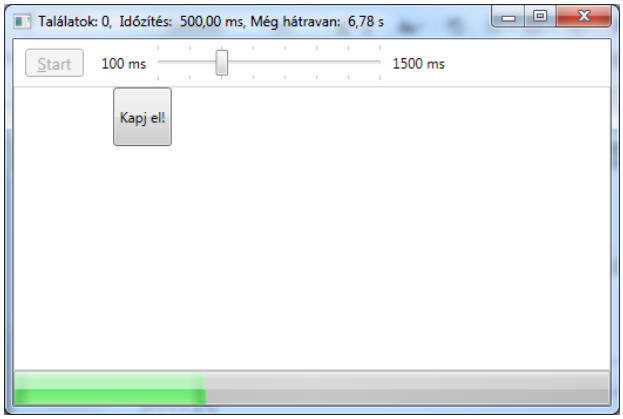
Johanyák Zsolt Csaba: **Ugráló gomb – oktatási segédlet**
http://www.johanyak.hu
e-mail: johanyak.csaba@gamf.kefo.hu

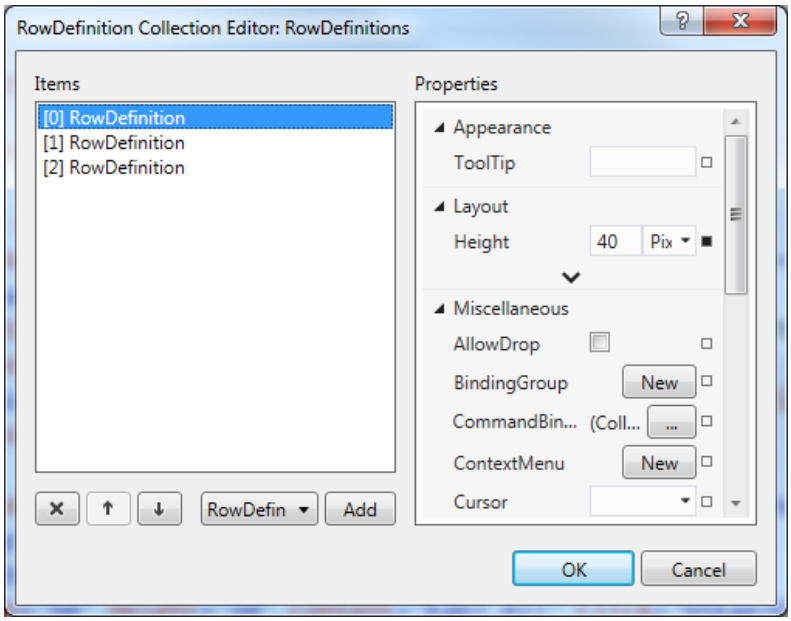
Ugráló gomb



Készítsünk egy egyszerű játékprogramot, ami egy mozgó nyomógombot tartalmaz. A nyomógomb beállított ideig marad egy helyben, majd az ablakon számára elhatárolt terület (panel) egy véletlenszerűen kiválasztott pozíciójában jelenik meg újra. A játék során az a feladat, hogy minél többször kattintsunk a gombon az egér segítségével. A játék előre beállított ideig tart, az ablak alján elhelyezett végrehajtásjelző tájékoztat az eltelt időről.

A feladat egy lehetséges megoldása a következő:

1. Az alkalmazás vázának automatikus generálása
   Fájl menü, New, Project …, Installed, Templates, Visual C# , Windows, WPF Application
   Name: UgraloGomb
   OK



2. Felület kialakítása

- Az ablak osztályának neve legyen wndUgrálóGomb, az őt definiáló állomány nevét is változtassuk meg wndUgraloGomb.xaml-re. Majd módosítsuk az App.xaml állományban A StartupUri attribútum értékét StartupUri="wndFoablak.xaml"-re.
- A felületmenedzser neve legyen grRács. Hozzunk létre benne három sort. Az első sor magassága legyen 40, a harmadiké 30.
- Helyezzünk el egy StackPanel-t az első sorban spEszköztár névvel, úgy, hogy töltse ki a területet.
- Egy nyomógombot (Button) helyezünk a a StackPanelre . x:Name=btStart, Text=&Start. Ezzel lehet majd elindítani a játékot.
- Egy csúszkát (Slider) helyezünk el a nyomógomb jobb oldalára. x:Name=slCsúszka. Ezzel lehet majd beállítani, hogy mennyi ideig maradjon egy helyben a mozgó nyomógomb.
- Egy-egy címkét (Label) helyezünk el a csúszka bal és jobb oldalára. Feladatuk a csúszkával beállítható minimális és maximális időérték kijelzése. Nevük: x:Name=llMin és x:Name=llMax.
- Egy végrehajtásjelzőt (ProgressBar) helyezünk el az ablak aljába (a rács harmadik sora). Ez fog tájékoztatni az eltelt játékidőről. Tulajdonságai: x:Name=pbVégrehajtásJelző.
- Egy keretet helyezünk el a rács második sorába, ebbe kerül a játéktér. Erre azért van szükség, hogy jól láthatóan elkülönítsük a játékteret az ablak többi részétől. Tulajdonságai:

x:Name=brKeret.

- A mozgó nyomógomb helyzetét a bal felső sarkainak koordinátáival szeretnénk szabályozni, ezért a játékteret egy Canvas komponenssel valósítjuk meg. Tulajdonságai: Name=cvLap, HorizontalAlignment="Stretch"
- Egy nyomógombot (Button) helyezünk a játéktérre, ezen kell kattintani játék közben. Fontosabb tulajdonságai: Name=btKapjEl, Text=Kapj el!

A felület teljes definíciója az alábbi.

```
<Window x:Class="UgraloGomb.wndUgrálóGomb"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid x:Name="grRács">
        <Grid.RowDefinitions>
            <RowDefinition Height="40" />
            <RowDefinition />
            <RowDefinition Height="30" />
        </Grid.RowDefinitions>
        <StackPanel x:Name="spEszköztár" Grid.Row="0" HorizontalAlignment="Stretch"
              Height="30" Orientation="Horizontal">
            <Button x:Name="btStart" Width="70"
              FontSize="14" Margin="10,0" Click="btStart_Click">_Start</Button>
            <Label x:Name="llMin" Content="ms" VerticalAlignment="Center"  />
            <Slider x:Name="slCsúszka" VerticalAlignment="Center" Width="200"
                    TickPlacement="Both" ValueChanged="slCsúszka_ValueChanged" />
            <Label x:Name="llMax" Content="ms" VerticalAlignment="Center"  />
        </StackPanel>
        <Border x:Name="brKeret" Grid.Row="1" BorderThickness="1" BorderBrush="LightGray" >
            <Canvas x:Name="cvLap" >
                <Button x:Name="btKapjEl" Canvas.Left="100" Canvas.Top="100"
                Width="50" Height="30" Content="Kapj el!" Click="btKapjEl_Click"
                        MouseEnter="btKapjEl_MouseEnter" MouseLeave="btKapjEl_MouseLeave" />
            </Canvas>
        </Border>
        <ProgressBar x:Name="pbVégrehajtásJelző" Grid.Row="2" />
    </Grid>
</Window>
```

3. Adattagok definiálása.
   Hozzunk létre az ablak osztályában egy adattagot az elért pontszám tárolására.

```
/// <summary>
/// Az elért pontszám.
/// </summary>
private int Eredmény;
```

   Hozzunk létre az ablak osztályában egy adattagot véletlenszámok előállítására szolgáló objektum referenciájának tárolására.

```
/// <summary>
/// Véletlenszámok előállítására szolgáló objektum.
/// </summary>
private Random Véletlen;
```

Hozzunk létre egy adattagot a játék kezdő időpillatatának tárolásásra.

```
/// <summary>
/// A játék kezdete.
/// </summary>
```

```
private DateTime KezdőIdő;
```

Hozzunk létre egy adattagot a megengedett játékidő tárolásásra.

```
/// <summary>
/// Megengedett játékidő másodpercben.
/// </summary>
private int MaxJátékIdő;
```

A játékidő méréséhez egy Timer objektumra lesz szükségünk. Mivel az időzítés lejártakor a felhasználói felületen kell végrehajtanunk változtatásokat, ezért a DispatcherTimer-t válasszuk a feladathoz.

```
/// <summary>
/// Időzítő a játékidő méréséhez és a gomb mozgatásához.
/// </summary>
private DispatcherTimer dtIdőzítő;
```

Hozzunk létre egy logikai adattagot, ami a későbbiekben a találatok érvényességének ellenőrzéséhez lesz szükséges.

```
/// <summary>
/// Meghatározza, hogy találatot jelent-e a Click esemény.
/// </summary>
private bool Érvényes;
```

4. Kezdőérték adás a konstruktorban. Az alábbi utasításokkal beállítjuk az ablakon elhelyezett komponensek tulajdonságait. Az időzítő Tick eseményéhez kapcsolódó eseménykezelő (`dtIdőzítő_Tick`) vázát a Visual Studioval generáltatjuk le automatikusan.

```
/// <summary>
/// Konstruktor. A koponensek egyes tulajdonságainak beállítása.
/// </summary>
public wndUgrálóGomb()
{
  InitializeComponent();
  // Timer objektum létrehozása a játékidő követésére és a gomb mozgatásához.
  // Az időzítő alapbeállításként 0,5 másodpercenként jelez.
  // Időzítő kezdetben leállítva.
  dtIdőzítő = new DispatcherTimer
    { Interval = new TimeSpan(0, 0, 0, 0,500), IsEnabled = false };
  // Eseménykezelő az  időzítőhöz.
  dtIdőzítő.Tick+= dtIdőzítő_Tick;
  // Alsó és felső határérték ezredmásodpercben arra, hogy mennyi ideig
  // maradhat egy helyben a mozgó nyomógomb.
    slCsúszka.Minimum = 100;
    slCsúszka.Maximum = 1500;
    // A csúszka jelzővonalainak távolsága.
    slCsúszka.TickFrequency = 200;
    // Mekkora elmozdulást jelent a csúszkán a le/fel nyíl billentyű
    // lenyomása?
    slCsúszka.SmallChange = 100;
    // Mekkora elmozdulást jelent a csúszkán a Page Up/Page Down
    // billentyű lenyomása?
    slCsúszka.LargeChange = 500;
    // A csúszka kezdeti pozíciója.
    slCsúszka.Value = 500;
```

```
      // A csúszka bal és jobb oldali címkének (feliratok) szövege.
      llMin.Content = slCsúszka.Minimum + " ms";
      llMax.Content = slCsúszka.Maximum + " ms";
      // A mozgó gomb kezdetben letiltva.
      btKapjEl.IsEnabled = false;
    // A megengedett játékidő másodpercben.
    MaxJátékIdő = 10;
      // Végrehajtásjelző szélsőértékeihez társított számértékek.
      pbVégrehajtásJelző.Minimum = 0;
    pbVégrehajtásJelző.Maximum = MaxJátékIdő;  // A játékidő másodpercben.
      // Végrehajtásjelző kezdőértéke.
      pbVégrehajtásJelző.Value = 0;
    // Véletlenszámokat előállító objektum létrehozása.
    Véletlen = new Random();
  }
```

5. Eseménykezelő készítése a mozgásidőzítő Tick eseményéhez. Az eseménykezelő utasításblokkjában frissítjük a feliratot az ablak fejlécében, beállítjuk a végrehajtásjelzőt, és új pozícióba helyezzük a KapjEl gombot. A pozíciót meghatározó adattagok (`LeftProperty`, `TopProperty`) DependencyProperty típusúak, ezért értéküket nem tudjuk közvetlenül beállítani. Az értékadás a a nyomógomb objektum SetValue() metódusa segítségével történik. Ha lejárt a játékidő, akkor le kell állítani az időzítőt. Az ablak fejlécébe történő feliratkiírás egy külön metódus feladata (`FeliratKiír()`), itt csak meghívjuk a metódust és generáltatjuk a vázát a Visual Studio segítségével.

```
    /// <summary>
    /// Eseménykezelő: lejárt az időzítő időegysége. Lépteti a
    /// végrehajtásjelzőt. Frissíti a feliratot az ablak fejlécében.
  /// Beállítja a végrehajtásjelzőt. Új pozícióba helyezi a KapjEl gombot.
  /// Ha lejárt a játékidő, akkor leállítja az időzítőt, letiltja a
  /// KapjEl gombot, és engedélyezi a Start gombot.
    /// </summary>
  void dtIdőzítő_Tick(object sender, EventArgs e)
  {
    // Állapot kiírása az ablak fejlécébe.
    FeliratKiír();
    // Beállítjuk a végrehajtásjelzőt.
    pbVégrehajtásJelző.Value = ElteltIdő();
    // Ha még nem járt le a játékidő, gomb mozgatása új pozícióba.
    if (ElteltIdő() < MaxJátékIdő)
    {
      // A gomb bal felső sarkának új x koordinátája.
      btKapjEl.SetValue(LeftProperty,
        Véletlen.NextDouble() * (cvLap.ActualWidth - btKapjEl.ActualWidth));
      // A gomb bal felső sarkának új y koordinátája.
      btKapjEl.SetValue(TopProperty,
        Véletlen.NextDouble() * (cvLap.ActualHeight - btKapjEl.ActualHeight));
    }
    else // Ha lejárt a játékidő.
    {
      // Időzítő leállítása.
      dtIdőzítő.IsEnabled = false;
      // Start gomb engedélyezése.
      btStart.IsEnabled = true;
      // Mozgó nyomógomb letiltása.
      btKapjEl.IsEnabled = false;
    }
  }
```

6. Az ablak fejlécében feliratot megjelenítő metódus definiálása. Az ablak fejlécében ki akarjuk jelezni, hogy mennyi az eddig elért találatok száma, mennyi időközönként mozdul el a nyomógomb, és mennyi idő van még hátra a játékból. A feladat megoldásához szükségünk van a játék kezdetétől eltelt másodpercek számára. ennek kiszámítását egy külön metódusban (`ElteltIdő()`) helyezzük el.

```
/// <summary>
/// Friss információkat jelenít meg a játék állásáról az ablak
/// fejlécében.
/// </summary>
private void FeliratKiír()
{
   Title = string.Format(
      "Találatok: {0},  Időzítés: {1,7:F2} ms, Még hátravan: {2,5:F2} s",
      Eredmény, slCsúszka.Value, Math.Max(0,MaxJátékIdő-ElteltIdő()));
}

/// <summary>
/// Kiszámolja a játék kezdetétől eltelt időt másodpercben.
/// </summary>
/// <returns>Eltelt idő.</returns>
double ElteltIdő()
{
   // Lekérdezzük az aktuális időt.
   DateTime Most = DateTime.Now;
   // A játék kezdete óta eltelt idő.
   return Most.Subtract(KezdőIdő).TotalSeconds;
}
```

7. Eseménykezelő készítése a Kapj El! Gombhoz

```
/// <summary>
/// Eseménykezelő: a felhasználó kattintott a KapjEl gombon.
/// Növeli eggyel az eredményt és frissíti a feliratot az
/// ablak fejlécében. Csak akkor számol találatot, ha az
/// Érvényes adattag értéke igaz.
/// </summary>
private void btKapjEl_Click(object sender, RoutedEventArgs e)
{
   // Ha az érvényesség ellenőrzést nem építenénk be, akkor az
 // Enter gomb lenyomása is pontot eredményezne.
 if (!Érvényes) return;
   // Ha érvényes a kattintás, azaz a nyomógomb felett volt az egér az esemény
   // keletkezésekor.
   Eredmény++;
   // Eredmény megjelenítése az ablak fejlécében.
   FeliratKiír();
}
```

Csak az egérrel elért találatokat tekintjük érvényesnek, ezért meg szeretnénk akadályozni, hogy a játékos az Enter gomb megnyomásával is pontot szerezzen. Az érvényesség ellenőrzést úgy oldjuk meg, hogy amikor az egér a KapjEl gomb felé kerül, akkor az Érvényes logikai adattagot igazra állítjuk, és amikor az egér elhagyja a nyomógomb területét, akkor az adattag értékét hamisra állítjuk.

```
/// <summary>
```

```
/// Igazra állítja az Érvényes adattagot, amikor az egérkurzor belép a
/// nyomógomb területére.
/// </summary>
private void btKapjEl_MouseEnter(object sender, MouseEventArgs e)
{
   Érvényes = true;
}

/// <summary>
/// Hamisra állítja az Érvényes adattagot, amikor az egérkurzor kilép a
/// nyomógomb területéről.
/// </summary>
private void btKapjEl_MouseLeave(object sender, MouseEventArgs e)
{
   Érvényes = false;
}
```

8. A játékot elindító Start gomb eseménykezelőjének elkészítése.

```
/// <summary>
/// A Start gombon történő kattintásra reagáló eseménykezelő.
/// Kinullázza az eredményt tároló változót és a végrehajtásjelzőt
/// alapállapotba állítja. Az időzítőt a csúszka állapotához
/// igazítja és indítja. Engedélyezi a btKapjEl gombot, tiltja
/// a btStart gombot. Kezdeti feliratot jelenít meg az ablak
/// fejlécében.
/// </summary>     private void btStart_Click(object sender, RoutedEventArgs e)
{
   // Lenullázzuk az eredményt.
   Eredmény = 0;
   // Játékidő elölről kezdődik.
   KezdőIdő = DateTime.Now;
   // Végrehajtásjelző a kezdő pozícióba.
   pbVégrehajtásJelző.Value = 0;
   // Az időzítő beállítása a csúszka állása alapján.
   dtIdőzítő.Interval = new TimeSpan(0,0,0,0,(int)slCsúszka.Value);
   // Start gomb letiltása.
   btStart.IsEnabled = false;
   // Időzítő indítása.
   dtIdőzítő.IsEnabled=true;
   // Eredmény megjelenítése az ablak fejlécében.
   FeliratKiír();
   // Kapj el nyomógomb engedélyezése.
   btKapjEl.IsEnabled = true;
}
```

9. Eseménykezelő készítése a csúszka mozgatásához

```
/// <summary>
/// Eseménykezelő: a felhasználó elmozdította a csúszkát.
/// Leállítjuk az időzítőt. A csúszka értékének megfelelően
/// beállítjuk az időzítés idejét. Ha mindez játékidőben történt,
/// akkor engedélyezzük az időzítőt. Frissítjük a feliratot az
/// ablak fejlécében.
/// </summary>
private void slCsúszka_ValueChanged(object sender,
   RoutedPropertyChangedEventArgs<double> e)
{
```

```
        // Tároljuk, hogy most folyik-e játék.
        bool VanJáték = dtIdőzítő.IsEnabled;
        // Időzítő letiltása az intervallum módosítás miatt.
        dtIdőzítő.IsEnabled = false;
        // Időzítő intervallumának beállítása.
        dtIdőzítő.Interval = new TimeSpan(0,0,0,0,(int)slCsúszka.Value);
        // A játék közben mozgatja a felhasználó a csúszkát, akkor
        if (VanJáték)
           dtIdőzítő.IsEnabled = true;
        // Eredmény megjelenítése az ablak fejlécében.
        FeliratKiír();
     }
```

Feladat

Tegyük lehetővé a felhasználó számára, hogy állítsa be a játékidő nagyságát. A játék közben ezen ne lehessen változtatni. Tehát a beállítás csak két játék közötti időben legyen lehetséges.