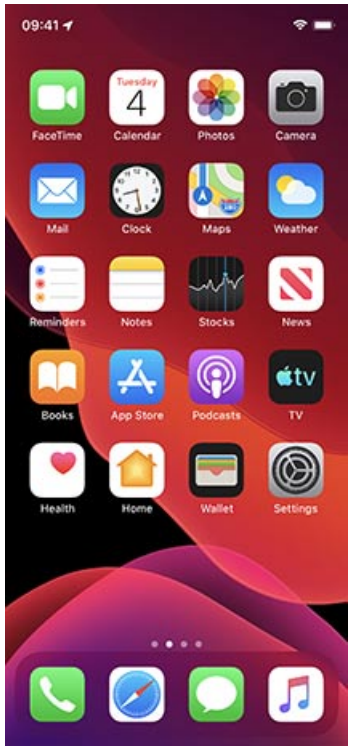
iOS SwiftUI -OHJELMOINTIHARJOITUKSIA

Kari Laitinen
http://www.naturalprogramming.com
2019-10-19 Tiedosto luotu.
2019-10-21 Viimeisin muutos.



HARJOITUKSIA OHJELMALLA SquareBallRectangleSUI

iOS SwiftUI -esimerkkiohjelmissa on sovellus nimeltä SquareBallRectangleSUI, joka näyttää neliötä, palloa, tai suorakaidetta sen mukaan mikä kuvio on ns. Segmented Controllin avulla valittuna.

Saat sovelluksen käyttöösi kun kopioit kurssin sivulta **SquareBallRectangleSUI.zip**-tiedoston ja purat sen paikallisesti. Purkaminen eli unzippaus tapahtuu tiedostonimen tuplaklikkauksella tai Safari voi tehdä sen myös automaattisesti.

Varmista ensin että saat ohjelman toimimaan ennenkuin alat tekemään seuraavia harjoituksia.

Huomioi että tästä sovelluksesta on olemassa myös UIKit-teknologialla tehty versio sekä myös vielä vanhempi Objective C -pohjainen versio. Ole siis tarkkana että otat käyttöön oikean sovelluksen. SwiftUI-teknologialla tehdyt versiot esimerkkiohjelmista tunnistaa niiden lopussa olevasta lyhenteestä SUI.

Harjoitus 1:

Muuta ohjelma sellaiseksi että se tekee suorakaiteen (rectangle) tilalle kolmion (triangle).

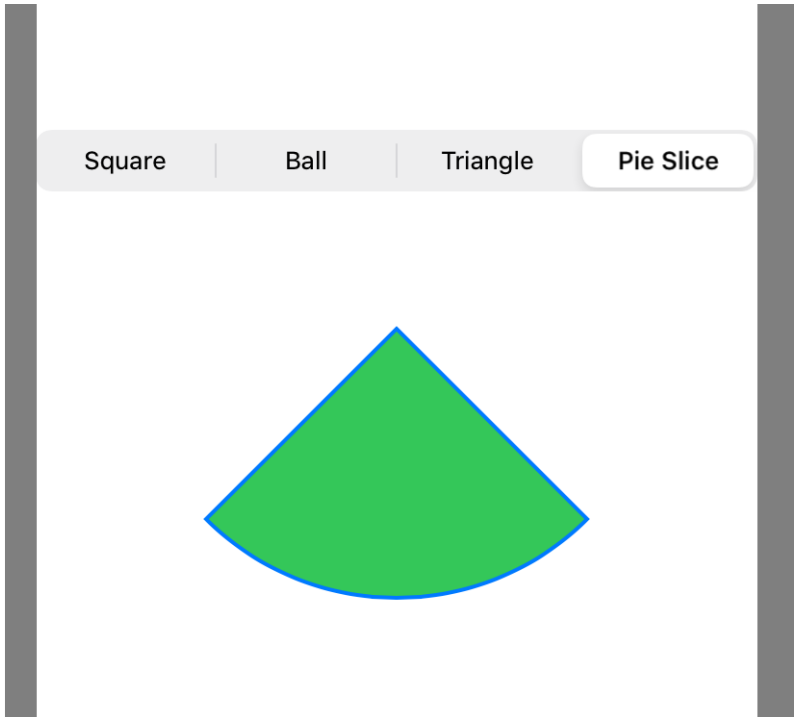
Tutkimalla sovellusta ShapesDemoSUI saat selville kuinka kolmio saadaan aikaiseksi. Voit kopioida tuosta sovelluksesta suoraan `struct`-määrittelyn nimeltä `Triangle`, joka sisältää valmiin piirtopolun joka toteuttaa kolmion. ShapesDemoSUI-sovelluksessa on myös kommenteissa kolmion tekemiseksi tarvittavat ohjelmarivit.

Tässä osatehtävässä sinun tulee määritellä käyttöliittymään sanan "Rectangle" tilalle "Triangle".

Harjoitus 2:

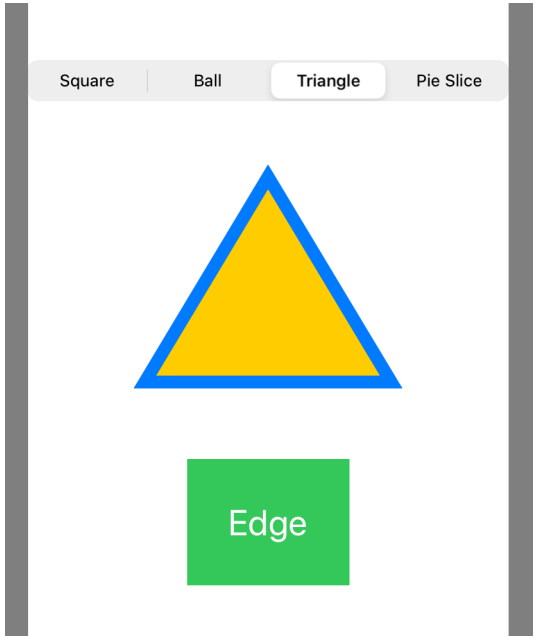
Muuta ohjelmaa siten että valittavaksi tulee uusi kuvio nimeltä "Pie Slice", eli eräänlainen piirakkapala. Tätä varten tulee käyttöliittymän Segmented Control muuttaa sellaiseksi että siinä on mahdollisuus neljään valintaan. Sovelluksessa ShapesDemoSUI on valmiina `struct` nimeltä `Pacman`, jonka pohjalta voit helpohkosti tehdä `struct`in nimeltä `PieSlice`.

Tässä on tarkoitus että kun valitaan "Pie Slice", ohjelma tuottaa suurinpiirtein seuraavan kuvan kaltaisen näkymän



Harjoitus 3:

Muuta ohjelmaa siten että siihen tulee painonappi (Button), jonka avulla voi säätää piirrettävien kuvioiden reunan paksuutta. Reunan paksuus voi aina nappia painettaessa kasvaa esim. kahdella (pikselillä) kunnes se saavuttaa arvon 10, minkä jälkeen se palaa arvoon 2. Käyttöliittymä voi tämän muutoksen jälkeen näyttää seuraavanlaiselta, eli siinä on tekstillä "Edge" varustettu painonappi:



Reunaviivan leveyden hallintaa varten voidaan tehdä `@State`-tyyppinen datajäsen, joka voi olla määritelty esim. seuraavasti

```
@State var edge_line_width : CGFloat = 2.0
```

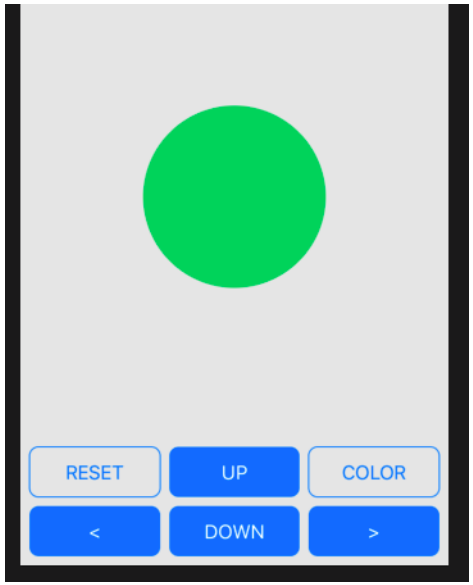

Tässä kannattaa tutustua esimerkkisovellukseen nimeltä ButtonDemoSUI.

SwiftUI-teknologiaa käytettäessä voidaan määritellä @State-muuttujia, jotka ovat sellaisia että muuttujan arvon muuttuessa systeemi automaattisesti suorittaa uudelleen ne ohjelman osat joissa ko. muuttujaa on käytetty.

HARJOITUKSIA OHJELMALLA MovingBallSUI

Harjoitus 1:

Nykyisellään ohjelmassa on neljä painonappia pallon liikutteluun ja sen lisäksi kaksi muuta nappia. Jos ajatellaan käytettävyyttä, niin olisi hyvä jos nuo pallon liikutteluun tarkoitetut napit jotenkin erottuisivat muista napeista. Muuta siis ohjelmaa siten että pallon liikuttelun napit ovat tummapohjaisia valkoisin tekstein seuraavaan tapaan:



Alkuperäisessä ohjelmassa painonappien tyyli on määritelty `struct`-määrittelyllä nimeltä `BallButtonStyle`. Voit kopioida tuon esim. nimelle `MovementButtonStyle`, ja tehdä siihen

tarvittavat muutokset, ja käyttää tuota uutta tyyliä pallon liikutteluun tarkoitetuille napeille. Taustavärin saa helposti tehtyä painonapeille kun käyttää `.fill()` -määrittelyä `.stroke()` -määrittelyn sijaan.

Harjoitus 2:

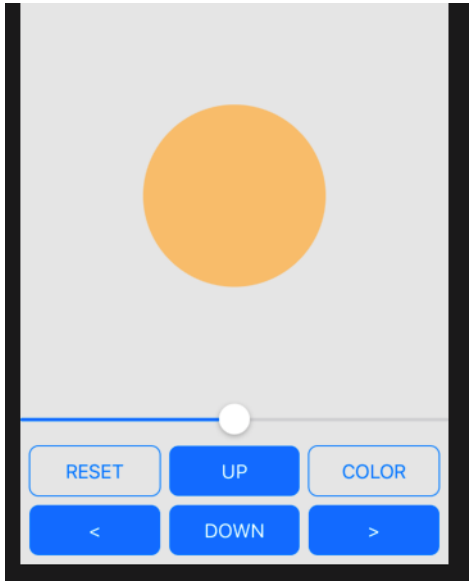
Nykyisellään ohjelmassa RESET-napin painallus aiheuttaa `Alert`in näyttämisen. Poista tuo käytöstä ja tee oikea toiminto RESET-napille. Poista vain ominaisuus että `Alert` näytetään. Keksitään tuolle `Alert`ille myöhemmin käyttöä.

RESET-operaatiossa pitää pallon siirtyä alkuperäiseen paikkaan ruudulla ja sen pitää muuttua sen väriseksi kuin se on ohjelman käynnistyessä.

Harjoitus 3:

Tutkimalla sovellusta nimeltä TurningArrowSUI saat selville kuinka `Slider`-oliota käytetään. `Slider` on käyttöliittymissä käytettävä olio jolla voidaan säätää jotain numeroarvoa annetuissa rajoissa.

Nyt tehtäväsi on lisätä `Slider` MovingBallSUI-sovellukseen siten että sen käyttöliittymä näyttää seuraavanlaiselta. Voit laittaa `Slider`in käyttöliittymään ensin ja vasta sitten laittaa se varsinaisesti toimimaan.



Tässä osatehtävässä on päämääränä että `Slider`illa pitää voida säätää pallon värin transparenttisuutta eli läpinäkyvyyttä. Värin läpinäkyvyys voidaan määritellä ns. alfa-arvolla. Läpinäkyvyydestä puhuttaessa käytetään myös opacity-käsitettä, mikä tarkoittaa peittävyyttä. alfa-arvo siis kuvaa myös värin peittävyyttä.

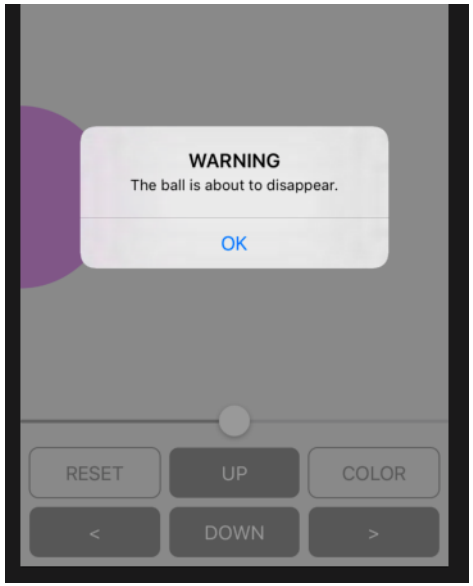
iOS-ohjelmoinnissa alfa-arvot ovat desimaalisia arvoja välillä 0 ... 1, mikä tarkoittaa liikkumista täysin läpinäkyvän ja täysin peittävän värin välillä. Sovelluksen pitää muuttaa pallon täyttövärin alfa-arvoa aina kun `Slider`in arvo muuttuu. Alfa-arvo voi olla alussa 0.5.

Tässä osatehtävässä kannattaa määritellä `@State`-muuttuja peittävyyttä kuvaamaan. Värin peittävyys voidaan asettaa `.opacity()`-metodilla väriä käyttöön otettaessa seuraavaan tapaan

```
Circle().fill( ball_filling_color.opacity( ball_color_opacity ) )
```

Harjoitus 4:

Jos olet tehnyt harjoitukset ohjeiden mukaan tähän saakka, on `Alert`-ominaisuus edelleen olemassa mutta sitä ei koskaan käynnistetä. Nyt tehtäväsi on näyttää `Alert` siinä tapauksessa että pallo on jo puoliksi ulkona liikkumisalueeltaan, kuten esim. seuraavassa kuvassa



Koska pallo voi liikkua koko ruudun alueella, voit verrata pallon offsettia koko näytön kokoon. Tutkimalla sovellusta ShapesDemoSUI saat selville erään tavan ottaa selville koko näytön koko.

Tässä siis pitää erikseen tutkia tilanteet jokaiseen suuntaan liikuttaessa. Kaikille ’seiniin

törmäyksille’ voidaan kuitenkin näyttää sama varoitusilmoitus. Kun opettaja teki tätä harjoitusta, hän kiinnitti seuraavan komennon alimman `VStack`-määrittelyn loppuun

```
.alert( isPresented: $showingAlert )
{
    Alert( title: Text( "WARNING" ),
           message: Text( "The ball is about to disappear." ),
           dismissButton: .default( Text( "OK" ) ) )
}
```

