### OYUN PROGRAMLAMA PROJE

Ahmet Ali Aşık

171816005

LİBGDX

FLAPPY BİRD

bu proje ödevinde libgdx oyun motorunu kullanarak android studio üzerinden hem desktop hemde android telefonlarda kullanılmak için gerekli konfigrasyonlar ve oyunun programlanması yapılacaktır.

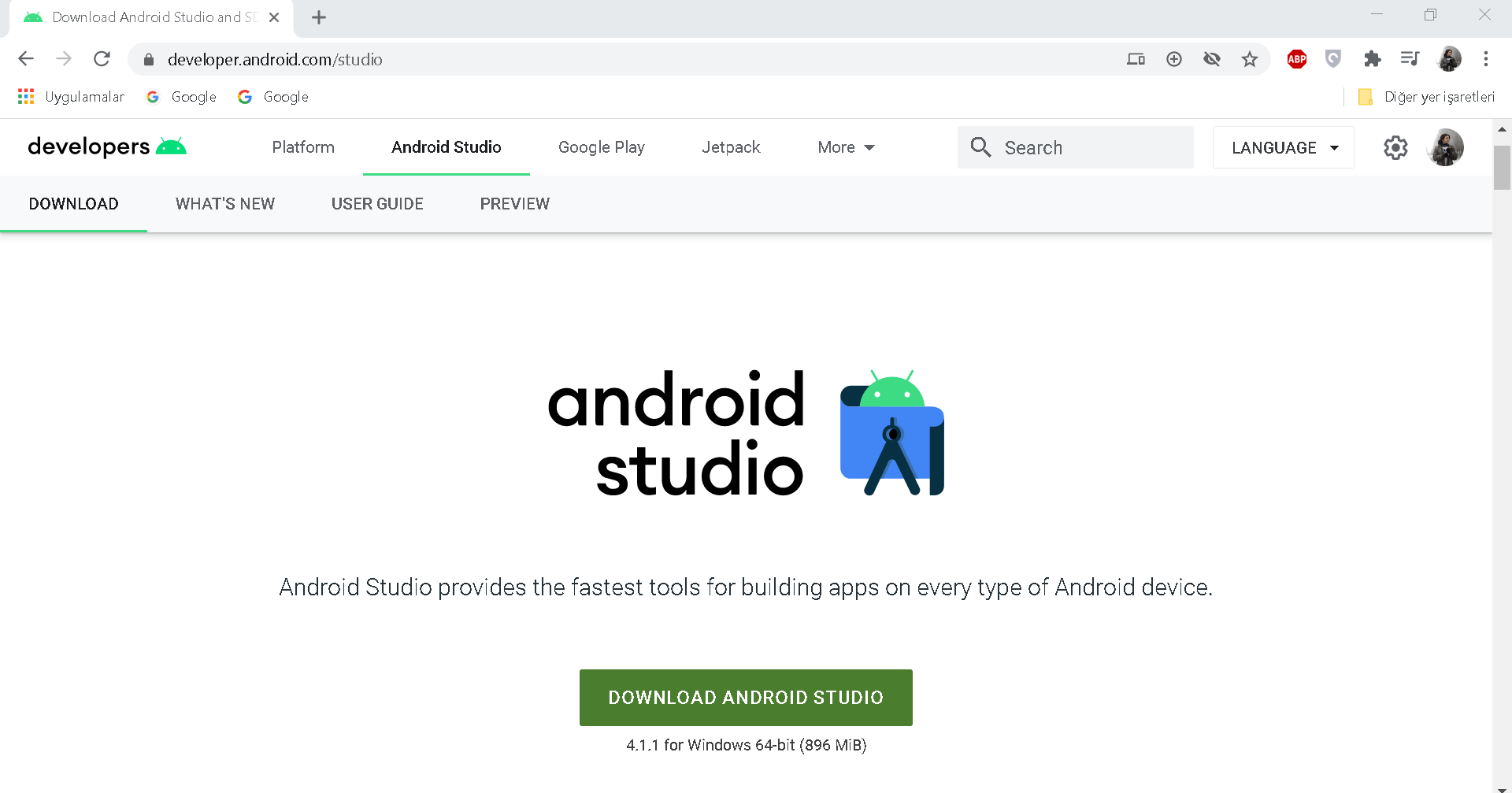
Giriş

1.Gerekenler:

1.Bilgisayarınızda android studio ide kurulu olmalıdır

<https://developer.android.com/studio>

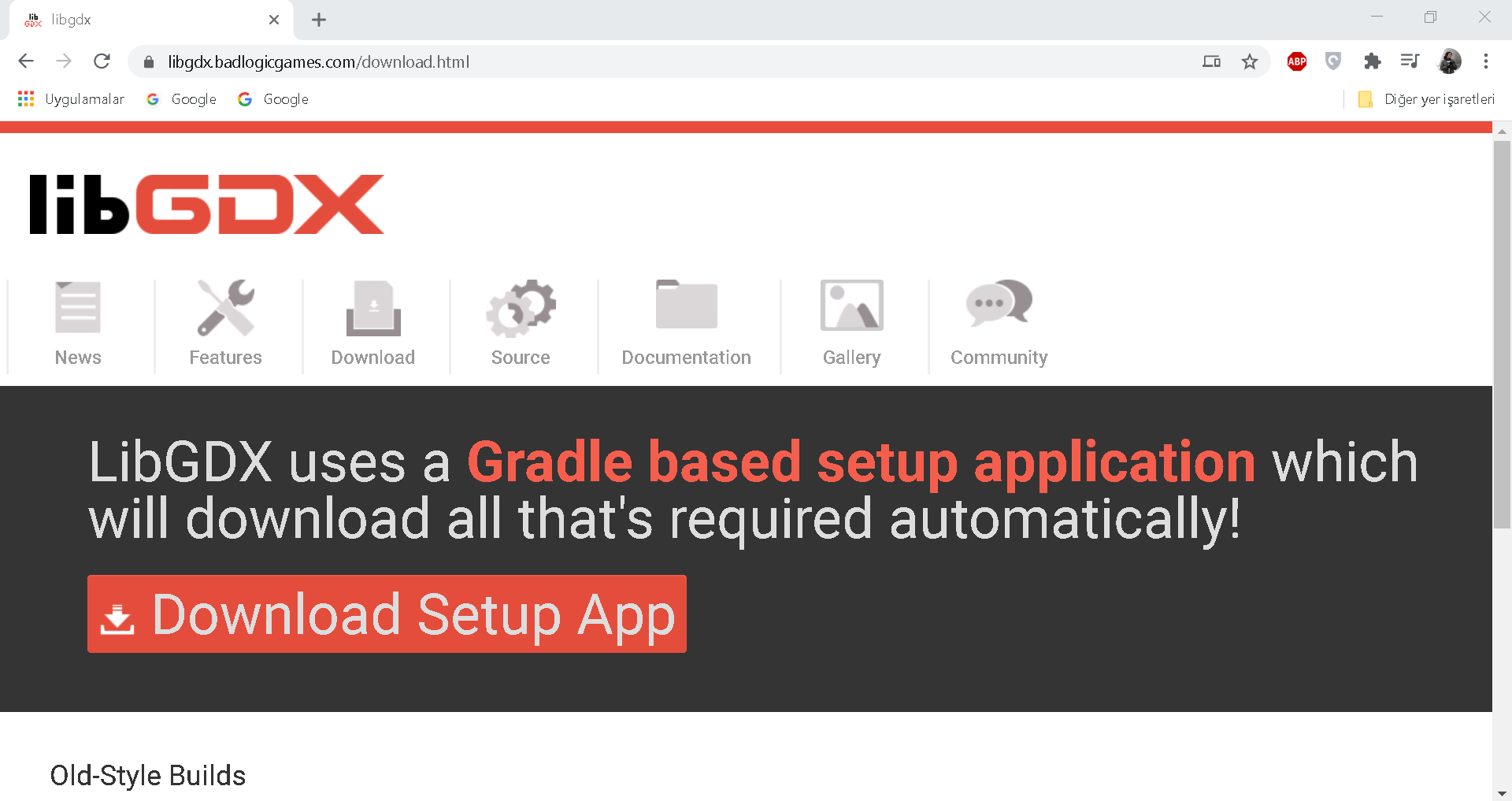
buradaki linkten android studio indirme işlemini yapabilirsiniz.



Download andorid studio butonuna bastıktan sonra next next diyerek kurulumu tamamlayın.

2.Libgdx kurulumu ve başlarken:  
 <https://libgdx.badlogicgames.com/download.html>

Linke tıklayarak çıkan sayfada Download Setup app butonuna tıklayınız. Sizin bulunduğunuz zaman zarfında görsel kısım değişiklik gösterebilir bunu önemsemeyin.



İndirilen gdx-setup.jar dosyasını açın.

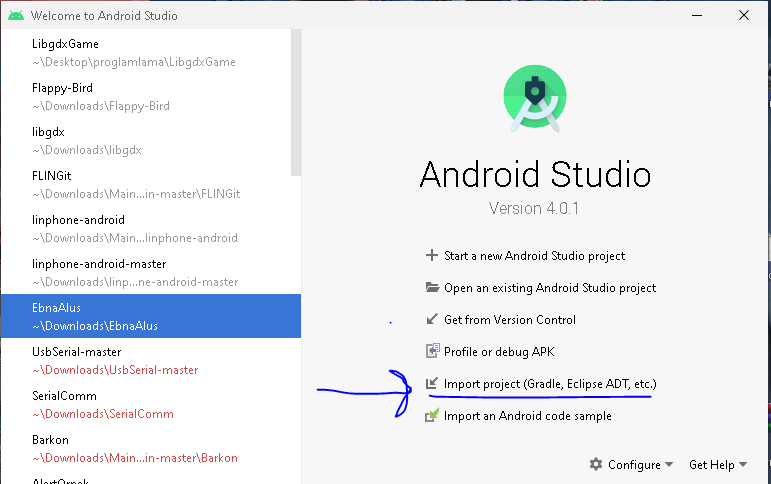


Burada gerekli isimlendirmeleri, proje konumunu, main class ismini, android sdk locationu’nu, oyunun hangi platformlarda kullanılması ayarlamalarını kendimize göre yapıyoruz ve en alttaki kutucuklardan ihtiyacımız olan kütüphaneleri dahil ediyoruz.

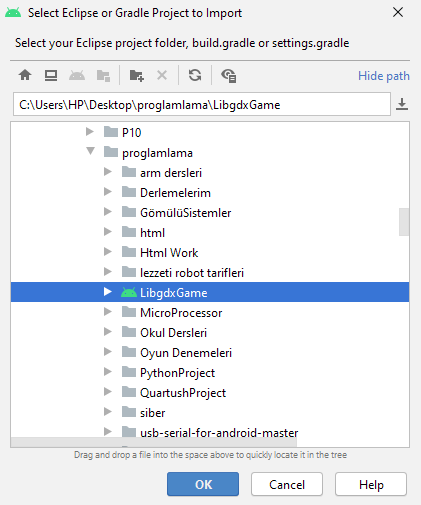


Benim ayarlamalarım bu şekilde sizde bu şekilde yapabilirsiniz.

Ardından android studioyu açıyoruz.

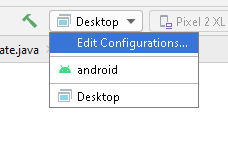


İmport procext seçeneğini seçtikten sonra libgdx setup panelinde vermiş olduğumuz lokasyona gidiyoruz.

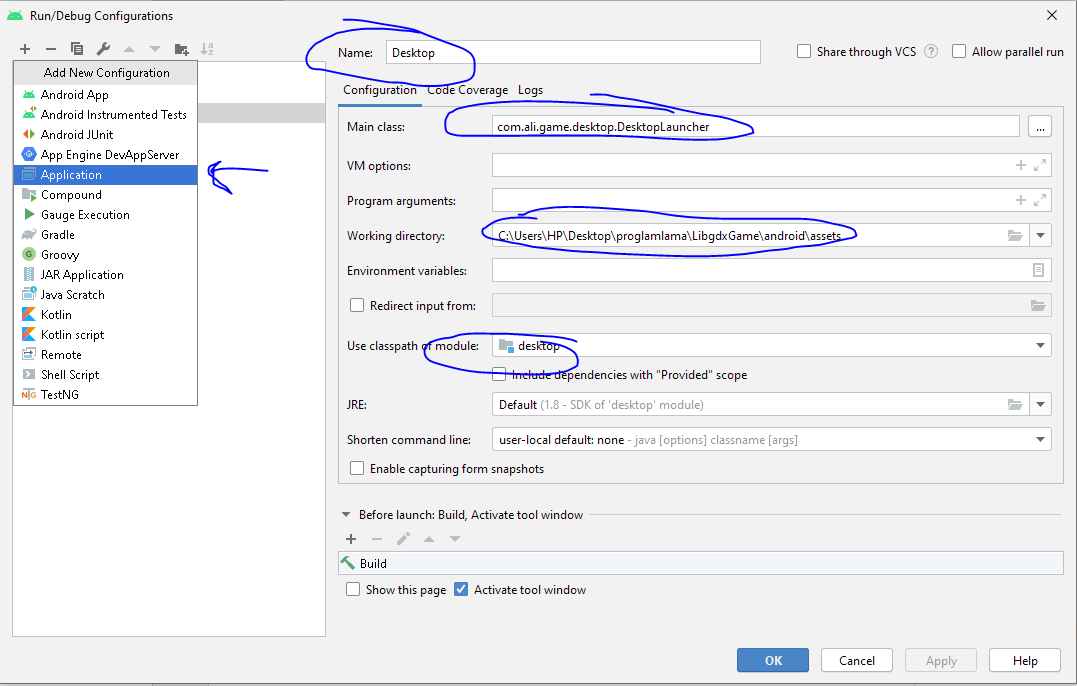


Lokasyonu bulup seçiyoruz ve OK butonuna basıyoruz

Projeyi desktop için de kullanacak arkadaşlar için son bir aşamamız kaldı



İdenin üst kısmında bulunan edit configration’s a tıklıyoruz.



Sırasıyla önce sol üstteki + simgesine tıklayıp Application seçiyoruz ve gelen ekranda ilgili yerleri dolduruyoruz.

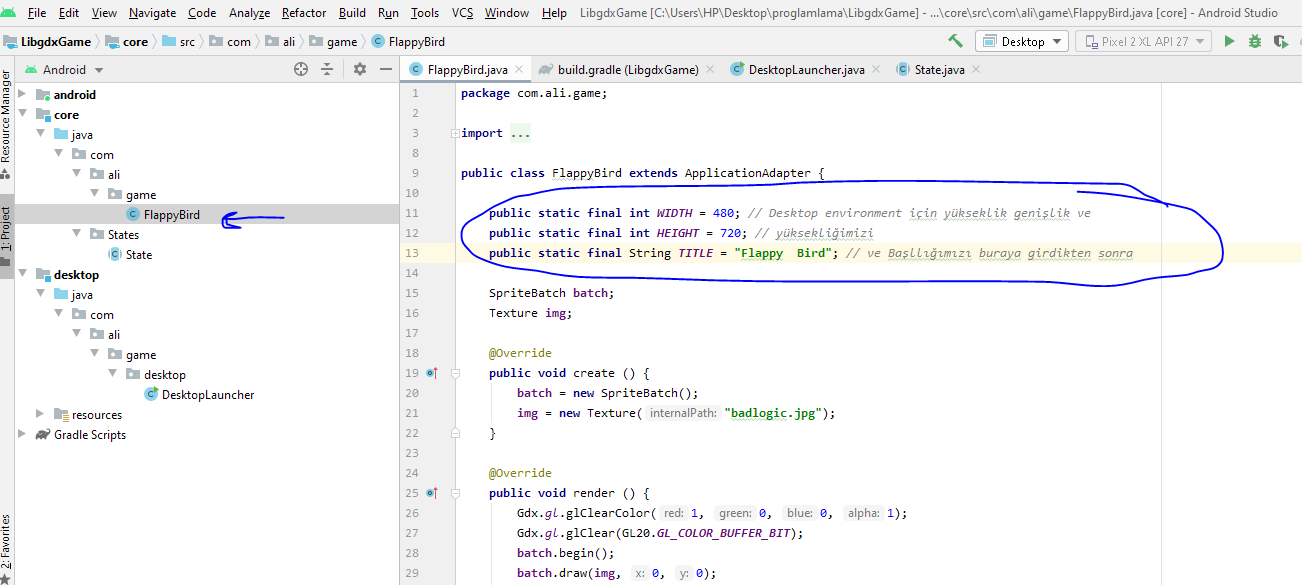
Ben doldurduğum kısımları mavi balon içine aldım sizde benim yaptıklarımın aynısını yapabilirsiniz.

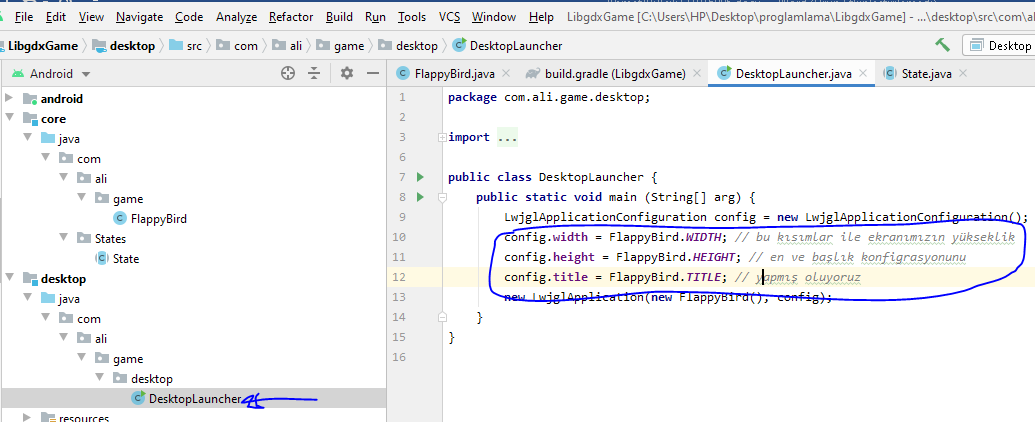
Okey tuşuna bastıktan sonra şablonumuz hazır isterseniz projeyi çalıştırarak bun görebilirsiniz.

Projeyi çalıştırdığınızda

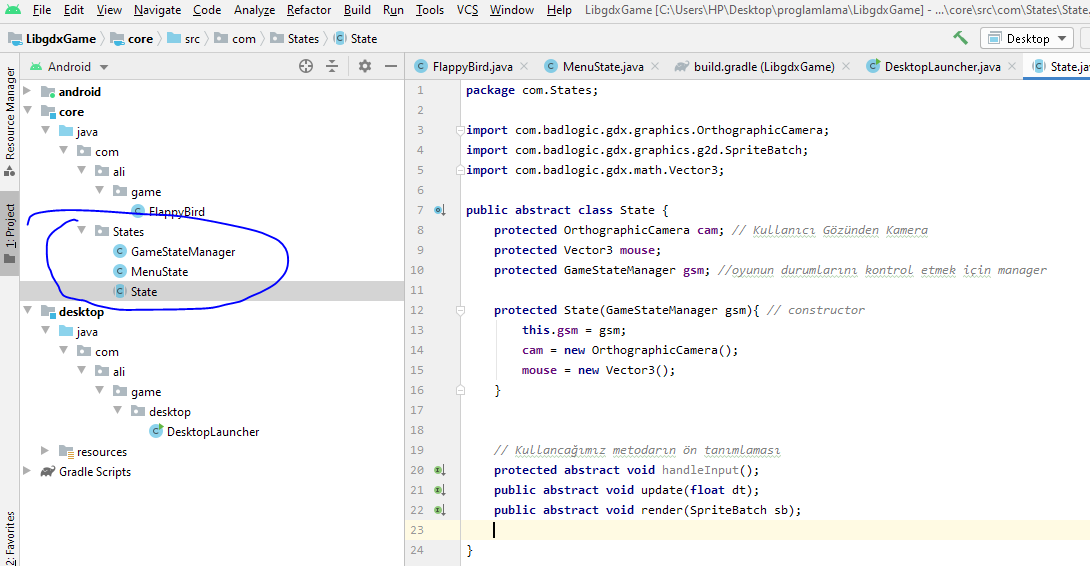


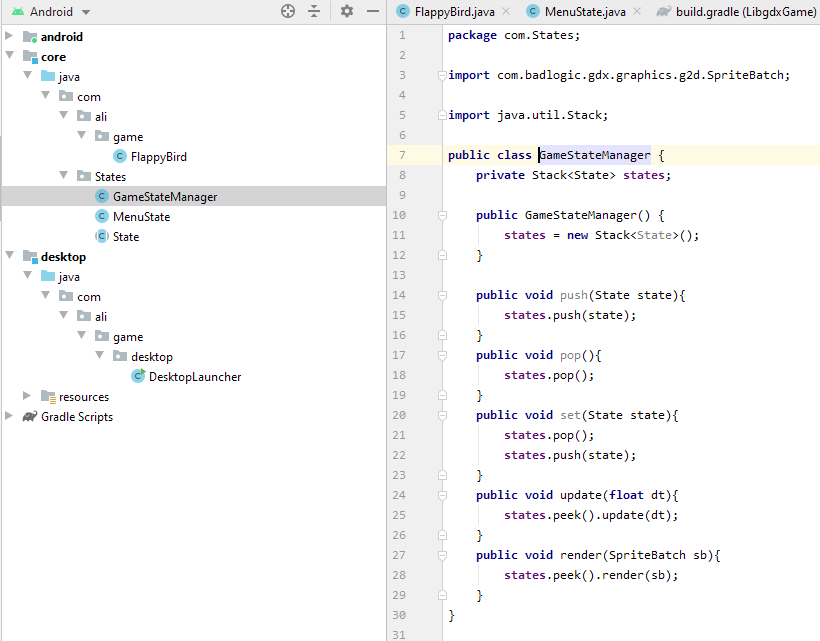
Bu şekidle bir şablon geliyor karşımıza.



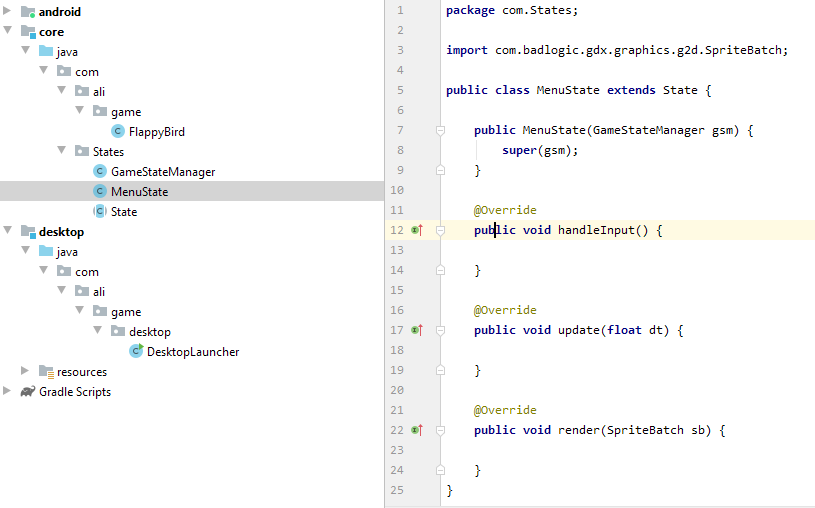
Açıklama satırlarını takip edin.

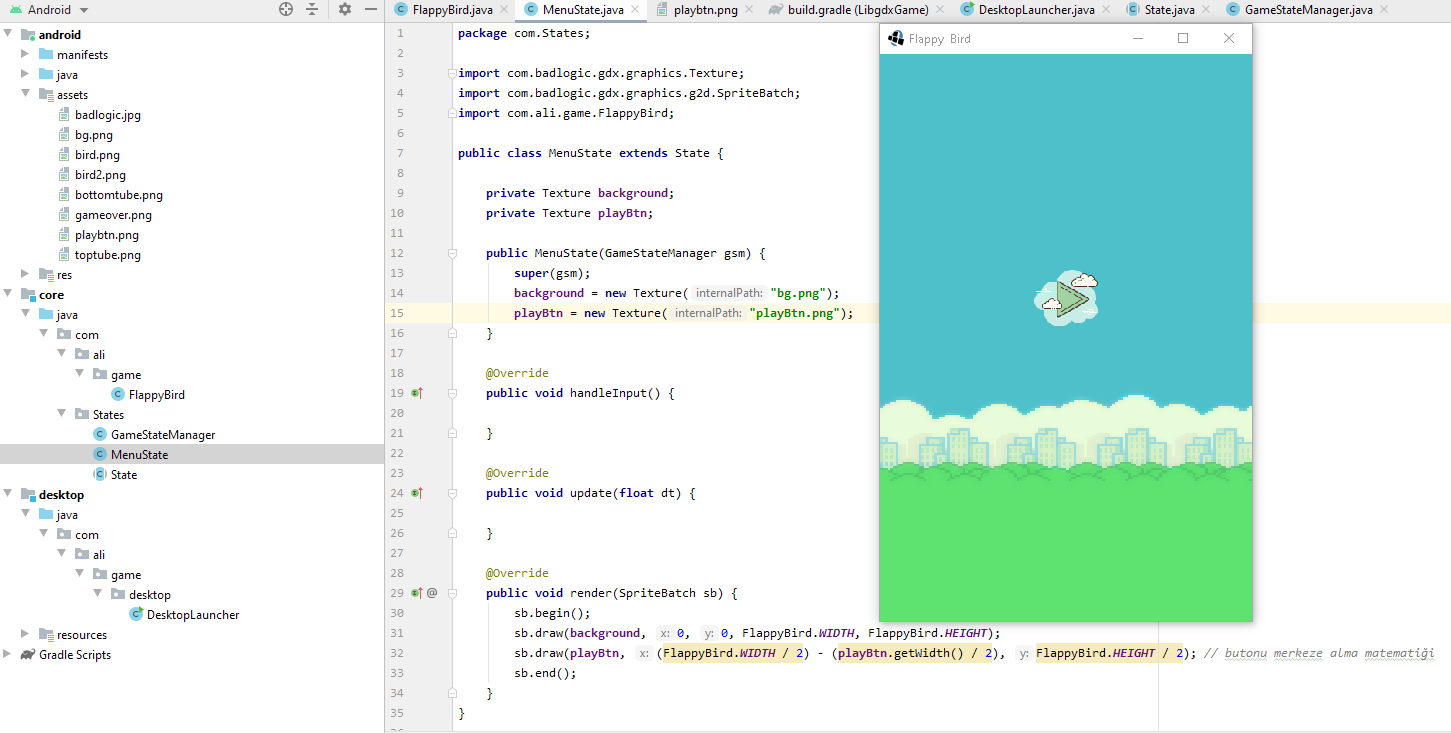
Temel şablonumuz hazır şimdi kodlamaya geçelim.

İlk önce States folder oluşturuyoruz ve içinde state Class’ı oluşturuyoruz bu Class oyuncunun hangi durumda olduğunu kontrol etmemize yarıyor örneğim: menü durumu, Oyun durumu gibi gibi…



Game state manager class ı oyunun durumunu yönetmek içiin kullandığımız class şu an bişey yapmıyor fakat temel şablonu bu şekilde.

Menu state class ının temel şablonunu bu şekidle oluşturuyoruz.



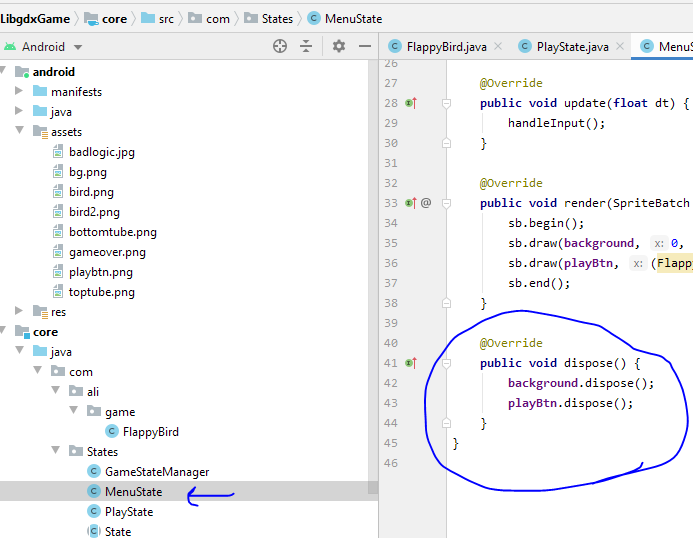
Menü durumunun şablonu bu şekilde buton ve arka plan resmi dosyalarını internetten indirebilirsiniz bir sürü bedava .png dosyası bulabileceğiniz site var örnek olarak bu projede

<https://icons8.com/icons>

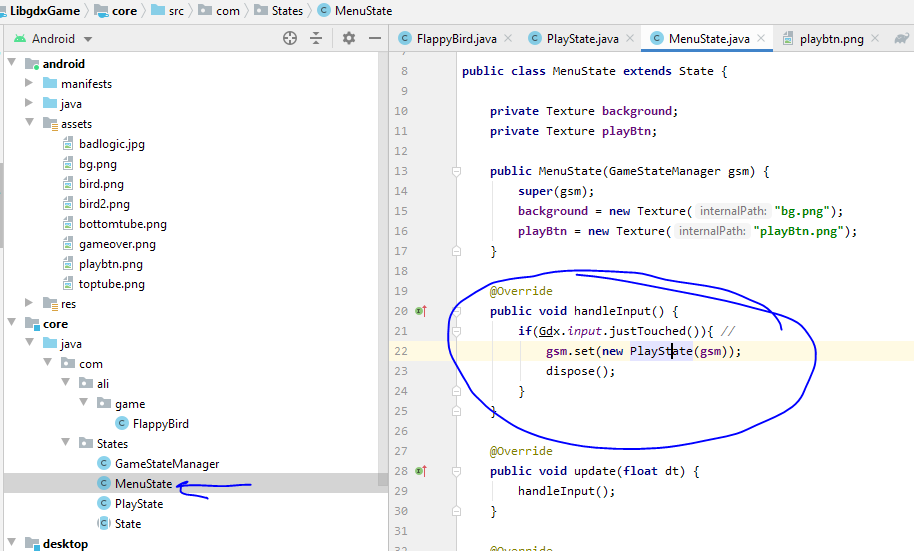
bu siteden yararlandım

oyundaki png dosyaları olsun ses dosyaları olsun bu tarz dosyalar herzaman android/assets dizininde bulunur. Bu dizine direk kopyala yapıştır yaparak kullanmak istediğiniz resimleri koyabilirsiniz.

Ardından menü de bulunan butona bir action vermemiz gerekiyor bunu da handle methodu içerisinde yapacağız.

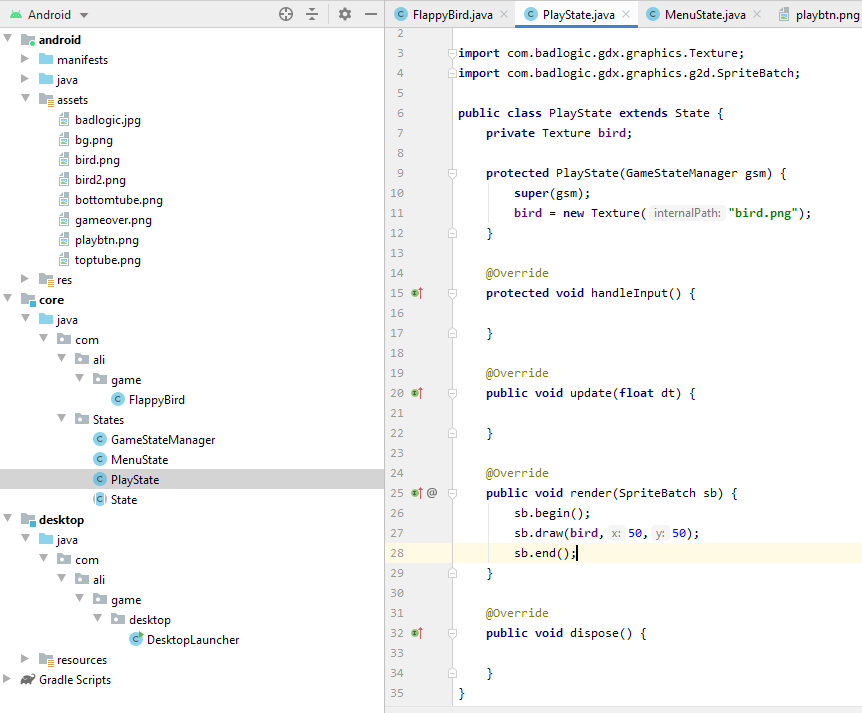


Öncelikle mevcut pencereleri kaptmak için dispose methodu tanımlıyoruz handle methodunda input girişi var mı diye kontrol ettiğimizde bu method çalışacak.

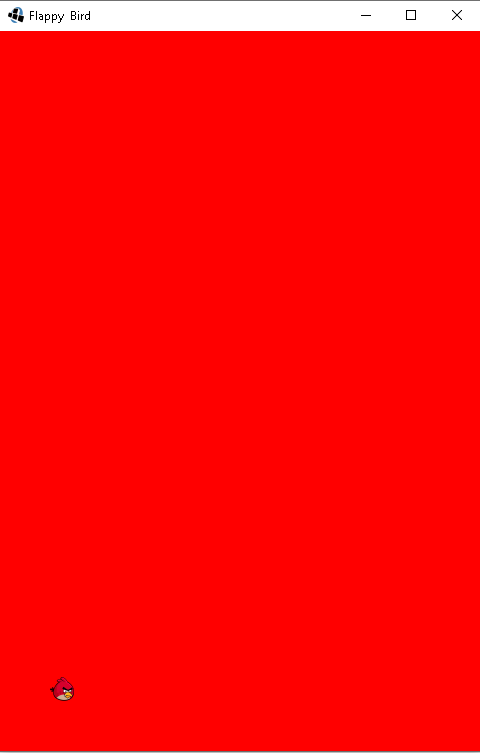


handleInput methodunda ekrana dokunulursa mevcut ekranları kapat ve oyun ekranına geç komutunu yazıyoruz.

Yeni açılacak ekran ise yeni yazdığımız State Class ını extend eden PlayState Yani oyun anı durumu Class ını yazıyoruz oda şu şekilde.



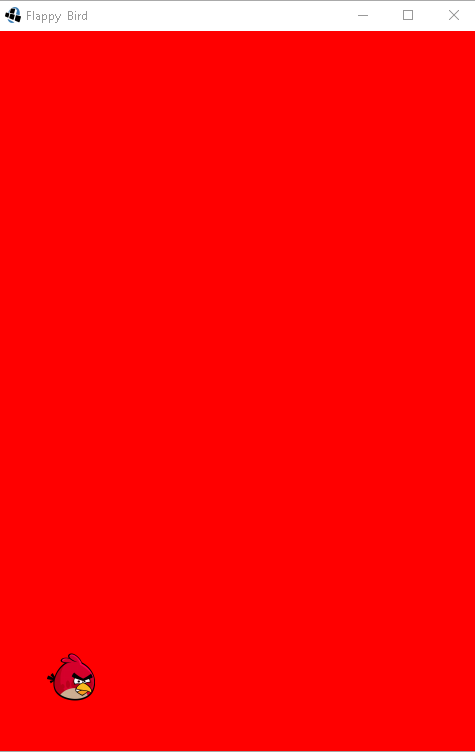
Render methodunda görülen değer kuşun anlık olarak yerini belirtiyor orası değişecek.

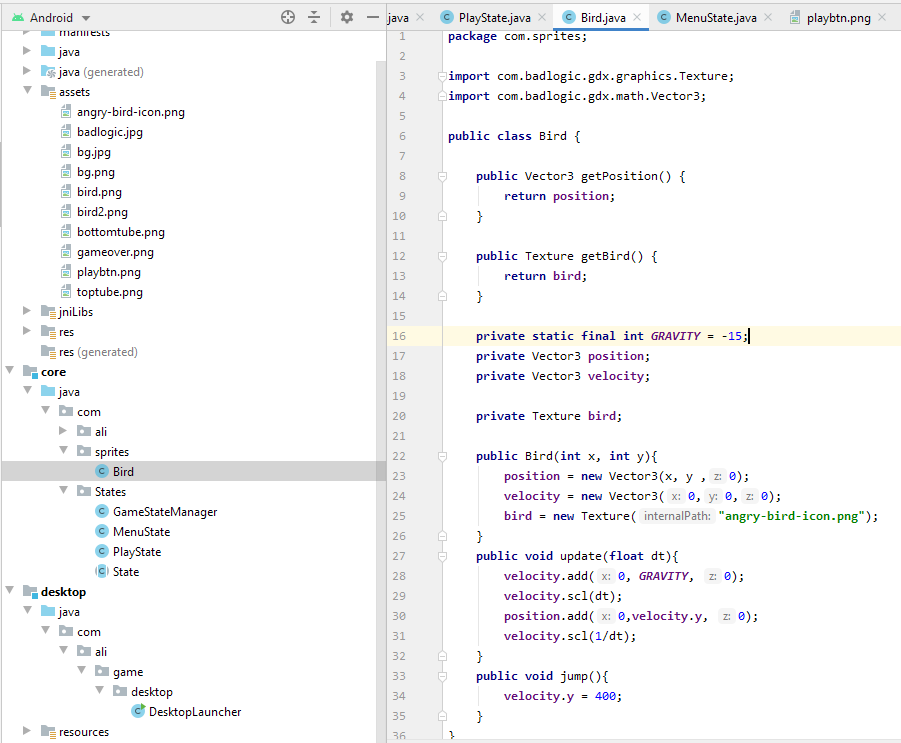


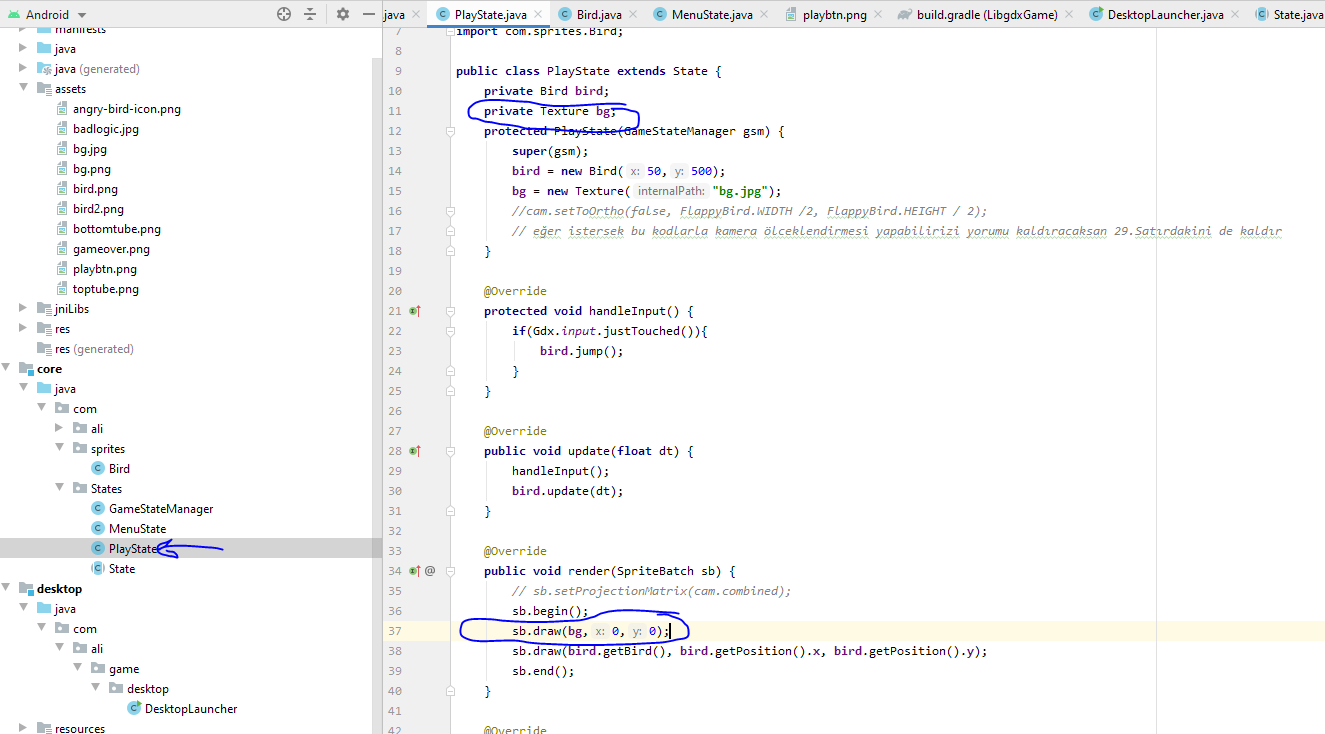
Ekrana tıklandığı anda şu anlık olarak gelen görüntü bu görünüyor ki kuş resmini daha büyük seçmeliyim Ya da kamerayı ölçeklendirmeliyim. Ben kamerayı ölçeklendirmeyi seçiyorum çünkü öğrenmiş olalım bunu da bir gün kullanırız elbet.

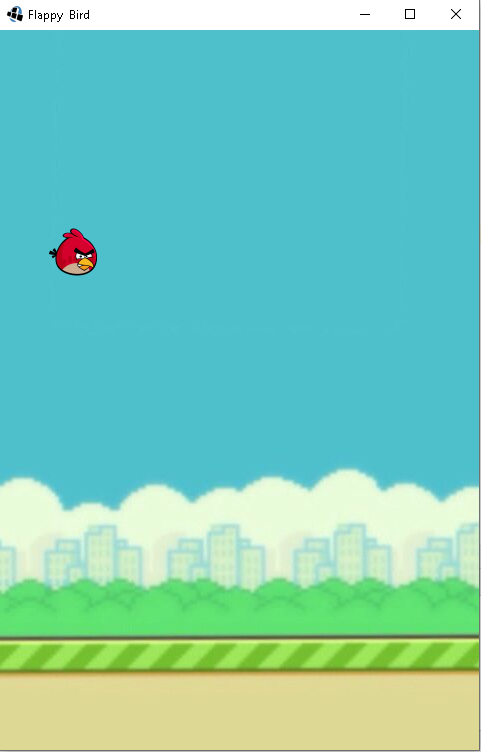
Bu işlemden sonra kuşun pixel pixel göründüğü için yeni resim seçtim fakat bu bilgide burda bulunsun

Yeni görünüm şu şekilde

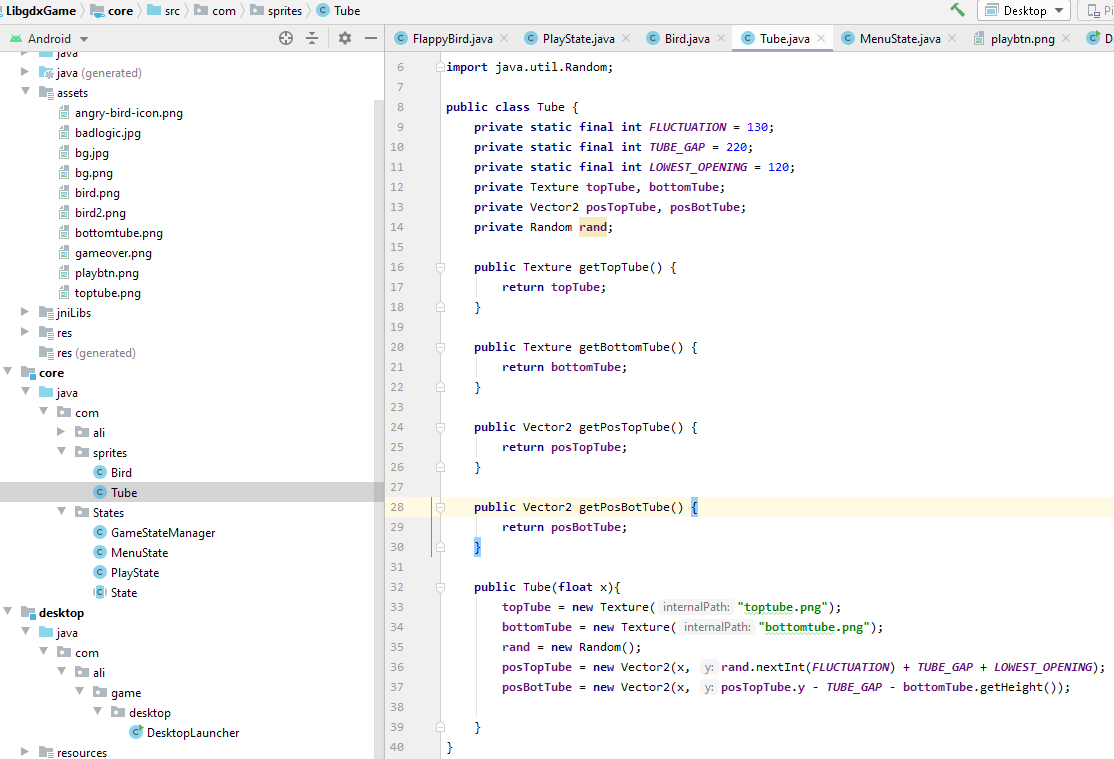


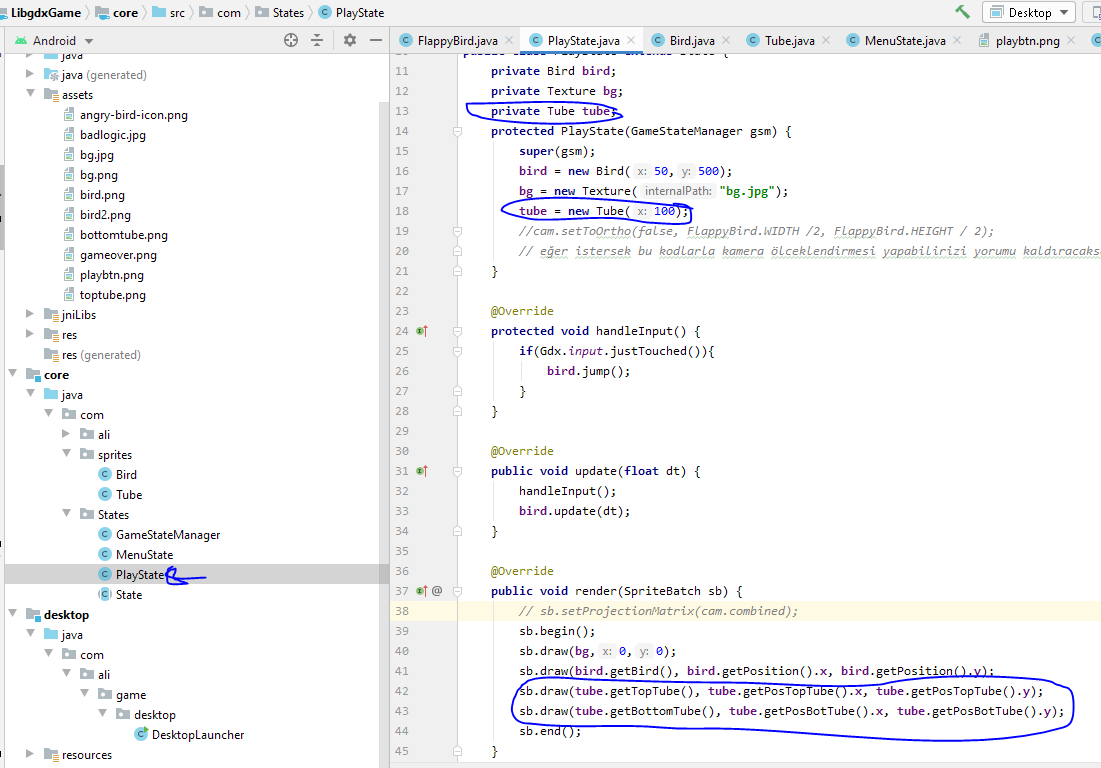
Kuşun hareket etmesi için gerekli class yeni bir splite dizini oluşturup altına yazdık Buradan değişkenlerin değeriyle oynayabilirsiniz.

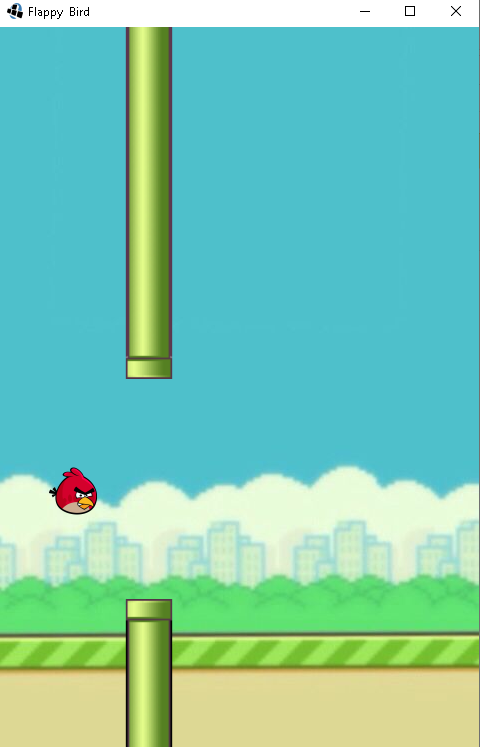
Arka planı da iki satır ile ekledik. Gerekli dosyaları internetten indiriyoruz.



Şu anda kuş bastığımda hareketini yapıyor ve arkaplan gelmiş durumda.

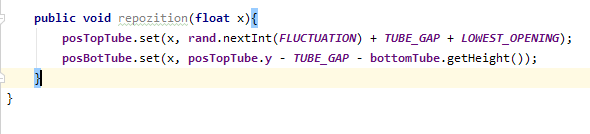
Tüp klasımızı oluşturduk ve pozisyon matematiğini kodladık.

PlayState class ının içinde tüplerimizi yerleştirdik ve render methoduna ekledik.

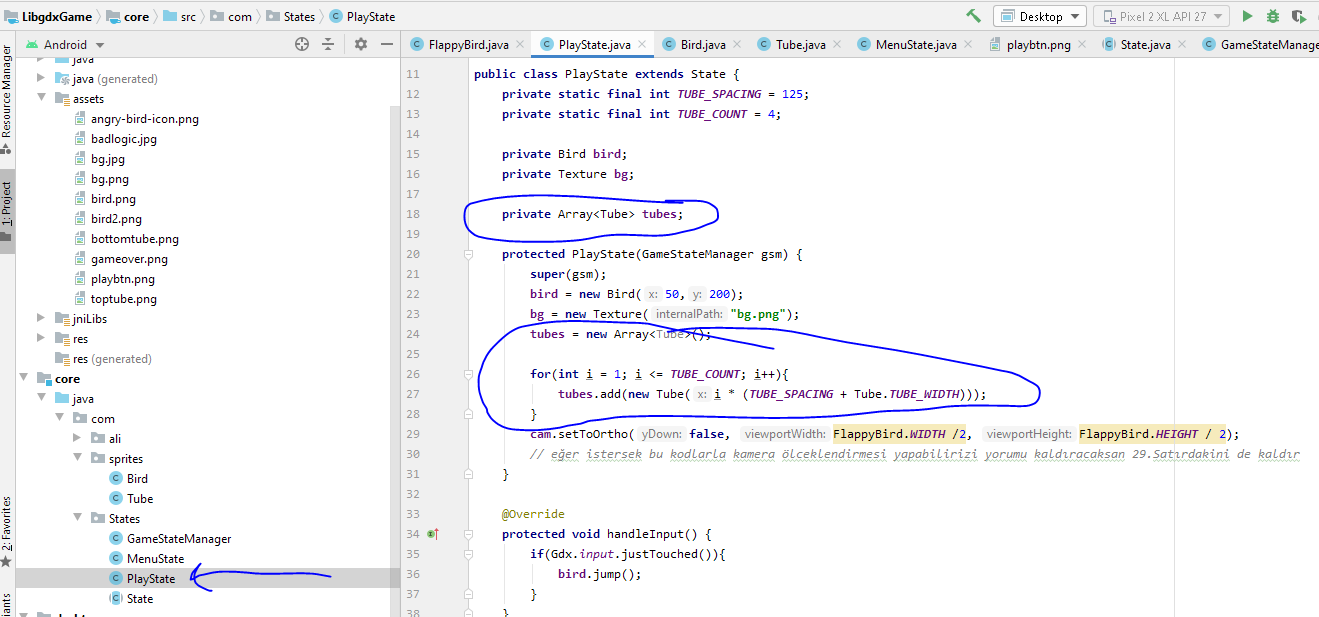


Oyunun görüntüsü anlık olarak bu şekilde

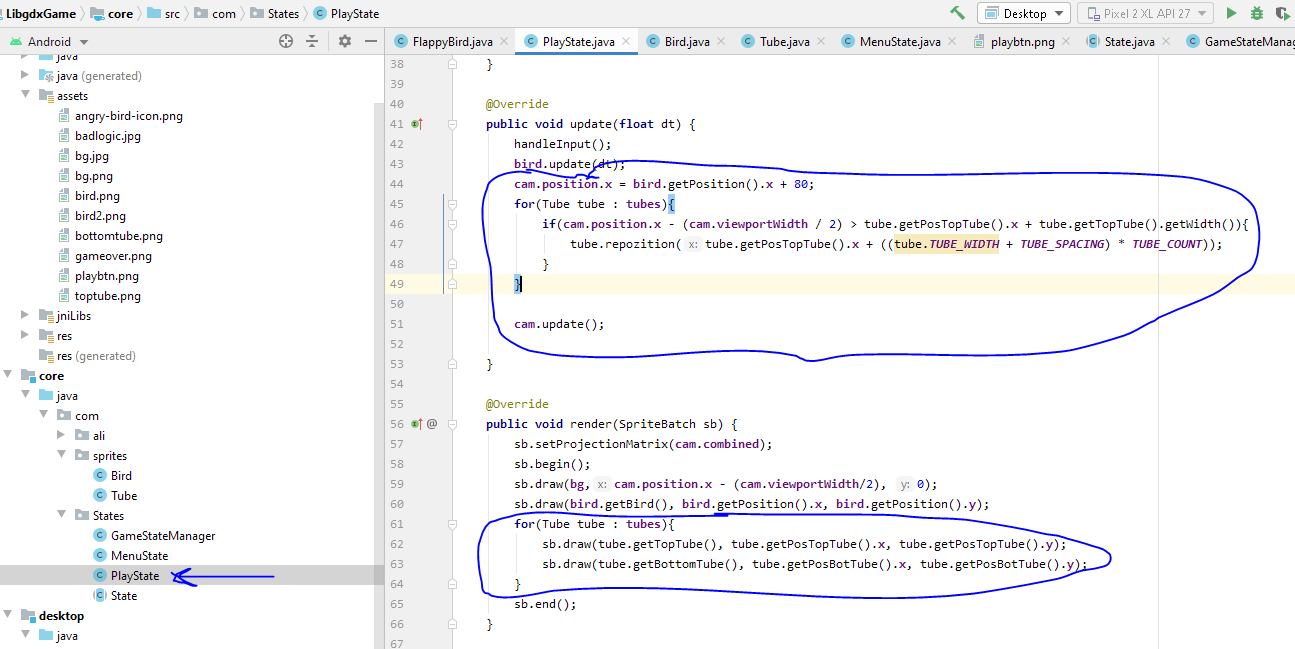
Şimdi tüp klasına yeni bir metod ekleyelim.

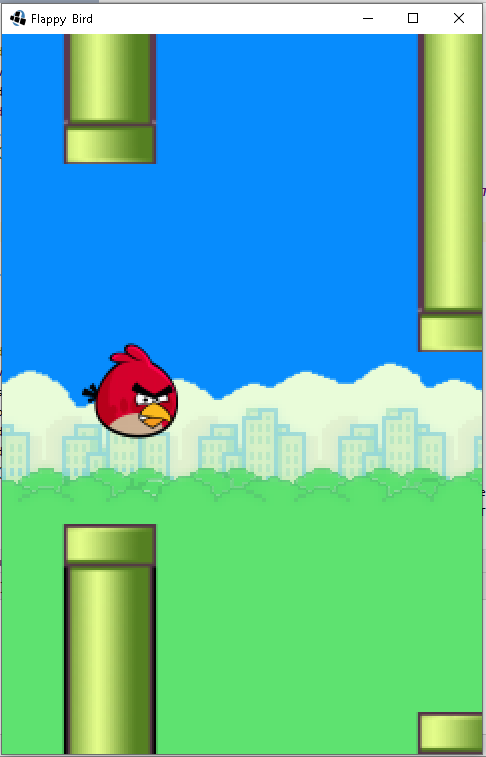


Bu metod ile oluşturulan yeni tüplerin konumları belirleniyor. Bu şekilde art arda aoluşturduğumuz tüpler birbirinin ardı sıra gelecek ve üst tüp ile alt tüp arasındaki mesafe her seferinde random olarak atancak.

Hadi PlayState klasına gidelim ve orda birkaç değişiklik yapalım.

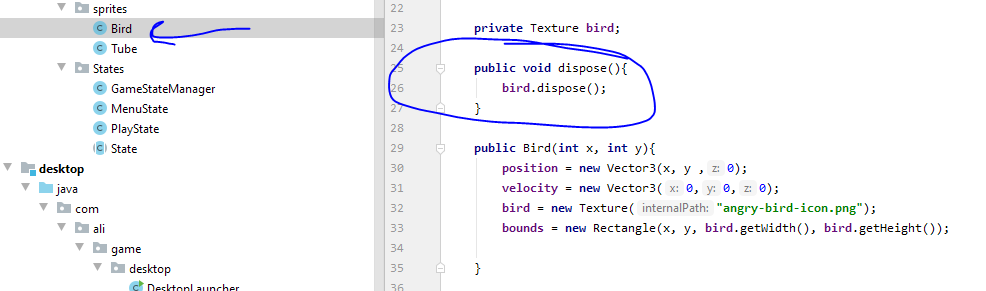
Tek tüp oluşturan kısmı sildik ve yerine tüp tipinden veri depolaayan bir arraylist oluşturduk.

PlayState class ında update kısmına tüp arrayinin konumlandurmasını girdik ve render methodunda bunu görüntüledik.

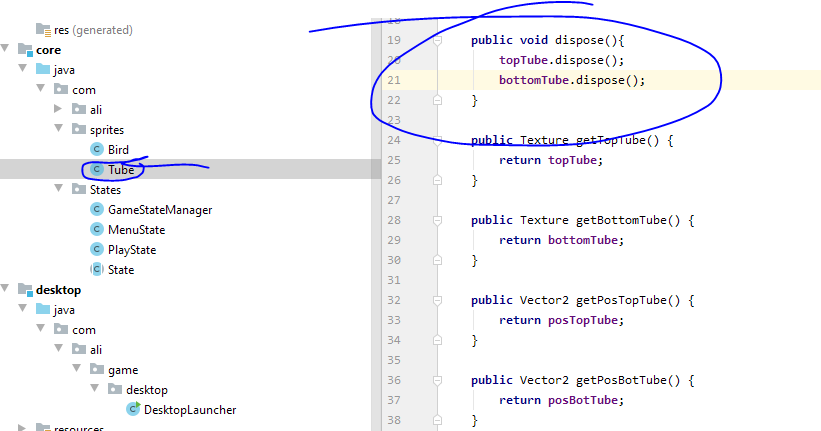


Anlık olarak oyunun görüntüsü bu şekilde.

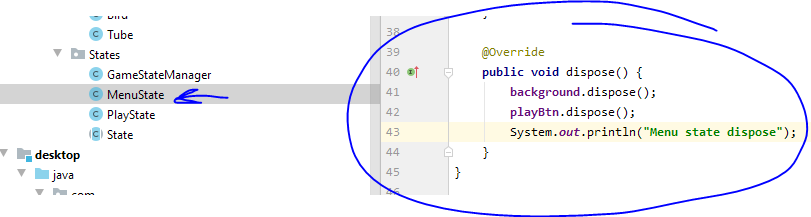
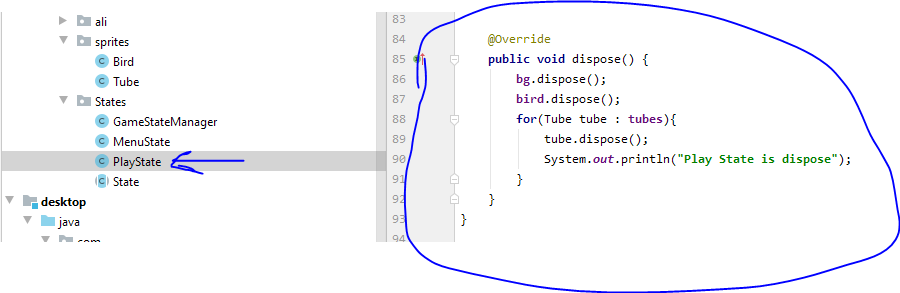
(Arka plan resmini değiştirdim toprak ksımını sonradan ekleyeceğim ve kameranın geri gitmemesi gibi bir sorunla karşılaştım bu hata kamera ölçeklendirmesi yaptığımda yok oldu o yüzden oyunumuz artık kamera ölçeklendirmeli görünüyor.)

Şimdi dispose methoduna geliyoruz. Oyunda her yandığımızda veye bir durum değiştirdiğimizde mevcut veriable ları dispose edersek belleğimizi yormamış oluruz.

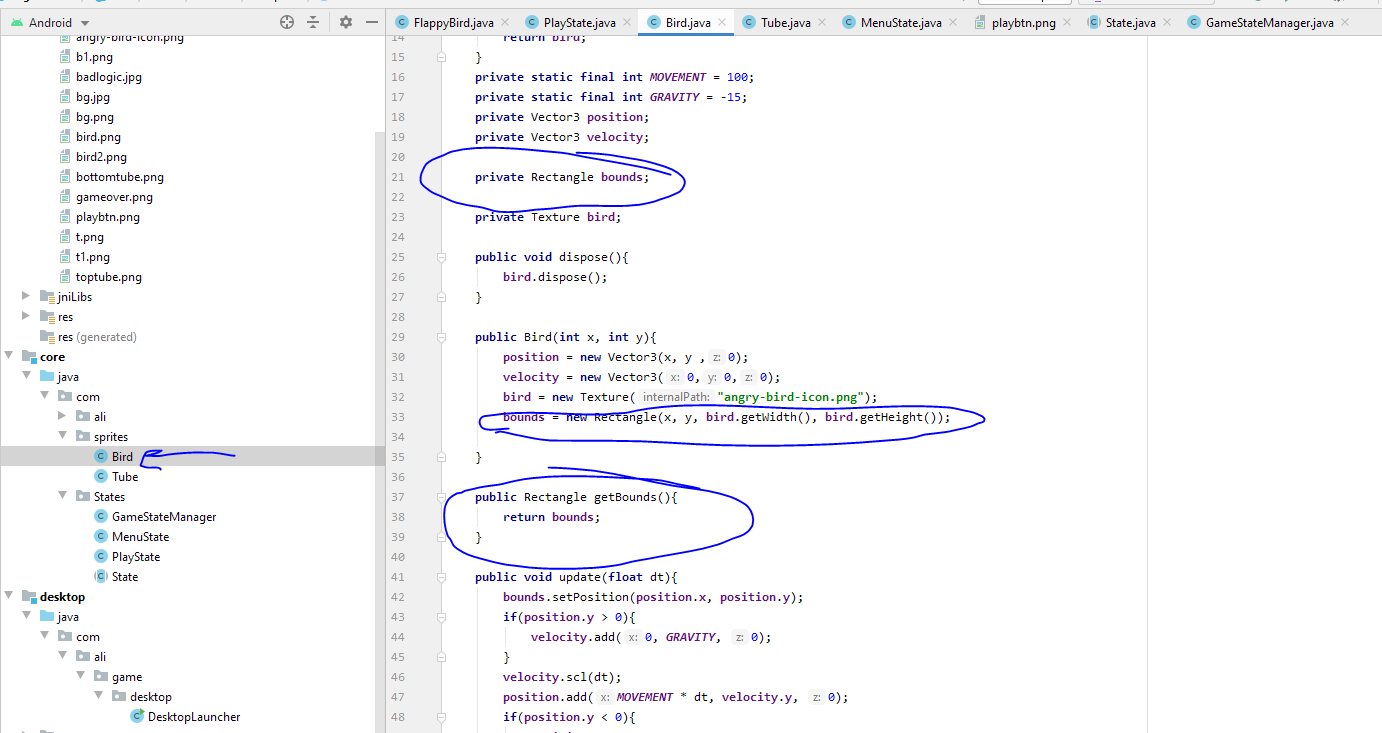
Bird class ına dispose metodu ekledik



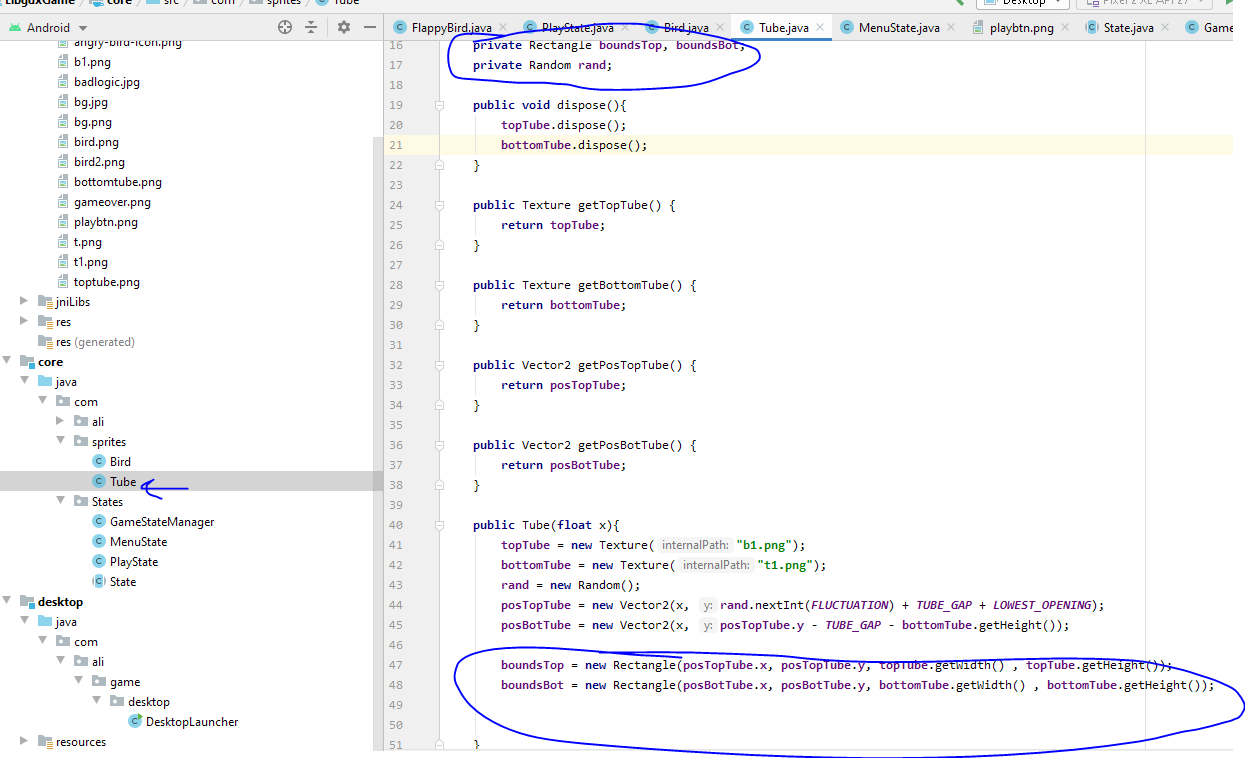
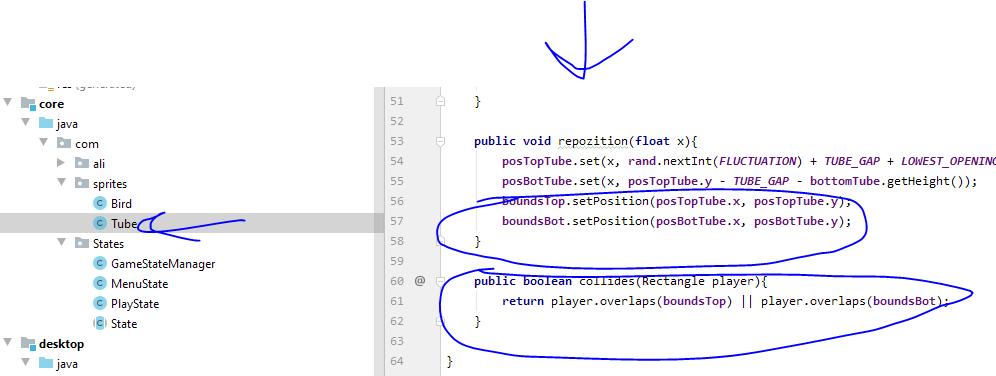
Tube Class ına dispose methodu ekledik

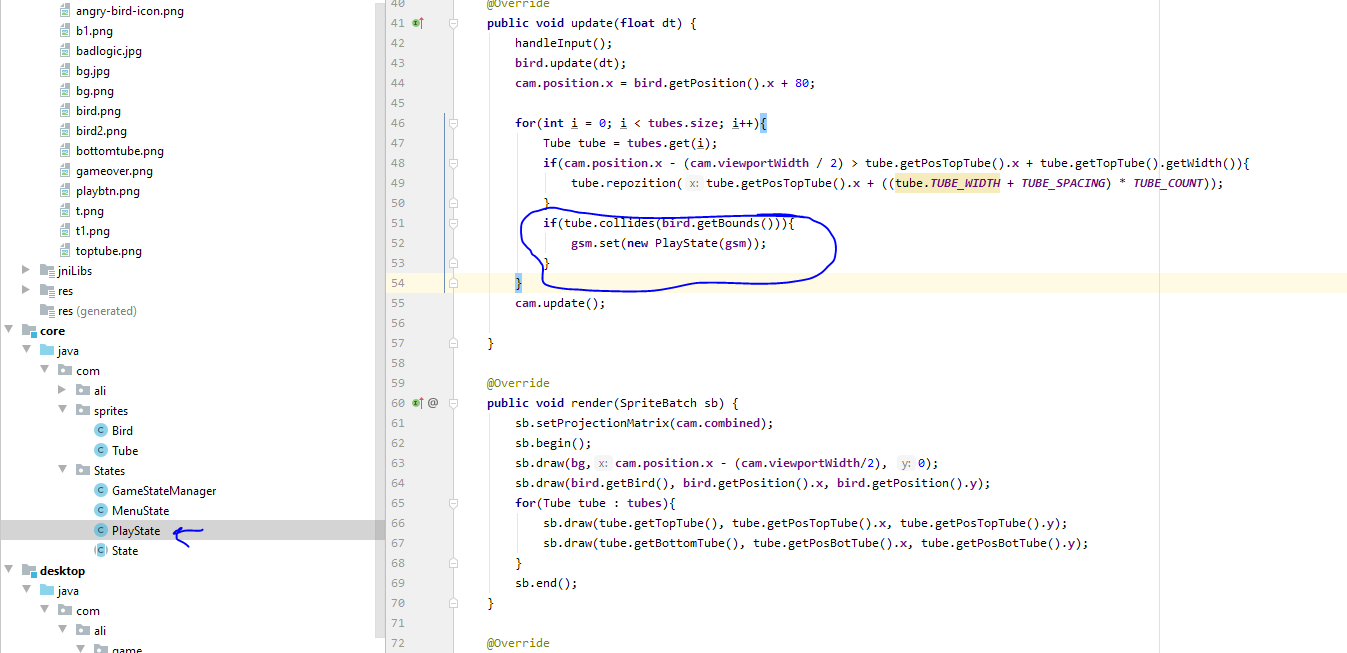
MenuState class ına dispose methodu ekledik ve çalıştığında konsol ekranından bilgilendirme mesajı yolladık.

PlayState class ına dispose methodu ekledik ve konsol ekranından bunu kontrol ettik.

Şimdide kuşumuzun tüplere çarptığında oluşacak olan işlemleri yapalım

Kuşumuzun tam üstüne görünmeyen bir dikdörtgen koyuyoruz.

Ardından Tube Class ına gidip aynı işlemleri orada da gerçekleştiriyoruz. Ve collides methoduyla direk tüplerin konumunu göndermiş oluyoruz.

Son olarak PlayState Class ının içerisindeki update methodundaki for döngüsüne bir if koşulu ile kuşun tüplere çarpıp çarpmadığını kontrol ediyoruz. Şu anda eğer kuş çarparsa oyun yeniden başlıyor.

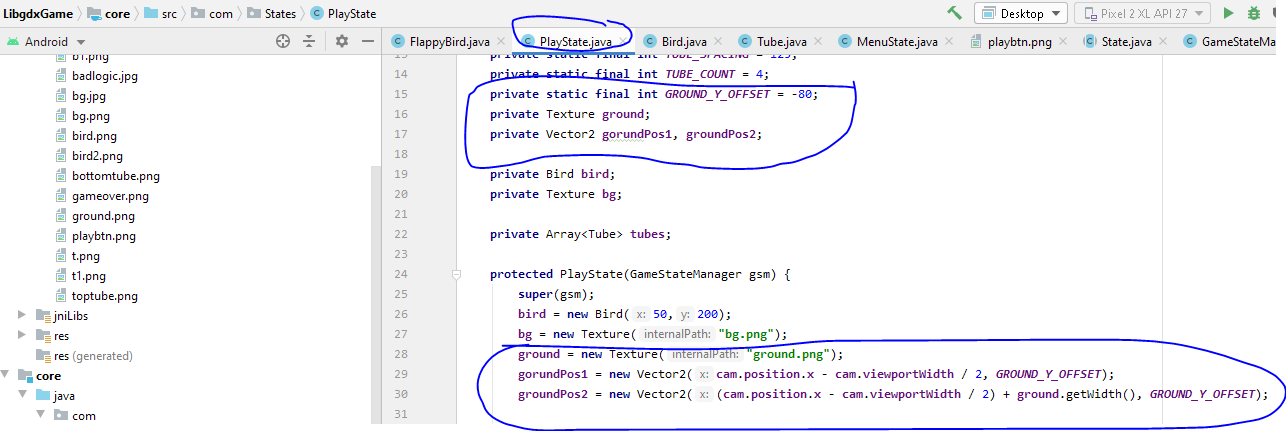
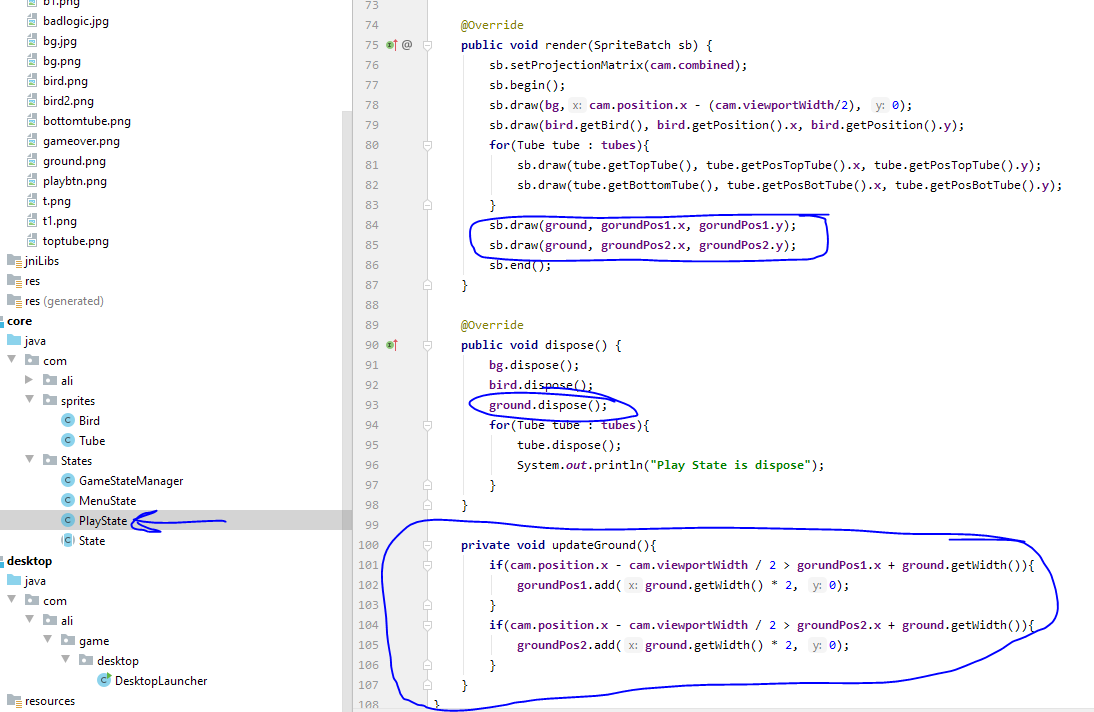


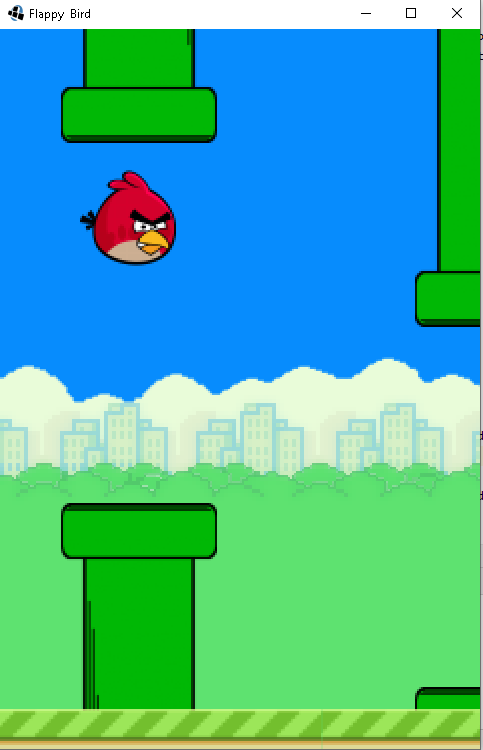
Anlık oyun görüntüsü(Bu tüplerin görünümü hoşuma gitti tüpleri değiştirdim) Konsol ekranında görüldüğü üzere menü durumunu başarıyla dispose ettiğimiz görünüyor.



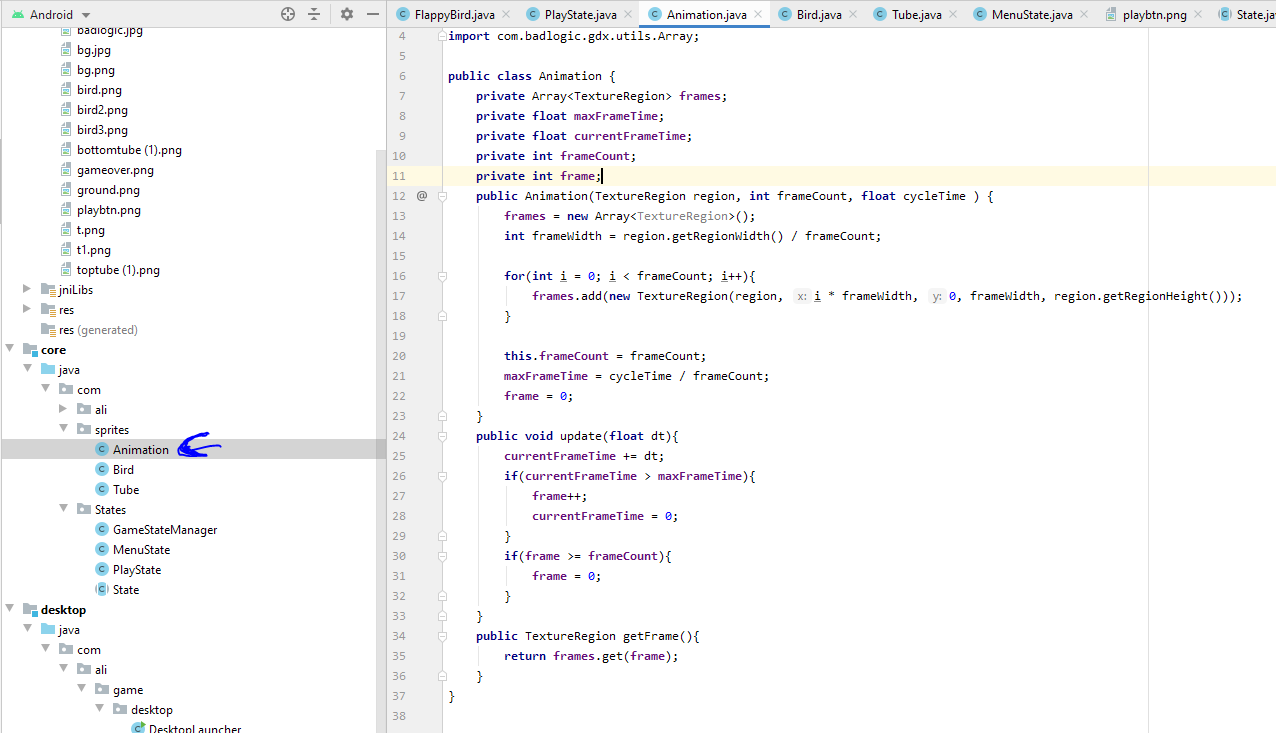
Her tüpe çarptığımızda oyun son buluyor ve kuş tüpler arkaplan dispose edilip oyun tekrar başlatılıyor.

Şimdi oyuna toprak ekleyelim ve toprağa çarpınca kuş ölsün ve tabiki oyun kapantığında toprağı dispose etmeyi unutmayalım.

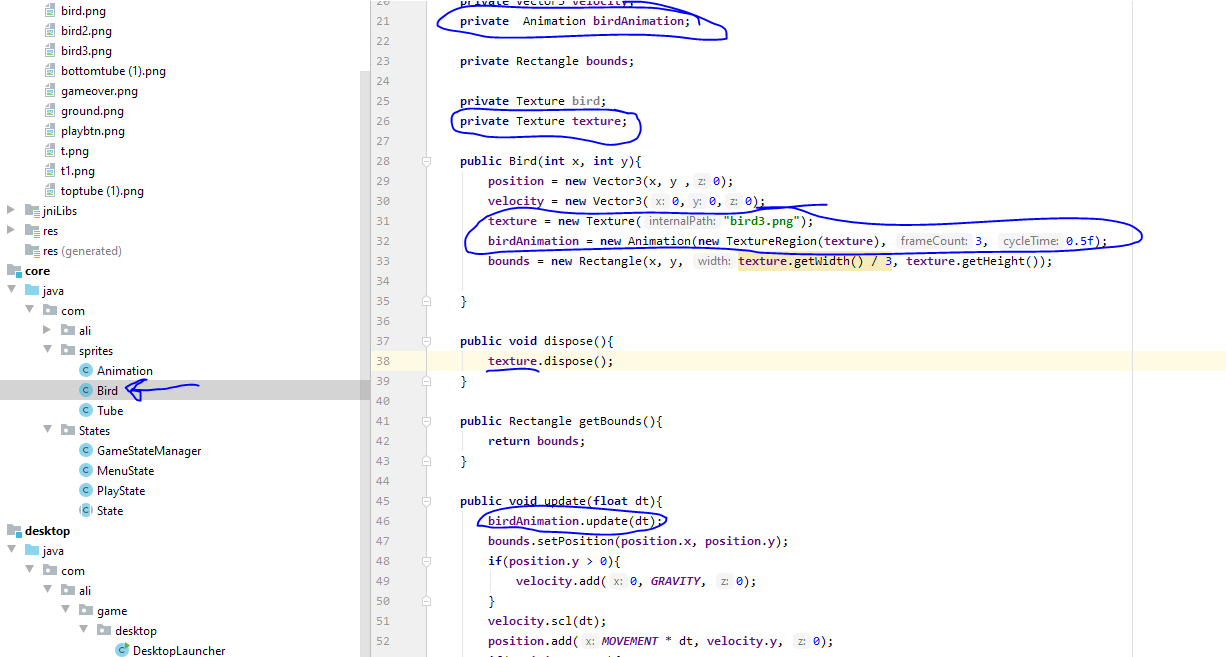
Play State class ında Texture aldık konum için vectorlerimizi tanımladık toplam 2 adet toprağımız var ve kamera açısından çıktıktan sonra ileri atılıyor. GROUND\_Y\_OFFSET değişkenini değiştirerek toprağın yüksekliğini ayarlayabiliyoruz ben -80 yaptım güzel görünüyor. Render methodunda toprağımızı sürekli çizme işlemini gerçeklerştirdik updateGround metodu ise render methodunda çizilen konumları sürekli olark güncelleme işini yapıyor kameradan çıktığında toprak yeni toprağı kameranın birazdan göreceği kısımda yeniden oluşturuyor.



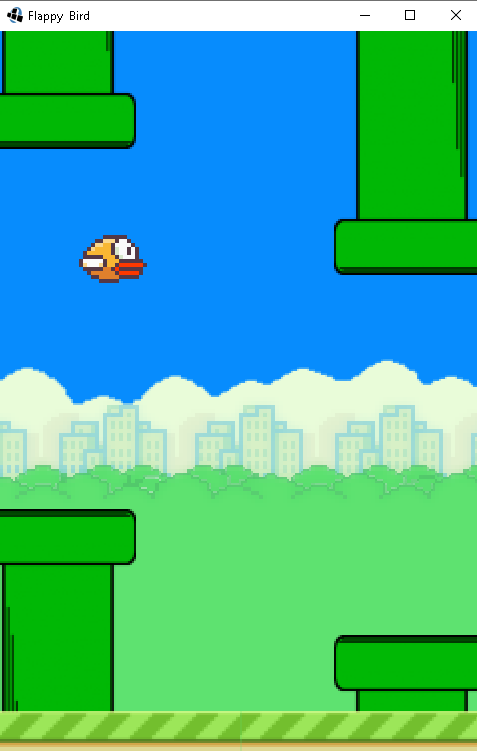
Anlık olarak oyunun görüntüsü bu şekilde



Şimdi de kuşumuz kanatlarını çırpma animasyonu verelim bunun için öncelikle internetten uygun boyutlarda yan yana animasyon şekli belirtilen bir dosya indirmeliyiz bu işlemden sonra Animation classını oluşturuyoruz ve içeriğini bu şekilde ayarlıyouz Artık Texture değil TextureRegion kullanacağız ve resim üzerinden bölgeleri bize sırası ile gösterecek bu class ta mevcut resmi bölme işlemi gerçekleştirdik.



Texture bird olan değişkeni sildik ve yerine Texture texture değişkenini koyduk mevcut sayfa da bird olan kısımları texture olarak değiştiriyoruz. Ardından yeni oluşturduğumuz Animation class ından bir nesne oluşturuyoruz ve önceden belirttiğimiz parametlreleri içine giriyoruz. Bizim animasyonumuz 3 resimden oluşuyor ve hızı 0.5 saniye de bir değişmek üzere yapıldı. Update methodumuzda Animation klasında yazdığımız update methodunu çağırıyoruz.

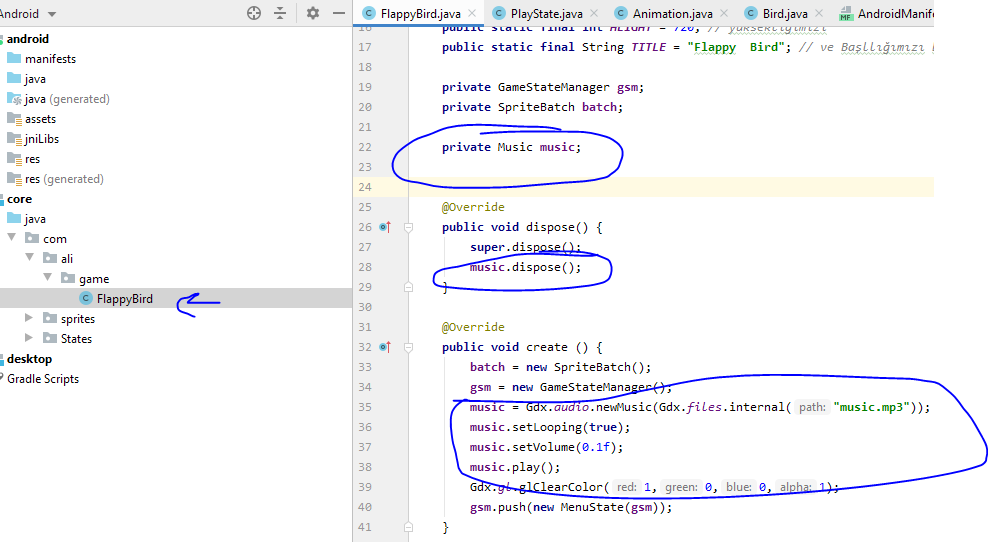


(Oyunun anlık görüntüsü)

Resimde animasyon görüntülenmediği için kısa bir video çekip youtube’a yükledim

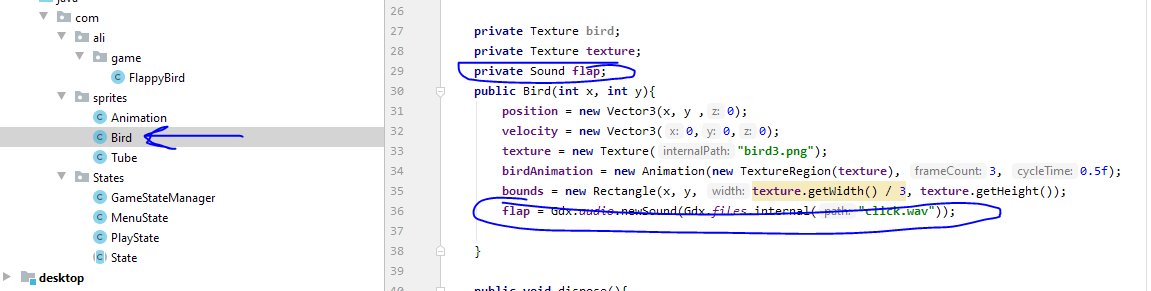
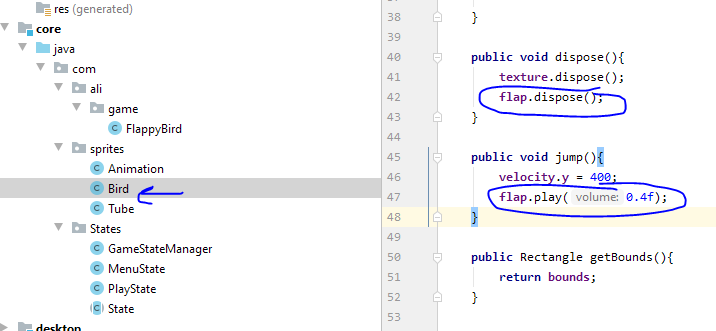
<https://youtu.be/r9IMeli8BIU>

linki tarayıcınızda açarak animasyonu görebilirsiniz.

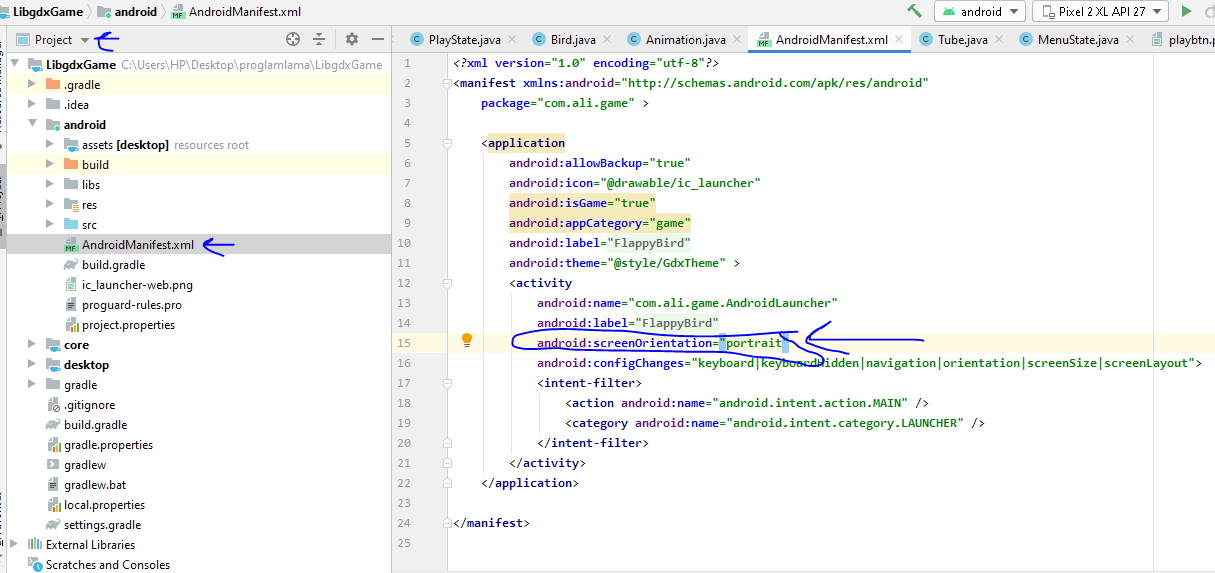
Şimdi oyunumuza ses ekleyelim

Muzik dosyası büyük olduğundan dolayı bu işlemi Ana methodumuzda yapıyoruz

Müziğimizi ekledikten sonra ses efekti ekleyelim.

Ses dosyası küçük boyutlu oldğu için onu ram belekte tutabiliriz. Veri tipimizi belirtip constructor methodunda nesnemizi yaratıyoruz.

Her zıplama metodu çalıştığında ses efektimizi veriyoruz ve dispose etmeyi unutmuyoruz.

Şimdi oyunumuzu android telefonumuza nasıl uygun hale getireceğimizi görelim.

Android manifest dosyasında Screen orientation kısmına portrait yazıyoruz çünkü oyunumuz dikey oynanıyor.

Ardnıdan telefonunuza yükleyebilirsiniz.

Kodlara github’ımdan ulaşabilirsiniz.

https://github.com/justBeHonest/FlappyBird

İyi Oyunlar…