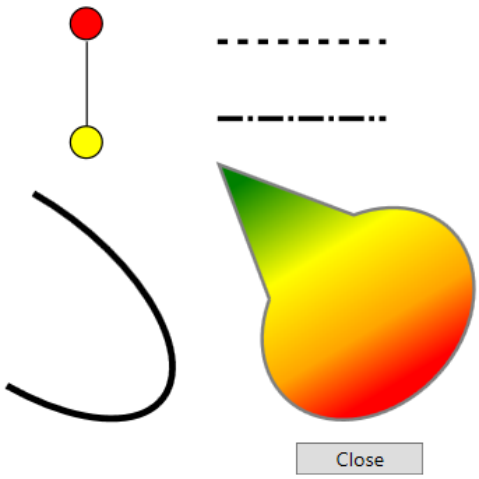
Napredna 2D grafika

Skripta objašnjenja primera za gradivo 5. nedelje:

- **Grafičke primitive**, podsećanje na neke osnovne koncepte sa prvog termina vežbi, gde se dodaju neki novi koncepti,
- **2D Transformacije**, i
- **2D Animacije**

U Visual Studio projektu Primer_2.1, prvo će biti objašnjen sadržaj prozora **Primitive1.xaml** (prikazan na **slici 1**).



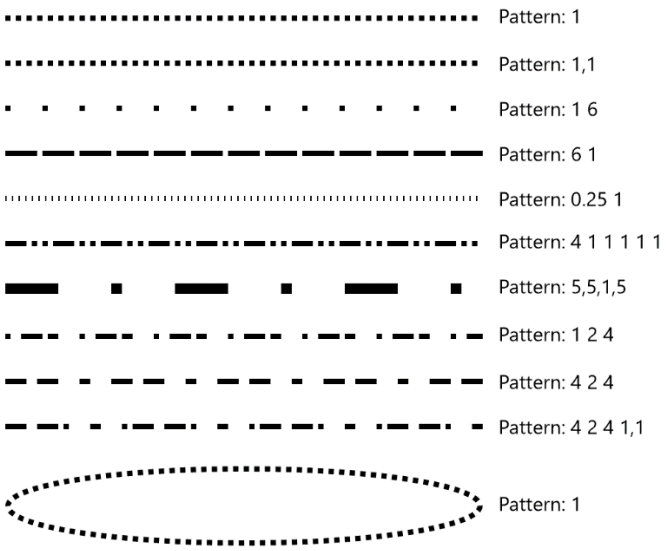
Slika 1. Sadržaj prozora Primitive1.xaml

Prvi crtež čine dve elipse postavljene jedna ispod druge i između kojih je nacrtana uspravna linija. Kroz ovaj crtež se podsećamo osnovnih property-ja kada se iscrtavaju objekti koji nasleđuju klasu ***Shape***. U slučaju da je potrebno da objekat koji se iscrtava treba da započne iscrtavanje iz određene tačke, zadaje se položaj te tačke preko property-ja *Canvas.Left* (pomeraj u odnosu na levu ivicu prozora, brojna vrednost) i *Canvas.Top* (pomeraj u odnosu na gornju ivicu prozora, brojna vrednost). Property-ji koji bliže određuju izgled objekta klase ***Ellipse***, su dimenzije (*Width*, *Height* - ponašaju se kao prečnici, brojne vrednosti), boja konturne linije objekta (*Stroke,* vrednost iz klase Brushes), debljina konturne linije (*StrokeThickness,* brojna vrednost ili objekat klase Thickness ako se zadaje iz .cs koda) i boja kojom će biti obojena površina date elipse (*Fill,* vrednost iz klase Brushes).

Linija se iscrtava kao objekat klase ***Line***, kome se, pored tačke iz koje počinje iscrtavanje, dodeljuju koordinate dve tačke koje određuju datu liniju (X1, Y1 - property-ji koji određuju početnu tačku, X2, Y2 - property-ji koji određuju krajnju tačku). Kao i kod elipse mogu se zadati debljina linije i njena boja.

Naredni crtež su dve paralelne linije u gornjem desnom delu prozora, gde se dodaje samo property *StrokeDashArray* koji predstavlja niz vrednosti (ili samo jednu) koje određuju na koji način će linija biti izdeljena, prva vrednost određuje koji deo linije će biti iscrtan, a naredna vrednost koji deo linije će ostati neiscrtan, pa se tako zadana sekvenca primenjuje na celu liniju. Veličina ovih (ne)iscrtanih delova linije se na osnovu datih parametara određuje u odnosu na zadatu debljinu linije. Tako se prva linija iscrtava sa uniformnom raspodelom (u kodu 2,2), dok će se prikazivati kao linija „crta tačka crta“, definisana da se crta kao linija, potom jedinični razmak, pa jedinični prikaz linije, pa ponovo jedinični

razmak (u kodu 5,1,1,1). Neki primeri kako se može zadatavati ovaj property su dati na **slici 2**. Na ovaj način je takođe moguće zadati i kako će se crtati konturna linija oko nekog oblika.



Slika 2. Neke varijante zadavanja property-ja *StrokeDashArray*

U donjoj polovini prozora su prikazana dva načina da se definiše objekat klase ***Path***. Prvoj je pored debljine i boje linije definisan je property *Data,* koji određuje način na koji će se ova putanja/linija iscrtati. „M“ predstavlja početnu tačku i za njom se navode koordinate ove tačke (u zadatku). Slovo „M“ je izabrano tako da predstavlja komandu pomeranja (move) odnosno dovođenja početka crteža u datu tačku (u kodu 80,220). U nastavku sledi slovo „A“ koje predstavlja krivu liniju baziranu na elipsi. U skladu sa time, prvi par brojeva koji sledi određuje poluprečike (u kodu 100,50) na osnovu kojih se formira data kriva linija i odgovara *Size* property-ju klase ***ArcSegment.***

Sledeći broj u sekvenci odgovara uglu gde kriva linije dostiže maksimum (u kodu 45, odgovara property-ju *RotationAngle*) . Naredna dva broja određuju da li se crta kao kriva linija koja zahvata ugao veći od 180 stepeni (u kodu 1, property *IsLargeArc*) ili manji, dok drugi broj govori o smeru iscrtavanja linije, da li je u smeru kretanja kazaljke na satu ili suprotno (u kodu 0, property *SweepDirection*). Poslednji par brojeva određuje koordinate tačke u kojoj treba da se završi iscrtavanje ove krive linije (u kodu 100,50, property *EndPoint*).

U nastavku se može primetiti da su ovoj putanji zadani visina i širina, čime se ograničava prostor u kojem bi crtež trebao biti nacrtan, međutim, tek postavljanjem property-ja *Stretch* na vrednost „Fill“ će crtež biti „uglavljen“ u zadate dimenzije. Uklanjanjem ovog propertyja će se videti pravilna veličina ove linije/putanje.

Drugi primer zadavanja objekta klase ***Path*** je dat na primeru iscrtavanja gradijentno obojenog oblačića, gde je način crtanja putanje definisan kroz geometriju (objekat ***PathGeometry***) gde se koriste segmenti (***LineSegment***, ***ArcSegment***) za crtanje kompletne figure (***PathFigure***, definiše se sa početnom tačkom iscrtavanja). Pri formiranju figure segmenti se navode sekvencijalno kako se nastavljaju jedan na drugi, gde se liniji zadaje krajnja tačka do koje će prostirati, a način definicije krive linije (***ArcSegment***) je objašnjen u prethodnom delu.

Prozor **Primitive2.xaml** prikazuje kako se putanji/liniji zadaje grupa geometrijskih oblika, kojoj je među property-jima objekta klase ***Path*** definisana boja kojom će biti obojena cela površina definisana geometrijom. Geometrija je predstavljena kao grupa nekoliko geometrijskih oblika (klasa ***GeometryGroup***). Kada se definiše grupa oblika, inicijalno podešavanje je da će svaka zajednička

površina, odnosno presek ovih oblika biti providna odnosno neobojena. Međutim, zadavanjem *FillRule* property-ja na vrednost „NonZero“ ovo će se izbeći i sve će biti popunjeno. Alternativna i inicijalna vrednost je uvek „EvenOdd“, koja ne smatra presek za površinu ove putanje.

Sami geometrijski oblici su zadati kao objekti klasa:

- ***LineGeometry*** – definišu se početna i krajnja tačka (*StartPoint* i *EndPoint*), odnosno njihove koordinate,
- ***EllipseGeometry*** – definiše se pomoću centra (*Center*, tačka oko koje se crta elipsa) i poluprečnika po obe ose (*RadiusX*, *RadiusY*, brojevne vrednosti), i
- ***RectangleGeometry*** – definiše se pomoću property-ja *Rect* koji kao vrednost dobija gornji levi ugao odakle se iscrtava, širinu i visinu.

Još jedan tip geometrijskog oblika koji se može iscrtavati je poligon, opisan klasom ***Polygon***. Da bi se on iscrtao, dodeljuju mu se koordinate tačaka koje određuju konturu njegove površine.

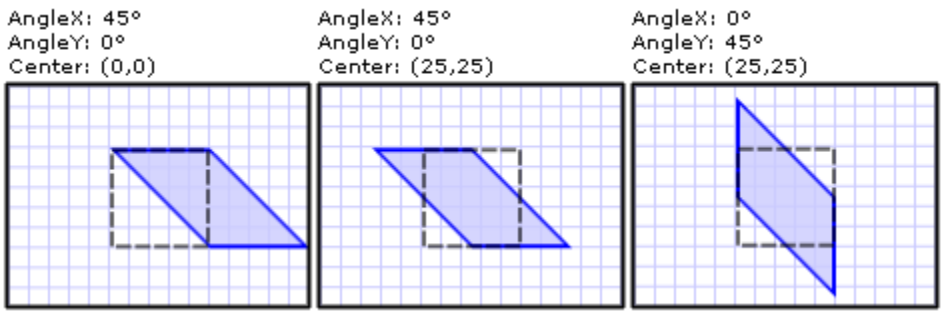
Transformacije se koriste da se objekti, potomci klase ***FrameworkElement*** mogu rotirati, skalirati, zakrivljivati ili translirati (pomerati).

Transformacija rotacije, koja se modeluje pomoću klase ***RotateTransform***, zahteva definisanje samo jednog property-ja, a to je ugao pod kojim će se objekat rotirati (*RotationAngle*).

Transformacija skaliranja, koja se modeluje pomoću klase ***ScaleTransform***, očekuje povećanja po X i Y osi i vrednosti koje se tu prosleđuju odgovaraju vrednostima koliko puta treba uvećati/smanjiti objekat po toj osi (*ScaleX*, *ScaleY*; dakle ako je potrebno da se on uveća 2 puta, dodeliće se vrednost 2, a ako je potrebno da se on smanji za polovinu svoje veličine, dodeliće se vrednost 0.5)

Transformacija zakrivljivanja, koja se modeluje pomoću klase ***SkewTransform***, očekuje da joj se zadaju uglovi pod kojima će se zakrivljivati element po X i Y osi (*AngleX*, zakrivljenje po X osi, meri u smeru kontra od kretanja kazaljke na satu u odnosu na Y osu; *AngleY*, zakrivljenje po Y osi, meri u smeru kontra od kretanja kazaljke na satu u odnosu na X osu; ovo je prikazano na **slici 3.**)

Transformacija translacije (pomeranja), koja se modeluje pomoću klase ***TranslateTransform***, očekuje da joj se zadaju pomeraji po svakoj osi (property-jem *X* se zadaje pomeraj po X osi – pozitivna vrednost pomera u pozitivnom smeru X ose, a negativna vrednost pomera objekat u negativnom smeru iste ose; property-jem Y se zadaje pomeraj po Y osi – pozitivna vrednost pomera u pozitivnom smeru Y ose **(raste na dole)**, a negativna vrednost pomera objekat u negativnom smeru iste ose).



Slika 3. Primer zakrivljenja objekta po X i Y osi

Na prozoru **Transformacije.xaml** prikazano je kako se koriste ove transformacije, gde se na promene vrednosti slajdera reaguje događajima i zavisno od izabrane vrednosti se primenjuje data

transformacija (xaml.cs kod). Moguće je primeniti više transformacija nad istim objektom kada se sve transformacije stave u grupu koja se dodeljuje konkretnom UI elementu. Da bi se transformacije primenile, dodeljuje se property *RenderTransform* (transformacije utiču na poziciju na kojoj se renderuje element).

Da bi se transformacije primenile u okviru XAML koda, nakon instanciranja konkretnog oblika (na primeru prozora **TransformacijeXAML.xaml**), obliku se ponovo dodeljuje objekat koji određuje transformaciju pod property-jem *RenderTransform*. U listingu 1, prikazan je XAML kod kojim se definiše transformacije rotacije, zavisno od vrednosti slajdera.

```
       <Rectangle
            Canvas.Left="80" Canvas.Top="200"
            Width="50" Height="50"
            Fill="RoyalBlue" Opacity="1.0">
            <Rectangle.RenderTransform>
                <RotateTransform Angle="{Binding ElementName=SliderAngle,
Path=Value}"/>
            </Rectangle.RenderTransform>
        </Rectangle>
```

Listing 1. XAML definicija transformacije

Drugi definisani oblik u ovom prozoru je još jedna instanca klase ***Rectangle*** nad kojom se primenjuje grupa transformacija, koja se zadaje identično kao i u .cs kodu. Jedina novina je u definiciji transformacijerotacije, kojoj se ovde zadaju još dva property-ja, *CenterX* i *CenterY*, koji predstavljaju koordinate tačke oko koje će se vršiti rotacija.

U prozoru **Transformacije2.xaml**, prikazano je kako se trasnformacije mogu primeniti nad platnom i uz to nad svim oblicima nacrtanim unutar njega. To se postiže pomoću klase ***LayoutTransform***. Ona će menjati izgled/raspored (layout) datog ***FrameworkElement***-a kroz rotaciju, skaliranje ili zakrivljivanje, ali negira translaciju jer će zbog offset-a ostalih transformacija, ***LayoutTransform*** automatski da poređati elemente. Slučaj u kojem se ***LayoutTransform*** najviše koristi je implementacija zoom funkcionalnosti što je u ovom prozoru i prikazano.

Animacije predstavljaju iluziju koja se stvara brzom smenom različitih slika, gde je svaka naredna blago različita od prethodne. U WPF-u, animacije se primenjuju na konkretne property-je objekata i omogućavaju promenu vrednosti datih property-ja u određenim vremenskim trenucima. Da bi animacija počela da se izvršava, mora da postoji neki događaj koji će je pokrenuti.

U prozoru **Animations1.xaml**, izvršavanje animacije počinje kada se učita prozor (klasi ***EventTrigger*** se zadaje ime događaja koji se čeka) posle čega se instancira ***Storyboard*** objekat. On predstavlja kontejner za animaciju i sadrži vremensku liniju (Timeline) kako bi animacija mogla da se izvršava u definisanim vremenskim trenucima.

Storyboard-u se može zadati da li će se animacija ponavljati i koliko puta (property *RepeatBehavior*). Može se uneti vreme koliko će se animacija ponavljati, koje, ako je kraće od vremena izvršavanja animacije, će je prekinuti pre nego što se završi, a ako je duže, animacija će trajati (ponavljati se) koliko vremena joj je dato ovim property-jem. Ovom property-ju se može dati posebna vrednost “Forever”, čime se postiže da se animacija konstantno ponavlja.

Klasom ***DoubleAnimation*** se definiše animacija koja svojim izvršavanjem menja vrednost jednog property-ja. Koji je to property joj se mora definisati kroz atribut *Storyboard.TargetProperty*, gde se unosi ime property-ja objekta nad kojim se izvršava animacija (u ovom slučaju na Window-u). Promena vrednosti se zadaje zadavajući property-je *From* i *To*, kojima se definiše vrednost željenog property-ja

na početku i kraju animacije. Trajanje animacije se zadaje pomoću property-ja *Duration* kojim se u formatu "sati:minuti:sekunde" zadaje trajanje. Za kraj, *AutoReverse* property animacije određuje da li će se nakon završetka ona izvršiti unazad za isto trajanje.

Alternative DoubleAnimation klasi u WPF-u su ColorAnimation (animira boju ***SolidColorBrush***-a ili ***GradientStop***-a) i PointAnimation (animira centar elipse)

U prozoru **Animations2.xaml**, tri ljubičasta kvadratna crteža – objekti klase ***Border***, imaju za sebe definisanu transformaciju translacije (pomeranja) sa vrednostima nula (ne pomeraju se od zadate pozicije). Kod svih se levim klikom miša na njihovu površinu pokreće *DoubleAnimationUsingKeyFrames,* klasična animacija tokom koje se menja neka vrednost, koja traje 15 sekundi utiče na objekat transformacije i njegov property *X*, ali se promene vrednosti dešavaju u konkretnim vremenskim trenucima (KeyFrame) i u sva slučaju svakog od ova tri objekta će se izvršavati drugačije, zavisno od tipa KeyFrame objekata:

- ***LinearDoubleKeyFrame***, zadaje se na koju vrednost se prelazi (*Value*) i u kom trenutku (*KeyTime*), a vrednost se animira od prošlog KeyFrame-a do vrednosti date u tekućoj instanci koristeći linearnu interpolaciju,
- ***DiscreteDoubleKeyFrame***, zadaje se na koju vrednost se prelazi (*Value*) i u kom trenutku (*KeyTime*), a vrednost se animira od prošlog KeyFrame-a do vrednosti date u tekućoj instanci koristeći diskretnu interpolaciju, i
- ***SplineDoubleKeyFrame***, zadaje se na koju vrednost se prelazi (*Value*) i u kom trenutku (*KeyTime*), a vrednost se animira od prošlog KeyFrame-a do vrednosti date u tekućoj instanci koristeći splajn interpolaciju. Property *KeySpline* predstavlja objekat istoimene klase kojom se definiše napredak animacije. Da bi se *KeySpline* mogao razumeti, potrebno je poznavati ***Bezier***-ove krive. Kada se opisuju *KeySpline*-ovi, početna tačka krive je uvek 0, a krajnja je 1. Zadavanjem dveju kontrolnih tačaka (njihovih X i Y koordinata), određuje se interpolacija animacije u odnosu na deo vremena (kako će se menjati vrednost zavisno od vremena – sve vrednosti se definišu u rasponu od 0 do 1).

Zeleni kvadratni crtež u donjem levom delu prozora je animiran tako da očekuje promenu property-ja *IsMouseOver* i u tom konkretnom slučaju da vrednost bude "True". Kada se desi ta promena, pozvaće se sadržaj *EnterActions* property-ja klase ***Trigger***, gde se u okviru ***Storyboard***-a koristi objekat ***ThicknessAnimation*** gde se *BorderThickness* property ovog objekta menja na vrednost 3. Kada vrednost property-ja *IsMouseOver* postane "False", pozvaće se sadržaj property-ja *ExitActions.*

Poslednji kvadrat je animiran tako da se levim klikom na njegovu površinu pozove sledeća funkcionalnost:

```
private void Border_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
     DoubleAnimation da = new DoubleAnimation();
     da.From = 0;
     da.To = 360;
     da.Duration = new Duration(TimeSpan.FromSeconds(3));
     da.RepeatBehavior = RepeatBehavior.Forever;
     RotateTransform rt = new RotateTransform();
     border2.RenderTransform = rt;
     rt.BeginAnimation(RotateTransform.AngleProperty, da);
}
```

Listing 2. Definisanje animacije i transformacije u .cs kodu i potom pokretanje animacije

Poslednji prozor u projektu **Brushes.xaml** prikazuje različite načine za bojenje površine oblika:

- ***SolidColorBrush***, boji jednom bojom celu površinu,
- ***LinearGradientBrush***, boje se linearno prelamaju, definisanjem objekata ***GradientStop***, kojima se određuju mesta prelamanja,
- ***RadialGradientBrush***, boje se prelamaju radijalno (šireći se kružno), gde je *GradientOrigin* pozicija (u procentima) odakle kreće da se širi gradijent (mesta prelamanja boja se zadaju kao i kod linearnog gradijenta),
- ***ImageBrush***, sa datom slikom popunjava površinu oblika, i
- ***DrawingBrush***, boji površinu crtežom koji mogu činiti oblici tekst, slike ili drugi crteži, prikazano na **slici 4**.

Slika 4. Primer bojenja kvadrata objektom klase DrawingBrush

Efekat "šahovske table" je postignut pomoću definicije grupe geometrijskih oblika koju čine kvadrat 100x100, i dva kvadrata 50x50 koji se dodiruju dijagonalnim temenima obojeni gradijentno tako da je boja na početku crna a na kraju siva. Tako zadato, ovu površinu bi trebao da oboji samo jedna pojava ovih kvadrata. Zadavanjem property-ja *TileMode* na vrednost "Tile", površinu ovog pravougaonika bi trebalo da popuni više pojava ove boje, a *Viewport* definiše se koliko će prostora zauzimati svaka, što je četvrtina (0.25 predstavlja 25%).

Na drugom priemeru sa **slike 4**, efekat je postignut tako što se koristi ***RadialGradientBrush***, a grupa geometrijskih oblika koja će biti obojena na dati način (ostatak se ne renderuje i ostaje providno), te obojeni ostaju samo drugi i četvrti kvadrant, a preko svega se ispisuje reč "Chocolate" (definisano ***TextBlock***-om).