUp используется для операций, требуемых при нажатии и отпускании пользователем какой-либо кнопки мыши. Обработчики этих событий имеют параметры Sender, Shift, Button, X и Y. Значения параметра Button определяют, какая кнопка мыши нажата: mbLeft — левая, mbRight — правая, mbMiddle — средняя. Параметры X и Y определяют координаты указателя мыши в клиентской области компонента.

* OnMouseMove

Событие OnMouseMove наступает при перемещении курсора мыши над компонентом. Обработчик события OnMouseMove вставляется в программу, если необходимо произвести какие-то операции при перемещении курсора мыши над компонентом. Параметр Shift, являющийся множеством, содержит элементы, позволяющие определить, какие кнопки мыши и какие вспомогательные клавиши (<Shift>, <Ctrl> или <Alt>) нажаты в этот момент. Параметры X и Y определяют координаты указателя мыши в клиентской области компонента.

6) Объекты типа TPoint, TRect.

Тип TPoint – это тип записи, содержащий целочисленные значения X и Y. Он обычно используется для хранения двумерных координат (например, запись координат точки).

Тип TRect является записью, содержащей значения координат X и Y точек верхнего левого и нижнего правого углов прямоугольника.

7) Методы Form1->Refresh() и ::InvalidateRect(…).

Метод Refresh приводит к немедленной перерисовке изображения на экране. Refresh вызывает метод Repaint. Методы Refresh и Repaint взаимозаменяемы.

InvalidateRect(…) указывает прямоугольник для перерисовки окна:

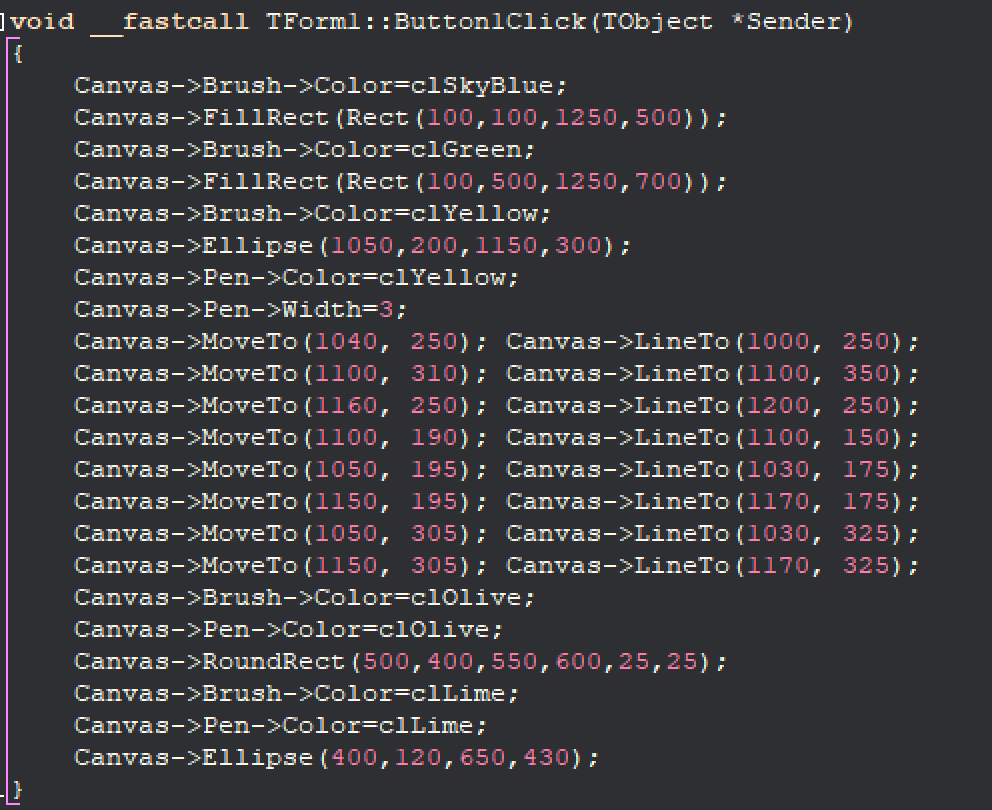
BOOL InvalidateRect (HWND hWnd, CONST RECT \*lpRect, BOOL bErase);

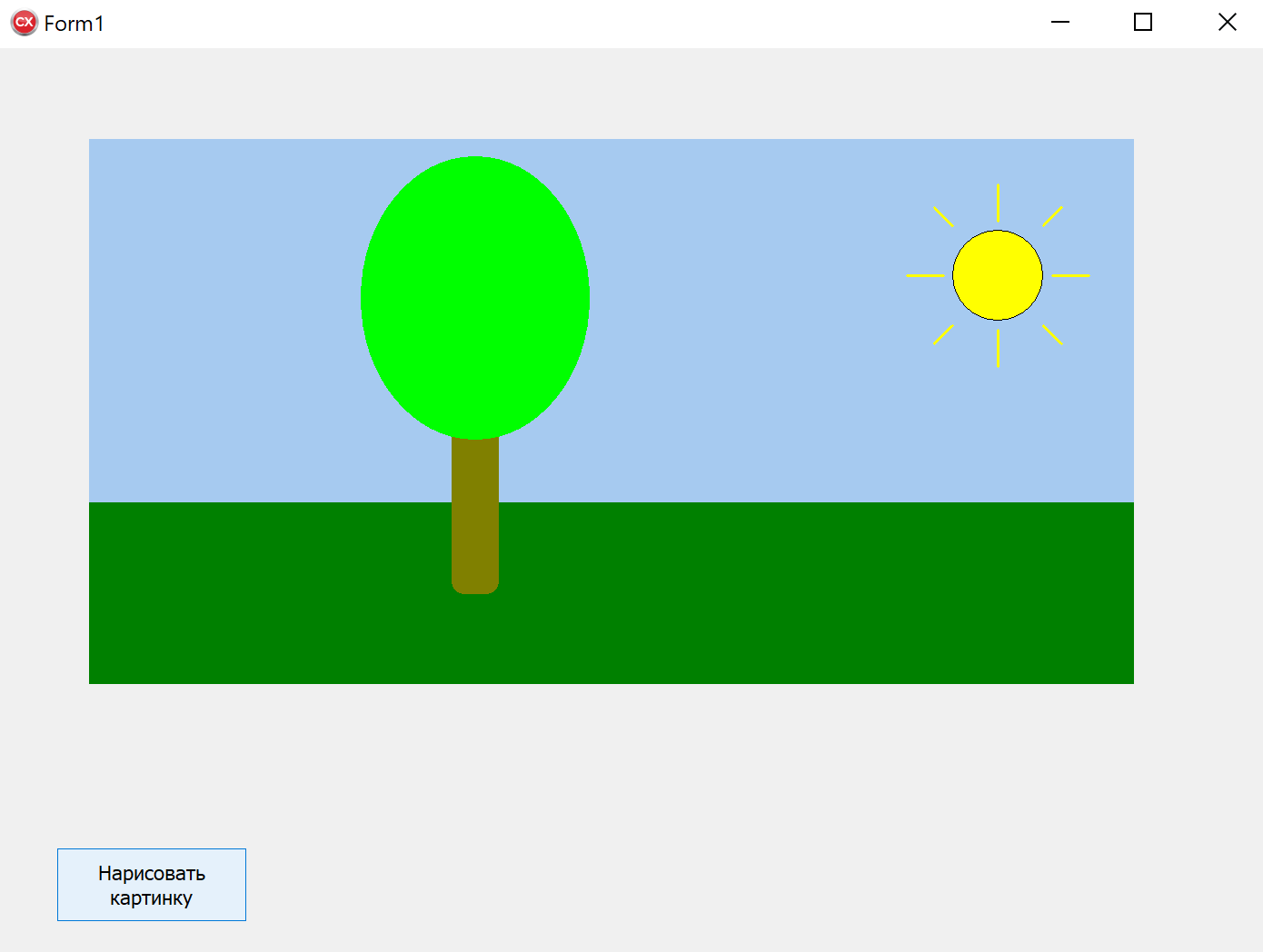
Если первый параметр NULL, то будут перерисованы все окна. Если второй параметр NULL, то будет перерисована вся область окна. Последний параметр указывает на действия с фоном, если TRUE, то фон будет стерт, иначе оставлен. При удачном выполнении этой функции результат отличен от нуля.

Указанные области перерисовки накапливаются, пока не будет получено сообщение WM\_PAINT или пока эта функция не будет отменена противоположной функцией ValidateRect(). Само сообщение WM\_PAINT посылается при необходимости изменить область, но только когда нет в очереди других сообщений. При установке нескольких областей производится их объединение и обработка одним WM\_PAINT.

**2. Выполнение работы.**

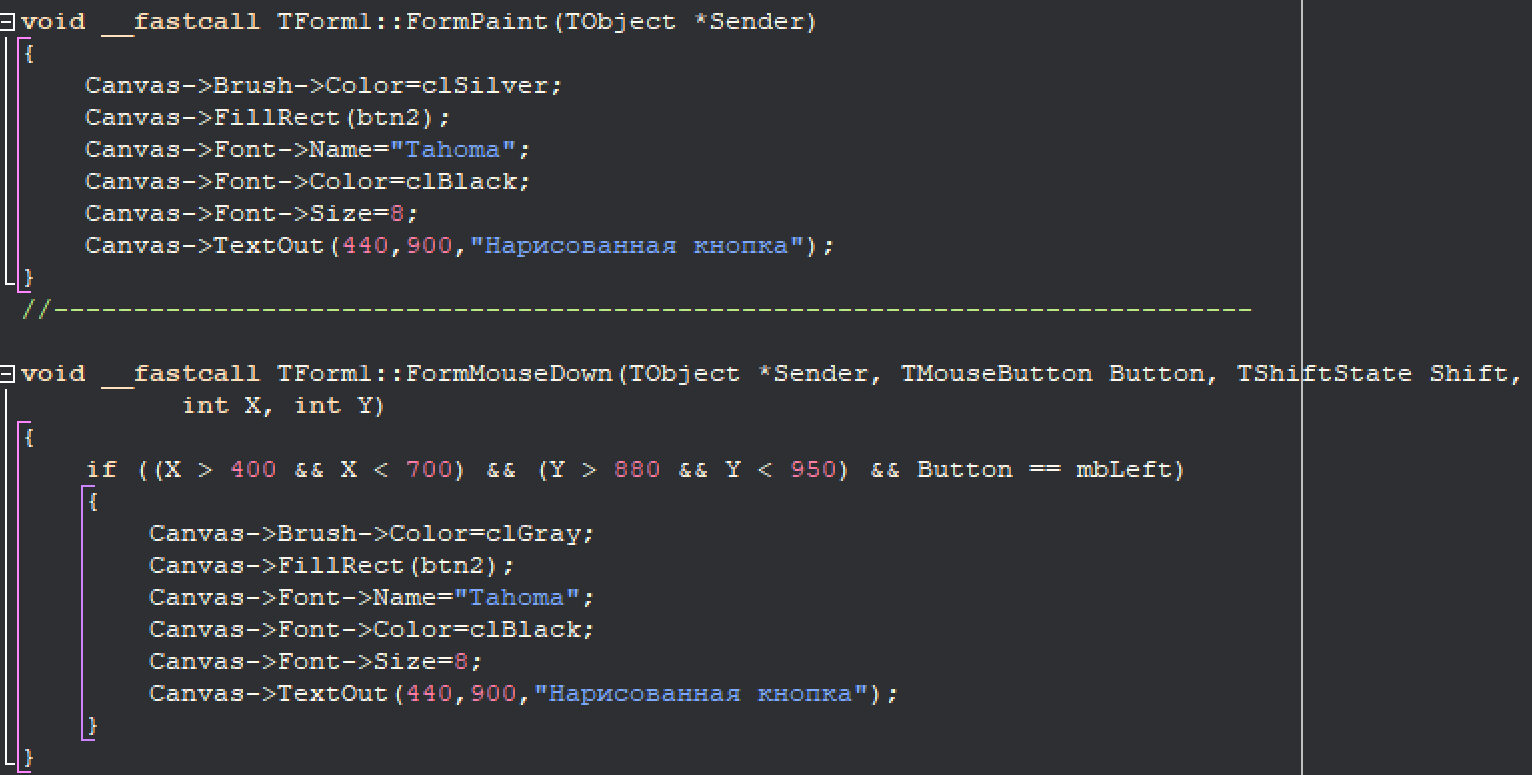
1. По щелчку на кнопке 1 нарисовать на канве формы картинку с использованием всевозможных графических примитивов, карандашей, кистей и заливок (элементы картинки должны составлять единый сюжет).

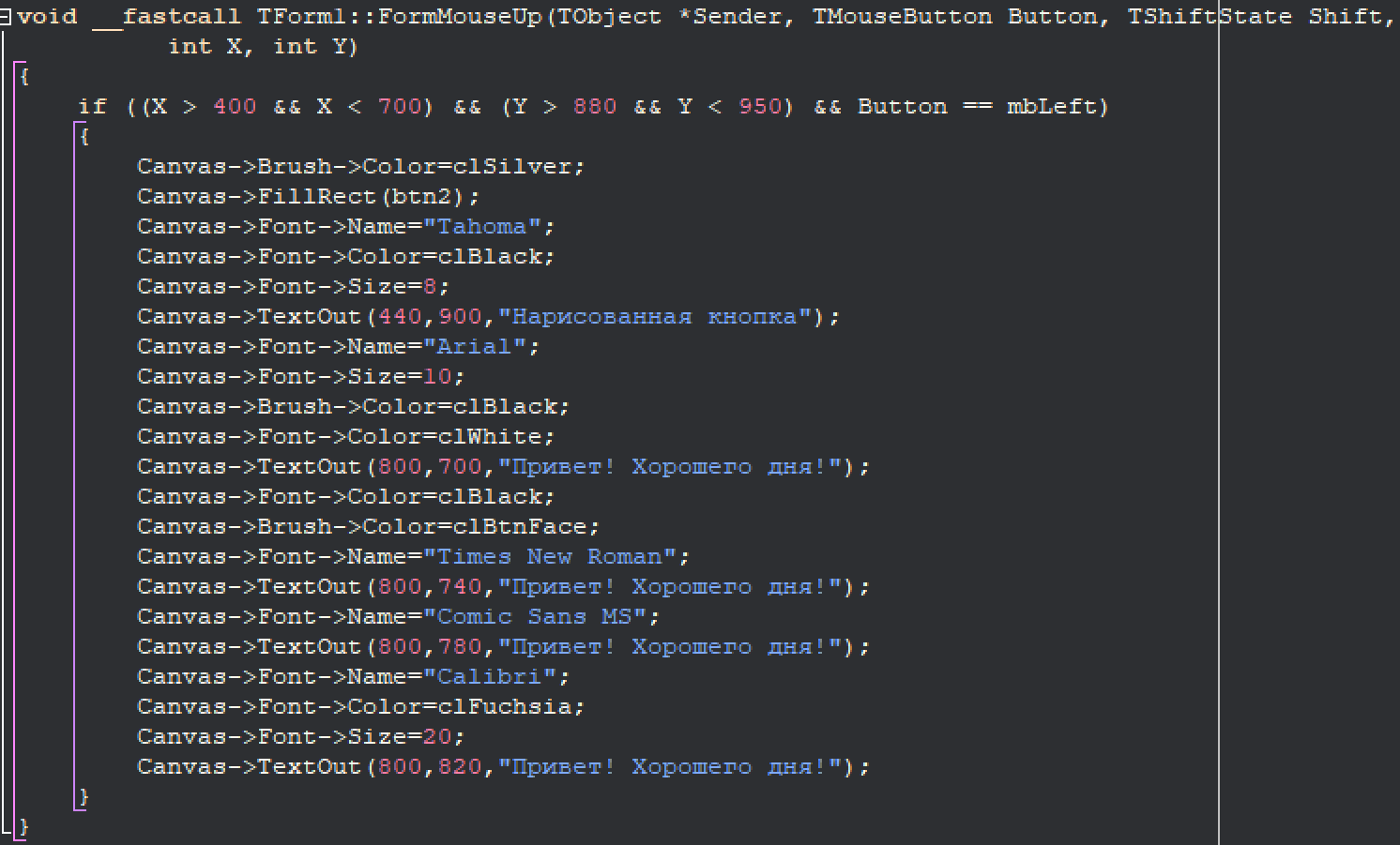


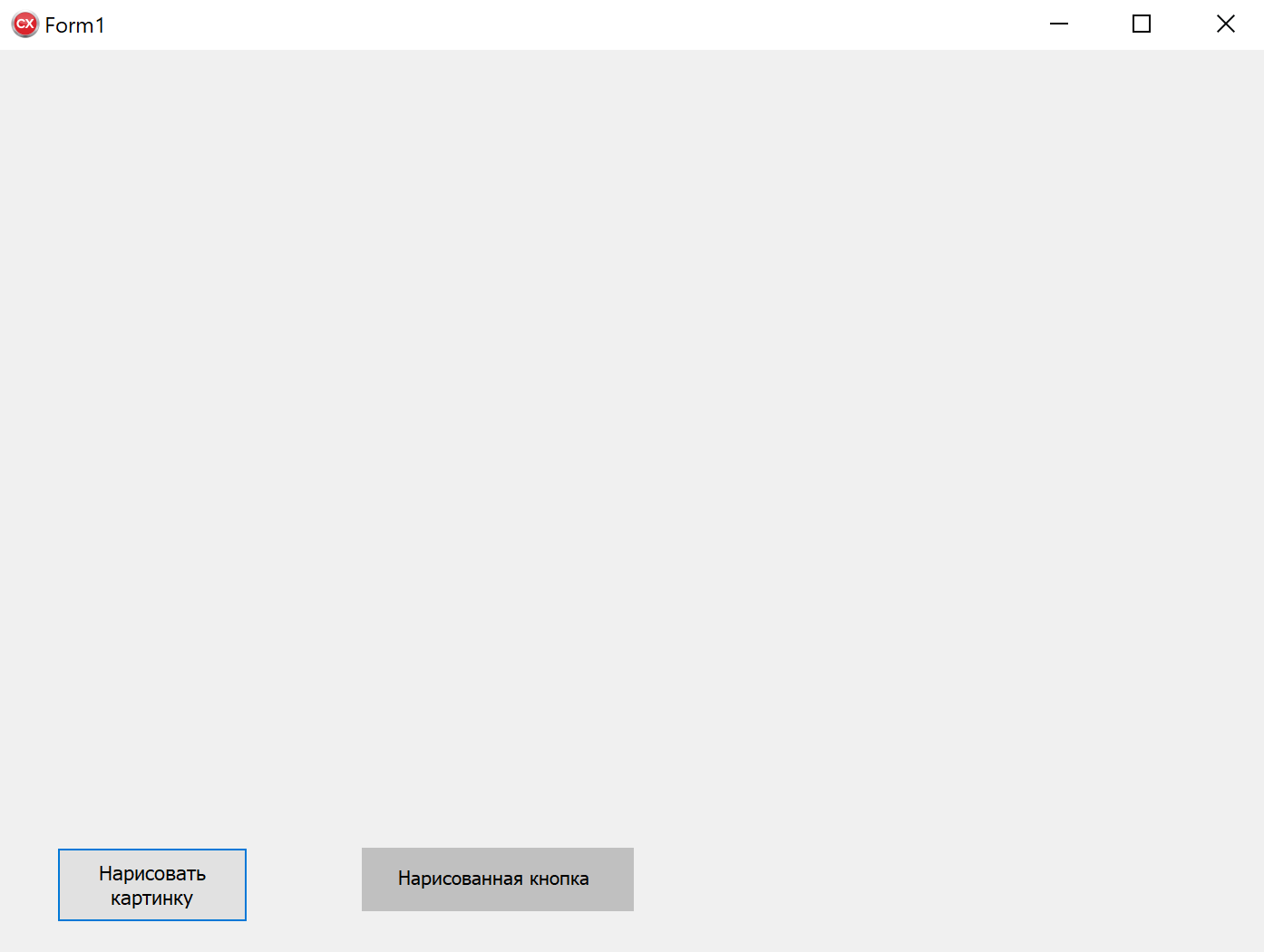


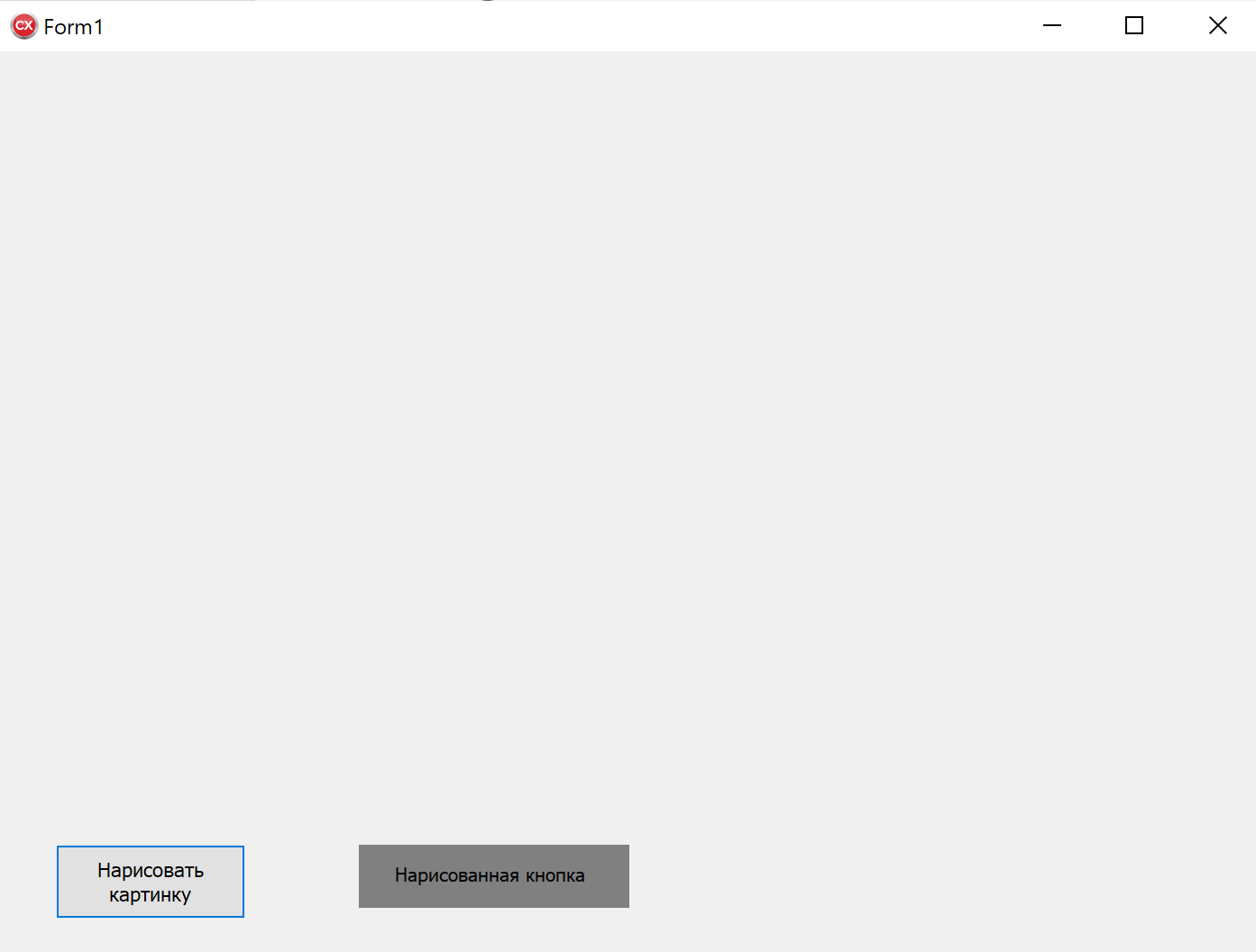
1. Нарисовать на канве формы кнопку (это должен быть рисунок, выполненный карандашом, кистью и графическими функциями). Пользователь должен воспринимать рисунок как элемент управления. При щелчке мышью по кнопке она должна мигнуть и должен выполниться код пункта 3.

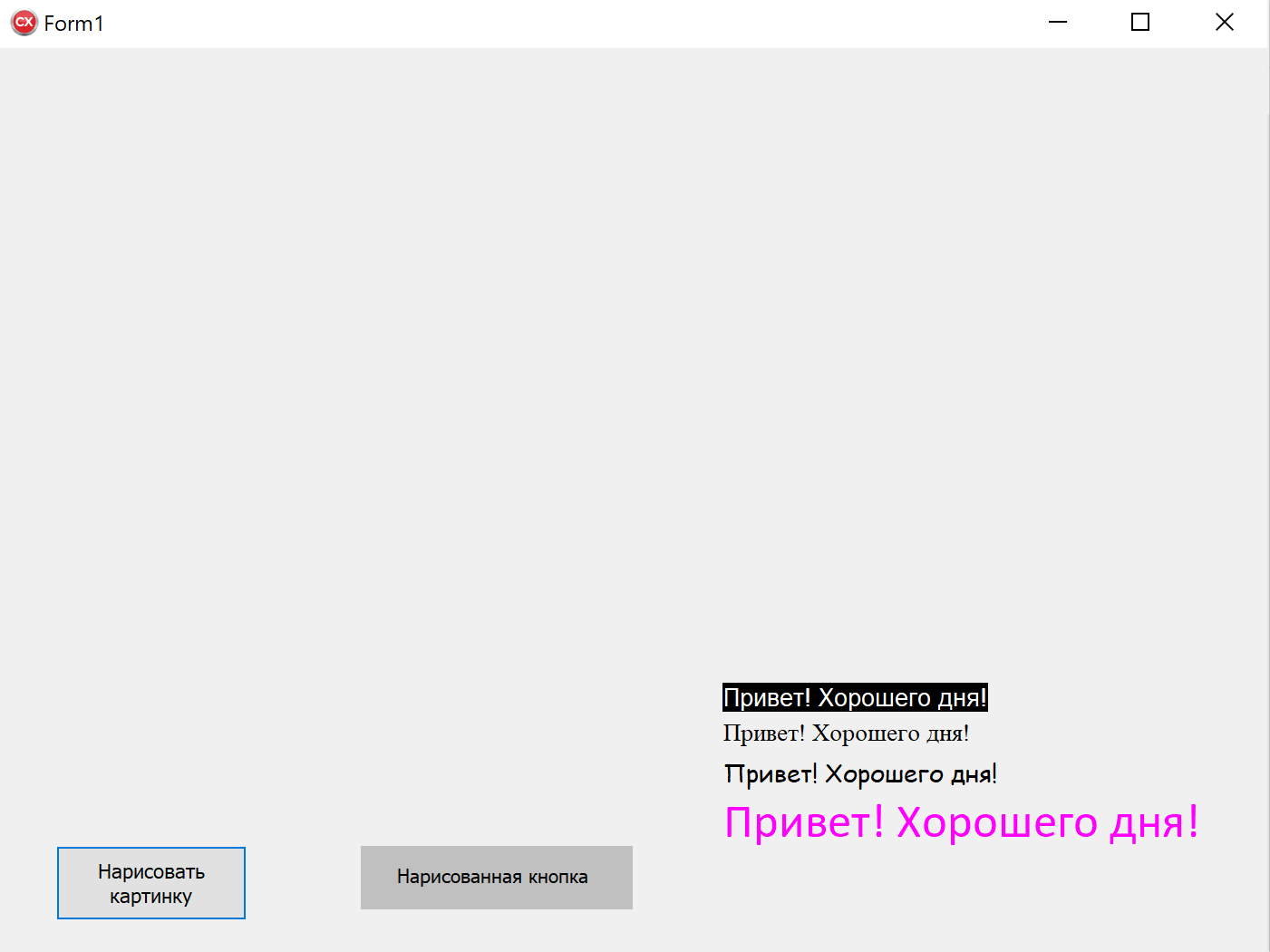
TRect btn2 = Rect(400,880,700,950);









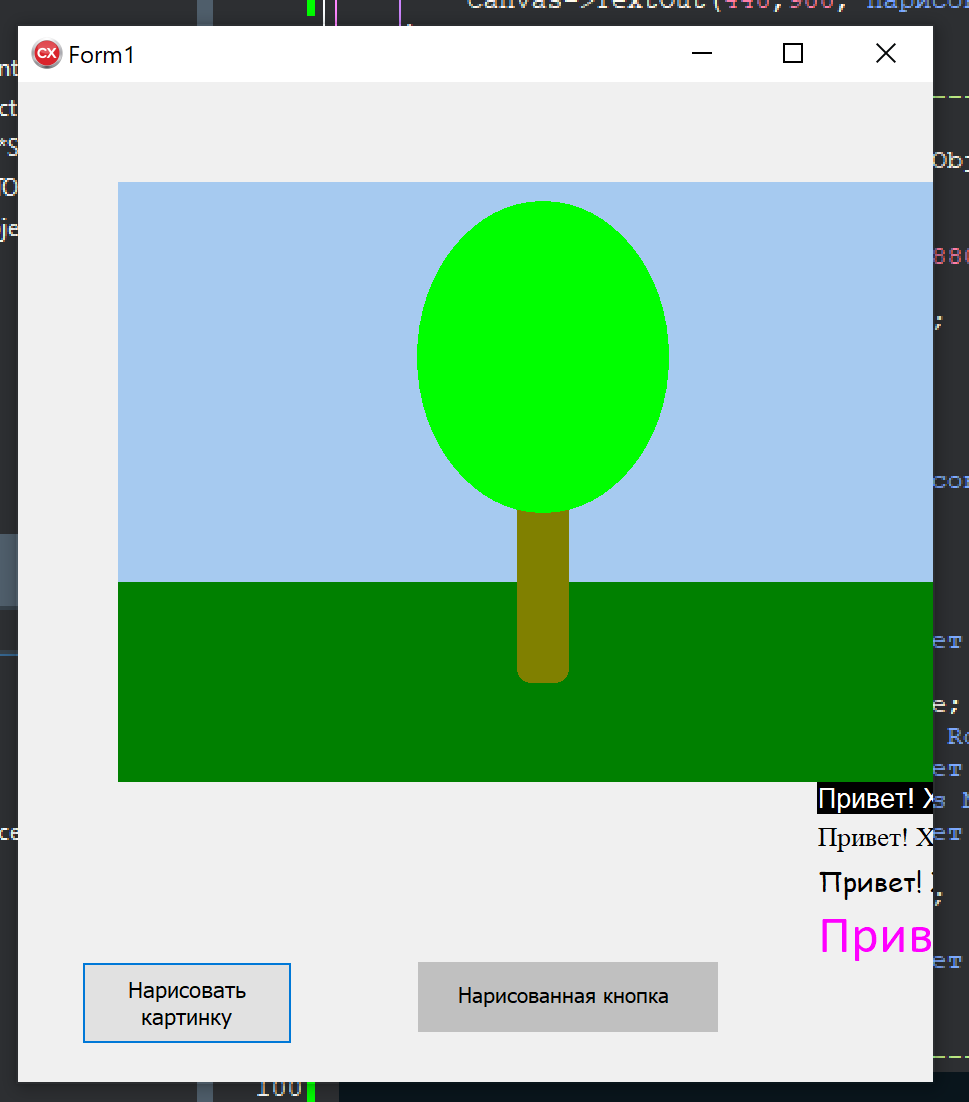
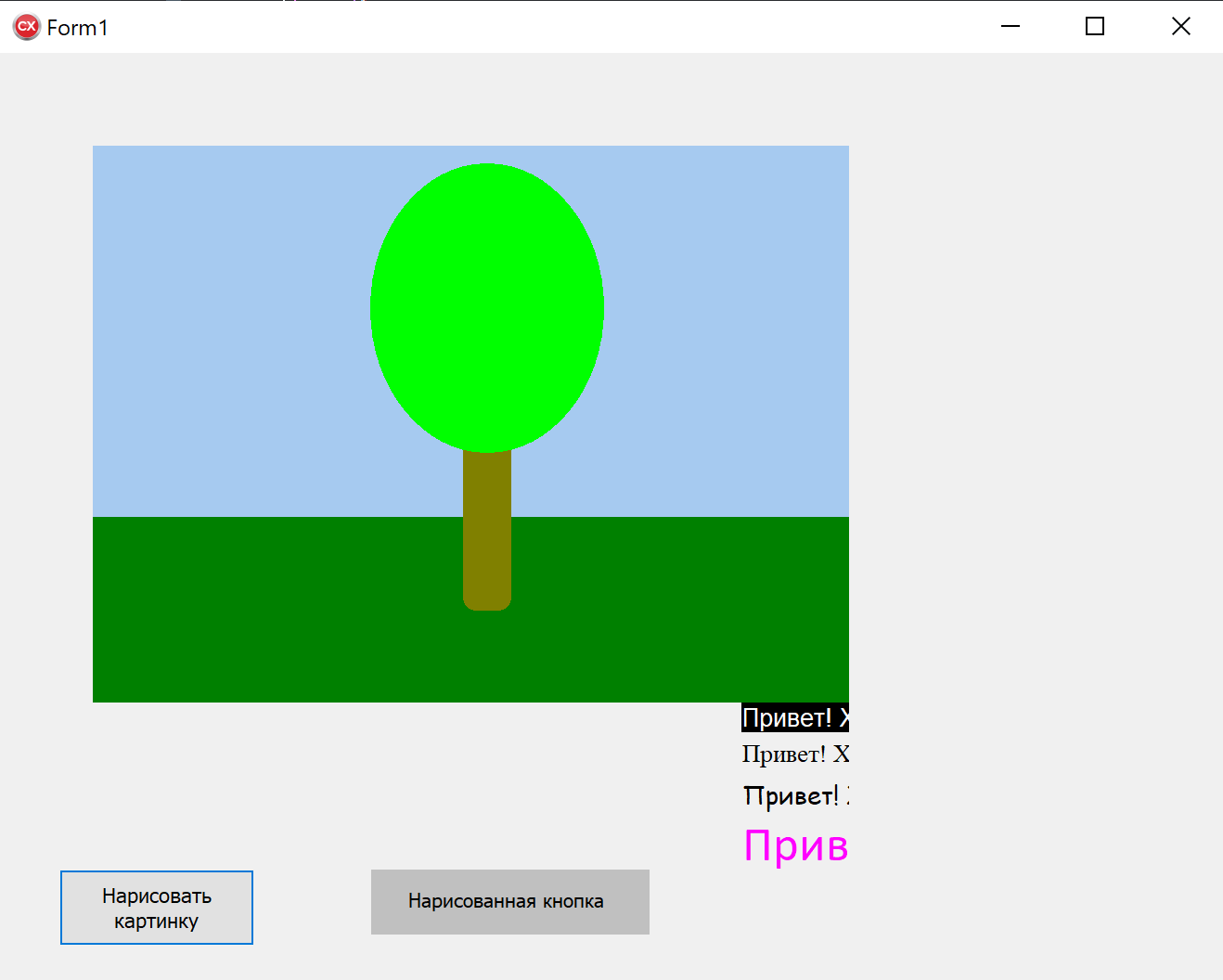


1. По щелчку на кнопке создать на канве формы другое изображение (график функции или вывод текста различными шрифтами и цветами).

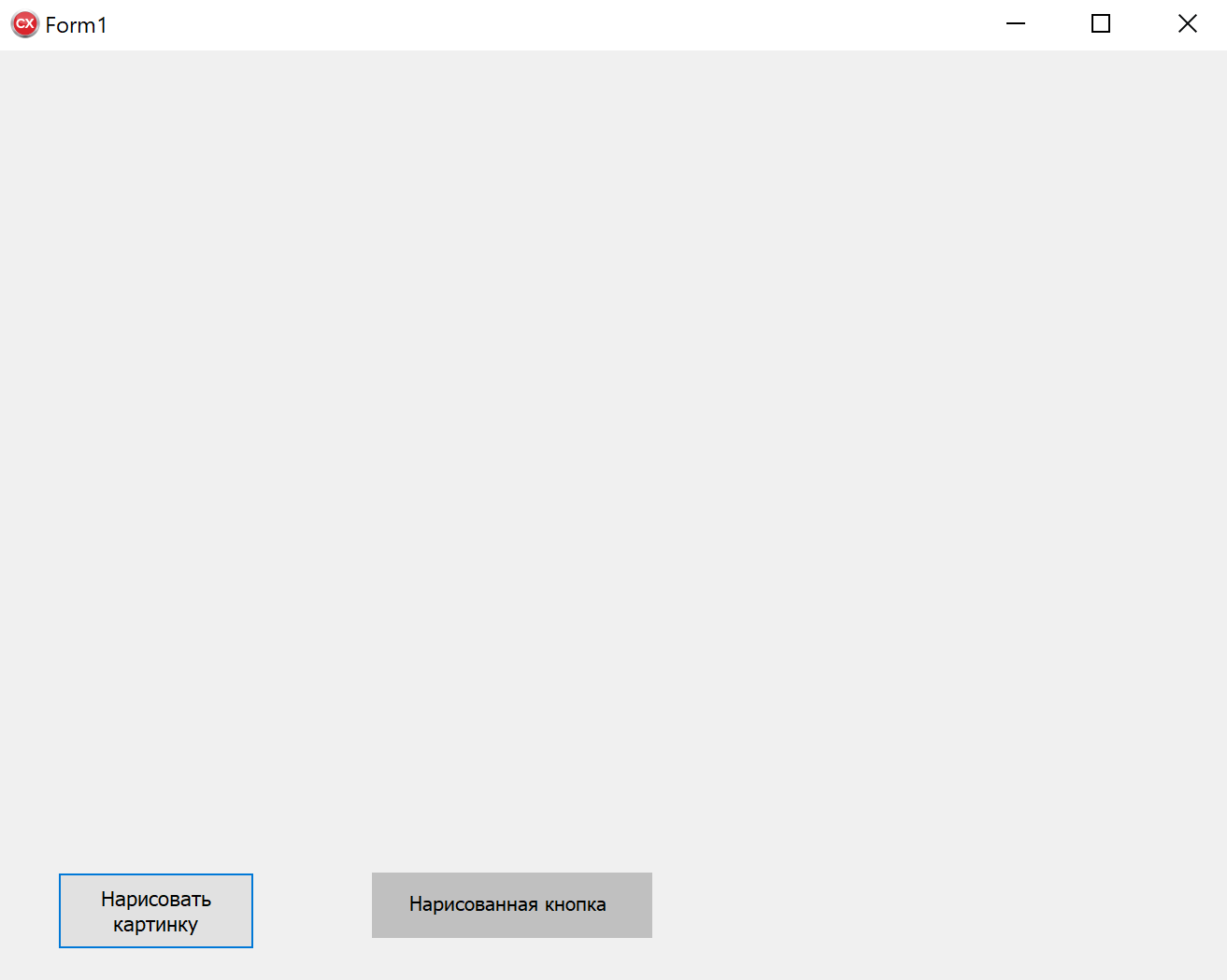
Код и скриншоты см. в предыдущем пункте.

1. Убедиться, что после изменения размеров окна (а также при сворачивании/разворачивании) картинка исчезает.

При изменении размеров окна картинка просто обрезается:

После сворачивания и разворачивания окно возвращается в исходное состояние:

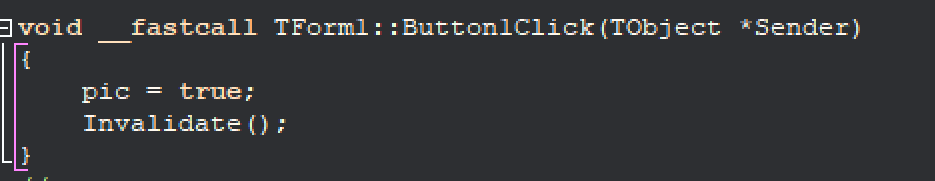


1. Перераспределить код приложения таким образом, чтобы картинки не исчезали с экрана.

Для этого завел 2 флага (для картинки и для текста):

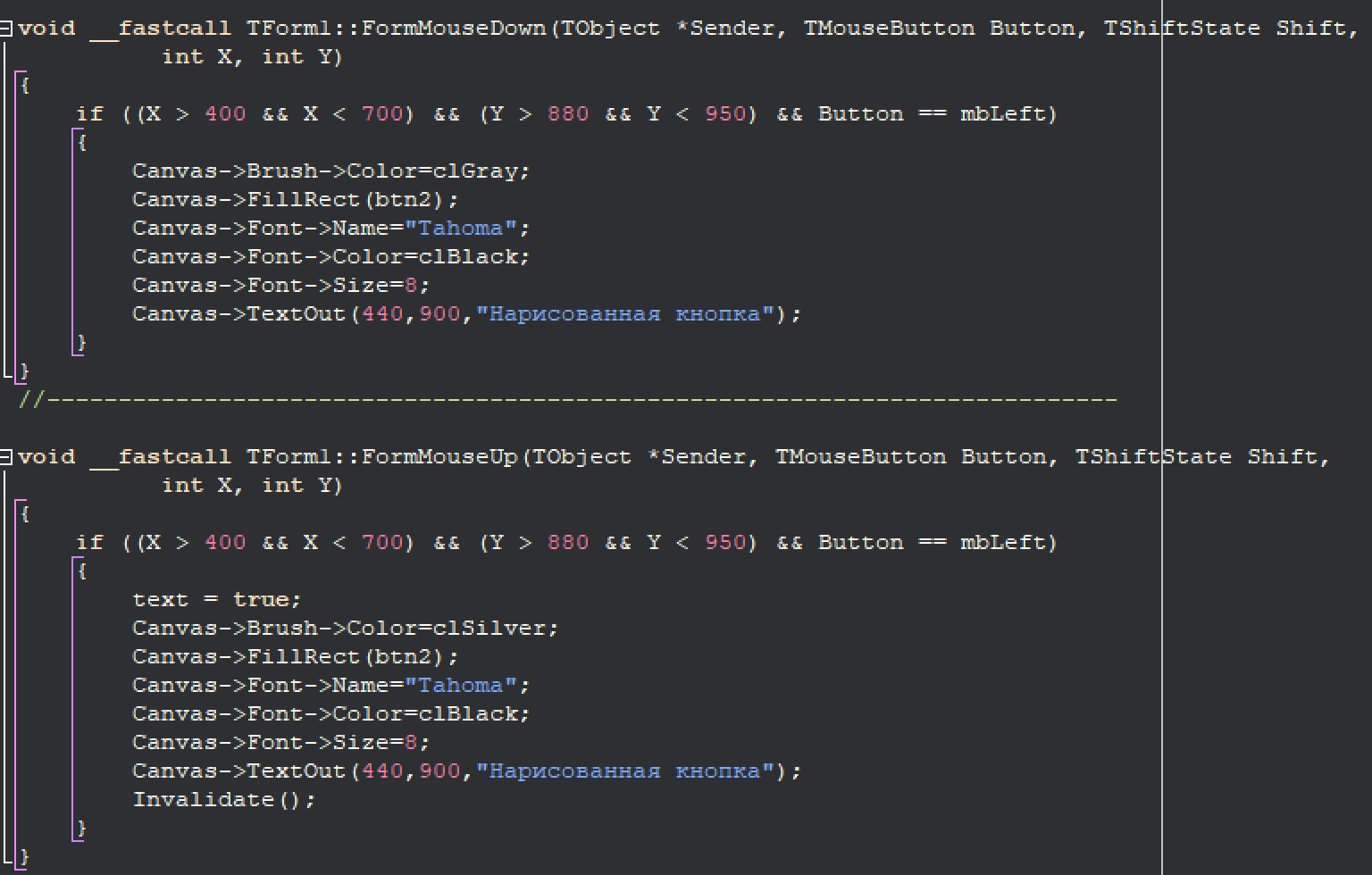
bool pic = false, text = false;

Изменил реакцию на событие нажатия кнопки:

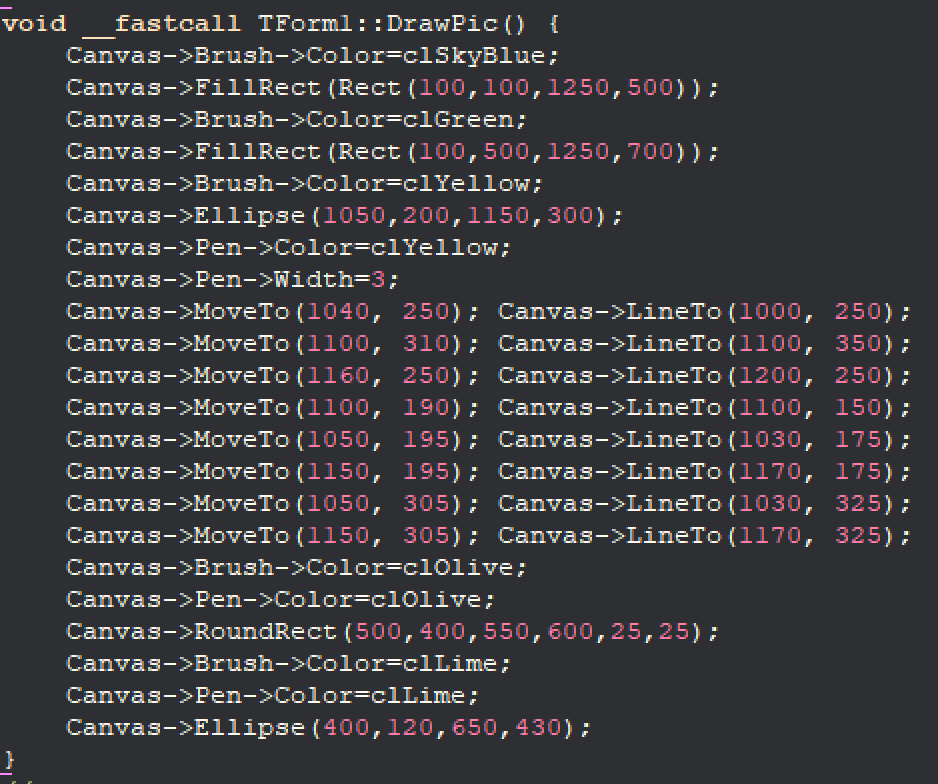


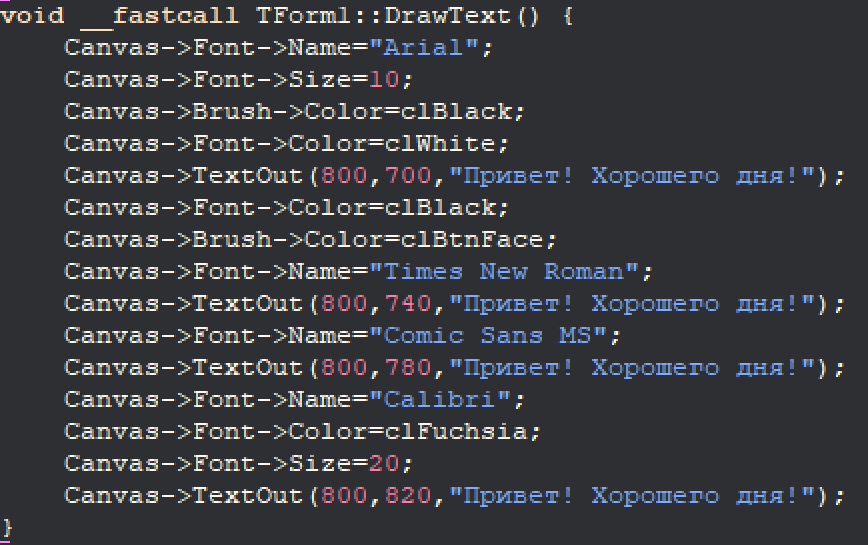
Invalidate() вызываю для перерисовки окна (вызов реакции на событие OnPaint).

Изменил события нажатия и отжатия кнопки мыши (в нашем случае левой):

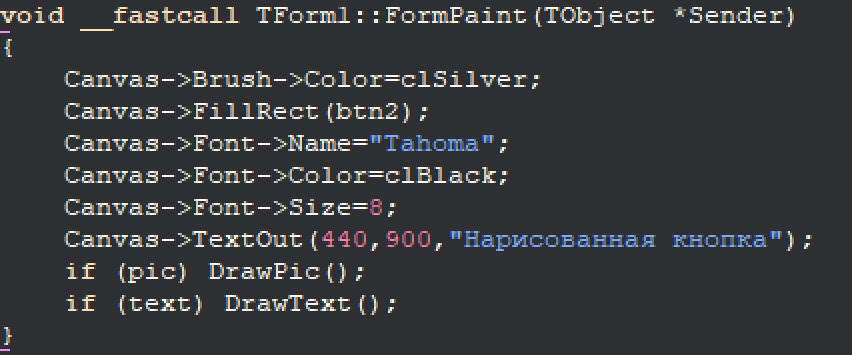


И выделил в отдельные функции отрисовку изображения и текста:





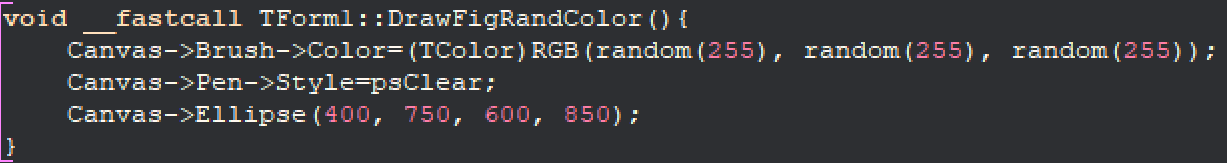
И наконец, отредактировал реакцию на событие OnPaint, добавив флаги:



1. Добавить к программе функции (можно добавить еще две кнопки):

* Нарисовать геометрическую фигуру случайным цветом (так чтобы фигура не исчезала) и проверить, что происходит. Объяснить, почему при изменении размеров окна появляются цветные полосы. Продумать, как избавиться от такого эффекта.

Функция для отрисовки фигуры с заливкой случайным цветом:



Добавил кнопку и написал реакцию на клик мышью по ней:

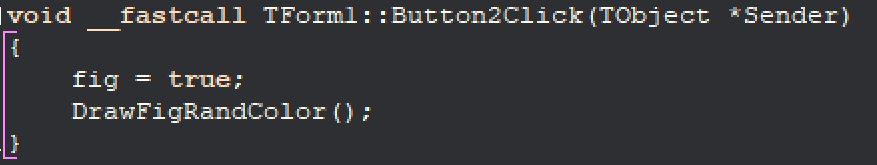
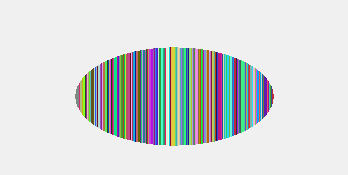


fig – это еще один флаг, теперь уже для фигуры (bool fig = false;)

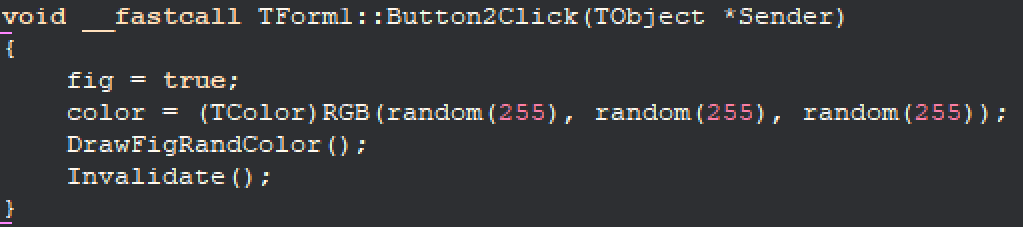
И добавил 1 строчку в FormPaint: if (fig) DrawFigRandColor();

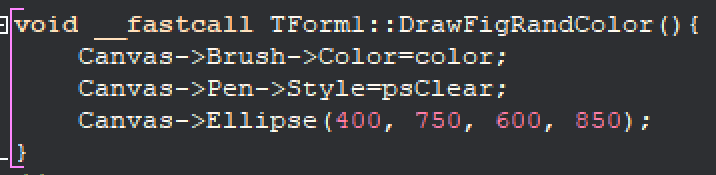
При изменении размера окна, когда фигура остается за рамками клиентской области, а потом возвращается в нее, наблюдается эффект цветных полос:



Это происходит, так как Windows перерисовывает только «испорченные» области, при этом каждый раз генерируя случайный цвет. Оптимальное решение – сохранять цвет фигуры при нажатии на кнопку и использовать его в дальнейшем при перерисовке.

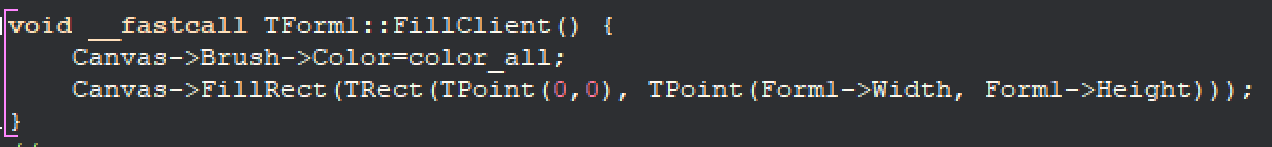
Объявляем переменнную: TColor color;

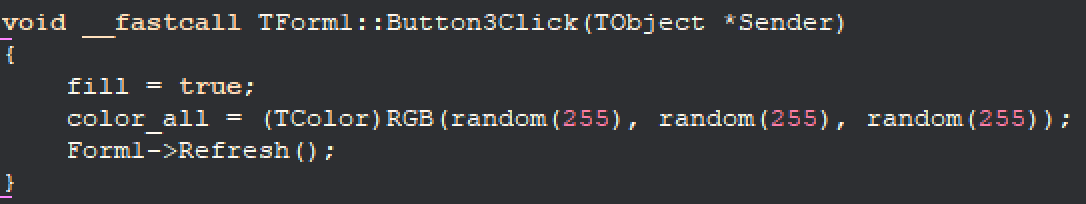




* Окрасить случайным цветом всю клиентскую область окна и, убедившись, что при изменении размеров окна появляются полоски, избавиться от подобного эффекта.

Работаем так же, как и с фигурой – запоминаем цвет формы до изменения размера: TColor color\_all;



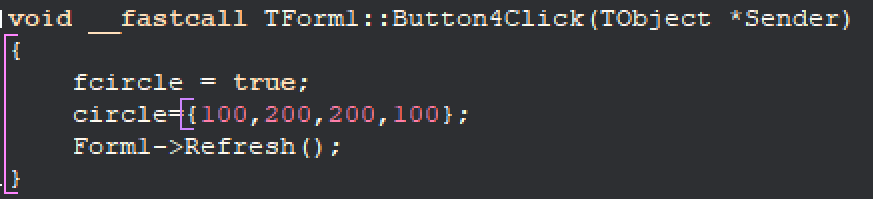


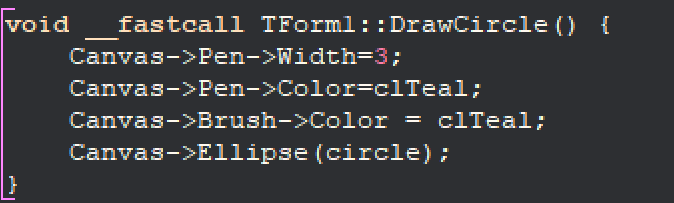
1. Добавить к приложению следующие функции:

* По щелчку на кнопке рисуем круг и заставляем его исчезнуть, если щелкаем мышью внутри круга.

TRect circle;

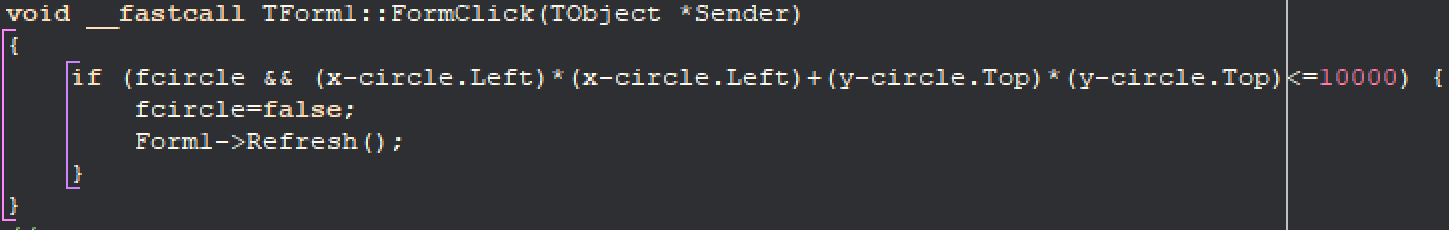
bool fcircle = false;



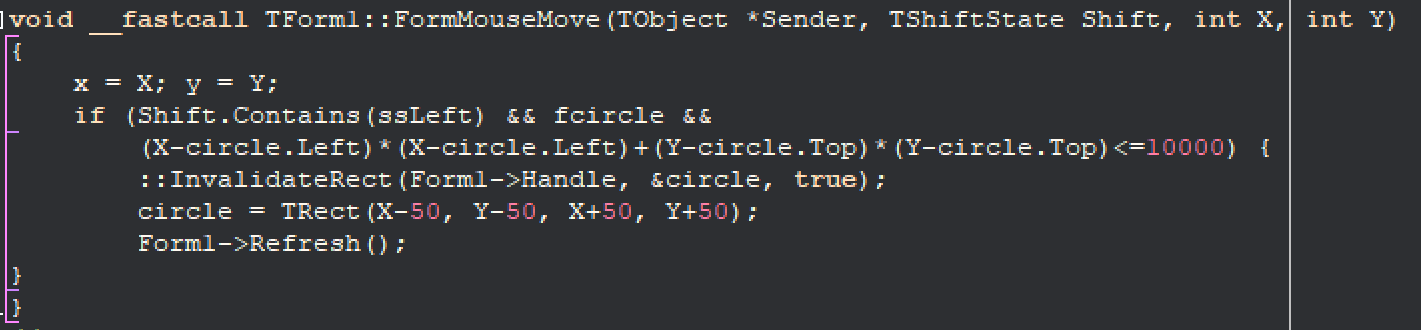


Добавляем в FormPaint: if (fcircle) DrawCircle();

А в FormClick добавляем код для удаления круга при щелчке внутри него:



* Круг рисуется по событию OnPaint. С помощью мыши круг перемещаем по поверхности формы.



Полный код программы доступен по ссылке: <https://github.com/programzan/SystemProgramming/blob/lab4/Unit1.cpp>

Здесь же размещены все сопутствующие файлы: <https://github.com/programzan/SystemProgramming/tree/lab4>