Michał Sakowicz 175907, Mateusz Woźniak 175610

1. Sformułowanie zadania wraz z przyjęciem założeń szczegółowych

Nasze zadanie opierało się na stworzeniu aplikacji spełniającej funkcję testera sprawności psychomotorycznej. Przyjęliśmy, iż aplikacja będzie składać się z trzech testów – 2 testów optycznych oraz jednego testu akustycznego.

Pierwszy wymyślony przez nas test optyczny polega na jak najszybszym kliknięciu przycisku gaz/hamulec po zaobserwowaniu odpowiedniego światła. Drugi test polega na kliknięciu opowiadającego podświetlanym uprzednio „lampkom" przycisku. Natomiast trzeci test skupia się na szybkości reakcji na bodziec akustyczny. Użytkownik musi jak najszybciej kliknąć przycisk „STOP" po usłyszeniu dźwięku.

Pomiar czasu odbywa się poprzez obliczenie różnicy między czasem od wystąpienia sygnału do momentu prawidłowej interakcji użytkownika. Czas zdefiniowany jest jako ilość ms, które upłynęły od 01.01.1970 r.

2. Opis przyjętych rozwiązań programowych

Program podzieliśmy na 5 podstawowych klas. Klasa główna – Form1, klasy testów – Test1, Test2, Test3 oraz jedną klasę która jest odpowiedzialna za obsługę wyników.

W Formularzu Form1 po kliknięciu w dany przycisk wywoływane są kolejne testy, natomiast po kliknięciu w przycisk „WYNIKI" ukazywane są wyniki w reprezentacji analitycznej oraz syntetycznej.

Formularz Test1 oparliśmy głównie na asynchronicznej funkcji changeButtonColor(), odpowiedzialnej za zmianę koloru światła. Losuje ona przycisk który w danym momencie ma zostać zapalony. Po wyborze koloru czerwonego lub zielonego rozpoczyna się odliczanie czasu. Odliczanie czasu trwa aż do chwili wciśnięcia odpowiednio hamulca i gazu. Niestandardową opcja jest ustawienie komponentu graficznego symbolizującego sygnalizację świetlną na kształt elipsy (przykład w jaki sposób to osiągnęliśmy wstawiam na kolejno drugim zrzucie ekranu)

```
private async void changeButtonColor()
        {
            buttonNumber = new Random().Next(3);
            await Task.Delay(new Random().Next(500, 2000));     //delay of changing colors
            if (buttonNumber == 0)
            {
                pictureBox1.BackColor = Color.Red;
                time = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            }
            else if (buttonNumber == 1)
            {
                pictureBox2.BackColor = Color.Orange;
                await Task.Delay(new Random().Next(500, 2000));
                pictureBox2.BackColor = Color.Black;
                addToTotalMeasurements();
                if (results.Count != (totalNumberOfAttempts-numberOfMainAttempts))
                {
                    changeButtonColor();
                }
            }
            else if (buttonNumber == 2)
            {
                pictureBox3.BackColor = Color.Green;
                time = DateTimeOffset.Now.ToUnixTimeMilliseconds();
            }
            addToTotalMeasurements();
        }
```

```
public class RoundButton : Button
    {
        protected override void OnPaint(System.Windows.Forms.PaintEventArgs e)
        {
            GraphicsPath grPath = new GraphicsPath();
            grPath.AddEllipse(4, 4, ClientSize.Width-10, ClientSize.Height-10);
            this.Region = new System.Drawing.Region(grPath);
            base.OnPaint(e);
        }
    }
```

Formularz Test2 opiera się na przyporządkowywaniu tagów do przycisków, które odpowiadają współrzędnym zapalających się diod. Główne funkcje realizujące to zadanie zostały ukazane na poniższych zrzutach ekranu.

```
  private async void turnOnDiode()
        {
            await Task.Delay(new Random().Next(500, 2000));
            Random rnd = new Random();
            leftDiode = rnd.Next(4);
            bottomDiode = rnd.Next(4, 8);
            pictureBox[leftDiode].BackColor = Color.Red;
            pictureBox[bottomDiode].BackColor = Color.Red;
            time = DateTimeOffset.Now.ToUnixTimeMilliseconds();
        }
```

```
  private void button_Click(object sender, EventArgs e)
      {
          Button button = (Button)sender;
          if (pictureBox[leftDiode].BackColor == Color.Red && pictureBox[bottomDiode].BackColor ==
              Color.Red && button.Tag.ToString() == (leftDiode.ToString()+bottomDiode.ToString()))
          {
              pictureBox[leftDiode].BackColor = Color.Black;
              pictureBox[bottomDiode].BackColor = Color.Black;
              timeDiff = DateTimeOffset.Now.ToUnixTimeMilliseconds() - time;
              results.Add(timeDiff);
              addToTotalMeasurements();
              if (results.Count != (totalNumberOfAttempts - numberOfMainAttempts))
              turnOnDiode();
          }
      }
```

W teście trzecim zamiast obsługi zapalania się świateł obsługiwaliśmy dźwięk – klasa System.Media.SoundPlayer. Gdy użytkownik był gotowy do rozpoczęcia pomiaru dźwięk był odtwarzany natomiast po kliknięciu odpowiedniego przycisku muzyka była stopowana – wywoływana była instancja klasy SoundPlayer która uprzednio odtwarzała dźwięk, a w momencie kliknięcia przycisku go pauzowała. Klasa w dużej mierze odpowiedzialną za prawidłowe działanie testu były metody playMusic() oraz stopMusic() ukazana na poniższym screenie.

```
private async void playMusic()
        {
            this.button1.Enabled = false;
            await Task.Delay(new Random().Next(1000, 2500));     //delay of changing colors
            this.button1.Enabled = true;
            soundPlayer.Play();
            time = DateTimeOffset.Now.ToUnixTimeMilliseconds();
        }

        private void stopMusic()
        {
            soundPlayer.Stop();
            long timeDiff = DateTimeOffset.Now.ToUnixTimeMilliseconds() - time;
            results.Add(timeDiff);
        }
```

Cechą wspólną wszystkich testów są funkcje odpowiadające za obliczanie średniej i dodanie tej wartości do globalnych wyników.

Klasa wyników opiera się na zastosowaniu komponentu Chart i manipulacji nim – dodanie opisu osi Y, osi X, maksimów tych osi, a także ustawienie kursora na progu zaliczeniowym. Za odpowiedni próg zaliczeniowy uznaliśmy wartość 900ms. Dla testów przeprowadzanych w normalnych warunkach tj. w gabinecie na rzeczywistych urządzeniach ten próg wynosi 1000ms. Ze względu na prostszą formę testów komputerowych nieznacznie zmniejszyliśmy owy próg.

3. Dyskusja osiągniętych wyników

Ogromną zaletą naszej aplikacji jest to, że symuluje on realne testy wykonywane podczas psychotestów. Sami mieliśmy do czynienia z takimi testami, aby mieć możliwość wykonywania pracy sezonowej (instalacje teletechniczne na wysokościach). Z tego powodu tworzenie oprogramowania symulującego te testy sprawiło nam dużą satysfakcję.

Trudności jakie napotkaliśmy to przesłanie wyników do zmiennej globalnej – ostatecznie zrealizowane zostało to za pomocą zmiennej statycznej z getterem seterem. Inną, aczkolwiek banalną trudnością było dobranie czytelnego tła – każdy z twórców miał inną wizję na ten temat, jednak ostatecznie udało nam się dojść do porozumienia. Po wykonaniu pierwszego testu postęp naszych prac zdecydowanie przyspieszył ze względu na analogie zastosowanych technologii w kolejnych testach. Kolejną trudnością było także opóźnienie w wykonywaniu danych metod końcowo zrealizowane dzięki funkcji asynchronicznej – mieliśmy wielki problem z samodzielnym napisaniem funkcji spełniającej nasze wymagania, jednakże ostatecznie udało nam się zaznajomić z pojęciem funkcji asynchronicznej i dzięki temu wszystkie nasze problemy automatycznie się rozwiązały.