Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

Факультет Информационные системы и технологии

Отчёт по лабораторной работе № 3

по дисциплине Технология программирования

Выполнил студент ИСТб-20-3 М. Д. Ярков 29.09.2021

номер группы подпись И. О. Фамилия Дата

Принял З. А. Бахвалова . .2021

должность подпись И. О. Фамилия Дата

Иркутск 2021

### Постановка задачи

### Создать класс для цвета, заданного в пространстве HSV, а также под каждую характеристику отдельный тип. Реализовать операции:

### Добавление/вычитание красного цвета

### Добавление/вычитание синего цвета

### Добавление/вычитание зеленого цвета

### Добавление/вычитание насыщености

### Добавление/вычитание яркости

### Перевод в RGB

### Словесное описание алгоритма

1. При запуске программы создаётся объект color класса Color, инициализируется интерфейс и производятся изменения значений на форме
2. При нажатии кнопки «Изменить цвета» или нажатии Enter во время изменения красного, зеленого или синего цвета, вызывается метод ChangeRgb:
   1. Производится изменение полей R, G, B объекта color на те, которые были введены:
      1. При изменении R, G или B класс Color автоматически проверяет вхождение изменённого числа в пределы от 0 до 255, и если число недопустимо, появляется ошибка
      2. В случае вхождения в пределы введённое число добавляется во внутреннюю переменную, а также обновляются значения HSV в соответствии с формулой 1
   2. Вызывается метод UpdateLabels:
      1. Производится изменение цвета индикатора на цвет, который содержится в color
      2. Производится изменение надписей интерфейса для отображения значений HSV и RGB
3. При нажатии кнопки «Изменить насыщ. / яркость» или нажатии Enter во время изменения насыщенности или яркости, вызывается метод ChangeHsv:
   1. Производится изменение полей S, V объекта color на те, которые были введены:
      1. При изменении S, V класс Color автоматически проверяет вхождение изменённого числа в пределы от 0 до 100 (для H максимальное значение 360), и если число недопустимо, появляется ошибка
      2. В случае вхождения в пределы введённое число добавляется во внутреннюю переменную, а также обновляются значения RGB в соответствии с формулой 2
   2. Вызывается метод UpdateLabels

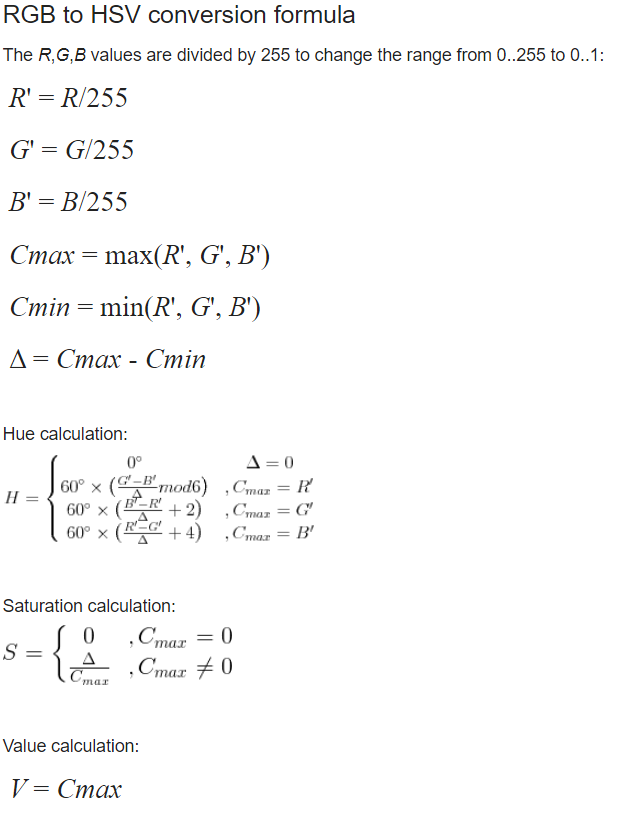


Рисунок 1 – формула 1

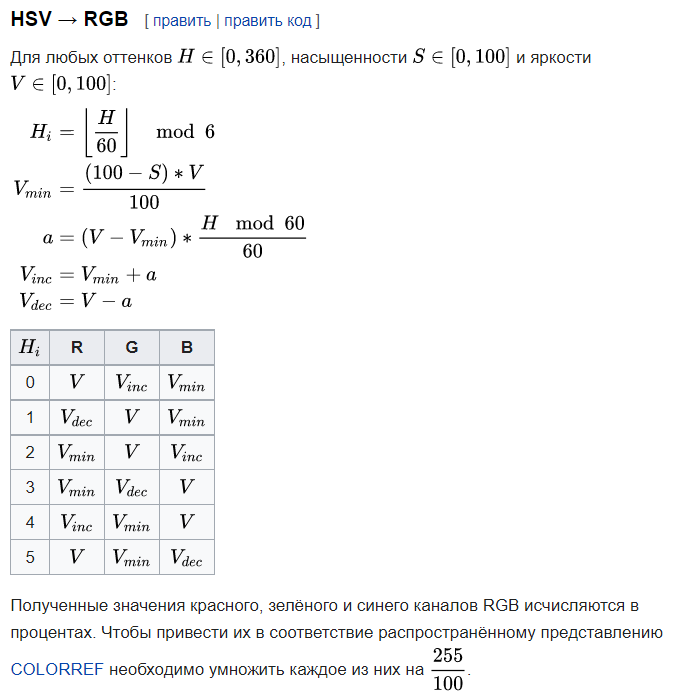


Рисунок 2 – формула 2

### Таблица спецификаций

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные/выходные величины | | |
| \_r, \_g, \_b, \_h, \_s, \_v | хранение цвета в схемах RGB и HSV | int |

### Таблица тестов

| Номер теста | Что проверяем | Входные данные | Выходные данные |
| --- | --- | --- | --- |
| 1 | Изменение любого цвета в разрешённых промежутках | 30  60  120 | Установлены соответствующие цвета, индикатор перекрашивается в другой цвет, обновлено представление в HSV |
| 2 | Изменение насыщенности/яркости в разрешённых промежутках | 95  60 | Установлена соответствующая насыщенность/яркость, индикатор перекрашивается в другой цвет, обновлено представление в RGB |
| 3 | Попытка установить значение за возможными пределами | -200 | Автоматическое изменение значения поля на ближайшее к введённому разрешённое число |

### Код программы

Основной файл Form1.cs:

|  |
| --- |
| using System;  using System.Windows.Forms;  namespace Lab3Var12 {  public partial class Form1 : Form {  Color color = new Color(); // создаём объект  public Form1() { // при запуске программы  InitializeComponent(); // инициализируем интерфейс  UpdateLabels(); // модифицируем поля в соответствии с введёнными данными  }  private void UpdateLabels() { // вывод данных  // отображаем текущий цвет в квадрате сверху  colorSquare.BackColor = System.Drawing.Color.FromArgb(255, color.R, color.G, color.B);  // изменение значений лейблов и изменяльщиков  hLabel.Text = "H: " + color.H;  sLabel.Text = "S: " + color.S; sChanger.Value = color.S;  vLabel.Text = "V: " + color.V; vChanger.Value = color.V;  rLabel.Text = "R: " + color.R; rChanger.Value = color.R;  gLabel.Text = "G: " + color.G; gChanger.Value = color.G;  bLabel.Text = "B: " + color.B; bChanger.Value = color.B;  }  private void ChangeRgb(object sender, EventArgs e) { // меняем RGB значения в зависимости от ввода  color.R = decimal.ToInt32(rChanger.Value);  color.G = decimal.ToInt32(gChanger.Value);  color.B = decimal.ToInt32(bChanger.Value);  UpdateLabels();  }  private void KeyPressedRgb(object sender, KeyEventArgs e) { // кнопка нажата в полях R/G/B  if (e.KeyCode == Keys.Enter) // при нажатии Enter  ChangeRgb(null, null); // устанавливаем введённые значения  }  private void ChangeHsv(object sender, EventArgs e) { // меняем SV значения в зависимости от ввода  color.S = decimal.ToInt32(sChanger.Value);  color.V = decimal.ToInt32(vChanger.Value);  UpdateLabels();  }  private void KeyPressedHsv(object sender, KeyEventArgs e) { // кнопка нажата в полях S/V  if (e.KeyCode == Keys.Enter) // при нажатии Enter  ChangeHsv(null, null); // устанавливаем введённые значения  }  }  } |

Дополнительный файл Color.cs:

|  |
| --- |
| using System;  namespace Lab3Var12 {  public enum ColorModels { HSV, RGB };  public class Color {  private int \_h, \_s, \_v, \_r, \_g, \_b;  public Color(ColorModels model = ColorModels.HSV, int var1 = 180, int var2 = 50, int var3 = 50) {  if (model == ColorModels.HSV) {  H = var1;  S = var2;  V = var3;  \_UpdateRgb();  }  else {  R = var1;  G = var2;  B = var3;  \_UpdateHsv();  }  }  public int H {  get => \_h;  set {  if (0 <= value && value <= 360) {  \_h = value;  \_UpdateRgb();  }  else  throw new ArgumentException("При изменении H получено недопустимое значение " + value);  }  }  public int S {  get => \_s;  set {  if (0 <= value && value <= 100) {  \_s = value;  \_UpdateRgb();  }  else  throw new ArgumentException("При изменении S получено недопустимое значение " + value);  }  }  public int V {  get => \_v;  set {  if (0 <= value && value <= 100) {  \_v = value;  \_UpdateRgb();  }  else  throw new ArgumentException("При изменении V получено недопустимое значение " + value);  }  }  public int R {  get => \_r;  set {  if (0 <= value && value <= 255) {  \_r = value;  \_UpdateHsv();  }  else  throw new ArgumentException("При изменении R получено недопустимое значение " + value);  }  }  public int G {  get => \_g;  set {  if (0 <= value && value <= 255) {  \_g = value;  \_UpdateHsv();  }  else  throw new ArgumentException("При изменении G получено недопустимое значение " + value);  }  }  public int B {  get => \_b;  set {  if (0 <= value && value <= 255) {  \_b = value;  \_UpdateHsv();  }  else  throw new ArgumentException("При изменении B получено недопустимое значение " + value);  }  }  private void \_UpdateRgb() {  int vMin = Convert.ToInt32(1.0 \* (100 - S) \* V / 100);  int a = Convert.ToInt32(1.0 \* (V - vMin) \* (H % 60) / 60);  int vInc = vMin + a;  int vDec = V - a;  switch ((H / 60) % 6) {  case 0:  \_r = V;  \_g = vInc;  \_b = vMin;  break;  case 1:  \_r = vDec;  \_g = V;  \_b = vMin;  break;  case 2:  \_r = vMin;  \_g = V;  \_b = vInc;  break;  case 3:  \_r = vMin;  \_g = vDec;  \_b = V;  break;  case 4:  \_r = vInc;  \_g = vMin;  \_b = V;  break;  case 5:  \_r = V;  \_g = vMin;  \_b = vDec;  break;  }  \_r = R \* 255 / 100;  \_g = G \* 255 / 100;  \_b = B \* 255 / 100;  }  private void \_UpdateHsv() {  double r = \_r / 255.0, g = \_g / 255.0, b = \_b / 255.0;  double min = Math.Min(r, Math.Min(g, b)), max = Math.Max(r, Math.Max(g, b));  if (max == min)  \_h = 0;  else if (max == r)  \_h = Convert.ToInt32(Math.Round(60.0 \* (1.0 \* (g - b) / (max - min) % 6)));  else if (max == g)  \_h = Convert.ToInt32(Math.Round(60.0 \* (b - r) / (max - min)) + 120);  else if (max == b)  \_h = Convert.ToInt32(Math.Round(60.0 \* (r - g) / (max - min)) + 240);  \_v = Convert.ToInt32(Math.Round(max \* 100));  \_s = max == 0 ? 0 : Convert.ToInt32(Math.Round((1 - (min / max)) \* 100));  }  }  } |

Файл с автотестами:

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using Lab3Var12;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Lab3Var12.Tests {  [TestClass()]  public class ColorTests {  [TestMethod()]  public void ColorsCreateTest1() {  Assert.ThrowsException<System.ArgumentException>(() => new Color(ColorModels.HSV, 100, 100, 101));  }  [TestMethod()]  public void ColorsCreateTest2() {  Assert.ThrowsException<System.ArgumentException>(() => new Color(ColorModels.HSV, 100, 100, -1));  }  [TestMethod()]  public void ColorsCreateTest3() {  Assert.ThrowsException<System.ArgumentException>(() => new Color(ColorModels.RGB, 100, 100, 256));  }  [TestMethod()]  public void ColorsCreateTest4() {  Assert.ThrowsException<System.ArgumentException>(() => new Color(ColorModels.RGB, -101, 100, 100));  }  [TestMethod()]  public void ColorsCreateTest5() {  Assert.ThrowsException<System.ArgumentException>(() => new Color(ColorModels.HSV, 361, -100, 0));  }  [TestMethod()]  public void ColorsCreateTest6() {  Assert.ThrowsException<System.ArgumentException>(() => new Color(ColorModels.HSV, -1, 100, 100));  }  [TestMethod()]  public void ColorsCreateTest7() {  Color color = new Color();  Assert.AreEqual(180, color.H);  }  [TestMethod()]  public void ColorsCreateTest8() {  Color color = new Color(ColorModels.HSV, 100, 100, 0);  Assert.AreEqual(100, color.S);  }  [TestMethod()]  public void ColorsCreateTest9() {  Color color = new Color(ColorModels.RGB, 100, 255, 0);  Assert.AreEqual(255, color.G);  }  }  } |

### Тесты

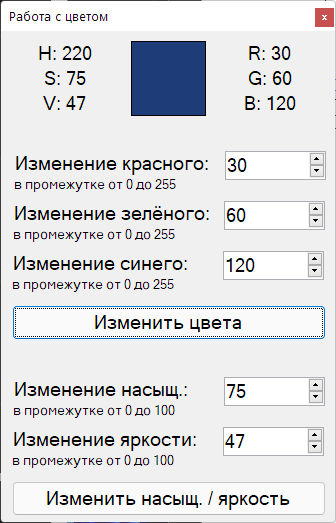


Рисунок 3 – тест 1

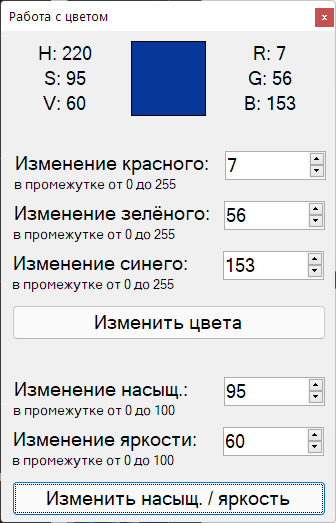


Рисунок 4 – тест 2

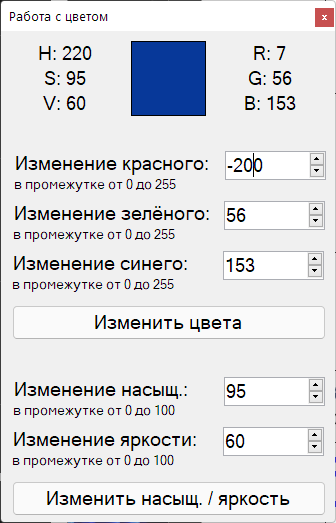


Рисунок 5 – тест 3 (введено число, не входящее в промежуток)

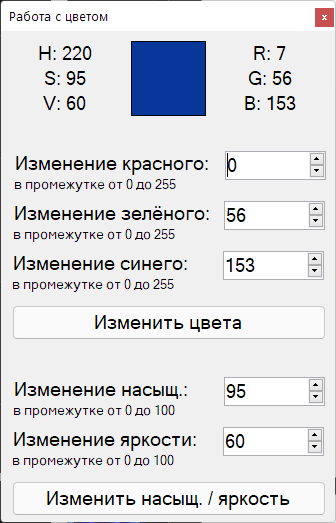


Рисунок 6 – тест 3 (введённое число автоматически изменилось на минимально возможное)