МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет прикладной математики, информатики и механики

Кафедра программного обеспечения  
и администрирования информационных систем

Направление 02.03.03 Математическое обеспечение  
и администрирование информационных систем

**Отчет**

по учебной практике (проектной)

Тема Работа с графическими компонентами на языке C#

Обучающийся 2 курс, 91 группа, Шелухин Д.В.

Руководитель от кафедры Меджидов Р.Г.

ВОРОНЕЖ

2025

# Введение

Работа посвящена изучению возможностей языка программирования C# при работе с графикой.

Целью является изучение классов и структур языка C# для работы с графикой (Graphics, Color, Brush и др.), элементы формы для работы с графикой и взаимодействия с пользователем (PictureBox и др.), методы рисования на Graphics, а также работы со звуком (SoundPlayer). В ходе работы будет создана программа в Windows Forms.

Результатом является получение новых теоретических знаний и практических навыков.

# 1. Теоретическая часть

## 1.1. Классы и структуры языка C# для работы с графикой

В работе были рассмотрены основные классы и структуры языка C#   
для работы с графикой. Они позволяют создавать и редактировать графические инструменты в Windows Forms.

### 1.1.1. Graphics

Класс Graphics предоставляет методы для рисования объектов на устройстве отображения. Этот класс не наследуется.

С помощью Graphics объекта можно нарисовать множество различных фигур, линий, изображений и значков. Кроме того, можно управлять системой координат, используемой Graphics объектом.

Примеры методов класса:

* Clear() — очищает поверхность рисования и заливает ее цветом, указанным в качестве параметра;
* DrawArc() — рисует дугу;
* DrawCurve() — рисует кривую;
* DrawClosedCurve() — рисует замкнутую кривую, конец которой будет соединен с началом кривой;
* DrawEllipse() — рисует контур эллипса;
* DrawLine() — рисует линию по двум точкам;
* DrawLines() — рисует серию линий по массиву точке;
* DrawPolygon() — рисует многоугольник по массиву точек;
* DrawRectangle() — рисует прямоугольник;
* FillEllipse() — заливает цветом эллипс;
* FillPolygon() — заливает цветом многоугольник;
* FillRectangle() — заливает цветом прямоугольник.

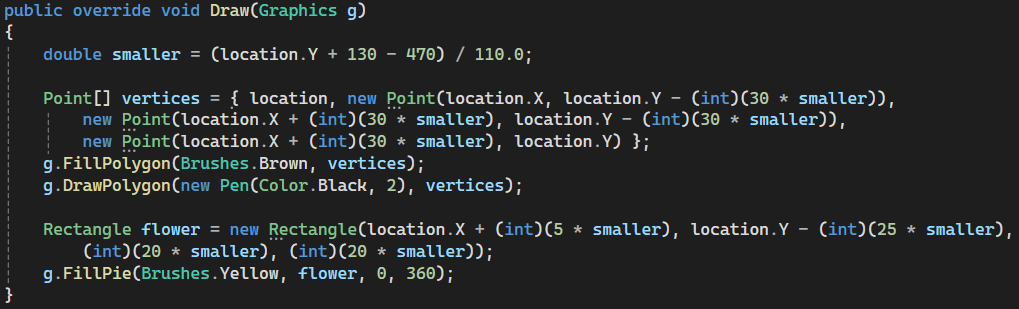


Рис. 1.1.1.1. Пример рисования с помощью Graphics

### 1.1.2. Color

Структура Color представляет цвета в RGB-формате. Она используется для определения цвета заливки, контура и текста.

Примеры методов класса:

* FromName() — создает цвет по имени;
* FromRgb() — создает цвет по компонентам RGB;
* ToInt() — возвращает числовое представление цвета;
* AddLuminosity() — увеличивает яркость цвета.

### 1.1.3. Brush

Класс Brush определяет объекты, которые используются для заливки внутри графических фигур, таких как прямоугольники, эллипсы, круги, многоугольники и дорожки. Этот класс является абстрактным, самым распространённым наследником является класс SolidBrush.

Примеры методов класса:

* Clone() — создаёт точную копию кисти;
* ToString() — возвращает строковое представление кисти.

### 1.1.4. Pen

Класс Pen определяет объект, используемый для рисования прямых и кривых линий. В отличие от класса Brush, Pen предназначен для рисования контура фигуры, а не для заливки фигуры. Этот класс не наследуется. Позволяет задавать цвет, толщину и стиль контура.

Примеры методов класса:

* Clone() — создаёт точную копию пера;
* ToString() — возвращает строковое представление пера.

Примеры свойств класса:

* Color — цвет пера;
* Width — ширина лини;
* LineJoin — способ соединения линий.

### 1.1.5. Image

Абстрактный класс Image, который предоставляет функциональные возможности для производных классов Bitmap и Metafile. Представляет собой изображение.

Примеры методов класса:

* FromFile() — создает объекта из файла;
* Save() — сохраняет изображение в файл в указанном формате;
* Clone() — копирует изображение;
* SetPropertyItem() — устанавливает метаданные изображения.

### 1.1.6. Icon

Класс Icon представляет значок Windows — это небольшое растровое изображение, используемое для визуального представления объектов, окон или приложений в интерфейсе Windows.

Есть возможность создания из файла, потока и готовой иконки, а также преобразования в Bitmap для использования в элементах управления.

Примеры методов класса:

* ToBitmap() — преобразует иконку в объект Bitmap;
* Save() — сохраняет иконку в указанный поток;
* Clone() — копирует иконку;
* FromHandle() — создает иконку из дескриптора нативного значка.

## 1.2. Элементы формы для работы с графикой и взаимодействия с пользователем

В данной работе были изучены элементы формы, предназначенные для работы с графикой и обеспечения взаимодействия с пользователем в приложениях Windows Forms.

### 1.2.1. PictureBox

Элемент PictureBox является контейнером для отображения графических объектов. PictureBox позволяет рисовать на своей поверхности с помощью класса Graphics, а также загружать изображения из файлового хранилища.

Примеры методов класса:

* Load() — загружает изображение из свойства ImageLocation;
* Refresh() — перерисовывает PictureBox.

Примеры свойств класса:

* Image — изображение в PictureBox;
* ImageLocation — строка с путем к изображению;
* SizeMode — способ отображения изображения в PictureBox;
* BorderStyle — стиль рамки вокруг PictureBox.



Рис. 1.2.1.1. Пример установки изображения в pictureBox

### 1.2.2. ColorDialog

Класс ColorDialog представляет общее диалоговое окно, в котором отображаются доступные цвета и элементы управления, позволяющие пользователю определять собственные цвета через цветовую модель RGB. Также есть настройки контраста, оттенка и яркости.

Примеры методов класса:

* ShowDialog() — отображает окно выбора цвета;
* Reset() — сбрасывает окно к настройкам по умолчанию.

Примеры свойств класса:

* Color — выбранный цвет;
* AllowFullOpen — разрешает или запрещает создавать цвета пользователю;
* AnyColor — указывает, отображать ли все доступные цвета в базовом наборе.

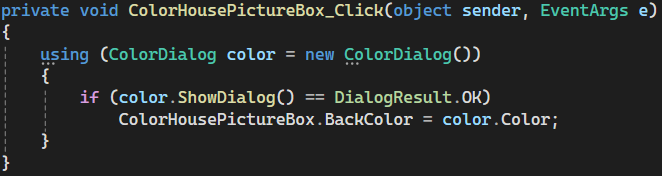


Рис. 1.2.2.1. Пример выбора цвета через ColorDialog

## 1.3. Работа с курсором

Класс Cursor предназначен для управления изображением указателя мыши в приложениях Windows Forms. Он позволяет задавать положение, внешний вид и область применения курсора. Можно создавать кастомный курсор через Graphics, а также устанавливать готовый через указание расположения файла типа .cur.

Класс Cursor не предназначен для изменения размера курсора — размер определяется самим файлом курсора или системными настройками.  
В дополнение, через статический класс Cursors доступна коллекция стандартных курсоров Windows: стрелка, крестик, рука и другие.

Примеры свойств класса:

* Size — размер курсора;
* HotSpot — фокус курсора.

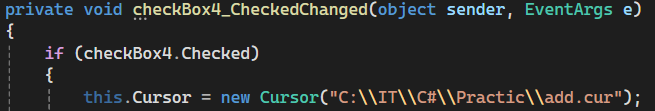


Рис. 1.3.1. Пример установки готового курсора при наступлении события

## 1.4. Работа со звуком

В данной работе была изучена работа со звуками с использованием класса SoundPlayer в языке C#. Класс SoundPlayer предоставляет простой способ воспроизведения звуковых файлов типа .wav в приложениях Windows Forms.

Примеры методов класса:

* Load() — загружает звук в синхронном режиме;
* Play() — воспроизводит звук асинхронно один раз;
* PlayLooping() — воспроизводит звук в цикле.

Примеры свойств класса:

* SoundLocation— хранит путь к звуковому файлу;
* Stream — поток, из которого загружается звук;
* LoadTimeout — время ожидания загрузки звука;
* IsLoadCompleted — указывает, завершена ли загрузка звука.



Рис. 1.4.1. Пример подключения звуков к проекту

# 2. Практическая часть

## 2.1. Постановка задачи

Общее условие: Здания нарисованы на PictureBox с помощью примитивов вроде отрезков, точек, многоугольников, кривых… Каждое здание имеет такую характеристику, как количество жильцов. Выводить суммарное число жителей населенного пункта. Объекты следует добавлять  
по координатам курсора по щелчку. Также следует учитывать перспективу. Добавить воспроизведение звуков при клике в зависимости от успешности добавления или, для некоторых задач, от типа действия и состояния здания. Использовать собственный нестандартный курсор в процессе добавления. Предоставить пользователю выбор цветов зданий (или отдельных элементов) при помощи ColorDialog.

Здания можно добавлять и удалять. Также можно добавлять объекты благоустройства, например, клумбу, фонарь или качели.

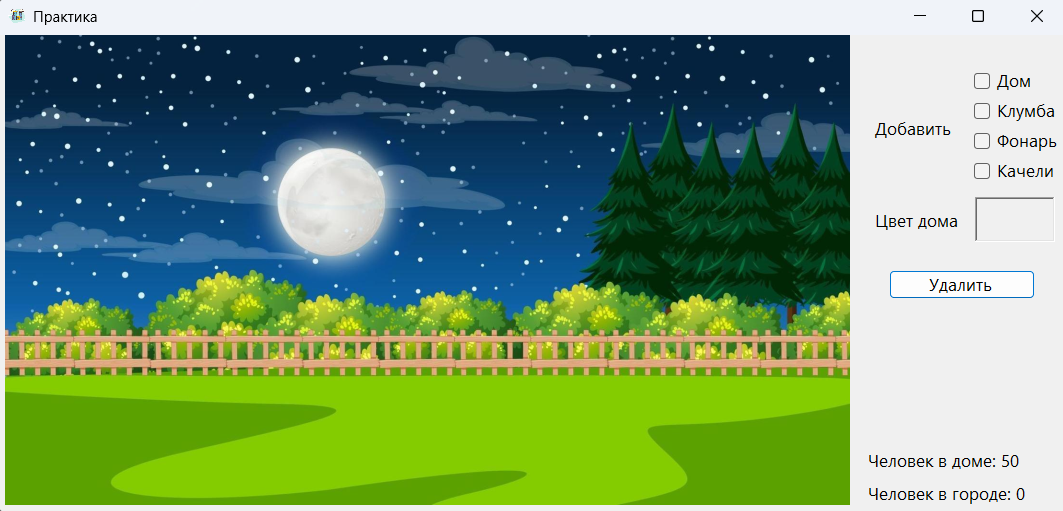
## 2.3. Структура программы

Для удобства внутри проекта выделено несколько модулей, помимо основной программы, каждый из которых содержит подпрограммы, выполняющие определенные группы действий:

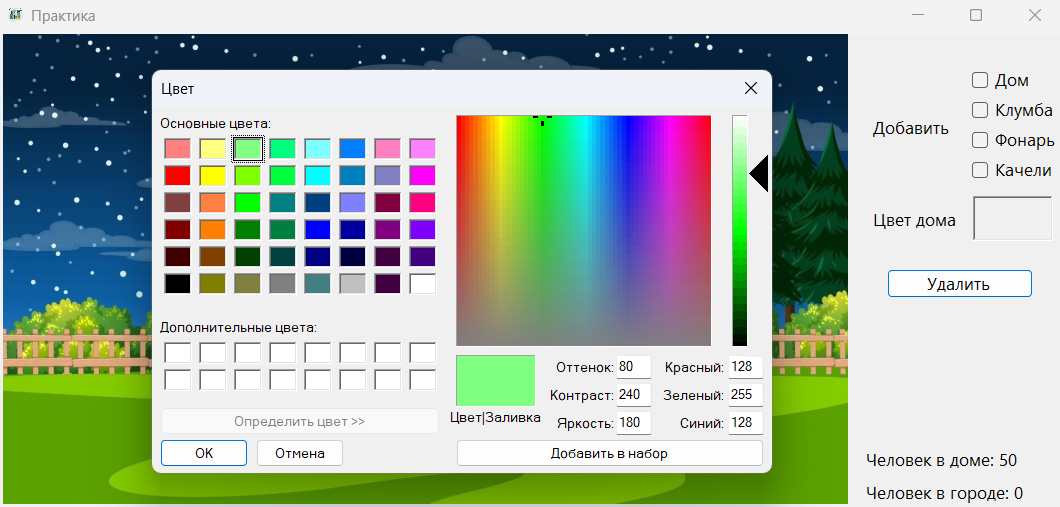
1. Модуль для работы с объектами благоустройства и домами. Должен содержать абстрактный класс и наследников. Каждый объект должен иметь координаты места рисования и своей формы, метод, рисующий этот объект на элемент отображения, и метод, показывающий входит ли заданная точка в область данного объекта.
2. Модуль для работы с главной формой: позволяет добавлять объекты по щелчку мыши, изменять цвет домов и вид курсора, воспроизводить звук и выводить на форму характеристики.

## 2.4. Интерфейс

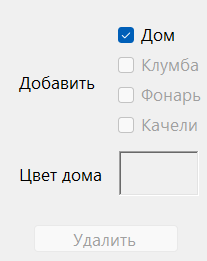
При запуске программы пользователь видит интерфейс, представленный на рисунке 2.4.1.

  
Рис. 2.4.1. Главный экран программы

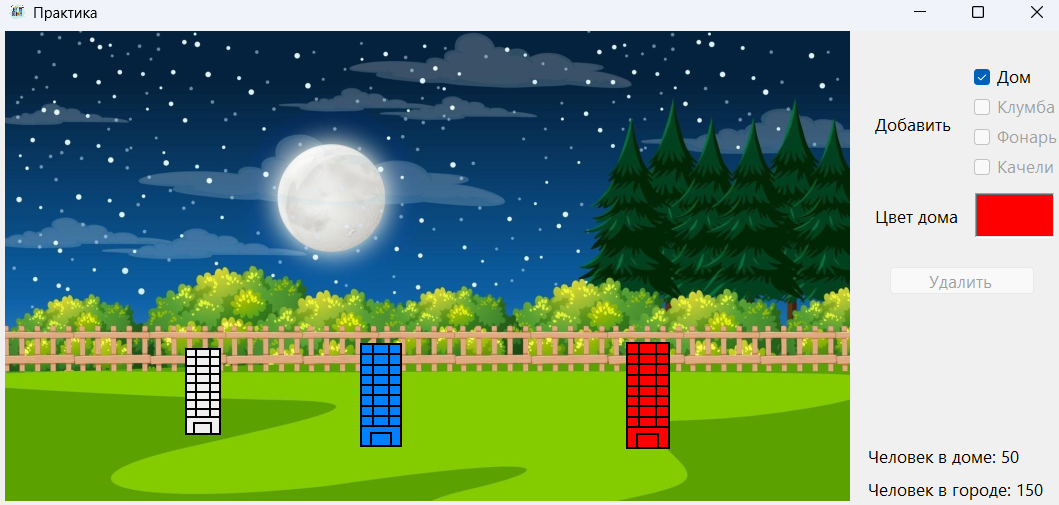
При нажатии на область, которая отвечает за цвет дома, всплывает окошко ColorDialog, позволяющее пользователю выбрать новый цвет. Он изменится после нажатия кнопки «ОК».

  
Рис. 2.4.2. Выбор цвета через ColorDialog

При нажатии на кнопки добавления или удаления остальные элементы формы, кроме области рисования, становятся недоступны для нажатия  
до момента повторного нажатия ранее выбранного элемента. Данное правило не распространяется на цвет дома, то есть при выборе рисования домов смена их цвета всегда доступна.

  
Рис. 2.4.3. Пример блокировки невыбранных элементов

В нижнем правом углу формы указано количество человек в городе  
и в одном доме. При добавлении домов количество жителей обновляется  
на форме.

  
Рис. 2.4.4. Пример вывода количества человек в городе

При добавлении объектов стандартный курсор меняется на кастомный  
в виде круга и плюса в нем зеленого цвета. При удалении курсор меняется  
на круг и минус в нем красного цвета.

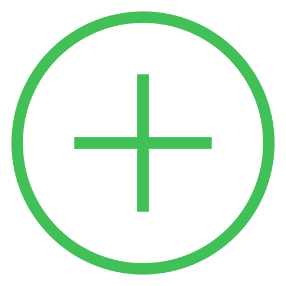
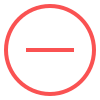
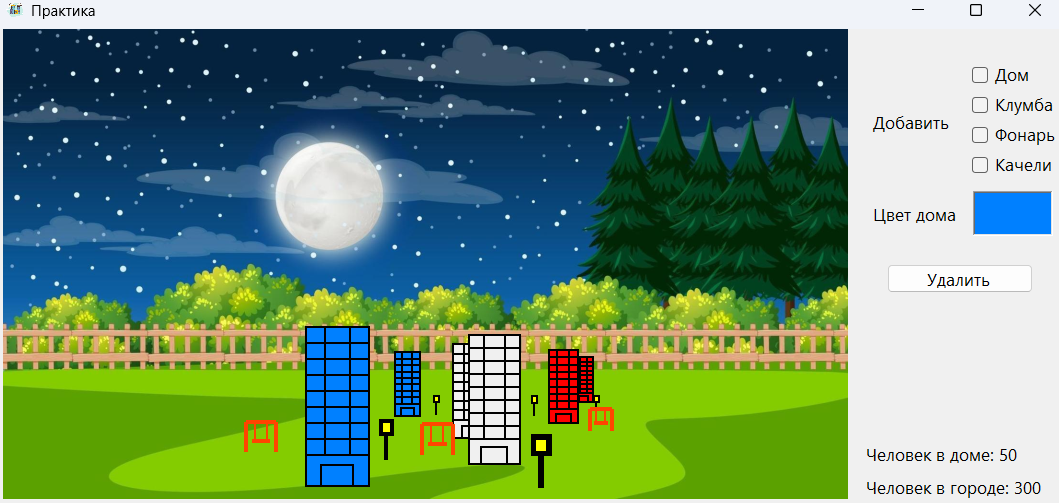
 

Рис. 2.4.5. Курсор добавления Рис. 2.4.6. Курсор удаления

При нажатии курсором на PictureBox на нём рисуется или удаляется выбранный объект. При этом происходит проверка на возможность удаления или добавления и воспроизводится соответствующий звук. В дополнение,  
на PictureBox учитывается перспектива объектов.

  
Рис. 2.4.7. Пример объектов с перспективой

## 2.5. Реализация

### 2.5.1. Модуль для работы с домами и объектами благоустройства

В данном модуле содержится абстрактный класс DrawObject с несколькими наследниками:

* House;
* StreetLight;
* Swing;
* FlowerBed.

Класс DrawObject содержит следующие методы:

* Draw(Graphics g) — на g рисуется объект одного из наследников;
* IsVisible(Point location) — проверяет вхождение заданной точки  
  в область объекта.

Свойства класса DrawObject:

* Location — координаты точки, в которой объект был нарисован;
* Path — область объекта.

### 2.5.3. Модуль для работы с главной формой

В данном модуле содержатся следующие методы:

* DelButton\_Click — обработчик события нажатия на кнопку «Удалить», блокирующий остальные элементы, которые можно нажать, и меняющий курсор;
* HouseCheckBox\_CheckedChanged — обработчик события отметки CheckBox для добавления дома, блокирующий остальные элементы, которые можно нажать, и меняющий курсор;
* FlowerBedCheckBox\_CheckedChanged — обработчик события отметки CheckBox для добавления клумбы, блокирующий остальные элементы, которые можно нажать, и меняющий курсор;
* StreetLightCheckBox\_CheckedChanged — обработчик события отметки CheckBox для добавления фонаря, блокирующий остальные элементы, которые можно нажать, и меняющий курсор;
* SwingCheckBox\_CheckedChanged — обработчик события отметки CheckBox для добавления качелей, блокирующий остальные элементы, которые можно нажать, и меняющий курсор;
* ColorHousePictureBox\_Click — обработчик события нажатия  
  на PictureBox для выбора цвета дома;
* DrawingPictureBox\_MouseClick — обработчик события нажатия  
  на PictureBox для выбора места рисования или удаления домов  
  и объектов благоустройства, а также воспроизведения соответствующих звуков;
* DrawingPictureBox\_Paint — обработчик события Paint, рисующий выбранный объект на PictureBox.

# Заключение

Работа позволила исследовать возможности языка программирования C# для работы с графикой.

В результате выполнения поставленных задач были изучены классы и структуры, необходимые для работы с графикой, элементы формы  
для работы с графикой и взаимодействия с пользователем, методы рисования на Graphics, а также методы работы со звуками. В рамках работы было создано приложение Windows Forms, которое рисует и удаляет объекты на PictureBox.

# Список литературы

1. Вирт Н. Алгоритмы и структуры данных / Н. Вирт; пер. с англ. Д. Б. Подшиваловой. – 2-е изд., испр. – Санкт-Петербург: Нев. диалект, 2001. – 352 с.
2. Стиль, разработка, эффективность, отладка и испытание программ / Д. Ван Тассел; пер. с англ. Е. К. Масловского – Москва: Мир, 1985. – 334 с.
3. Искусство программирования / Д. Э. Вирт; пер. с англ. С. Г. Тригуб. – Вильямс, 2019. – 720 с.
4. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Дж. Рихтер. – Издательство Питер, 2013. - 928 c.
5. Элементы PictureBox. – URL: <https://metanit.com/sharp/windowsforms/4.16.php> (дата обращения: 09.07.2025).
6. Класс Graphics. – URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.graphics?view=windowsdesktop-7.0> (дата обращения: 09.07.2025).
7. Графическая подсистема C#. – URL: <https://stepik.org/lesson/344238/step/1> (дата обращения: 09.07.2025).

# Приложение 1 Листинг DrawObject.cs

using System;

using System.Collections.Generic;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practic

{

abstract internal class DrawObject

{

protected GraphicsPath Path = new GraphicsPath();

public Point Location = new Point();

abstract public void Draw(Graphics g);

public bool IsVisible(Point point)

{

return Path.IsVisible(point);

}

}

}

**Приложение 2  
Листинг House.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practic

{

internal class House: DrawObject

{

Color color;

public House(Point Location, Color color)

{

this.Location = Location;

this.color = color;

double smaller = (Location.Y + 130 - 470) / 110.0;

Point[] vertices = { Location, new Point(Location.X,   
Location.Y - (int)(150 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(150 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y) };

Path.AddPolygon(vertices);

}

public override void Draw(Graphics g)

{

double smaller = (Location.Y + 130 - 470) / 110.0;

SolidBrush background = new SolidBrush(color);

Point[] vertices = { Location, new Point(Location.X,  
Location.Y - (int)(150 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(150 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y) };

g.FillPolygon(background, vertices);

Point[] door = { new Point(Location.X + (int)(15 \* smaller), Location.Y), new Point(Location.X + (int)(15 \* smaller),  
Location.Y - (int)(20 \* smaller)), new Point(Location.X + (int)(45 \* smaller), Location.Y - (int)(20 \* smaller)), new Point(Location.X + (int)(45 \* smaller), Location.Y) };

g.DrawPolygon(new Pen(Color.Black, 2), door);

g.DrawPolygon(new Pen(Color.Black, 2), vertices);

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(120 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(120 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(90 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(90 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(60 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(60 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(30 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(30 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X +  
(int)(18 \* smaller), Location.Y - (int)(150 \* smaller)),  
new Point(Location.X + (int)(18 \* smaller), Location.Y - (int)(30 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X +  
(int)(42 \* smaller), Location.Y - (int)(150 \* smaller)),  
new Point(Location.X + (int)(42 \* smaller), Location.Y - (int)(30 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(135 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(135 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(105 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(105 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(75 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(75 \* smaller)));

g.DrawLine(new Pen(Color.Black, 2), new Point(Location.X, Location.Y - (int)(45 \* smaller)), new Point(Location.X + (int)(60 \* smaller), Location.Y - (int)(45 \* smaller)));

}

}

}

**Приложение 3  
Листинг FlowerBed.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practic

{

internal class FlowerBed: DrawObject

{

public FlowerBed(Point Location)

{

this.Location = Location;

double smaller = (Location.Y + 130 - 470) / 110.0;

Point[] vertices = { Location, new Point(Location.X,  
Location.Y - (int)(30 \* smaller)), new Point(Location.X + (int)(30 \* smaller), Location.Y - (int)(30 \* smaller)), new Point(Location.X + (int)(30 \* smaller), Location.Y) };

Path.AddPolygon(vertices);

}

public override void Draw(Graphics g)

{

double smaller = (Location.Y + 130 - 470) / 110.0;

Point[] vertices = { Location, new Point(Location.X,  
Location.Y - (int)(30 \* smaller)), new Point(Location.X + (int)(30 \* smaller), Location.Y - (int)(30 \* smaller)), new Point(Location.X + (int)(30 \* smaller), Location.Y) };

g.FillPolygon(Brushes.Brown, vertices);

g.DrawPolygon(new Pen(Color.Black, 2), vertices);

Rectangle flower = new Rectangle(Location.X + (int)(5 \* smaller), Location.Y - (int)(25 \* smaller), (int)(20 \* smaller), (int)(20 \* smaller));

g.FillPie(Brushes.Yellow, flower, 0, 360);

}

}

}

**Приложение 4  
Листинг StreetLight.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practic

{

internal class StreetLight:DrawObject

{

public StreetLight(Point Location)

{

this.Location = Location;

double smaller = (Location.Y + 130 - 470) / 110.0;

Point[] vertices = { new Point(Location.X - (int)(7 \* smaller), Location.Y), new Point(Location.X - (int)(7 \* smaller), Location.Y -  
(int)(50 \* smaller)), new Point(Location.X + (int)(13 \* smaller), Location.Y - (int)(50 \* smaller)), new Point(Location.X + (int)(13 \* smaller), Location.Y)};

Path.AddPolygon(vertices);

}

public override void Draw(Graphics g)

{

double smaller = (Location.Y + 130 - 470) / 110.0;

Point[] vertices = { Location, new Point(Location.X, Location.Y - (int)(30\*smaller)), new Point(Location.X - (int)(7\*smaller), Location.Y - (int)(30\*smaller)), new Point(Location.X - (int)(7\*smaller), Location.Y - (int)(50\*smaller)), new Point(Location.X + (int)(13\*smaller), Location.Y - (int)(50\*smaller)), new Point(Location.X + (int)(13\*smaller), Location.Y - (int)(30\*smaller)), new Point(Location.X + (int)(6\*smaller), Location.Y - (int)(30\*smaller)), new Point(Location.X + (int)(6\*smaller), Location.Y) };

g.FillPolygon(Brushes.Black, vertices);

Point[] light = { new Point(Location.X - (int)(2 \* smaller), Location.Y - (int)(35 \* smaller)), new Point(Location.X - (int)(2 \* smaller), Location.Y - (int)(45 \* smaller)), new Point(Location.X + (int)(8 \* smaller), Location.Y - (int)(45 \* smaller)), new Point(Location.X + (int)(8 \* smaller), Location.Y - (int)(35 \* smaller)) };

g.FillPolygon(Brushes.Yellow, light);

}

}

}

**Приложение 5  
Листинг Swing.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Practic

{

internal class Swing:DrawObject

{

public Swing(Point Location)

{

this.Location = Location;

double smaller = (Location.Y + 130 - 470) / 110.0;

Point[] vertices = { Location, new Point(Location.X, Location.Y - (int)(40 \* smaller)), new Point(Location.X + (int)(40 \* smaller), Location.Y - (int)(40 \* smaller)), new Point(Location.X + (int)(40 \* smaller), Location.Y)};

Path.AddPolygon(vertices);

}

public override void Draw(Graphics g)

{

double smaller = (Location.Y + 130 - 470) / 110.0;

g.DrawLine(new Pen(Color.OrangeRed, 4), Location,  
new Point(Location.X, Location.Y - (int)(40 \* smaller)));

g.DrawLine(new Pen(Color.OrangeRed, 4), new Point(Location.X, Location.Y - (int)(40 \* smaller)), new Point(Location.X + (int)(40 \* smaller), Location.Y - (int)(40 \* smaller)));

g.DrawLine(new Pen(Color.OrangeRed, 4), new Point(Location.X + (int)(40 \* smaller), Location.Y - (int)(40 \* smaller)), new Point(Location.X + (int)(40 \* smaller), Location.Y));

g.DrawLine(new Pen(Color.OrangeRed, 2), new Point(Location.X + (int)(12 \* smaller), Location.Y - (int)(40 \* smaller)), new Point(Location.X + (int)(12 \* smaller), Location.Y - (int)(15 \* smaller)));

g.DrawLine(new Pen(Color.OrangeRed, 2), new Point(Location.X + (int)(28 \* smaller), Location.Y - (int)(40 \* smaller)), new Point(Location.X + (int)(28 \* smaller), Location.Y - (int)(15 \* smaller)));

g.DrawLine(new Pen(Color.OrangeRed, 4), new Point(Location.X + (int)(8 \* smaller), Location.Y - (int)(15 \* smaller)), new Point(Location.X + (int)(32 \* smaller), Location.Y - (int)(15 \* smaller)));

}

}

}

**Приложение 6  
Листинг Form1.cs**

using System.Windows.Forms;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Media;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices.Marshalling;

using System.Reflection.Metadata.Ecma335;

namespace Practic

{

public partial class Form1 : Form

{

List<DrawObject> drawObjects = new List<DrawObject>();

List<DrawObject> visibleObjects = new List<DrawObject>();

bool isDel = false;

int countHouse = 0;

Point location = new Point();

SoundPlayer goodSound =  
new SoundPlayer("C:\\IT\\C#\\Practic\\Sounds\\good.wav");

SoundPlayer badSound =  
new SoundPlayer("C:\\IT\\C#\\Practic\\Sounds\\bad.wav");

Cursor addCursor =  
new Cursor("C:\\IT\\C#\\Practic\\Cursors\\add.cur");

Cursor delCursor =  
new Cursor("C:\\IT\\C#\\Practic\\Cursors\\remove.cur");

public Form1()

{

InitializeComponent();

DrawingPictureBox.Image = Image.FromFile("C:\\IT\\C#\\Practic\\SourceImage.jpg");

DrawingPictureBox.SizeMode = PictureBoxSizeMode.StretchImage;

this.Icon = new Icon("C:\\IT\\C#\\Practic\\city.ico");

}

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

{

if (!isDel)

{

this.Cursor = delCursor;

isDel = true;

HouseCheckBox.Enabled = false;

FlowerBedCheckBox.Enabled = false;

StreetLightCheckBox.Enabled = false;

SwingCheckBox.Enabled = false;

ColorHousePictureBox.Enabled = false;

}

else

{

this.Cursor = Cursors.Default;

isDel = false;

HouseCheckBox.Enabled = true;

FlowerBedCheckBox.Enabled = true;

StreetLightCheckBox.Enabled = true;

SwingCheckBox.Enabled = true;

ColorHousePictureBox.Enabled = true;

}

}

private void HouseCheckBox\_CheckedChanged(object sender, EventArgs e)

{

if (HouseCheckBox.Checked)

{

this.Cursor = addCursor;

FlowerBedCheckBox.Enabled = false;

StreetLightCheckBox.Enabled = false;

SwingCheckBox.Enabled = false;

DelButton.Enabled = false;

}

else

{

this.Cursor = Cursors.Default;

FlowerBedCheckBox.Enabled = true;

StreetLightCheckBox.Enabled = true;

SwingCheckBox.Enabled = true;

DelButton.Enabled = true;

}

}

private void FlowerBedCheckBox\_CheckedChanged(object sender,  
EventArgs e)

{

if (FlowerBedCheckBox.Checked)

{

this.Cursor = addCursor;

HouseCheckBox.Enabled = false;

StreetLightCheckBox.Enabled = false;

SwingCheckBox.Enabled = false;

DelButton.Enabled = false;

ColorHousePictureBox.Enabled = false;

}

else

{

this.Cursor = Cursors.Default;

HouseCheckBox.Enabled = true;

StreetLightCheckBox.Enabled = true;

SwingCheckBox.Enabled = true;

DelButton.Enabled = true;

ColorHousePictureBox.Enabled = true;

}

}

private void StreetLightCheckBox\_CheckedChanged(object sender, EventArgs e)

{

if (StreetLightCheckBox.Checked)

{

this.Cursor = addCursor;

FlowerBedCheckBox.Enabled = false;

HouseCheckBox.Enabled = false;

SwingCheckBox.Enabled = false;

DelButton.Enabled = false;

ColorHousePictureBox.Enabled = false;

}

else

{

this.Cursor = Cursors.Default;

FlowerBedCheckBox.Enabled = true;

HouseCheckBox.Enabled = true;

SwingCheckBox.Enabled = true;

DelButton.Enabled = true;

ColorHousePictureBox.Enabled = true;

}

}

private void SwingCheckBox\_CheckedChanged(object sender, EventArgs e)

{

if (SwingCheckBox.Checked)

{

this.Cursor = addCursor;

FlowerBedCheckBox.Enabled = false;

StreetLightCheckBox.Enabled = false;

HouseCheckBox.Enabled = false;

DelButton.Enabled = false;

ColorHousePictureBox.Enabled = false;

}

else

{

this.Cursor = Cursors.Default;

FlowerBedCheckBox.Enabled = true;

StreetLightCheckBox.Enabled = true;

HouseCheckBox.Enabled = true;

DelButton.Enabled = true;

ColorHousePictureBox.Enabled = true;

}

}

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

{

using (ColorDialog color = new ColorDialog())

{

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

ColorHousePictureBox.BackColor = color.Color;

}

}

private void DrawingPictureBox\_MouseClick(object sender, MouseEventArgs e)

{

if (e.Location.Y >= 350 && e.Location.Y <= 470)

{

visibleObjects = drawObjects.Where(elem => elem.IsVisible(e.Location)).ToList();

if ((HouseCheckBox.Checked || FlowerBedCheckBox.Checked || StreetLightCheckBox.Checked || SwingCheckBox.Checked) && visibleObjects.Count() == 0 || isDel && visibleObjects.Count() != 0)

{

goodSound.Play();

location = e.Location;

this.InvokePaint(DrawingPictureBox,  
new PaintEventArgs(DrawingPictureBox.CreateGraphics(), DrawingPictureBox.ClientRectangle));

}

else badSound.Play();

}

else badSound.Play();

}

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

{

if (HouseCheckBox.Checked)

{

House house = new House(location, ColorHousePictureBox.BackColor);

drawObjects.Add(house);

PeopleInCityLabel.Text = $"Человек в городе: {++countHouse \* 50}";

}

else if (FlowerBedCheckBox.Checked)

{

FlowerBed flowerbed = new FlowerBed(location);

drawObjects.Add(flowerbed);

}

else if (StreetLightCheckBox.Checked)

{

StreetLight streetlight = new StreetLight(location);

drawObjects.Add(streetlight);

}

else if (SwingCheckBox.Checked)

{

Swing swing = new Swing(location);

drawObjects.Add(swing);

}

else if(isDel)

{

if(visibleObjects.Count == 1)

drawObjects.Remove(visibleObjects[0]);

else

{

DrawObject objectForDel = visibleObjects.OrderByDescending(elem => elem.Location.Y).First();

drawObjects.Remove(objectForDel);

}

}

if(!isDel) drawObjects.Sort((x, y) => x.Location.Y.CompareTo(y.Location.Y));

foreach (var i in drawObjects)

i.Draw(e.Graphics);

}

}

}