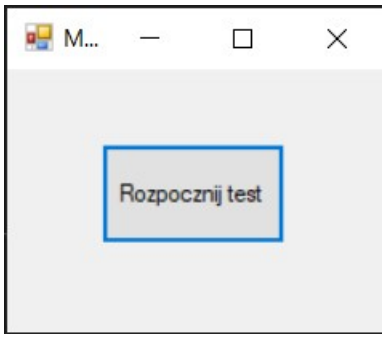
Sprawozdanie do zad. 2. w ramach laboratorium Organizacja Systemów Komputerowych

Paweł Szczepański (188640), Igor Szczyrbak (188661)

28 kwietnia 2024

1 Założenia programu

Aplikacja służy do testowania szybkości reakcji w odpowiedzi na bodźce wzrokowe i słuchowe oraz koordynacji ręka-oko. Służy do tego seria testów. Każdy z testów odbywa się w odrębnym oknie. Testy są poprzedzane oknem informującym o przebiegu testu oraz dwoma testami próbnymi. Właściwy test jest powtarzany czterokrotnie. Wciśnięcie przycisku "Rozpocznij test"rozpoczyna serię testów poprzez wywoływanie kolejnych okien.



Rysunek 1: Okno startowe aplikacji

```
private void button1_Click(object sender, EventArgs e)
{
    tests[0] = new Form2(this);
    tests[1] = new Form3(this);
    tests[2] = new Form4(this);
    tests[3] = new Form5(this);

    results = new double[testAmount, maxTests];
    currentTest = 0;
    while (currentTest < testAmount)
    {
        tests[currentTest].ShowDialog();
        currentTest++;
    }

    Form6 wykres = new Form6(results);
    wykres.ShowDialog();
}
```

2 Rozwiązania programowe

2.1 Pomiar czasu i przesyłanie danych

Okna poszczególnych testów zapisują referencję do okna głównego, co umożliwia przesyłanie wyników testów do okna głównego poprzez funkcję publiczną AddResults().

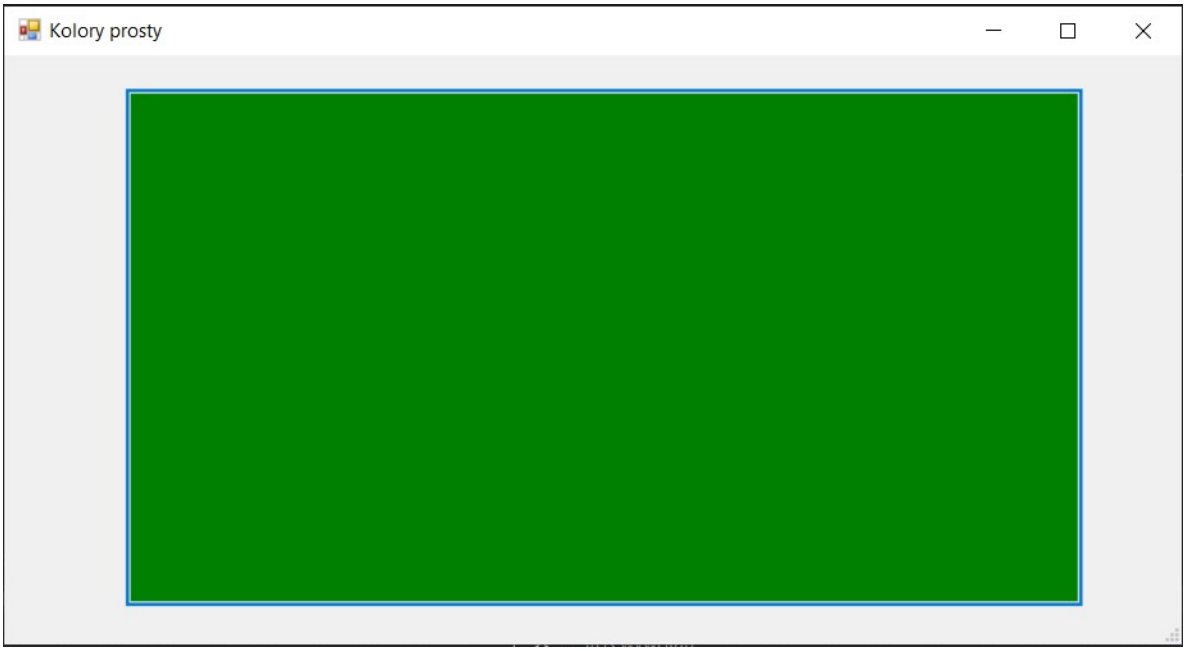
```
public void AddResults(int resultNum, double result) =>
    results[currentTest, resultNum] = (int)result;
```

Czas reakcji jest mierzony przy pomocy funkcji DateTime.Now. Umożliwia to dokładny pomiar czasu. Rezultaty są zapisywane i wyświetlane w milisekundach.

```
start = DateTime.Now;
.
.
double result = (DateTime.Now - start).TotalMilliseconds;
parent.AddResults(testsPassed - learningTests, result);
```

2.2 Prosty test reakcji na zmianę koloru

Okno testu zawiera duży kolorowy przycisk. Test polega na naciśnięciu przycisku lewym przyciskiem myszy, po tym jak przycisk zmieni kolor z czerwonego na zielony.



Rysunek 2: Okno testu

Okno informujące o przebiegu testu jest wyświetlane przy pomocy funkcji MessageBox.Show().

```
MessageBox.Show("W tym teście kliknij kolorowe pole przycisk lewym przyciskiem myszy, gdy
    zmieni kolor na zielony. " + "Wykonaj teraz " + learningTests.ToString() + " testy
    próbne.");
```

Po rozpoczęciu testu uruchamiany jest timer o losowym interwale. Służy on do odmierzenia czasu od rozpoczęcia testu do zmiany koloru pola.

```
private void Reset()
{
```

```
        timer1.Interval = r.Next(Form1.wait[0], Form1.wait[1]);
        timer1.Enabled = true;
        button1.BackColor = Color.Red;
        canClick = false;
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        timer1.Enabled = false;
        button1.BackColor = Color.Green;
        canClick = true;
        start = DateTime.Now;
    }
```

W pozostałych testach odmierzanie czasu do rozpoczęcia pomiaru oraz okna informacyjne są obsługiwane w taki sam sposób.

2.3 Złożony test reakcji na zmianę koloru

Okno testu zawiera duży kolorowy przycisk. Test polega na naciśnięciu przyciku lewym przyciskiem myszy, jeżeli ten zmieni kolor z czerwonego na zielony i prawym przyciskiem myszy jeżeli zmieni kolor na niebieski. Jest to sprawdzane poprzez flagę zmiennej typu MouseEventArgs, informującej o wciśniętym przycisku.

```
        private void button1_MouseDown(object sender, MouseEventArgs e)
    {
        if (!canClick)
            MessageBox.Show("Za szybko! Test zostanie powtórzony.");
        else
        {
            if (e.Button.HasFlag(MouseButtons.Left))
                //Obsługa lewego przyciksu
            else if (e.Button.HasFlag(MouseButtons.Right))
                //Obługa prawego przycisku
        }

        Reset();
    }
```

2.4 Test czasu reakcji na dźwięk

Okno testu zawiera duży zielony przycisk. Test polega na naciśnięciu przycisku po usłyszeniu dźwięku. Dźwięk jest odgrywany przy pomocy funckji SoundPlayer.Play(). Plik audio formut .wav jest zapisany w zasobach aplikacji.

```
using System.Media;
    .
    .
    SoundPlayer sound1 = new SoundPlayer(Properties.Resources.click);
    .
    .
    sound1.Play();
```

2.5 Test koordynacji ręka-oko

Okno testu jest puste, po rozpoczęciu odliczania czasu pojawia się w nim ruchomy zielony przycisk. Test polega na naciśnięciu ruchomego przycisku po jego pojawieniu.

Rysunek 3: Okno testu ręka-oko

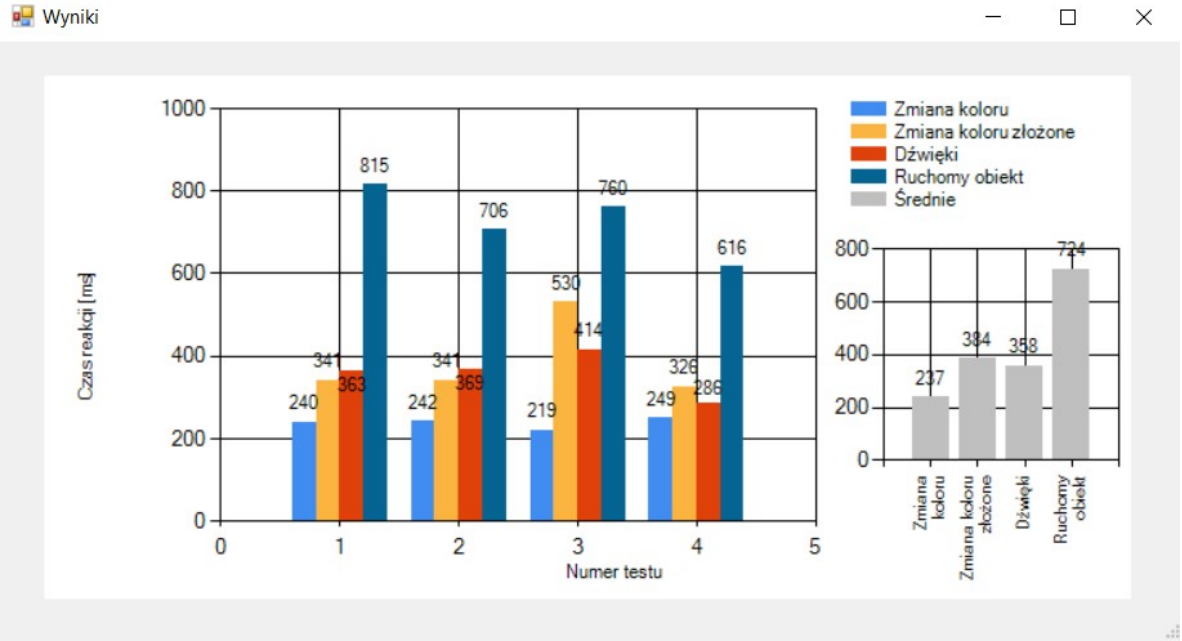
Do obliczania i zmiany pozycji przycisku wykorzystywany jest timer2 wywołujący odpowiednią funkcję co 30ms (około 30 klatek na sekundę). Zmienna direction opisuje kierunek ruchu przycisku w radianach, a jej wartość jest losowana.

```
private void timer2_Tick(object sender, EventArgs e)
{
    //Obsługa "zderzeń" z krawędziemi okna
    if (position[1] < 0)
    {
        position[1] = 0;
        direction = r.NextDouble() * -Math.PI - Math.PI;
    }
    else if (position[0] < 0)
    {
        position[0] = 0;
        direction = r.NextDouble() * Math.PI - Math.PI / 2;
    }
    else if (position[1] > 325)
    {
        position[1] = 325;
        direction = r.NextDouble() * -Math.PI;
    }
    else if (position[0] > 700)
    {
        position[0] = 700;
        direction = r.NextDouble() * Math.PI + Math.PI / 2;
    }

    //speed jest podana w punktach na sekudę, a frameTime w milisekundach, stąd dzielenie
        przez 10^3
    position[0] += Math.Cos(direction) * speed * frameTime / 1000;
    position[1] += Math.Sin(direction) * speed * frameTime / 1000;
    button1.Location = new Point((int)(position[0]), (int)(position[1])); //Pozycja musi
        być podana w liczbach całkowitych
}
```

2.6 Prezentacja wyników testu

Ostatnie okno prezentuje wyniki testów w formie wykresu słupokowego. Wykres po lewej informuje o wynikach w poszczególnych prób, wykres po prawej o średnich wynikach danych kategorii testów.



Rysunek 4: Okno prezentacji wyników

Wykresy realizowane są przy pomocy komponentu klasy Chart.Przy inicjalizacji okna przesyłane są do niego zebrane wyniki, które następnie są dzielone na tablice według rodzajów testu, obliczane są średnie i tablice są przypisane konkretnym seriom danym wykresu.

```
public Form6(double[,] data)
   {
       InitializeComponent();

       x = new double[Form1.maxTests];
       form2 = new double[Form1.maxTests];
       form3 = new double[Form1.maxTests];
       form4 = new double[Form1.maxTests];
       form5 = new double[Form1.maxTests];
       average = new double[4];

       for (int i = 0; i < Form1.maxTests; i++)
       {
           x[i] = i + 1;
           form2[i] = data[0, i];
           form3[i] = data[1, i];
           form4[i] = data[2, i];
           form5[i] = data[3, i];

           average[0] += data[0, i];
           average[1] += data[1, i];
           average[2] += data[2, i];
           average[3] += data[3, i];
       }

       for (int i = 0; i < 4; i++)
       {
```

```
            average[i] /= Form1.maxTests;
            average[i] = (int)average[i];
        }

        string[] avTitles = new string[4];
        avTitles[0] = "Zmiana koloru";
        avTitles[1] = "Zmiana koloru złożone";
        avTitles[2] = "Dźwięki";
        avTitles[3] = "Ruchomy obiekt";


        chart1.Series.ElementAt(0).Points.DataBindXY(x, form2);
        chart1.Series.ElementAt(1).Points.DataBindXY(x, form3);
        chart1.Series.ElementAt(2).Points.DataBindXY(x, form4);
        chart1.Series.ElementAt(3).Points.DataBindXY(x, form5);
        chart1.Series.ElementAt(4).Points.DataBindXY(avTitles, average);
    }
```

3 Dyskusja osiągniętych wyników

Aplikacja pozwala na testowanie i wyświetlanie czasu reakcji na bodźce wzrokowe i słuchowe, ale nie przechowuje tych danych po wyświetleniu wykresów. Struktura kodu pozwala na łatwe dodawanie kolejnych testów.