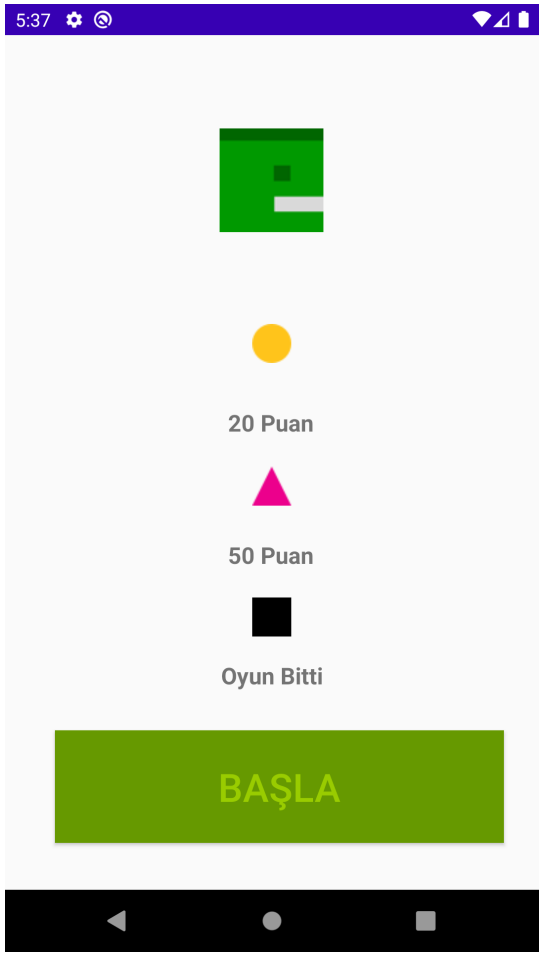
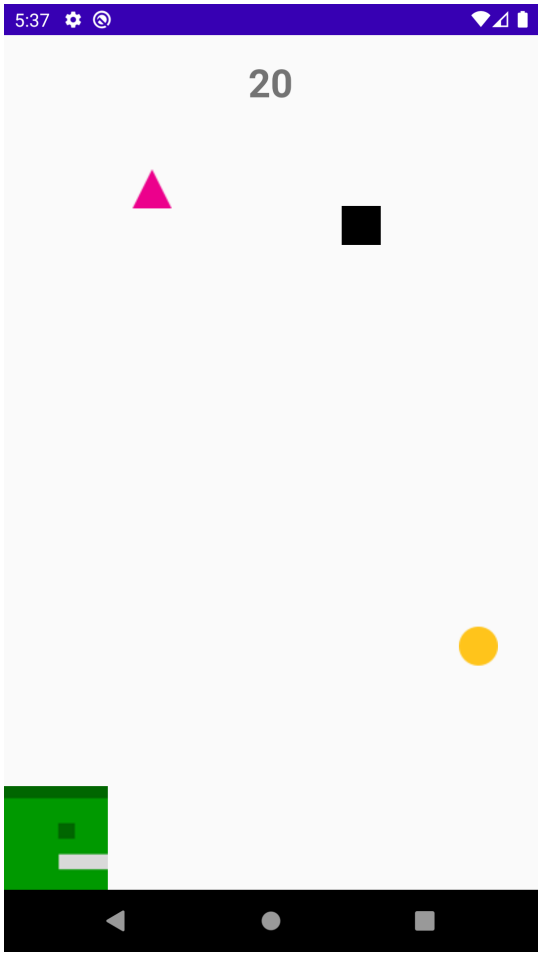
Android Mobil Uygulama Geliştirme Eğitimi | Kotlin

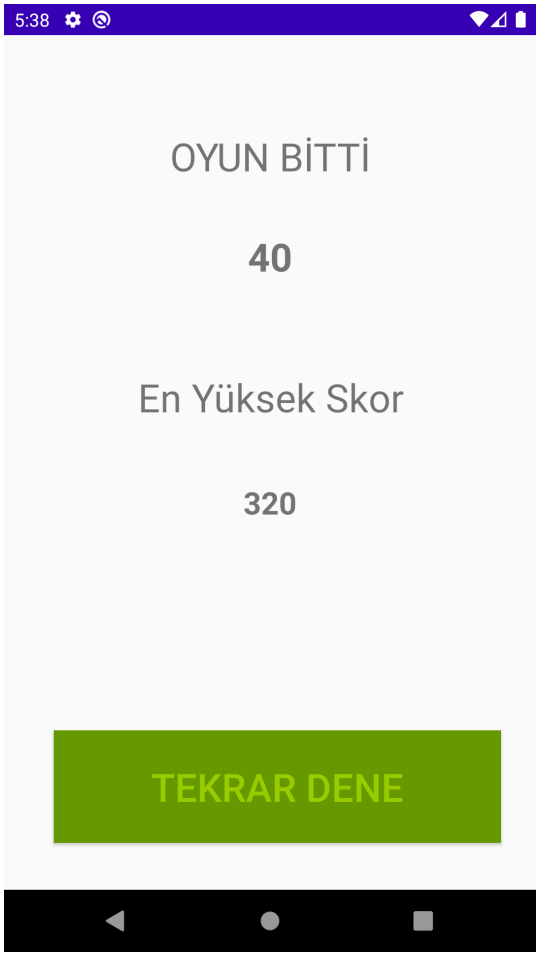
Android 2D Oyun Yapımı

Kasım ADALAN
Elektronik ve Haberleşme Mühendisi
Android - IOS Developer and Trainer

Tasarım


5:37
20 Puan
50 Puan
Oyun Bitti
BAŞLA


5:37
20


5:38
OYUN BİTTİ
40
En Yüksek Skor
320
TEKRAR DENE

Dokunma Özelliği

on Touch Event

- On touch event sayfa layoutuna bağlamak daha iyidir.
- Ekranın kendisine bağlarsak status bar yüksekliği nesnelerin yerini y ekseninde fazla gösterir.
- Motion event bağlı olduğu layoutun veya nesnenin koordinatlarını verir.
- getX()
- getY() metodları ile anlık dokunulan koordinatlar alınır.

```
cl.setOnTouchListener(object:View.OnTouchListener{
    override fun onTouch(v: View?, event: MotionEvent?): Boolean {

        if(event?.action == MotionEvent.ACTION_DOWN){
            Log.e( tag: "MotionEvent",  msg: "ACTION_DOWN : Ekrana dokundu")
            //Tek parmak ekrana dokunulduğunda
        }

        if(event?.action == MotionEvent.ACTION_UP){
            Log.e( tag: "MotionEvent",  msg: "ACTION_UP : Ekranı bıraktı")
            //Tek parmak ekrana dokunduktan sonra ellini ekrandan çekince
        }

        return true
    }
})
```

Örnek Kodlama

```
class OyunEkraniActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_oyun_ekrani)

        Log.e( tag: "Yükseklik1", (cl.height).toString())
        Log.e( tag: "Genişlik2", (cl.width).toString())

        cl.setOnTouchListener(object:View.OnTouchListener{
            override fun onTouch(v: View?, event: MotionEvent?): Boolean {

                Log.e( tag: "Yükseklik2", (cl.height).toString())
                Log.e( tag: "Genişlik2", (cl.width).toString())

                if(event?.action == MotionEvent.ACTION_DOWN){
                    Log.e( tag: "MotionEvent", msg: "ACTION_DOWN : Ekrana dokundu")
                    //Tek parmak ekrana dokunulduğunda
                }

                if(event?.action == MotionEvent.ACTION_UP){
                    Log.e( tag: "MotionEvent", msg: "ACTION_UP : Ekranı bıraktı")
                    //Tek parmak ekrana dokunduktan sonra ellini ekrandan çekince
                }

                return true
            }
        })
    }
}
```

Anakarakterin Yukarı Aşağı Serbest Hareketi

Timer ve Handler

Kullanımı

Timer - Handler

- Timer belli aralıklarla iş yapmak için kullanılır.
- Timer başta çalıştırılır ve oyuna hareket veren ve sürekli çalışan bir yapı olur.
- Timer bir kere çalıştığı anda biz durdurana kadar çalışır.
- Handlerın görevi android ara yüzündeki yapıları güncellemektir.

Timer - Handler

```
import kotlin.concurrent.schedule
```

Timer için kullanılan schedule metodu importu

```
private val timer = Timer()
```

```
//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
timer.schedule( delay: 0, period: 20){ this: TimerTask
    Handler(Looper.getMainLooper()).post {
        if(dokunmaKontrol){
            anakarakterY-=20.0f
        }else{
            anakarakterY+=20.0f
        }

        anakarakter.y = anakarakterY
    }
}
```

```kotlin
class OyunEkraniActivity : AppCompatActivity() {

    //Pozisyonlar
    private var anakarakterX = 0.0f
    private var anakarakterY = 0.0f

    //Kontroller
    private var dokunmaKontrol = true

    private val timer = Timer()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_oyun_ekrani)

        cl.setOnTouchListener(object:View.OnTouchListener{
            override fun onTouch(v: View?, event: MotionEvent?): Boolean {

                if(event?.action == MotionEvent.ACTION_DOWN){
                    Log.e( tag: "MotionEvent",  msg: "ACTION_DOWN : Ekrana dokundu")
                }
                if(event?.action == MotionEvent.ACTION_UP){
                    Log.e( tag: "MotionEvent",  msg: "ACTION_UP : Ekranı bıraktı")
                }

                //Anakarakterin ekrandaki başlama konumuna göre x ve y konumu alındı
                anakarakterX = anakarakter.x
                anakarakterY = anakarakter.y

                //0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
                timer.schedule( delay: 0, period: 20){ this: TimerTask
                    Handler(Looper.getMainLooper()).post {
                        if(dokunmaKontrol){
                            anakarakterY-=20.0f
                        }else{
                            anakarakterY+=20.0f
                        }

                        anakarakter.y = anakarakterY
                    }
                }

                return true
            }
        })
    }
}
```

dokunmaKontrol değişkeni true olursa yukarı , false olursa aşağı hareket eder.

Anakarakterin Yukarı Aşağı Dokunarak Kontrolü

```
cl.setOnTouchListener(object:View.OnTouchListener{
    override fun onTouch(v: View?, event: MotionEvent?): Boolean {

        if(event?.action == MotionEvent.ACTION_DOWN){
            Log.e( tag: "MotionEvent",  msg: "ACTION_DOWN : Ekrana dokundu")
            dokunmaKontrol = true
        }
        if(event?.action == MotionEvent.ACTION_UP){
            Log.e( tag: "MotionEvent",  msg: "ACTION_UP : Ekranı bıraktı")
            dokunmaKontrol = false
        }

        //Anakarakterin ekrandaki başlama konumuna göre x ve y konumu alındı
        anakarakterX = anakarakter.x
        anakarakterY = anakarakter.y

        timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
            Handler(Looper.getMainLooper()).post {

                if(dokunmaKontrol){
                    anakarakterY-=20.0f
                }else{
                    anakarakterY+=20.0f
                }

                anakarakter.y = anakarakterY
            }
        }

        return true
    }
})
```

dokunmaKontrol değişkenini dokunma eventleri ile true false yapabiliriz ve karakterin yönünü değiştirebiliriz.

Bu kodlama yapısında eksiğimiz her dokunma işlemi olduğunda timer yeniden oluşturulur ve karakter çok fazla hızlanır. Bunu çözebilmek için timerı bir kere çalıştırmamız gereklidir.

Çözüm :

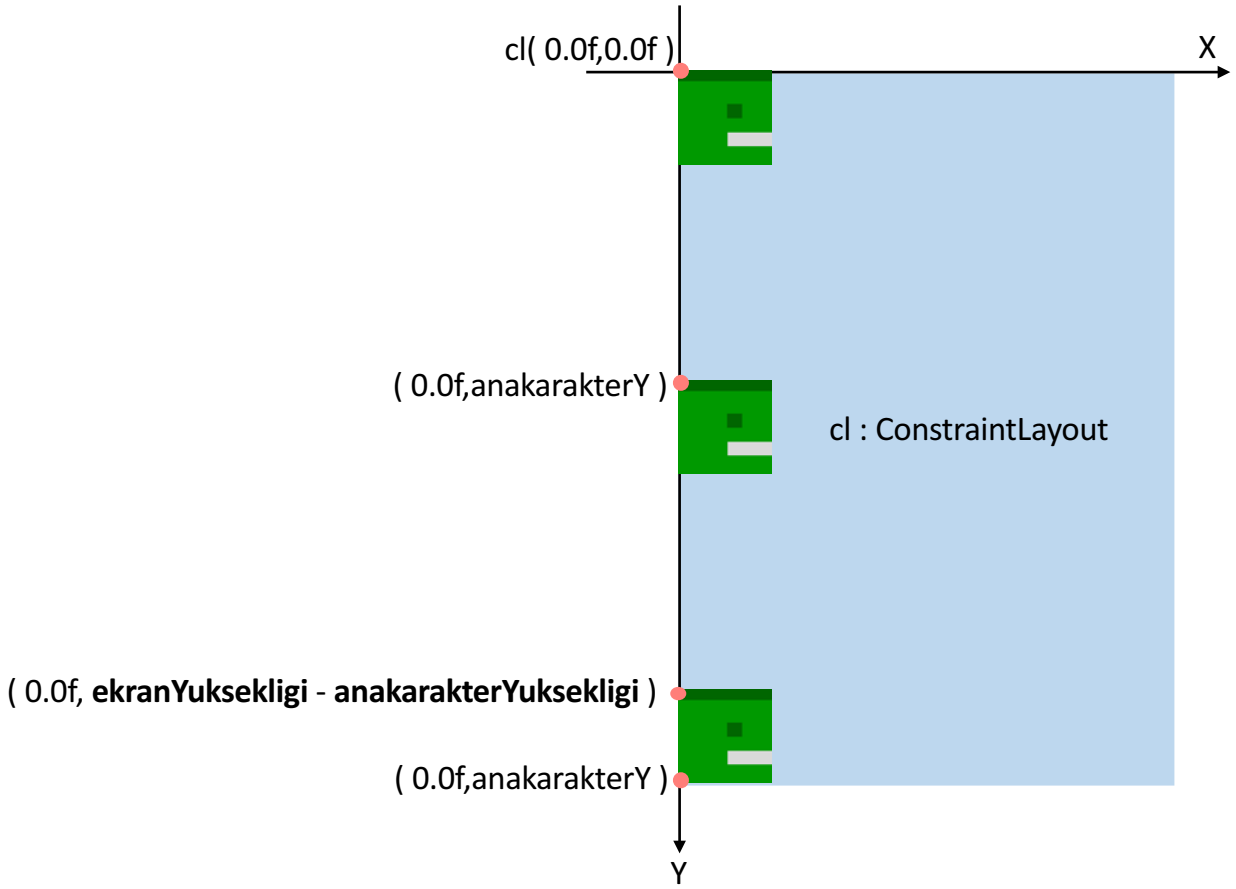
```
class OyunEkraniActivity : AppCompatActivity() {

    //Pozisyonlar
    private var anakarakterX = 0.0f
    private var anakarakterY = 0.0f

    //Kontroller
    private var dokunmaKontrol = false
    private var baslangicKontrol = false

    private val timer = Timer()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_oyun_ekrani)

        cl.setOnTouchListener(object:View.OnTouchListener{
            override fun onTouch(v: View?, event: MotionEvent?): Boolean {

                if(baslangicKontrol){
                    if(event?.action == MotionEvent.ACTION_DOWN){
                        Log.e( tag: "MotionEvent", msg: "ACTION_DOWN : Ekrana dokundu")
                        dokunmaKontrol = true
                    }
                    if(event?.action == MotionEvent.ACTION_UP){
                        Log.e( tag: "MotionEvent", msg: "ACTION_UP : Ekranı bıraktı")
                        dokunmaKontrol = false
                    }
                }else{
                    baslangicKontrol = true

                    //Anakarakterin ekrandaki başlama konumuna göre x ve y konumu alındı
                    anakarakterX = anakarakter.x
                    anakarakterY = anakarakter.y

                    timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
                        Handler(Looper.getMainLooper()).post {

                            if(dokunmaKontrol){
                                anakarakterY-=20.0f
                            }else{
                                anakarakterY+=20.0f
                            }
                            anakarakter.y = anakarakterY
                        }
                    }
                }
                return true
            }
        })
    }
}
```

ekrana daha sonraki dokunmalarımızda baslangicKontrol true olduğu için dokunmaKontrol değişkenini true veya false yapabilicez.Timer'da dokunmaKontrol değerine göre yukarı ve aşağı hareketi sağlayacak.

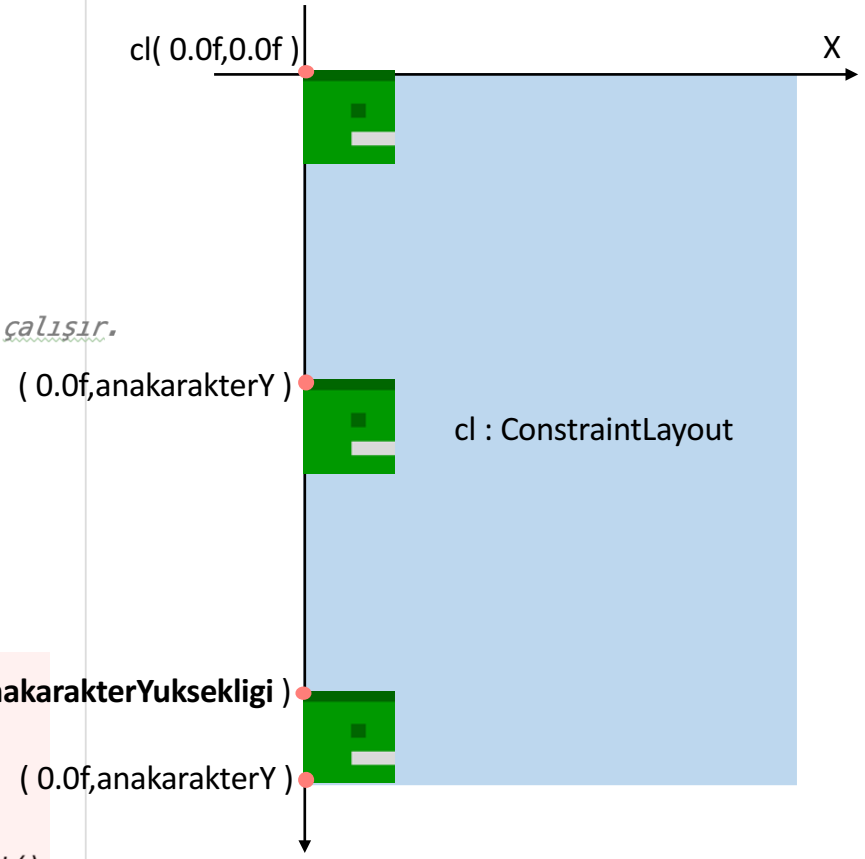
baslangicKontrol değişkeni ilk durumu false olduğu için ekrana ilk kez dokunduğumuzda true olacak ve timer çalışmaya başlıyacak. Bu sayede her dokunma hareketinde timer yeniden oluşturulmayacak ve sadece ilk dokunmada oluşturulacak.

Ana karakterin Ekran Dışına Çıkmasını Engelleme

Ekranın dışına taşmayı engelleme



```
//Anakarakterin ekrandaki başlama konumuna göre x ve y konumu alındı
anakarakterX = anakarakter.x
anakarakterY = anakarakter.y

anakarakterGenisligi = anakarakter.width
anakarakterYuksekligi = anakarakter.height
ekranGenisligi = cl.width
ekranYukseligi = cl.height

timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
    Handler(Looper.getMainLooper()).post {

        if(dokunmaKontrol){
            anakarakterY-=20.0f
        }else{
            anakarakterY+=20.0f
        }

        if(anakarakterY <= 0.0f){
            anakarakterY = 0.0f
        }

        if(anakarakterY >= ekranYukseligi - anakarakterYuksekligi){
            anakarakterY = (ekranYukseligi - anakarakterYuksekligi).toFloat()
        }

        anakarakter.y = anakarakterY
    }
}
```



```
class OyunEkraniActivity : AppCompatActivity() {

    //Pozisyonlar
    private var anakarakterX = 0.0f
    private var anakarakterY = 0.0f

    //Boyutlara
    private var ekranGenisligi = 0
    private var ekranYukseligi = 0
    private var anakarakterGenisligi = 0
    private var anakarakterYuksekligi = 0

    //Kontroller
    private var dokunmaKontrol = false
    private var baslangicKontrol = false

    private val timer = Timer()
```

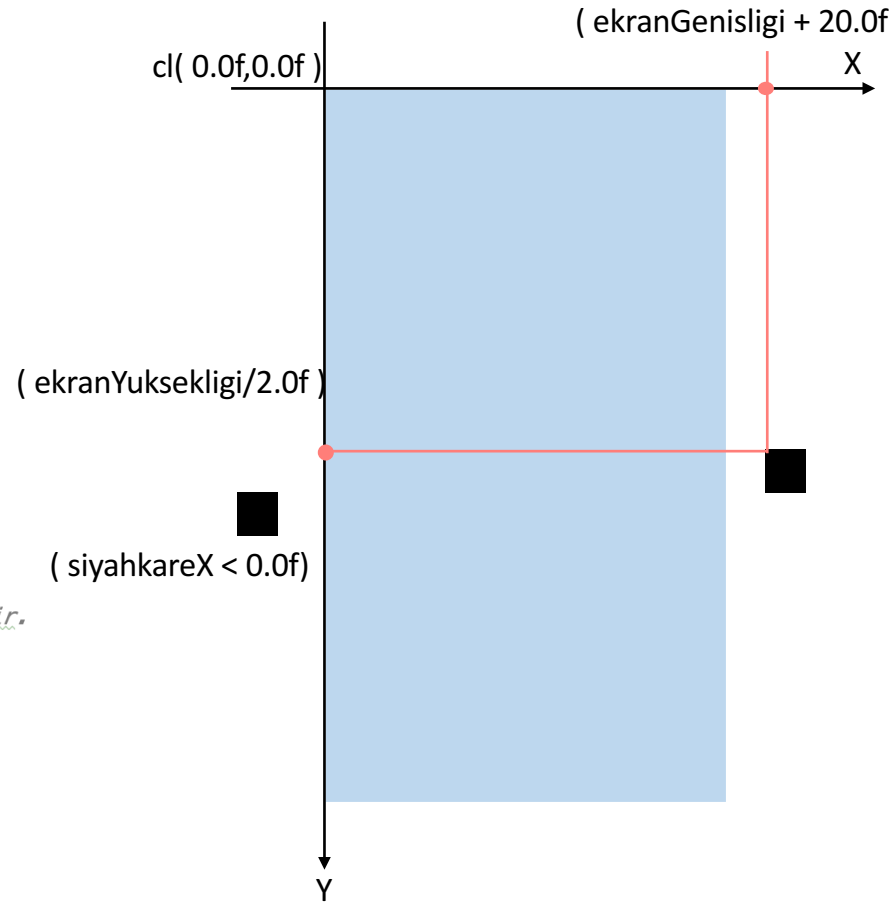
```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_oyun_ekrani)

    cl.setOnTouchListener(object:View.OnTouchListener{
        override fun onTouch(v: View?, event: MotionEvent?): Boolean {

            if(baslangicKontrol){
                if(event?.action == MotionEvent.ACTION_DOWN){
                    Log.e( tag: "MotionEvent", msg: "ACTION_DOWN : Ekrana dokundu")
                    dokunmaKontrol = true
                }
                if(event?.action == MotionEvent.ACTION_UP){
                    Log.e( tag: "MotionEvent", msg: "ACTION_UP : Ekranı bıraktı")
                    dokunmaKontrol = false
                }
            }else{
                baslangicKontrol = true

                //Anakarakterin ekrandaki başl
                anakarakterX = anakarakter.x
                anakarakterY = anakarakter.y

                anakarakterGenisligi = anakarakter.width
                anakarakterYuksekligi = anakarakter.height
                ekranGenisligi = cl.width
                ekranYukseligi = cl.height

                timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
                    Handler(Looper.getMainLooper()).post {

                        if(dokunmaKontrol){
                            anakarakterY-=20.0f
                        }else{
                            anakarakterY+=20.0f
                        }

                        if(anakarakterY <= 0.0f){
                            anakarakterY = 0.0f
                        }

                        if(anakarakterY >= ekranYukseligi - anakarakterYuksekligi){
                            anakarakterY = (ekranYukseligi - anakarakterYuksekligi).toFloat()
                        }

                        anakarakter.y = anakarakterY
                    }
                }
            }
            return true
        }
    })
}
```

Kodlamayı modüler hale getirme

```
timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
    Handler(Looper.getMainLooper()).post {
        anakarakterHareketEttirme()
    }
}

fun anakarakterHareketEttirme(){
    if(dokunmaKontrol){
        anakarakterY-=20.0f
    }else{
        anakarakterY+=20.0f
    }

    if(anakarakterY <= 0.0f){
        anakarakterY = 0.0f
    }

    if(anakarakterY >= ekranYukseligi - anakarakterYuksekligi){
        anakarakterY = (ekranYukseligi - anakarakterYuksekligi).toFloat()
    }

    anakarakter.y = anakarakterY
}
```

Cisimlerin Serbest Hareketi

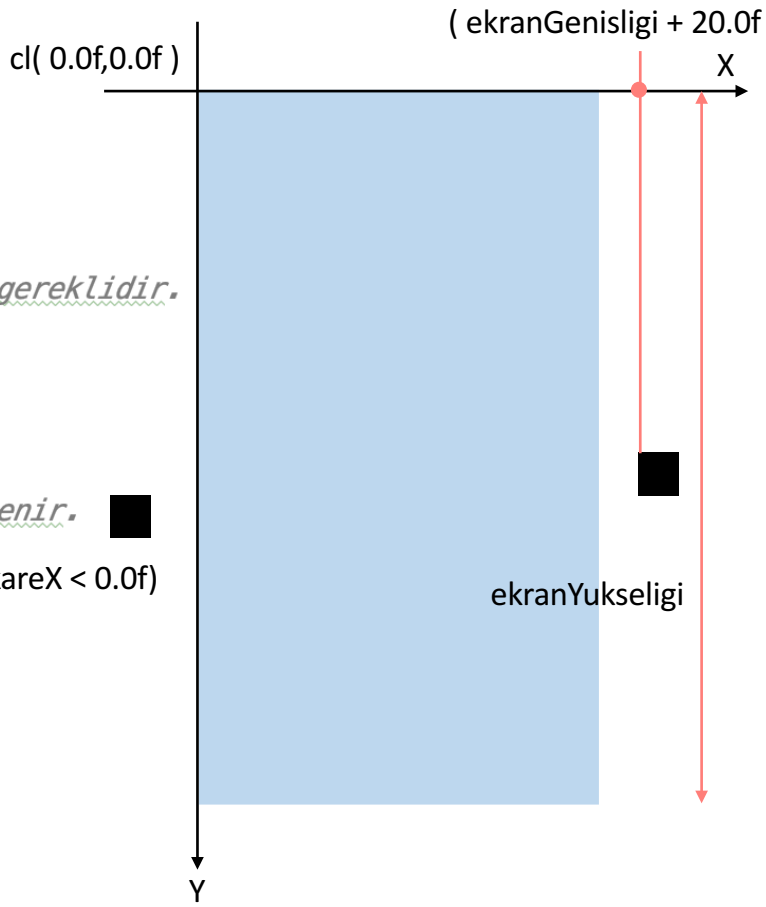
```
class OyunEkraniActivity : AppCompatActivity() {

    //Pozisyonlar
    private var anakarakterX = 0.0f
    private var anakarakterY = 0.0f
    private var siyahkareX = 0.0f
    private var siyahkareY = 0.0f
```




```
fun cisimlerinHareketEttir(){
    siyahkareX-=30.0f //Hızı bu ifade belirler
    // - ise sağdan-sola ,+ ise sağdan-sola hareket sağlar.

    //Eğer cisim en solda ise yani ekranın sonunda ise baştan başlaması gereklidir.
    if (siyahkareX < 0.0f ){
        siyahkareX = ekranGenisligi + 20.0f
        //Cismi ekranın en sağına yani ekranın dışında yönlendirir.
    }

    //Cismin anlık x ve y konumu belirlenir.
    siyahkare.x = siyahkareX
    siyahkare.y = ekranYukseligi/2.0f//Y ekseni dikeydeki konumunu belirler.
}
```

Cisimlerin Rasgele Hareketi

```
fun cisimlerinHareketEttir(){
    siyahkareX-=30.0f //Hızı bu ifade belirler
    // - ise sağdan-sola ,+ ise sağdan-sola hareket sağlar.

    //Eğer cisim en solda ise yani ekranın sonunda ise baştan başlaması gereklidir.
    if (siyahkareX < 0.0f ){
        siyahkareX = ekranGenisligi + 20.0f
        //Cismi ekranın en sağına yani ekranın dışında yönlendirir.
        siyahkareY = floor( x: Math.random() * ekranYukseligi).toFloat()
        //Ekran yüksekligi sınır değerlerinde rasgele dikey konum belirlenir.
    }

    //Cismin anlık x ve y konumu belirlenir.
    siyahkare.x = siyahkareX
    siyahkare.y = siyahkareY
}
```



```
class OyunEkraniActivity : AppCompatActivity() {

    //Pozisyonlar
    private var anakarakterX = 0.0f
    private var anakarakterY = 0.0f
    private var siyahkareX = 0.0f
    private var siyahkareY = 0.0f
    private var saridaireX = 0.0f
    private var saridaireY = 0.0f
    private var kirmiziucgenX = 0.0f
    private var kirmiziucgenY = 0.0f



    timer.schedule( delay: 0, period: 20){//0ms ge
        Handler(Looper.getMainLooper()).post {
            anakarakterHareketEttirme()
            cisimlerinHareketEttir()
        }
    }
```

```
fun cisimlerinHareketEttir(){
    siyahkareX-=25.0f
    saridaireX-=20.0f
    kirmiziucgenX-=30.0f

    if (siyahkareX < 0.0f ){
        siyahkareX = ekranGenisligi + 20.0f
        siyahkareY = floor( x: Math.random() * ekranYukseligi).toFloat()
    }
    siyahkare.x = siyahkareX
    siyahkare.y = siyahkareY

    if (saridaireX < 0.0f ){
        saridaireX = ekranGenisligi + 20.0f
        saridaireY = floor( x: Math.random() * ekranYukseligi).toFloat()
    }
    saridaire.x = saridaireX
    saridaire.y = saridaireY

    if (kirmiziucgenX < 0.0f ){
        kirmiziucgenX = ekranGenisligi + 20.0f
        kirmiziucgenY = floor( x: Math.random() * ekranYukseligi).toFloat()
    }
    kirmiziucgen.x = kirmiziucgenX
    kirmiziucgen.y = kirmiziucgenY
}
```

Ekran Arayüzünün Optimize Edilmesi

Arayüzdeki Görsellerin Konum ve Görünümünü Ayarlama

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_oyun_ekrani)

    //Cisimleri ekranın dışına çıkarma
    siyahkare.x = -80.0f
    siyahkare.y = -80.0f
    saridaire.x = -80.0f
    saridaire.y = -80.0f
    kirmiziucgen.x = -80.0f
    kirmiziucgen.y = -80.0f
```

Oyun başladığında ekrandaki görseller ekran dışında olsun

Oyun uyarısını oyuna başlamak için ekrana dokunulduğunda kaldırılması

```
}else{
    baslangicKontrol = true

    //Kullanıcı başlama uyarısını görünmez yapar.
    textViewOyunaBasla.visibility = View.INVISIBLE

    //Anakarakterin ekrandaki başlama konumuna göre x ve y konumu alındı
    anakarakterX = anakarakter.x
    anakarakterY = anakarakter.y

    anakarakterGenisligi = anakarakter.width
    anakarakterYuksekligi = anakarakter.height
    ekranGenisligi = cl.width
    ekranYukseligi = cl.height

    timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
        Handler(Looper.getMainLooper()).post {
            anakarakterHareketEttirme()
            cisimlerinHareketEttir()
        }
    }
}
```

Nesne hızlarının her ekrana göre uyumlu yapılması .

- Optimize for multiple devices
- Nesnelerin hızı pixel artışı ile ilgilidir.
- Örneğin : Yatay giden bir nesnenin pixel artışını 30 diye statik yaparsak.Yatay uzunluğu farklı olan cihazlarda hız değişik olacaktır.

```
fun cisimlerinHareketEttir(){
    siyahkareX-=30.0f // bu ifade belirler
    // - ise sağdan-sola ,+ ise sağdan-sola hareket sağlar.

    //Eğer cisim en solda ise yani ekranın sonunda ise baştan başlaması gereklidir.
    if (siyahkareX < 0.0f ){
        siyahkareX = ekranGenisligi + 20.0f
        //Cismi ekranın en sağına yani ekranın dışında yönlendirir.
        siyahkareY = floor( x: Math.random() * ekranYukseligi).toFloat()
        //Ekran yüksekligi sınır değerlerinde rasgele dikey konum belirlenir.
    }

    //Cismin anlık x ve y konumu belirlenir.
    siyahkare.x = siyahkareX
    siyahkare.y = siyahkareY
}
```

Hızların optimize edilmesi

- Hızların dinamik olarak optimize edilmesini isteniyorsa.
- Ekran genişliğine veya uzunluğuna göre oranlama yapmalıyız.

```
val anakarakterHiz = ekranYukseligi/60.0f//1280 / 60.0f  = 20.0f

    siyahkareX-= ekranGenisligi/44.0f//1080 / 44.0f  = 25.0f
    saridaireX-= ekranGenisligi/54.0f//1080 / 54.0f  = 20.0f
    kirmiziucgenX-= ekranGenisligi/36.0f//1080 / 36.0f  = 30.0f
```

Yatayda hareket edenler için ekranGenisligi , dikeyde hareket edenler için ekranYuksekligi baz alınır.

Final Kod

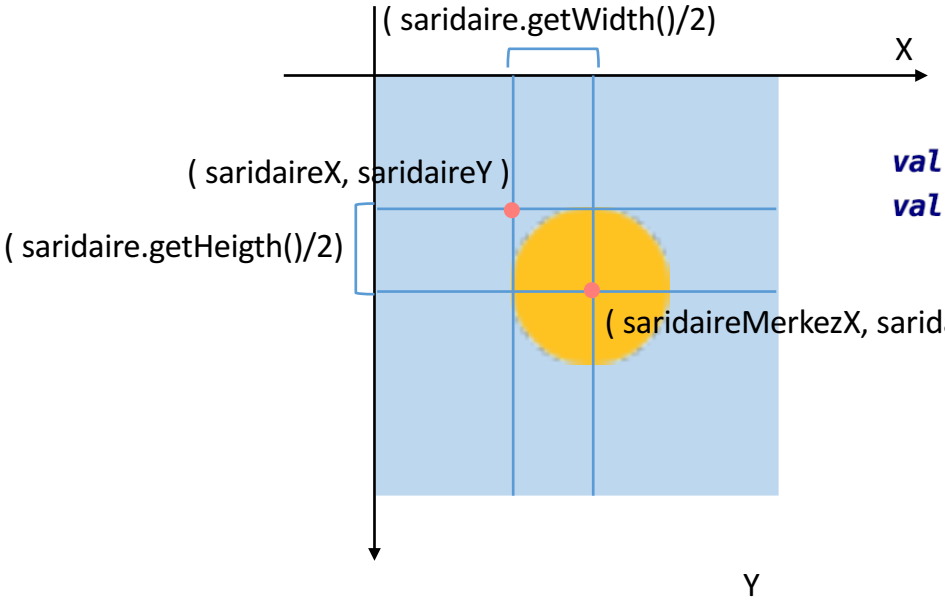
```
fun anakarakterHareketEttirme(){

    val anakarakterHiz = ekranYukseligi/60.0f//1280 / 60.0f  = 20.0f

    if(dokunmaKontrol){
        anakarakterY-=anakarakterHiz
    }else{
        anakarakterY+=anakarakterHiz
    }

    if(anakarakterY <= 0.0f){
        anakarakterY = 0.0f
    }

    if(anakarakterY >= ekranYukseligi - anakarakterYukseklig
        anakarakterY = (ekranYukseligi - anakarakterYukseklig
    }

    anakarakter.y = anakarakterY
}
```

```
fun cisimlerinHareketEttir(){

    siyahkareX-= ekranGenisligi/44.0f//1080 / 44.0f  = 25.0f
    saridaireX-= ekranGenisligi/54.0f//1080 / 54.0f  = 20.0f
    kirmiziucgenX-= ekranGenisligi/36.0f//1080 / 36.0f  = 30.0f

    if (siyahkareX < 0.0f ){
        siyahkareX = ekranGenisligi + 20.0f
        siyahkareY = floor( x: Math.random() * ekranYukseligi).toFloat()
    }
    siyahkare.x = siyahkareX
    siyahkare.y = siyahkareY

    if (saridaireX < 0.0f ){
        saridaireX = ekranGenisligi + 20.0f
        saridaireY = floor( x: Math.random() * ekranYukseligi).toFloat()
    }
    saridaire.x = saridaireX
    saridaire.y = saridaireY

    if (kirmiziucgenX < 0.0f ){
        kirmiziucgenX = ekranGenisligi + 20.0f
        kirmiziucgenY = floor( x: Math.random() * ekranYukseligi).toFloat()
    }
    kirmiziucgen.x = kirmiziucgenX
    kirmiziucgen.y = kirmiziucgenY
}
```

Çarpışma Kontrol

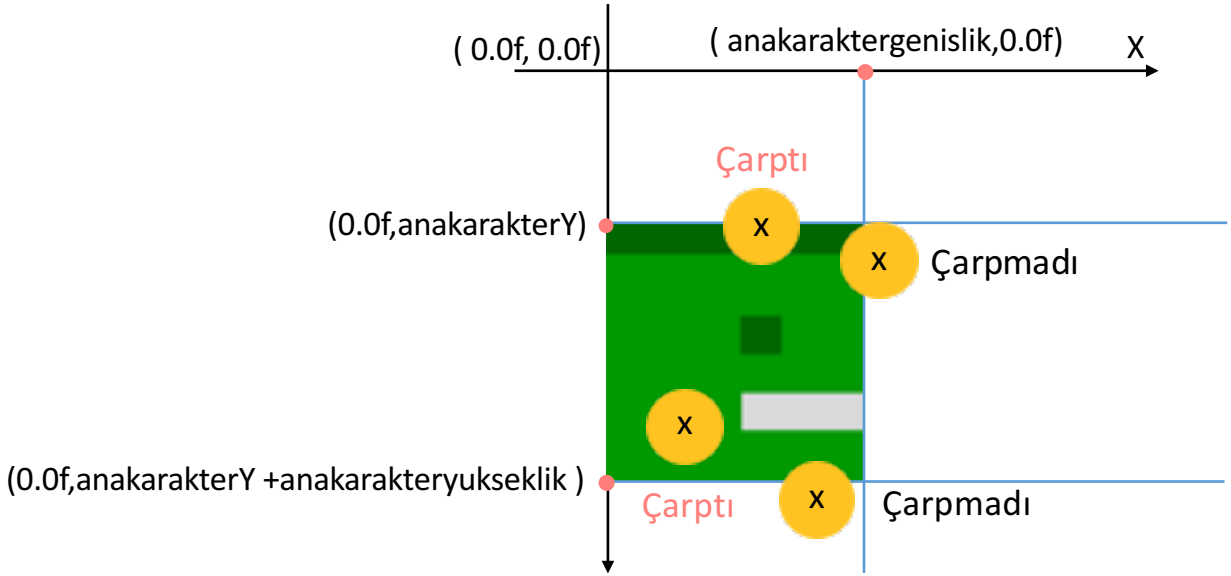
Çarpışma Kontrolü

- Çarpışma gerçekleşecek alanlar belirlenmelidir.
- Örneğin merkez noktası bulunmalıysa ;




```
val saridaireMerkezX = saridaireX + saridaire.width/2.0f
val saridaireMerkezY = saridaireY + saridaire.height/2.0f
```

Çarpışma Kontrolü



Gelen topun merkez noktası kutunun anlık olarak X ve Y koordinatları arasında olmalı ki çarpışma olsun.

```
val saridaireMerkezX = saridaireX + saridaire.width/2.0f
val saridaireMerkezY = saridaireY + saridaire.height/2.0f

if (0.0f <= saridaireMerkezX && saridaireMerkezX <= anakarakterGenisligi
        && anakarakterY <= saridaireMerkezY && saridaireMerkezY <= anakarakterY+anakarakterYukseklig){

    skor+=20
    saridaireX = -10.0f   Çapışma olursa görsel nesneyi ekranın sağına geçirki kaybolsun.

}
```

```
fun carpismaKontrol(){

    val saridaireMerkezX = saridaireX + saridaire.width/2.0f
    val saridaireMerkezY = saridaireY + saridaire.height/2.0f

    if (0.0f <= saridaireMerkezX && saridaireMerkezX <= anakarakterGenisligi
            && anakarakterY <= saridaireMerkezY && saridaireMerkezY <= anakarakterY+anakarakterYuksekligi){
        skor+=20
        saridaireX = -10.0f
    }

    val kirmziucgenMerkezX = kirmiziucgenX + kirmiziucgen.width/2.0f
    val kirmziucgenMerkezY = kirmiziucgenY + kirmiziucgen.height/2.0f

    if (0.0f <= kirmziucgenMerkezX && kirmziucgenMerkezX <= anakarakterGenisligi
        && anakarakterY <= kirmziucgenMerkezY && kirmziucgenMerkezY <= anakarakterY + anakarakterYuksekligi) {
        skor += 50
        kirmiziucgenX = -10.0f
    }

    val siyahkareMerkezX = siyahkareX + siyahkare.width/2.0f
    val siyahkareMerkezY = siyahkareY + siyahkare.height/2.0f

    if (0.0f <= siyahkareMerkezX && siyahkareMerkezX <= anakarakterGenisligi
        && anakarakterY <= siyahkareMerkezY && siyahkareMerkezY <= anakarakterY + anakarakterYuksekligi) {
        siyahkareX = -10.0f

        timer.cancel()//Timer durdur.

        val intent = Intent( packageContext: this@OyunEkraniActivity, SonucEkraniActivity::class.java)
        intent.putExtra( name: "skor", skor)
        startActivity(intent)
    }

    textViewSkor.text = skor.toString()
    }
}
```

```
timer.schedule( delay: 0, period: 20){//0ms geçikmeli 20ms aralıkla tekrarlı çalışır.
    Handler(Looper.getMainLooper()).post {
        anakarakterHareketEttirme()
        cisimlerinHareketEttir()
        carpismaKontrol()
    }
}
```

Çarpışma Kontrolü

- Oyunu bitirecek çarpışma olursa hareket sonlanmalıdır.
- Bunun için timer durdurulur.

```
val siyahkareMerkezX = siyahkareX + siyahkare.width/2.0f
val siyahkareMerkezY = siyahkareY + siyahkare.height/2.0f

if (0.0f <= siyahkareMerkezX && siyahkareMerkezX <= anakarakterGenisligi
    && anakarakterY <= siyahkareMerkezY && siyahkareMerkezY <= anakarakterY + anakarakterYuksekligi) {
    siyahkareX = -10.0f

    timer.cancel()//Timer durdur.

    val intent = Intent( packageContext: this@OyunEkraniActivity, SonucEkraniActivity::class.java)
    intent.putExtra( name: "skor", skor)
    startActivity(intent)
}
```

En Yüksek Skor

En Yüksek Skor

- En yüksek skor bilgisini telefonun hafızasına kalıcı olarak kayıt edebiliriz.
- Bunun için en pratik yol *SharedPreferences* kullanmaktır.

```
val sp = getSharedPreferences( name: "Sonuc", Context.MODE_PRIVATE)
val enYuksekSkor = sp.getInt( key: "enYuksekSkor",  defValue: 0)
```

```kotlin
class SonucEkraniActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sonuc_ekrani)

        val skor = intent.getIntExtra( name: "skor", defaultValue: 0)
        textViewToplamSkor.text = skor.toString()

        val sp = getSharedPreferences( name: "Sonuc", Context.MODE_PRIVATE)
        val enYuksekSkor = sp.getInt( key: "enYuksekSkor",  defValue: 0)

        if (skor > enYuksekSkor){

            val editor = sp.edit()
            editor.putInt("enYuksekSkor", skor)
            editor.commit()

            textViewEnYuksekSkor.text = skor.toString()
        }else{
            textViewEnYuksekSkor.text = enYuksekSkor.toString()
        }

        buttonTekrarDene.setOnClickListener { it: View!
            startActivity(Intent( packageContext: this@SonucEkraniActivity,MainActivity::class.java))
            finish()
        }
    }
}
```

En yüksek skorun kayıt edilen yerden okunması

Son skor en yüksek skordan büyük ise son alınan skor en yüksek skor olarak kayıt edilir.

Teşekkürler...


kasım-adalan


kasimadalan@gmail.com


kasimadalan