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, Миа Музамил Ахмад

Руководитель от кафедры Меджидов Р.Г.

ВОРОНЕЖ

2025**Введение**

Разработка пользовательских графических интерфейсов с возможностью взаимодействия с элементами на экране является одной из актуальных задач в программировании. Целью данной курсовой работы является создание Windows Forms приложения, позволяющего пользователю изменять внешний вид виртуальной куклы с помощью графических средств, настроек и событий мыши.

Проект демонстрирует принципы работы с графикой в C#, обработки событий, использования ресурсов, воспроизведения звуков и взаимодействия с формой. Программа ориентирована на наглядность, интерактивность и простоту в использовании.

## **Постановка задачи**

Необходимо создать приложение Windows Forms на языке C#, в котором отображается кукла, нарисованная с помощью изображений и графических элементов. Пользователь должен иметь возможность:

* изменять цвета одежды (майки и брюки) с помощью ColorDialog;
* добавлять стразы по клику мыши, но только в допустимую область;
* выбирать аксессуары на лицо (очки, маски и т.д.) из списка;
* слышать звуковое сопровождение при действиях;
* видеть кастомный курсор при наведении на область майки.

Также необходимо соблюсти принципы модульности, разделения логики и оформления интерфейса в соответствии с требованиями к разработке программ.

# Теоретическая часть

## **Классы и структуры языка C# для работы с графикой**

В ходе выполнения проекта были использованы ключевые графические классы из пространства имён System.Drawing, предназначенные для работы с изображениями, цветами и отрисовкой фигур в среде Windows Forms.

### **1.1.1. Graphics**

Класс Graphics представляет графический контекст и предоставляет методы для рисования элементов на форме или в других элементах управления, таких как PictureBox. В проекте он используется для отрисовки базового изображения куклы, а также наложения страз и аксессуаров:

* DrawImage() – отрисовка изображений (кукла, маска, очки и пр.);
* FillPolygon() и FillEllipse() – отрисовка геометрических фигур (стразы);
* DrawPolygon() и DrawEllipse() – отрисовка контуров страз.

### **1.1.2. Color**

Структура Color используется для определения цвета элементов. В приложении цветовая схема куклы может быть изменена пользователем через ColorDialog. Пример использования можно увидеть на рисунке 1.1.2.1.

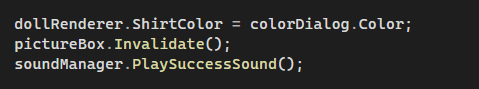


Рис. 1.1.2.1. Пример работы с ColorDialog

### **1.1.3. Brush**

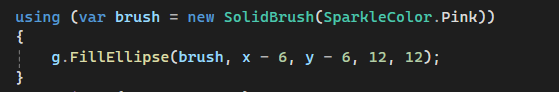
****Класс SolidBrush используется для заливки фигур цветом. Например, стразы разных форм отрисовываются розовым цветом на рисунке 1.1.3.1.

Рис. 1.1.3.1. Пример работы с Brush

### **1.1.4. Image и Bitmap**

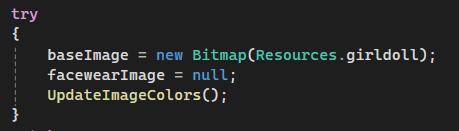
Класс Bitmap, производный от Image, используется для загрузки и обработки изображений куклы и аксессуаров (маски, очки и др.). Изображения загружаются из встроенных ресурсов и масштабируются под размеры PictureBox. Пример использования можно увидеть на рисунке 1.1.4.1.

Рис. 1.1.4.1. Пример работы с Image и Bitmap

## **1.2. Элементы формы для графики и взаимодействия**

В проекте активно используются элементы Windows Forms для взаимодействия пользователя с графическим интерфейсом.

### **1.2.1. PictureBox**

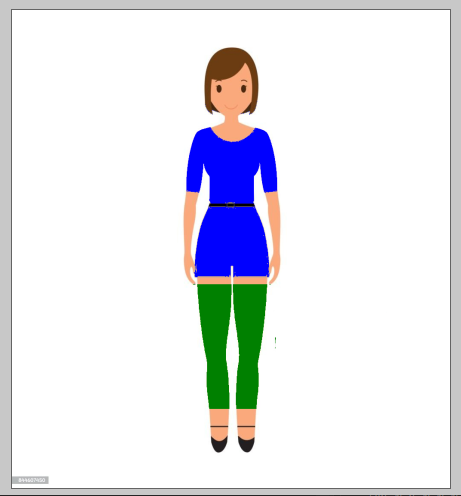
Элемент PictureBox используется для отображения куклы. На нём осуществляется вся отрисовка с помощью Graphics и Paint-события. Также через MouseClick добавляются стразы в определённую область, а через MouseMove меняется курсор. Пример использования можно увидеть на рисунке 1.2.1.1.

Рис. 1.2.1.1. Пример работы с PictureBox

### **1.2.2. ColorDialog**

Стандартный диалог выбора цвета (ColorDialog) используется для изменения цвета майки и брюки куклы. Это делает интерфейс более интуитивным и визуально гибким. Пример использования можно увидеть на рисунке 1.2.2.1.

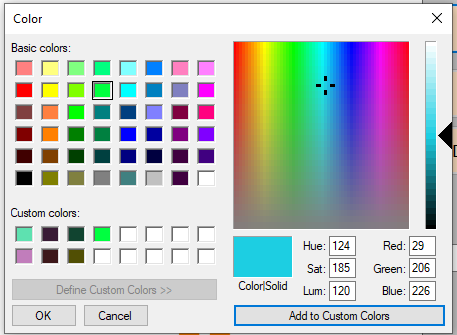


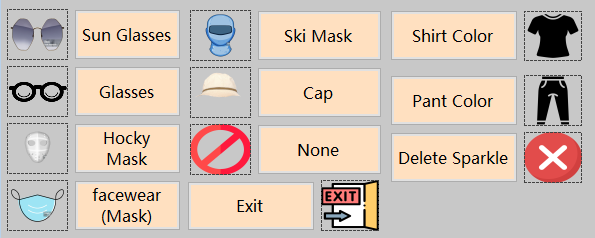
Рис. 1.2.2.1. Пример работы с ColorDialog

### **1.2.3. Button (Кнопки)**

Приложение содержит несколько кнопок, каждая из которых выполняет конкретное действие:

* **btnShirtColor / btnSkirtColor** — открытие диалога выбора цвета;
* **btnToggleRhinestones** — удаление последнего добавленного элемента (стразы);
* **btnExit** — завершение работы приложения;
* **btnsunglasses, btnskimask, btncap** и др. — установка соответствующего аксессуара на лицо куклы.

Все кнопки визуально настроены, имеют понятные надписи и обработчики событий Click, связанные с логикой интерфейса. Пример использования можно увидеть на рисунке 1.2.3.1.

Рис. 1.2.3.1. Пример работы с Button

### **1.2.4. ComboBox (Комбинированный список)**

Программа содержит два комбинированных списка:

* **comboFacewear** — выбор одного из вариантов лицевого аксессуара (очки, маски, шапка, лыжная маска и пр.);
* **cmboSparkle** — выбор формы страз (ромб, круг, сердце).

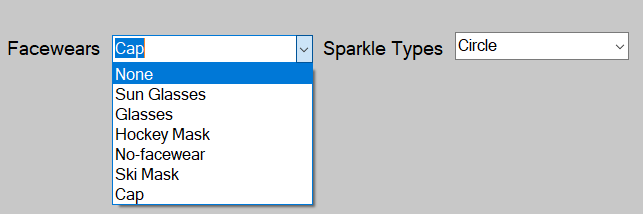
При выборе значения из списка вызывается соответствующий обработчик (SelectedIndexChanged), который обновляет состояние отображения и может сопровождаться звуком. Пример использования можно увидеть на рисунке 1.2.4.1.

Рис. 1.2.4.1. Пример работы с ComboBox

## **1.3. Методы рисования и геометрия**

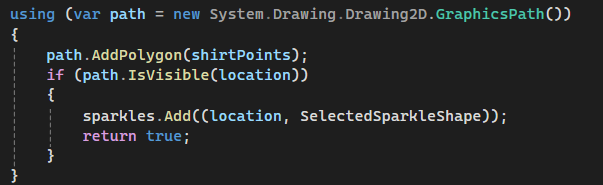
В рамках проекта были использованы как стандартные методы отрисовки, так и дополнительные инструменты для проверки геометрических условий.

### **1.3.1. Методы класса Graphics**

* DrawImage() — отрисовка базового изображения;
* FillPolygon() — отрисовка страз в форме ромба;
* FillEllipse() — отрисовка круглых страз;
* DrawPath() — отрисовка сердец (пользовательская форма).

### **1.3.2. GraphicsPath и AddPolygon**

Класс GraphicsPath используется для задания сложных форм — например, полигона, ограничивающего область майки, куда разрешено добавлять стразы. Метод IsVisible(Point) позволяет определить, находится ли точка внутри полигона: пример использования можно увидеть на рисунке 1.3.2

Рис. 1.3.2. Пример работы с GraphicsPath и AddPolygon 

## **1.4. Работа со звуком**

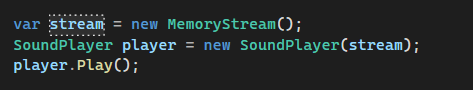
Для воспроизведения звуков использован класс SoundPlayer, встроенный в .NET. В проекте задействованы два звука: успех (success.wav) и ошибка (failed.wav). Воспроизведение происходит через Play(): пример использования можно увидеть на рисунке 1.4.1

Рис. 1.4.1. Пример работы с MemoryStream

Звуки загружаются из встроенных ресурсов через MemoryStream: рисунк 1.4.2

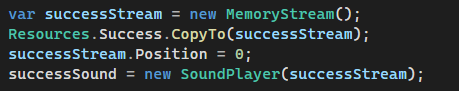
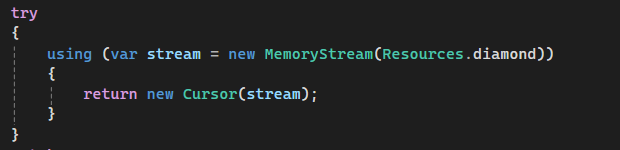


Рис. 1.4.2. Пример работы с MemoryStream

## **1.5. Работа с ресурсами**

Все изображения и звуковые файлы загружаются из встроенных ресурсов проекта, что упрощает переносимость и исключает необходимость обращения к внешним файлам. Используется пространство имён Doll\_Managment\_Project.Properties, а доступ осуществляется через Resources.имя.

## **1.6. Пользовательский курсор**

В проекте реализована замена стандартного курсора на кастомный, когда курсор находится над майкой куклы. Курсор загружается из .cur файла через поток MemoryStream: пример использования можно увидеть на рисунке 1.6.1, а графическое использование на рисунке 1.6.2.

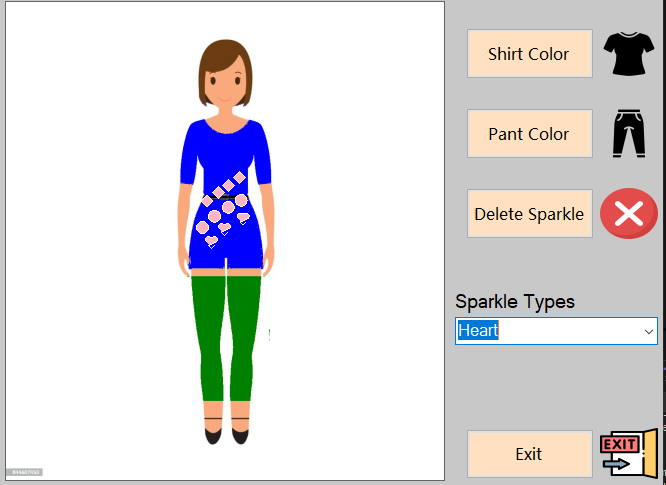
Рис. 1.6.1. Пример работы с пользовательским курсором

Рис. 1.6.2. Графическое использование курсора

**2. Практическая часть**

## **2.1. Структура программы**

Программа организована модульно. Основные компоненты: Рис. 2.1.1.

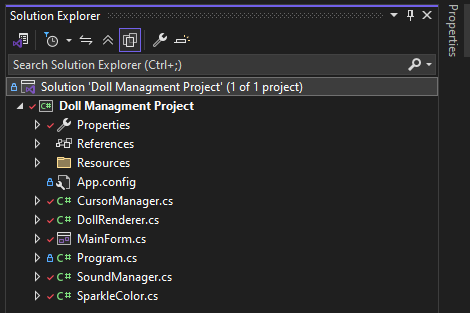
* DollRenderer.cs — отвечает за отрисовку куклы, нанесение одежды, аксессуаров и страз. Включает в себя масштабирование, управление цветами и проверку допустимости координат;
* SoundManager.cs — модуль, управляющий воспроизведением звуков успеха и неудачи при действиях пользователя;
* CursorManager.cs — отдельный модуль, загружающий и устанавливающий нестандартный курсор в нужных областях (на майке);
* SparkleColor.cs — хранит перечисление типов страз (ромб, круг, сердце) и цвет;
* MainForm.cs — основной модуль управления интерфейсом, включает события мыши, обновление PictureBox, обработку кнопок и ComboBox.

Рис. 2.1.1. Структура программы

## **2.2. Интерфейс приложения**

После запуска пользователь видит интерфейс с изображением куклы (см. рис. 2.2.1). Справа и слева находятся кнопки и выпадающие списки для настройки:

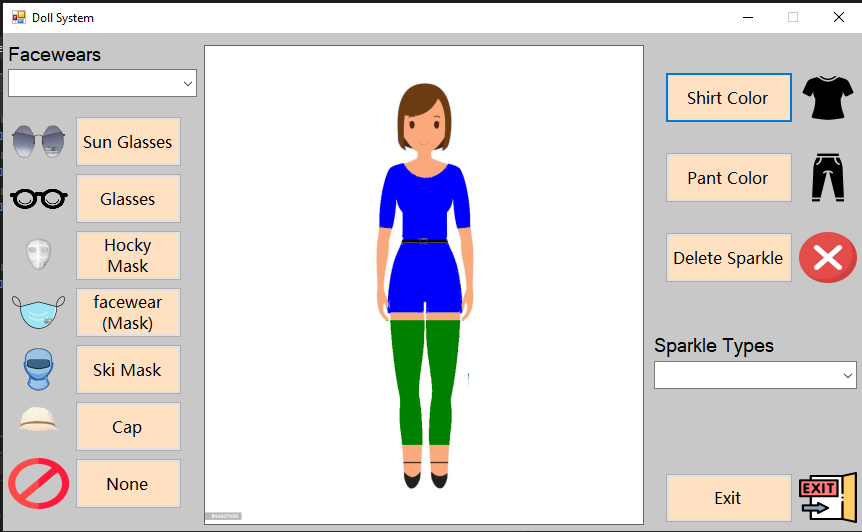
* выбор цвета майки и брюки (ColorDialog вызывается при нажатии на соответствующие кнопки);
* Combobox для выбора аксессуара на лицо (очки, маски, шапка и т.д.);
* Combobox для выбора формы страз (Diamond, Circle, Heart);
* кнопка для удаления последнего страза (btnToggleRhinestones);
* при наведении курсора на майку — курсор меняется на **кастомный** (иконка diamond.cur);
* добавление страз осуществляется кликом **в пределах области майки**;
* звуки добавления/ошибки проигрываются при действиях.

Рис. 2.2.1. Главный экран программы

## **2.3. Реализация**

### **2.3.1. Модуль DollRenderer.cs**

Модуль DollRenderer отвечает за визуализацию куклы и обработку её внешнего вида. В этом классе сосредоточена логика отрисовки одежды, аксессуаров и декоративных элементов (стразов). Также здесь реализованы функции изменения цветов и масштабирования изображения в зависимости от размеров окна.

Этот класс не связан напрямую с формой Windows Forms — он универсален и может быть использован в любом графическом приложении, где требуется визуализация подобного рода объектов.

**Основные функции и методы:**

* DrawDoll(Graphics g, int width, int height) — метод отвечает за отрисовку всей куклы, включая:
  + базовое изображение (тело куклы);
  + лицо и одежду;
  + аксессуары на лице (очки, маски, кепка);
  + стразы на одежде.

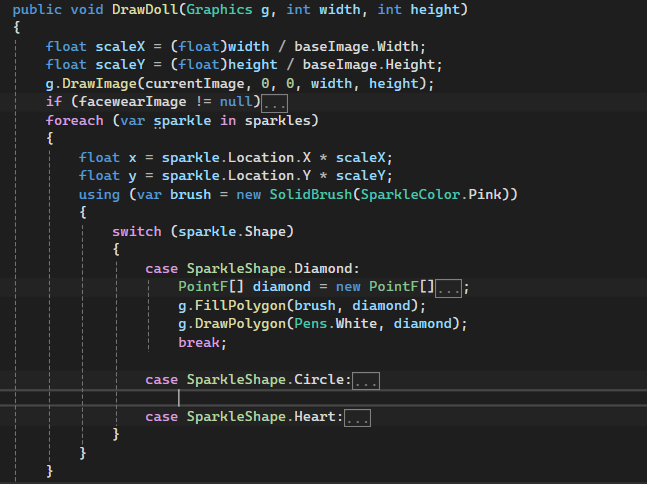
При этом соблюдается масштабирование — изображение корректно растягивается под размеры PictureBox, что важно для адаптивности интерфейса. рис. 2.3.1.1

Рис. 2.3.1.1. **Функция DrawDoll**

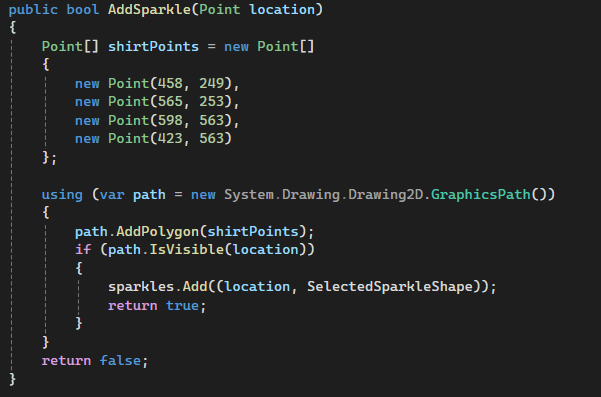
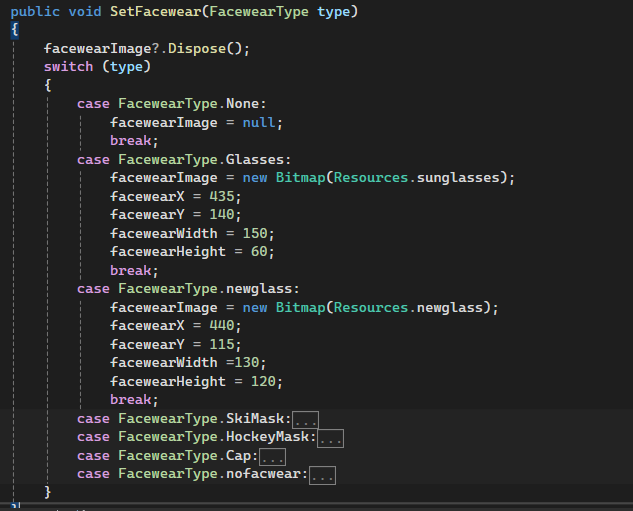
* AddSparkle(Point location) — метод добавляет страз в указанную пользователем точку (координаты курсора мыши), но только если точка попадает в **допустимую зону** — область майки куклы. Область проверяется с помощью GraphicsPath и функции IsVisible() по координатам. При успешном добавлении стразы отрисовываются вместе с куклой. Рис. 2.3.1.2

Рис. 2.3.1.2. **Функция AddSparkle**

* ClearSparkles()— метод удаляет **последний добавленный страз**. Это позволяет пользователю отменять действия поочерёдно, например, при ошибочном добавлении.
* SetFacewear(FacewearType type) — устанавливает аксессуар на лицо куклы. В зависимости от выбранного типа (очки, маска и т.д.) метод подбирает соответствующее изображение и накладывает его в нужной позиции и с нужными размерами. Рис. 2.3.1.3

Рис. 2.3.1.3. **Функция SetFacewear**

* UpdateImageColors() — обновляет цвета **майки и брюки** куклы в соответствии с выбором пользователя. Цвета определяются через ColorDialog, а замена происходит по алгоритму сравнения текущего цвета пикселя с базовым (по допуску RGB).

### **2.3.2. Модуль SoundManager.cs**

Работает с WAV-файлами, встроенными в ресурсы.  
Основные методы:

* PlaySuccessSound() — звук при успешном действии;
* PlayFailureSound() — звук при ошибке (например, клик вне майки).

### **2.3.3. Модуль CursorManager.cs**

Загружает пользовательский курсор (diamond.cur) из ресурсов.  
Применяется при наведении мыши на область майки куклы:

* GetCustomCursor() — возвращает объект Cursor из памяти.

### **2.3.4. MainForm.cs**

Класс MainForm представляет основную форму Windows Forms-приложения, с которой взаимодействует пользователь. Он содержит графический интерфейс (GUI), обработчики событий и вызывает методы из вспомогательных классов, таких как DollRenderer, SoundManager и CursorManager.

Основные обработчики событий:

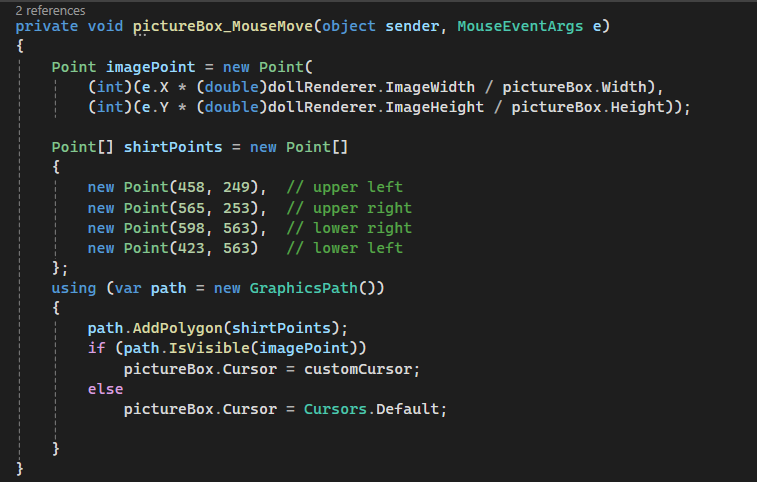
*  MouseMove — обрабатывает перемещение курсора по PictureBox.  
  Если курсор оказывается над зоной майки, то он **меняется на пользовательский (кастомный)** курсор — в данном случае, курсор в форме бриллианта. Это позволяет пользователю понять, где допустимо добавление стразов.MouseClick — попытка добавить страз. Рис. 2.3.4.1;

Рис. 2.3.4.1. **Функция MouseMove**

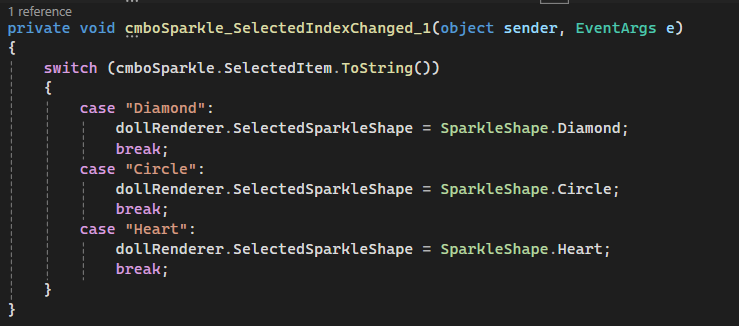
* MouseClick — обрабатывает щелчок мыши по PictureBox.  
  Выполняется попытка добавить **страз** в точку нажатия. Если точка попадает в допустимую зону (область майки), страз добавляется, и звучит **звук успеха**. В противном случае — звучит **звук ошибки**;
* btnShirtColor\_Click **и** btnSkirtColor\_Click — обрабатывают нажатие на кнопки изменения цвета одежды. Открывается диалоговое окно ColorDialog, позволяющее выбрать новый цвет. Цвет применяется к майке или юбке, и обновляется изображение;
* cmboSparkle\_SelectedIndexChanged — позволяет выбрать форму страза: **бриллиант**, **круг** или **сердце**. Пользователь может изменить форму добавляемых декоративных элементов; рис. 2.3.4.2.

Рис. 2.3.4.2. **Функция** cmboSparkle\_SelectedIndexChanged

* btnDeleteRhinestones\_Click — обрабатывает удаление последнего добавленного страза. При нажатии — последний элемент удаляется, воспроизводится звук успеха. Если стразов не осталось — звучит сигнал ошибки, и выводится сообщение. Рис. 2.3.4.3.

Рис. 2.3.4.3. **Функция btnDeleteRhinestones\_Click**

# Заключение

В рамках курсового проекта было разработано Windows Forms-приложение на языке C# для управления внешним видом куклы. Основной задачей являлось создание интерактивного интерфейса, позволяющего пользователю изменять цвет одежды, выбирать аксессуары, добавлять декоративные элементы (стразы) с визуальной и звуковой обратной связью.

В ходе выполнения проекта были достигнуты следующие результаты:

* **изучены и применены ключевые классы и структуры**:
  + Graphics — для отрисовки элементов на форме;
  + Color — для управления цветами одежды и декоративных элементов;
  + Cursor — для отображения кастомного курсора;PictureBox — как основное пространство для визуализации куклы;
  + SoundPlayer — для воспроизведения звуков при действиях пользователя.
* **реализована полноценная работа с графикой,** включая:
  + масштабирование изображения в зависимости от размеров PictureBox;
  + динамическую отрисовку элементов (аксессуары, стразы);
  + замену базовых цветов одежды с помощью ColorDialog;
  + наложение дополнительных графических слоёв (например, маски, очки и т.д.).

Таким образом, поставленные задачи были успешно выполнены, и проект демонстрирует практическое применение графических и интерактивных возможностей C# и Windows Forms.

# Список использованных источников

1. Microsoft Learn. Документация по NET. Graphics класс. – URL:  
   <https://learn.microsoft.com/ru/dotnet/api/system.drawing.graphics?view=windowsdesktop-7.0> (дата обращения: 12.07.2025).
2. Microsoft Learn. Документация по NET. Colors класс. – URL:  
   <https://learn.microsoft.com/ru/dotnet/api/system.windows.media.colors?view=windowsdesktop-8.0> (дата обращения: 13.07.2025).
3. Троелсен Э. «C# и .NET. Профессиональное программирование». — СПб.: Питер, 2022.
4. MSDN Library — Разработка Windows Forms.  – URL: [https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/](https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/" \t "_new) (дата обращения: 14.07.2025).
5. Richter J. «CLR via C#». — Microsoft Press, 2021.
6. Metanit. Программирование на C# — <https://metanit.com/sharp/> (дата обращения: 14.07.2025).

# Приложени

**Приложение 1. Листинг CursorManager.cs**

using Doll\_Managment\_Project.Properties;

using System.IO;

using System.Windows.Forms;

public class CursorManager

{

public Cursor GetCustomCursor()

{

try

{

using (var stream = new MemoryStream(Resources.diamond))

{

return new Cursor(stream);

}

}

catch

{

return Cursors.Default;

}

}

}

**Приложение 2. Листинг DollRenderer.cs**

using Doll\_Managment\_Project.Properties;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

namespace Doll\_Managment\_Project

{

public enum FacewearType

{

None,

Glasses,

SkiMask,

HockeyMask,

Cap,

newglass,

nofacwear

}

public class DollRenderer

{

private Color \_shirtColor = Color.Blue;

private Color \_skirtColor = Color.Green;

private Bitmap baseImage;

private Bitmap currentImage;

private readonly Color ShirtBaseColor = Color.FromArgb(150, 120, 190);

private readonly Color SkirtBaseColor = Color.FromArgb(215, 190, 160);

private readonly int ShirtYMin = 190, ShirtYMax = 850;

private readonly int SkirtYMin = 585, SkirtYMax = 850;

private readonly int ColorTolerance = 60;

private Bitmap facewearImage;

private int facewearX = 140;

private int facewearY = 80;

private int facewearWidth, facewearHeight;

private List<(Point Location, SparkleShape Shape)> sparkles = new List<(Point, SparkleShape)>();

public SparkleShape SelectedSparkleShape { get; set; } = SparkleShape.Diamond;

public int ImageWidth => baseImage.Width;

public int ImageHeight => baseImage.Height;

public DollRenderer()

{

try

{

baseImage = new Bitmap(Resources.girldoll);

facewearImage = null;

UpdateImageColors();

}

catch

{

baseImage = new Bitmap(400, 500);

using (Graphics g = Graphics.FromImage(baseImage))

{

g.Clear(Color.White);

g.DrawString("Image missing", new Font("Arial", 12), Brushes.Red, 10, 10);

}

}

}

public void SetFacewear(FacewearType type)

{

facewearImage?.Dispose();

switch (type)

{

case FacewearType.None:

facewearImage = null;

break;

case FacewearType.Glasses:

facewearImage = new Bitmap(Resources.sunglasses);

facewearX = 435;

facewearY = 140;

facewearWidth = 150;

facewearHeight = 60;

break;

case FacewearType.newglass:

facewearImage = new Bitmap(Resources.newglass);

facewearX = 440;

facewearY = 115;

facewearWidth =130;

facewearHeight = 120;

break;

case FacewearType.SkiMask:

facewearImage = new Bitmap(Resources.ski\_mask);

facewearX = 350;

facewearY = 60;

facewearWidth = 320;

facewearHeight = 250;

break;

case FacewearType.HockeyMask:

facewearImage = new Bitmap(Resources.hockey\_mask);

facewearX = 350;

facewearY = 60;

facewearWidth = 320;

facewearHeight = 280;

break;

case FacewearType.Cap:

facewearImage = new Bitmap(Resources.mycap);

facewearX = 390;

facewearY = 10;

facewearWidth = 250;

facewearHeight = 250;

break;

case FacewearType.nofacwear:

facewearImage = new Bitmap(Resources.no\_facewear);

facewearX = 420;

facewearY = 135;

facewearWidth = 180;

facewearHeight = 130;

break;

}

}

public bool AddSparkle(Point location)

{

Point[] shirtPoints = new Point[]

{

new Point(458, 249),

new Point(565, 253),

new Point(598, 563),

new Point(423, 563)

};

using (var path = new System.Drawing.Drawing2D.GraphicsPath())

{

path.AddPolygon(shirtPoints);

if (path.IsVisible(location))

{

sparkles.Add((location, SelectedSparkleShape));

return true;

}

}

return false;

}

public bool ClearSparkles()

{

if(sparkles.Count > 0)

{

sparkles.RemoveAt(sparkles.Count - 1);

return true;

}

return false;

}

public Color ShirtColor

{

get => \_shirtColor;

set { \_shirtColor = value; UpdateImageColors(); }

}

public Color SkirtColor

{

get => \_skirtColor;

set { \_skirtColor = value; UpdateImageColors(); }

}

public void DrawDoll(Graphics g, int width, int height)

{

float scaleX = (float)width / baseImage.Width;

float scaleY = (float)height / baseImage.Height;

g.DrawImage(currentImage, 0, 0, width, height);

if (facewearImage != null)

{

g.DrawImage(facewearImage,

facewearX \* scaleX,

facewearY \* scaleY,

facewearWidth \* scaleX,

facewearHeight \* scaleY);

}

foreach (var sparkle in sparkles)

{

float x = sparkle.Location.X \* scaleX;

float y = sparkle.Location.Y \* scaleY;

using (var brush = new SolidBrush(SparkleColor.Pink))

{

switch (sparkle.Shape)

{

case SparkleShape.Diamond:

PointF[] diamond = new PointF[]

{

new PointF(x, y - 6),

new PointF(x + 6, y),

new PointF(x, y + 6),

new PointF(x - 6, y)

};

g.FillPolygon(brush, diamond);

g.DrawPolygon(Pens.White, diamond);

break;

case SparkleShape.Circle:

g.FillEllipse(brush, x - 6, y - 6, 12, 12);

g.DrawEllipse(Pens.White, x - 6, y - 6, 12, 12);

break;

case SparkleShape.Heart:

using (GraphicsPath heartPath = CreateHeartShape(x, y))

{

g.FillPath(brush, heartPath);

g.DrawPath(Pens.White, heartPath);

}

break;

}

}

}

}

private System.Drawing.Drawing2D.GraphicsPath CreateHeartShape(float x, float y)

{

var path = new System.Drawing.Drawing2D.GraphicsPath();

RectangleF leftArc = new RectangleF(x - 6, y - 6, 6, 6);

RectangleF rightArc = new RectangleF(x, y - 6, 6, 6);

PointF bottom = new PointF(x, y + 6);

path.AddArc(leftArc, 135, 180);

path.AddArc(rightArc, 225, 180);

path.AddLine(x - 3, y + 3, x, y + 6);

path.AddLine(x + 3, y + 3, x, y + 6);

return path;

}

private void UpdateImageColors()

{

using (Bitmap original = new Bitmap(baseImage))

{

currentImage?.Dispose();

currentImage = new Bitmap(original.Width, original.Height);

for (int x = 0; x < original.Width; x++)

{

for (int y = 0; y < original.Height; y++)

{

Color pixel = original.GetPixel(x, y);

if (y >= ShirtYMin && y <= ShirtYMax &&

IsColorCloseTo(pixel, ShirtBaseColor, ColorTolerance))

{

currentImage.SetPixel(x, y, \_shirtColor);

}

else if (y >= SkirtYMin && y <= SkirtYMax &&

IsColorCloseTo(pixel, SkirtBaseColor, ColorTolerance))

{

currentImage.SetPixel(x, y, \_skirtColor);

}

else

currentImage.SetPixel(x, y, pixel);

}

}

}

}

private bool IsColorCloseTo(Color color, Color baseColor, int tolerance)

{

return Math.Abs(color.R - baseColor.R) < tolerance &&

Math.Abs(color.G - baseColor.G) < tolerance &&

Math.Abs(color.B - baseColor.B) < tolerance;

}

}

}

**Приложение 3. Листинг MainForm.cs**

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace Doll\_Managment\_Project

{

public partial class MainForm : Form

{

private readonly DollRenderer dollRenderer;

private readonly SoundManager soundManager;

private readonly CursorManager cursorManager;

private readonly Cursor customCursor;

public MainForm()

{

InitializeComponent();

dollRenderer = new DollRenderer();

dollRenderer = new DollRenderer();

soundManager = new SoundManager();

cursorManager = new CursorManager();

customCursor = cursorManager.GetCustomCursor();

colorDialog = new ColorDialog();

pictureBox.Paint += PictureBox\_Paint;

pictureBox.MouseClick += pictureBox\_MouseClick;

pictureBox.MouseMove += pictureBox\_MouseMove;

pictureBox.Image = Properties.Resources.girldoll;

this.Cursor = Cursors.Default;

}

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

{

dollRenderer.DrawDoll(e.Graphics, pictureBox.Width, pictureBox.Height);

}

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

{

Point imagePoint = new Point(

(int)(e.X \* (double)dollRenderer.ImageWidth / pictureBox.Width),

(int)(e.Y \* (double)dollRenderer.ImageHeight / pictureBox.Height));

bool success = dollRenderer.AddSparkle(imagePoint);

if (success)

soundManager.PlaySuccessSound();

else

soundManager.PlayFailureSound();

pictureBox.Invalidate();

}

private void pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

Point imagePoint = new Point(

(int)(e.X \* (double)dollRenderer.ImageWidth / pictureBox.Width),

(int)(e.Y \* (double)dollRenderer.ImageHeight / pictureBox.Height));

Point[] shirtPoints = new Point[]

{

new Point(458, 249), // upper left

new Point(565, 253), // upper right

new Point(598, 563), // lower right

new Point(423, 563) // lower left

};

using (var path = new GraphicsPath())

{

path.AddPolygon(shirtPoints);

if (path.IsVisible(imagePoint))

pictureBox.Cursor = customCursor;

else

pictureBox.Cursor = Cursors.Default;

}

}

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

{

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

{

dollRenderer.ShirtColor = colorDialog.Color;

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

else

soundManager.PlayFailureSound();

}

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

{

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

{

dollRenderer.SkirtColor = colorDialog.Color;

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

else

soundManager.PlayFailureSound();

}

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

{

this.Close();

}

private void comboFacewear\_SelectedIndexChanged(object sender, EventArgs e)

{

bool sound\_success = true;

switch (comboFacewear.SelectedItem.ToString())

{

case "None":

dollRenderer.SetFacewear(FacewearType.None);

break;

case "Sun Glasses":

dollRenderer.SetFacewear(FacewearType.Glasses);

break;

case "Ski Mask":

dollRenderer.SetFacewear(FacewearType.SkiMask);

break;

case "Hockey Mask":

dollRenderer.SetFacewear(FacewearType.HockeyMask);

break;

case "Cap":

dollRenderer.SetFacewear(FacewearType.Cap);

break;

case "Glasses":

dollRenderer.SetFacewear(FacewearType.newglass);

break;

case "No-facewear":

dollRenderer.SetFacewear(FacewearType.nofacwear);

break;

default:

sound\_success = false;

break;

}

pictureBox.Invalidate();

if (sound\_success)

soundManager.PlaySuccessSound();

else

soundManager.PlayFailureSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.Glasses);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.newglass);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.None);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.SkiMask);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.nofacwear);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.Cap);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

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

{

dollRenderer.SetFacewear(FacewearType.HockeyMask);

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

private void cmboSparkle\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

switch (cmboSparkle.SelectedItem.ToString())

{

case "Diamond":

dollRenderer.SelectedSparkleShape = SparkleShape.Diamond;

break;

case "Circle":

dollRenderer.SelectedSparkleShape = SparkleShape.Circle;

break;

case "Heart":

dollRenderer.SelectedSparkleShape = SparkleShape.Heart;

break;

}

}

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

{

bool removed = dollRenderer.ClearSparkles();

if (removed)

{

pictureBox.Invalidate();

soundManager.PlaySuccessSound();

}

else

{

soundManager.PlayFailureSound();

MessageBox.Show("No sparkles to remove.", "Notice", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

**Приложение 4. Листинг SoundManager.cs**

using System;

using System.IO;

using System.Media;

using System.Windows.Forms;

using Doll\_Managment\_Project.Properties;

namespace Doll\_Managment\_Project

{

public class SoundManager

{

private SoundPlayer successSound;

private SoundPlayer failureSound;

public SoundManager()

{

try

{

var successStream = new MemoryStream();

Resources.Success.CopyTo(successStream);

successStream.Position = 0;

successSound = new SoundPlayer(successStream);

var failureStream = new MemoryStream();

Resources.failed.CopyTo(failureStream);

failureStream.Position = 0;

failureSound = new SoundPlayer(failureStream);

}

catch (Exception ex)

{

MessageBox.Show($"Error loading sounds: {ex.Message}", "Sound Error");

}

}

public void PlaySuccessSound()

{

try

{

successSound?.Play();

}

catch (Exception ex)

{

MessageBox.Show($"Could not play success sound: {ex.Message}", "Sound Error");

SystemSounds.Beep.Play();

}

}

public void PlayFailureSound()

{

try

{

failureSound?.Play();

}

catch (Exception ex)

{

MessageBox.Show($"Could not play failure sound: {ex.Message}", "Sound Error");

SystemSounds.Beep.Play();

}

}

}

}

**Приложение 5. Листинг SparkleColor.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Doll\_Managment\_Project

{

public enum SparkleShape

{

Diamond,

Circle,

Heart

}

public static class SparkleColor

{

public static readonly Color Pink = Color.FromArgb(255, 182, 193);

}

}