Разработка автоматизированного рабочего места дизайнера на C++

Техническое задание

Листов 28

Пермь 2025

**Оглавление**

[1. Введение 3](#_heading=h.1fob9te)

[2. Основания для разработки АРМ 3](#_heading=h.3znysh7)

[3. Назначение АРМ 3](#_heading=h.2et92p0)

4. [Требования к](#_heading=h.3dy6vkm) АРМ 4

5. [UML-диаграмма классов](#_heading=h.1t3h5sf) 5

6. [Используемые инструменты и технологии](#_heading=h.1t3h5sf)  5

7. [Код программы 6](#_heading=h.2s8eyo1)

8. [Примеры работы программы 2](#_heading=h.17dp8vu)4

9. [Заключение 2](#_heading=h.17dp8vu)7

10. Список используемых источников 28

**1. Введение**

**1.1. Наименование программы**

Наименование программы: “ Разработка автоматизированного рабочего места дизайнера на C++”. Далее по тексту – АРМ.

**1.2. Краткая характеристика области применения**

АРМ предназначено для автоматизации подбора цветовых палитр в дизайнерских проектах. Программа позволяет дизайнеру быстро генерировать гармоничные цветовые сочетания на основе выбранного базового цвета, учитывая правила колористики и психологию восприятия цветов.

**2. Основания для разработки АРМ**

**2.1. Основания для проведения разработки**

“Разработка автоматизированного рабочего места дизайнера на C++”, утвержденного доцентом кафедры ИТАС Поляковой О. А.

Согласно требованиям Поляковой О. А., именуемой заказчиком, исполнитель Токарева А. В.

**3. Назначение АРМ**

**3.1. Функциональное назначение**

Программа предоставляет дизайнеру следующие возможности:

1. Генерация цветовых палитр на основе:
   * Монохромной схемы (оттенки одного цвета).
   * Триадной схемы (цвета, образующие равносторонний треугольник на цветовом круге).
   * Комплементарной схемы (контрастные противоположные цвета).
2. Регулировка яркости палитры в реальном времени.
3. Сохранение и загрузка палитр для повторного использования.
4. Копирование HEX-кодов цветов в буфер обмена.

**3.2. Эксплуатационное назначение**

Цель АРМ — сокращение времени на рутинные задачи подбора цветов и минимизация ошибок при работе с цветовыми сочетаниями.

**4. Требования к АРМ**

**4.1. Требования к функциональным характеристикам**

***4.1.1. Требования к составу выполняемых функций***

Программа должна обеспечивать возможность выполнения перечисленных ниже функций.

1. Управление цветовыми палитрами:
   * Ввод HEX-кода базового цвета.
   * Выбор типа цветовой схемы.
   * Генерация палитры.
2. Дополнительные функции:
   * Регулировка яркости.
   * Сохранение палитр в форматах .pal или .txt.
   * Загрузка ранее созданных палитр.

***4.1.2. Требование к организации входных данных***

*Пользователь системы*

* HEX-код цвета (строка, 6 символов).
* Тип цветовой схемы (монохромная, триадная, комплементарная).

***4.1.3. Требования к организации выходных данных***

* Сгенерированная палитра (массив HEX-кодов).
* Визуальное отображение цветов.

1. **UML-диаграмма классов**

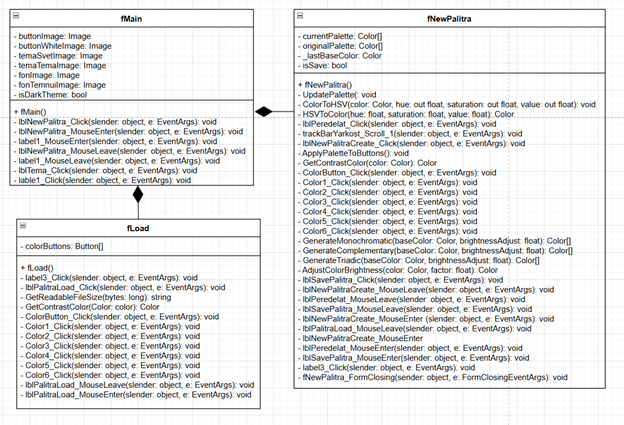
Система включает следующие основные классы:

1. fMain
   * Отвечает за запуск приложения и навигацию между формами.
   * Содержит кнопки для перехода к созданию палитры (fNewPalette) и загрузке сохранённых палитр (fLoad).
2. fNewPalette

* генерация цветов на основе алгоритмов колористики.
* регулировка яркости.

1. fLoad
   * загрузка палитры из .pal или .txt.

Отношения между классами: композиция. fMain создаёт и управляет объектами fNewPalette и fLoad.



1. **Используемые инструменты и технологии**

**6.1. Среды разработки и вспомогательные программы**

1. Microsoft Visual Studio 2022
   * Основная среда разработки для создания приложения на C# (Windows Forms).
   * Использованные компоненты:
     1. Windows Forms – для проектирования графического интерфейса.
     2. .NET Framework 4.8 – платформа для работы приложения.
2. SAI2 (Paint Tool SAI)
   * Использовалась для разработки дизайна интерфейса (логотип, иконки, цветовые схемы).
3. OBS Studio
   * Программа для записи видео с экрана монитора при демонстрации работы приложения.
4. CapCut
   * Видеоредактор для монтажа записанных материалов.

**6.2. Библиотеки и API**

System.Drawing - Стандартная библиотека .NET для работы с графикой (отрисовка цветов, управление изображениями).

1. **Код программы**

**7.1 fMain**

namespace Palitra

{

public partial class fMain : Form

{

private Image buttonImage;

private Image buttonWhiteImage;

private Image temaSvetImage;

private Image temaTemaImage;

private Image fonImage;

private Image fonTemnuiImage;

private bool isDarkTheme = false;

public fMain()

{

InitializeComponent();

buttonImage = Properties.Resources.Button;

buttonWhiteImage = Properties.Resources.ButtonWhite;

temaSvetImage = Properties.Resources.TemaSvet;

temaTemaImage = Properties.Resources.TemaTema;

fonImage = Properties.Resources.Fon;

fonTemnuiImage = Properties.Resources.FonTemnui;

}

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

{

fNewPalitra newPalitraForm = new fNewPalitra();

if (!isDarkTheme) newPalitraForm.BackgroundImage = fonImage;

else newPalitraForm.BackgroundImage = fonTemnuiImage;

newPalitraForm.ShowDialog();

}

private void lblNewPalitra\_MouseEnter(object sender, EventArgs e)

{

lblNewPalitra.Image = Properties.Resources.Button;

}

private void label1\_MouseEnter(object sender, EventArgs e)

{

lable1.Image = Properties.Resources.Button;

}

private void lblNewPalitra\_MouseLeave(object sender, EventArgs e)

{

lblNewPalitra.Image = Properties.Resources.ButtonWhite;

}

private void label1\_MouseLeave(object sender, EventArgs e)

{

lable1.Image = Properties.Resources.ButtonWhite;

}

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

{

isDarkTheme = !isDarkTheme;

if (isDarkTheme)

{

BackgroundImage = fonTemnuiImage;

lblTema.Image = temaTemaImage;

}

else

{

BackgroundImage = fonImage;

lblTema.Image = temaSvetImage;

}

}

private void lable1\_Click(object sender, EventArgs e)

{

fLoad LoadForm = new fLoad();

if (!isDarkTheme) LoadForm.BackgroundImage = fonImage;

else LoadForm.BackgroundImage = fonTemnuiImage;

LoadForm.ShowDialog();

}

}

}

**7.2 fNewPalitra**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Palitra

{

public partial class fNewPalitra : Form

{

private Color[] currentPalette = new Color[6];

private Color[] originalPalette = new Color[6];

private Color \_lastBaseColor = Color.Empty;

private bool isSave = false;

public fNewPalitra()

{

InitializeComponent();

trackBarYarkost.Minimum = 0;

trackBarYarkost.Maximum = 100;

trackBarYarkost.Value = 50;

lblProzent.Text = $"{trackBarYarkost.Value}%";

cbGarmonyColor.Items.AddRange(new string[] { "Монохромная", "Комплементарная", "Триадная" });

cbGarmonyColor.SelectedIndex = 0;

trackBarYarkost.Scroll += (s, e) => lblProzent.Text = $"{trackBarYarkost.Value}%";

}

private void UpdatePalette()

{

if (string.IsNullOrEmpty(tbRGB.Text) || tbRGB.Text.Length != 6)

{

MessageBox.Show("Введите 6-значный HEX-код цвета (без #)", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

try

{

\_lastBaseColor = ColorTranslator.FromHtml("#" + tbRGB.Text);

lblMainColor.BackColor = \_lastBaseColor;

lblMainColor.ForeColor = GetContrastColor(\_lastBaseColor);

float brightness = (trackBarYarkost.Value - 50) / 100f;

switch (cbGarmonyColor.SelectedItem?.ToString())

{

case "Монохромная":

currentPalette = GenerateMonochromatic(\_lastBaseColor, brightness);

break;

case "Комплементарная":

currentPalette = GenerateComplementary(\_lastBaseColor, brightness);

break;

case "Триадная":

currentPalette = GenerateTriadic(\_lastBaseColor, brightness);

break;

default:

MessageBox.Show("Выберите тип цветовой гармонии", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

originalPalette = (Color[])currentPalette.Clone();

ApplyPaletteToButtons();

lblPeredelat.Visible = true;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка создания палитры: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ColorToHSV(Color color, out float hue, out float saturation, out float value)

{

int max = Math.Max(color.R, Math.Max(color.G, color.B));

int min = Math.Min(color.R, Math.Min(color.G, color.B));

float delta = max - min;

if (delta == 0)

hue = 0;

else if (max == color.R)

hue = ((color.G - color.B) / (float)delta) % 6;

else if (max == color.G)

hue = ((color.B - color.R) / (float)delta) + 2;

else

hue = ((color.R - color.G) / (float)delta) + 4;

hue \*= 60;

if (hue < 0) hue += 360;

saturation = max == 0 ? 0 : delta / (float)max;

value = max / 255f;

}

private Color HSVToColor(float hue, float saturation, float value)

{

int hi = (int)(hue / 60) % 6;

float f = (hue / 60) - (int)(hue / 60);

value \*= 255;

int v = (int)value;

int p = (int)(value \* (1 - saturation));

int q = (int)(value \* (1 - f \* saturation));

int t = (int)(value \* (1 - (1 - f) \* saturation));

switch (hi)

{

case 0: return Color.FromArgb(v, t, p);

case 1: return Color.FromArgb(q, v, p);

case 2: return Color.FromArgb(p, v, t);

case 3: return Color.FromArgb(p, q, v);

case 4: return Color.FromArgb(t, p, v);

default: return Color.FromArgb(v, p, q);

}

}

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

{

if (originalPalette == null || originalPalette.Length == 0)

return;

Random rnd = new Random();

for (int i = 0; i < currentPalette.Length; i++)

{

float h, s, v;

ColorToHSV(originalPalette[i], out h, out s, out v);

float hueShift = h + (float)(rnd.NextDouble() \* 15 - 7.5);

h = (hueShift + 360) % 360;

float satShift = s \* (1f + (float)(rnd.NextDouble() \* 0.1 - 0.05));

float valShift = v \* (1f + (float)(rnd.NextDouble() \* 0.1 - 0.05));

satShift = Math.Max(0.2f, Math.Min(1f, satShift));

valShift = Math.Max(0.3f, Math.Min(1f, valShift));

currentPalette[i] = HSVToColor(h, satShift, valShift);

}

ApplyPaletteToButtons();

}

private void trackBarYarkost\_Scroll\_1(object sender, EventArgs e)

{

lblProzent.Text = $"{trackBarYarkost.Value}%";

if (\_lastBaseColor == Color.Empty)

return;

float brightnessAdjust = (trackBarYarkost.Value - 50) / 100f;

switch (cbGarmonyColor.SelectedItem?.ToString())

{

case "Монохромная":

currentPalette = GenerateMonochromatic(\_lastBaseColor, brightnessAdjust);

break;

case "Комплементарная":

currentPalette = GenerateComplementary(\_lastBaseColor, brightnessAdjust);

break;

case "Триадная":

currentPalette = GenerateTriadic(\_lastBaseColor, brightnessAdjust);

break;

}

originalPalette = (Color[])currentPalette.Clone();

ApplyPaletteToButtons();

}

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

{

UpdatePalette();

lblPeredelat.Visible = true;

}

private void ApplyPaletteToButtons()

{

System.Windows.Forms.Button[] colorButtons = { Color1, Color2, Color3, Color4, Color5, Color6 };

for (int i = 0; i < 6; i++)

{

colorButtons[i].BackColor = currentPalette[i];

colorButtons[i].Text = $"{currentPalette[i].R:X2}{currentPalette[i].G:X2}{currentPalette[i].B:X2}";

colorButtons[i].ForeColor = GetContrastColor(currentPalette[i]);

}

}

private Color GetContrastColor(Color color)

{

double brightness = (color.R \* 299 + color.G \* 587 + color.B \* 114) / 1000;

return brightness > 128 ? Color.Black : Color.White;

}

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

{

var btn = sender as System.Windows.Forms.Button;

if (btn != null && !string.IsNullOrEmpty(btn.Text))

{

Clipboard.SetText(btn.Text);

MessageBox.Show($"Цвет {btn.Text} скопирован в буфер обмена", "Скопировано",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void Color1\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color2\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color3\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color4\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color5\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color6\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private Color[] GenerateMonochromatic(Color baseColor, float brightnessAdjust)

{

Color[] palette = new Color[6];

float h, s, v;

ColorToHSV(baseColor, out h, out s, out v);

float hueVariation = 15f;

float satVariation = 0.4f;

float valVariation = 0.5f;

for (int i = 0; i < 6; i++)

{

float hueShift = h + hueVariation \* (i - 2.5f) / 2.5f;

float satShift = s \* (1f + satVariation \* (i - 3f) / 3f);

float valShift = v \* (1f + valVariation \* (i - 3f) / 3f);

hueShift = Math.Max(0, Math.Min(360, hueShift));

satShift = Math.Max(0.2f, Math.Min(1f, satShift));

valShift = Math.Max(0.3f, Math.Min(1f, valShift));

Color modifiedColor = HSVToColor(hueShift, satShift, valShift);

palette[i] = AdjustColorBrightness(modifiedColor, brightnessAdjust);

}

return palette;

}

private Color[] GenerateComplementary(Color baseColor, float brightnessAdjust)

{

Color[] palette = new Color[6];

Color complementary = Color.FromArgb(255 - baseColor.R, 255 - baseColor.G, 255 - baseColor.B);

palette[0] = AdjustColorBrightness(baseColor, brightnessAdjust);

palette[1] = AdjustColorBrightness(complementary, brightnessAdjust);

for (int i = 2; i < 6; i++)

{

float ratio = (i - 1) / 5f;

int r = (int)(baseColor.R \* (1 - ratio) + complementary.R \* ratio);

int g = (int)(baseColor.G \* (1 - ratio) + complementary.G \* ratio);

int b = (int)(baseColor.B \* (1 - ratio) + complementary.B \* ratio);

palette[i] = AdjustColorBrightness(Color.FromArgb(r, g, b), brightnessAdjust);

}

return palette;

}

private Color[] GenerateTriadic(Color baseColor, float brightnessAdjust)

{

Color[] palette = new Color[6];

palette[0] = AdjustColorBrightness(baseColor, brightnessAdjust);

palette[1] = AdjustColorBrightness(Color.FromArgb(baseColor.G, baseColor.B, baseColor.R), brightnessAdjust);

palette[2] = AdjustColorBrightness(Color.FromArgb(baseColor.B, baseColor.R, baseColor.G), brightnessAdjust);

palette[3] = AdjustColorBrightness(Color.FromArgb(

(baseColor.R + palette[1].R) / 2,

(baseColor.G + palette[1].G) / 2,

(baseColor.B + palette[1].B) / 2), brightnessAdjust);

palette[4] = AdjustColorBrightness(Color.FromArgb(

(palette[1].R + palette[2].R) / 2,

(palette[1].G + palette[2].G) / 2,

(palette[1].B + palette[2].B) / 2), brightnessAdjust);

palette[5] = AdjustColorBrightness(Color.FromArgb(

(palette[2].R + baseColor.R) / 2,

(palette[2].G + baseColor.G) / 2,

(palette[2].B + baseColor.B) / 2), brightnessAdjust);

return palette;

}

private Color AdjustColorBrightness(Color color, float factor)

{

float r = color.R / 255f;

float g = color.G / 255f;

float b = color.B / 255f;

if (factor > 0)

{

r = r + (1 - r) \* factor;

g = g + (1 - g) \* factor;

b = b + (1 - b) \* factor;

}

else if (factor < 0)

{

factor = 1 + factor;

r \*= factor;

g \*= factor;

b \*= factor;

}

return Color.FromArgb(

(int)(r \* 255),

(int)(g \* 255),

(int)(b \* 255));

}

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

{

if (currentPalette == null || currentPalette.Length == 0)

{

MessageBox.Show("Нет созданной палитры для сохранения.", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

using (SaveFileDialog sfd = new SaveFileDialog())

{

sfd.Filter = "Файлы палитр (\*.pal)|\*.pal|Текстовые файлы (\*.txt)|\*.txt";

sfd.FilterIndex = 1;

sfd.RestoreDirectory = true;

sfd.FileName = "MyPalette.pal";

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

{

try

{

using (StreamWriter writer = new StreamWriter(sfd.FileName))

{

foreach (Color color in currentPalette)

{

string hex = "#" + color.R.ToString("X2") +

color.G.ToString("X2") +

color.B.ToString("X2");

writer.WriteLine(hex);

}

}

MessageBox.Show("Палитра успешно сохранена!", "Сохранено",

MessageBoxButtons.OK, MessageBoxIcon.Information);

isSave = true;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении файла: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

private void lblNewPalitraCreate\_MouseLeave(object sender, EventArgs e)

{

lblNewPalitraCreate.Image = Properties.Resources.ButtonWhite;

}

private void lblPeredelat\_MouseLeave(object sender, EventArgs e)

{

lblPeredelat.Image = Properties.Resources.ButtonWhite;

}

private void lblSavePalitra\_MouseLeave(object sender, EventArgs e)

{

lblSavePalitra.Image = Properties.Resources.ButtonWhite;

}

private void lblNewPalitraCreate\_MouseEnter(object sender, EventArgs e)

{

lblNewPalitraCreate.Image = Properties.Resources.Button;

}

private void lblPeredelat\_MouseEnter(object sender, EventArgs e)

{

lblPeredelat.Image = Properties.Resources.Button;

}

private void lblSavePalitra\_MouseEnter(object sender, EventArgs e)

{

lblSavePalitra.Image = Properties.Resources.Button;

}

private void label3\_Click(object sender, EventArgs e)

{

Close();

}

private void fNewPalitra\_FormClosing(object sender, FormClosingEventArgs e)

{

if (isSave == false)

{

DialogResult result = MessageBox.Show("Ваша палитра не сохранена, вы точно хотите уйти? Изменения не сохранятся.","Внимание!",MessageBoxButtons.YesNo,MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

isSave = true;

Close();

}

else if (result == DialogResult.No)

{

e.Cancel = true;

}

}

}

}

}

**7.3 fLoad**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Palitra

{

public partial class fLoad : Form

{

private Button[] colorButtons;

public fLoad()

{

InitializeComponent();

}

private void label3\_Click(object sender, EventArgs e)

{

Close();

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Palette files (\*.pal;\*.txt)|\*.pal;\*.txt|All files (\*.\*)|\*.\*";

openFileDialog.Title = "Выберите файл палитры";

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

{

string filePath = openFileDialog.FileName;

string fileName = Path.GetFileName(filePath);

long fileSize = new FileInfo(filePath).Length;

lblNameFile.Text = "Файл: " + fileName;

lblRazmer.Text = $"Размер: {GetReadableFileSize(fileSize)}";

panel1.Size = new Size(lblNameFile.Width + 4, panel1.Height);

panel1.Visible = true;

try

{

string[] hexColors = File.ReadAllLines(filePath);

if (hexColors.Length != 6)

{

MessageBox.Show("Файл должен содержать ровно 6 HEX-цветов.", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

var colorButtons = new System.Windows.Forms.Button[]

{

Color1, Color2, Color3, Color4, Color5, Color6

};

for (int i = 0; i < 6; i++)

{

string hex = hexColors[i].Trim();

if (hex.StartsWith("#")) hex = hex.Substring(1);

if (hex.Length != 6)

{

MessageBox.Show($"Цвет #{i + 1} имеет неверный формат: {hex}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

Color color = ColorTranslator.FromHtml("#" + hex);

colorButtons[i].BackColor = color;

colorButtons[i].Text = hex.ToUpper();

colorButtons[i].ForeColor = GetContrastColor(color);

}

catch

{

MessageBox.Show($"Невозможно преобразовать цвет #{i + 1}: {hex}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при чтении файла: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private string GetReadableFileSize(long bytes)

{

string[] suffixes = { "B", "KB", "MB", "GB", "TB" };

int counter = 0;

double len = bytes;

while (len >= 1024 && counter < suffixes.Length - 1)

{

counter++;

len = len / 1024;

}

return $"{len:0.##} {suffixes[counter]}";

}

private Color GetContrastColor(Color color)

{

double brightness = (color.R \* 299 + color.G \* 587 + color.B \* 114) / 1000;

return brightness > 128 ? Color.Black : Color.White;

}

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

{

var btn = sender as System.Windows.Forms.Button;

if (btn != null && !string.IsNullOrEmpty(btn.Text))

{

Clipboard.SetText(btn.Text);

MessageBox.Show($"Цвет {btn.Text} скопирован в буфер обмена", "Скопировано",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private void Color1\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color2\_Click\_1(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color3\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color4\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color5\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void Color6\_Click(object sender, EventArgs e) => ColorButton\_Click(sender, e);

private void lblPalitraLoad\_MouseLeave(object sender, EventArgs e)

{

lblPalitraLoad.Image = Properties.Resources.ButtonWhite;

}

private void lblPalitraLoad\_MouseEnter(object sender, EventArgs e)

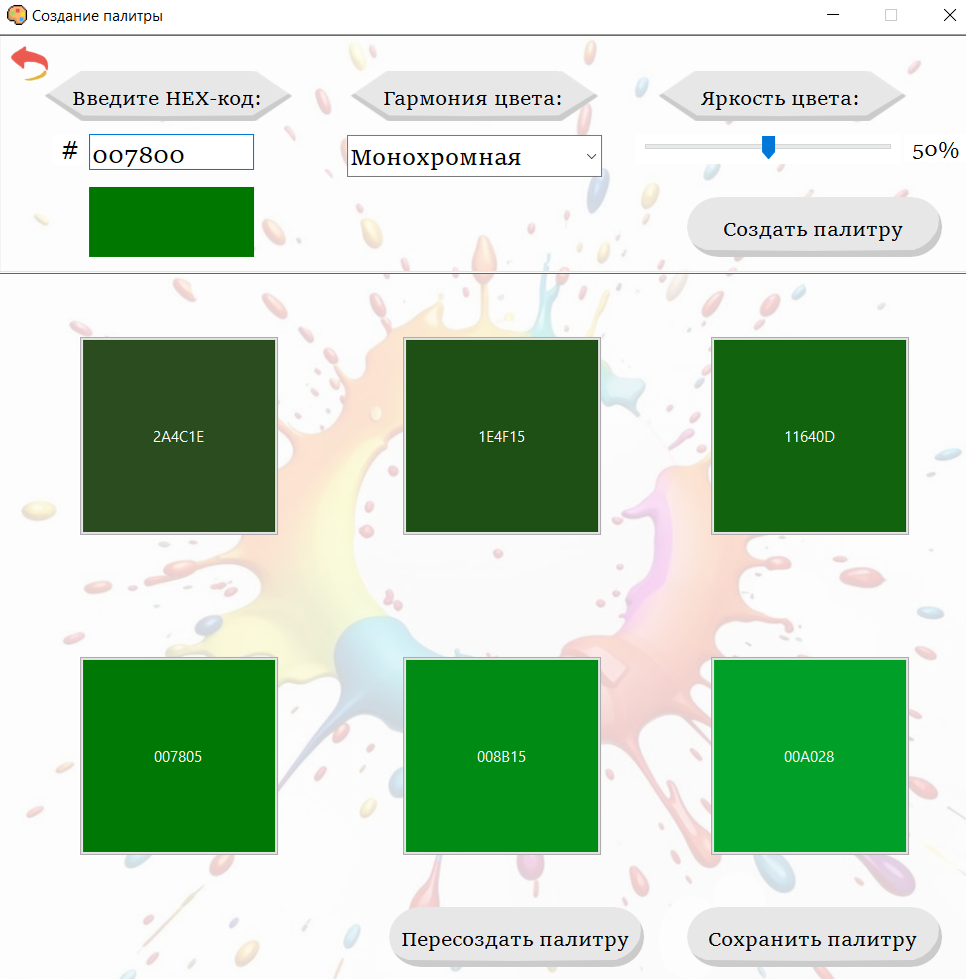
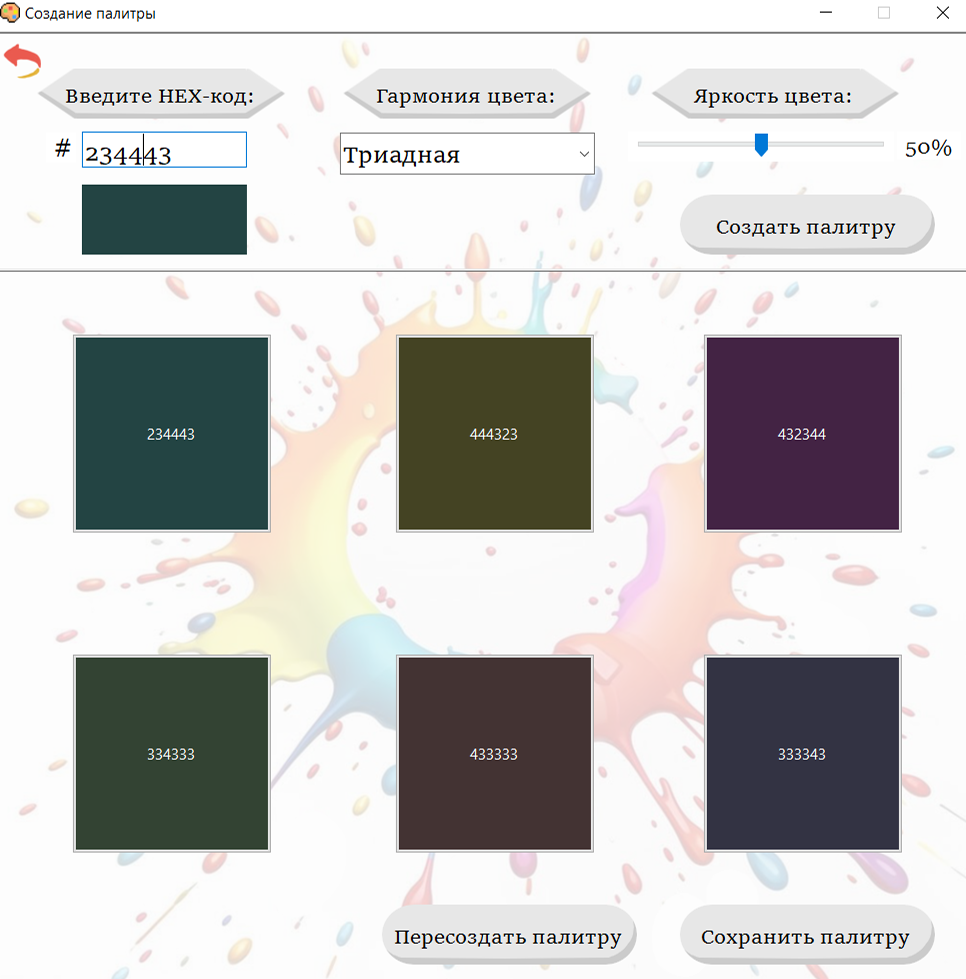
{

lblPalitraLoad.Image = Properties.Resources.Button;

}

}

}

1. **Примеры работы программы**
2. **Заключение**

Разработанное автоматизированное рабочее место дизайнера успешно решает поставленные задачи:

1. Автоматизация рутинных процессов – генерация гармоничных палитр занимает секунды вместо часов ручного подбора.
2. Гибкость настроек – поддержка 3 типов цветовых схем и регулировка яркости.
3. Удобство работы – интуитивный интерфейс и возможность сохранения результатов.

Перспективы развития:

* Добавление новых схем (аналоговая, квадратная).
* Интеграция с графическими редакторами (Photoshop, Figma).
* Мобильная версия для iOS/Android.

1. **Список используемых источников**
   1. [Документация по Windows Forms для .NET | Microsoft Learn](https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/)
   2. [usd3ng1tj8s3y9c1w9fh7dqvu6kbmkvu.pdf](https://www.obninskartcolledge.ru/upload/iblock/b65/usd3ng1tj8s3y9c1w9fh7dqvu6kbmkvu.pdf)
   3. [HEX-код: что это за кодировка для обозначения цветов](https://media.contented.ru/glossary/hex-kod/#:~:text=HEX-%D0%BA%D0%BE%D0%B4%D1%8B%20%E2%80%94%20%D1%8D%D1%82%D0%BE%20%D0%BE%D0%B1%D0%BE%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%20%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2,%D0%B1%D1%83%D0%BA%D0%B2%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%B0%D1%8F%20%D0%BE%D0%B1%D0%BE%D0%B7%D0%BD%D0%B0%D1%87%D0%B0%D0%B5%D1%82%20%D0%BA%D0%BE%D0%BD%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D1%8B%D0%B9%20%D1%86%D0%B2%D0%B5%D1%82)