Space Cowboy

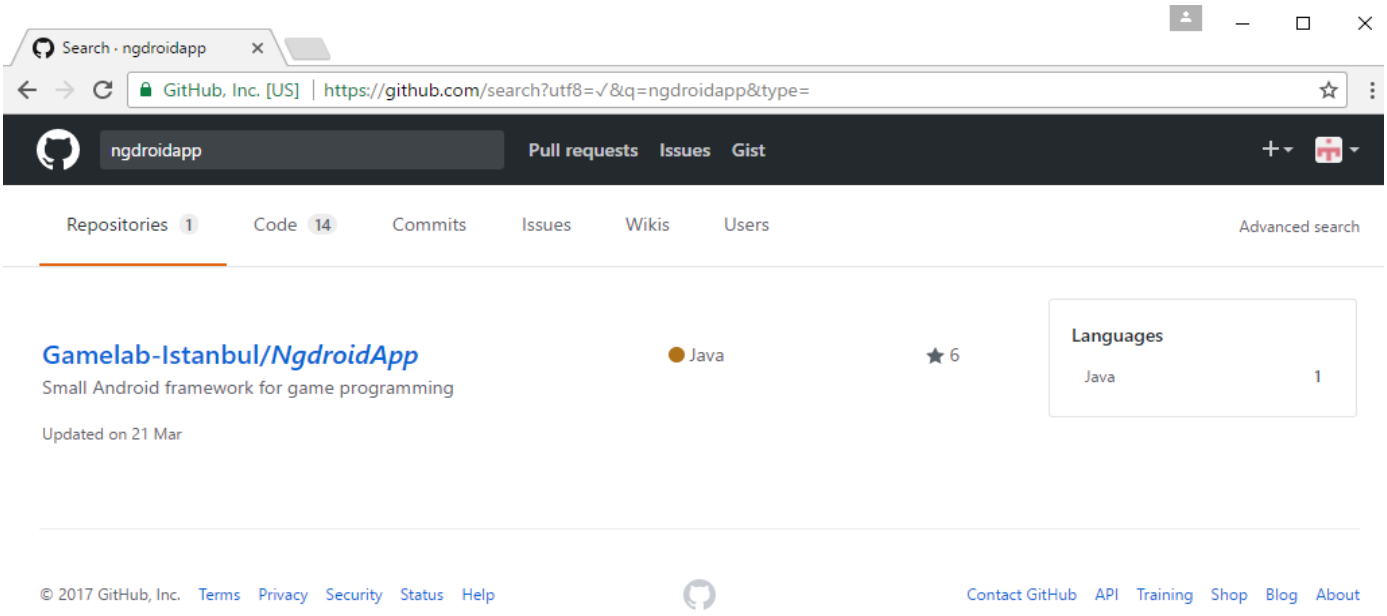
(Android Studio'da 2 boyutlu mobil oyun yazılımı eğitimi)

İçindekiler:

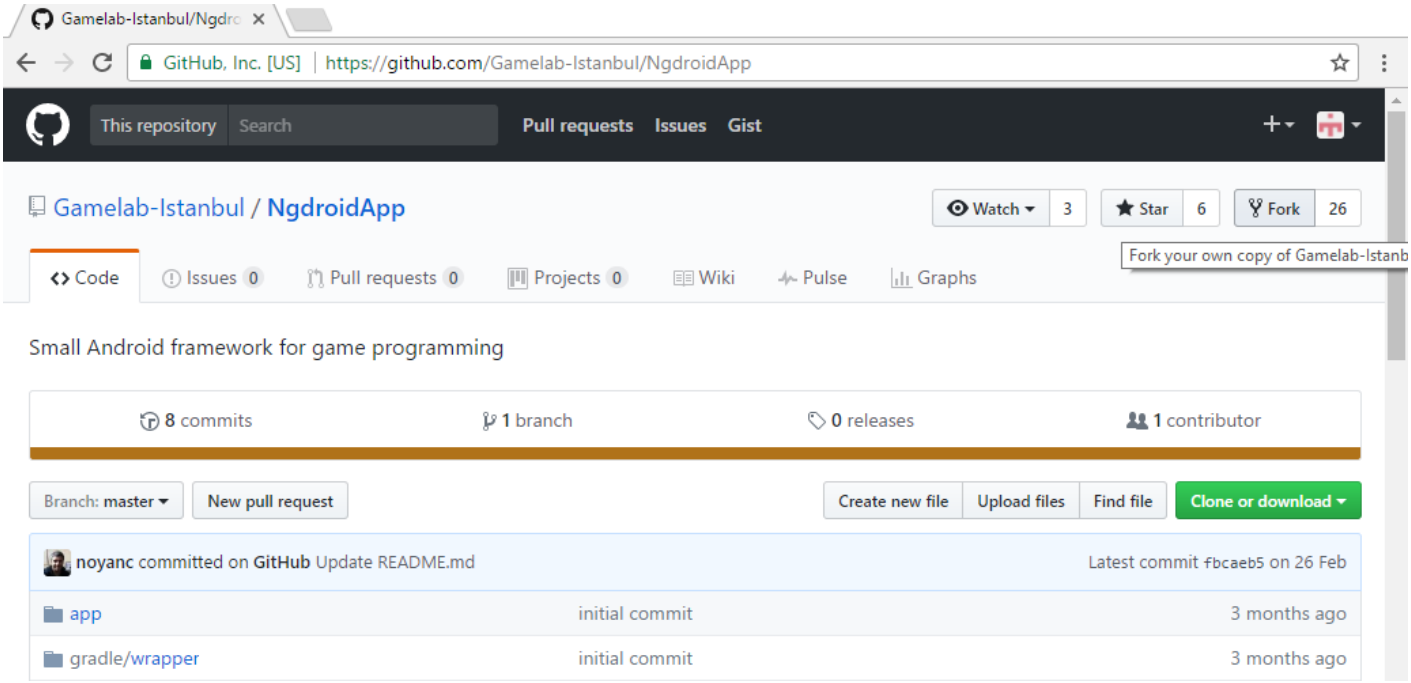
1- Gereken kurulumlar, GitHub'dan oyunun framework'unun indirilmesi (NgdroidApp), update/draw metodunun çalıştırılarak uygulamanın kontrol edilmesi
2- Animasyonun ekrana gösterilmesi
3- Animasyona kullanıcı kontrollerinin eklenmesi (User interaction)
4- Ateş etme, çarpışma ve patlama (çok boyutlu dizi, collision detection)
5- Menü ve ses (GUI Tasarımı)

1 – Hazırlık

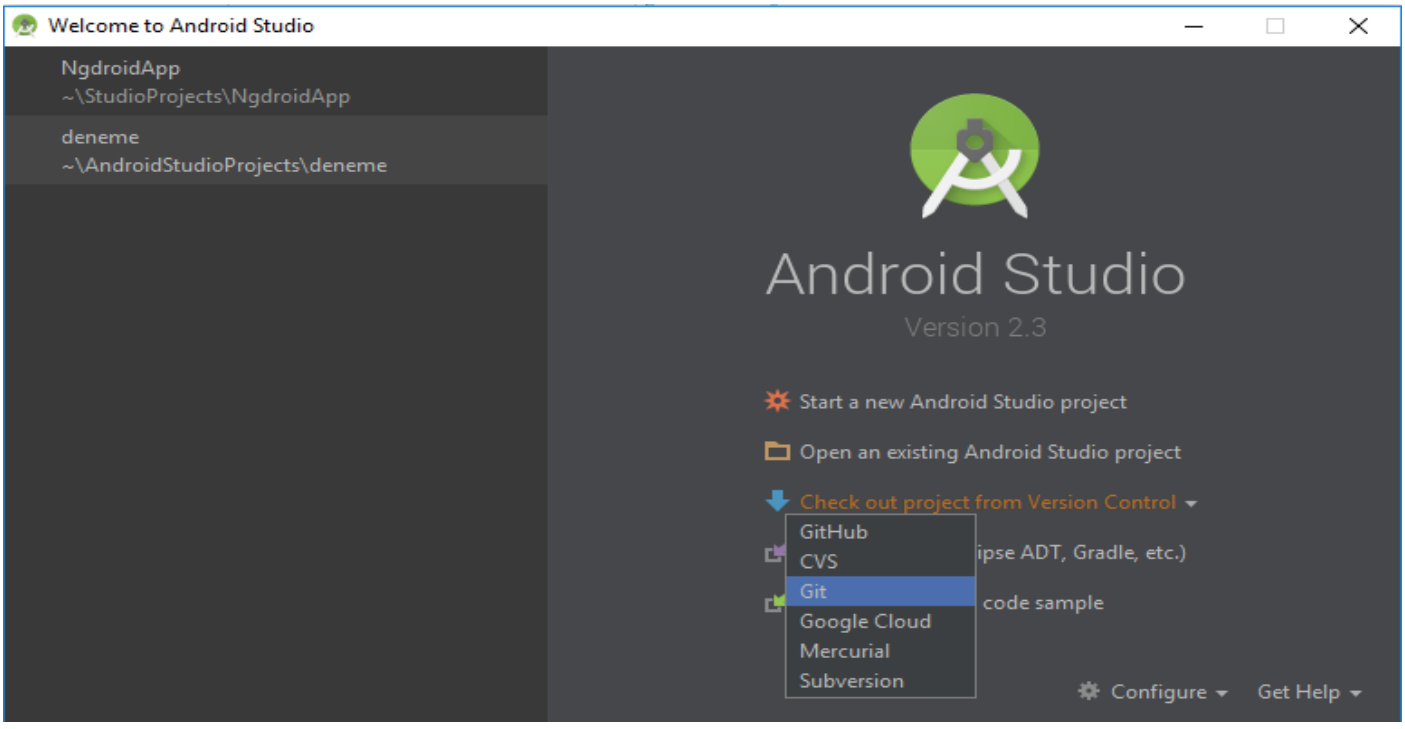
Programın çalışabilmesi için Android Studio'nun, Android SDK'nın, Android Sanal Aygıtının (Virtual Device) ve Git uygulamasının kurulu olması gerekir. Programı indirmek için de kendinize ait GitHub hesabınızdan projeyi fork etmeniz gereklidir.

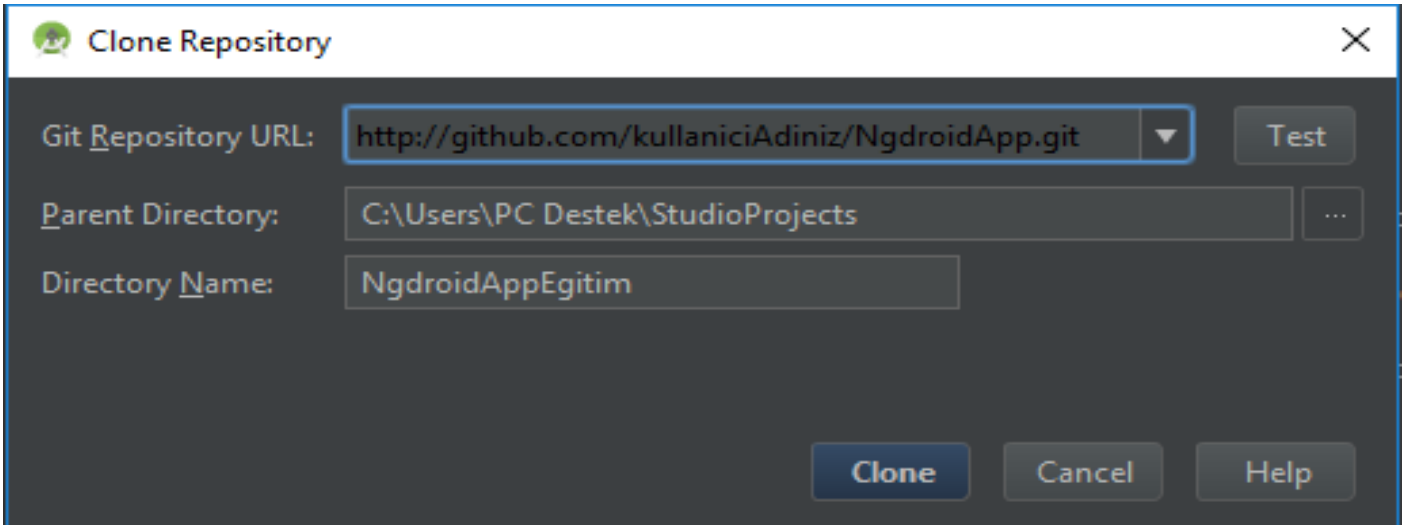


Gamelab–Istanbul / NgdroidApp adresinden projeyi fork edebilirsiniz.

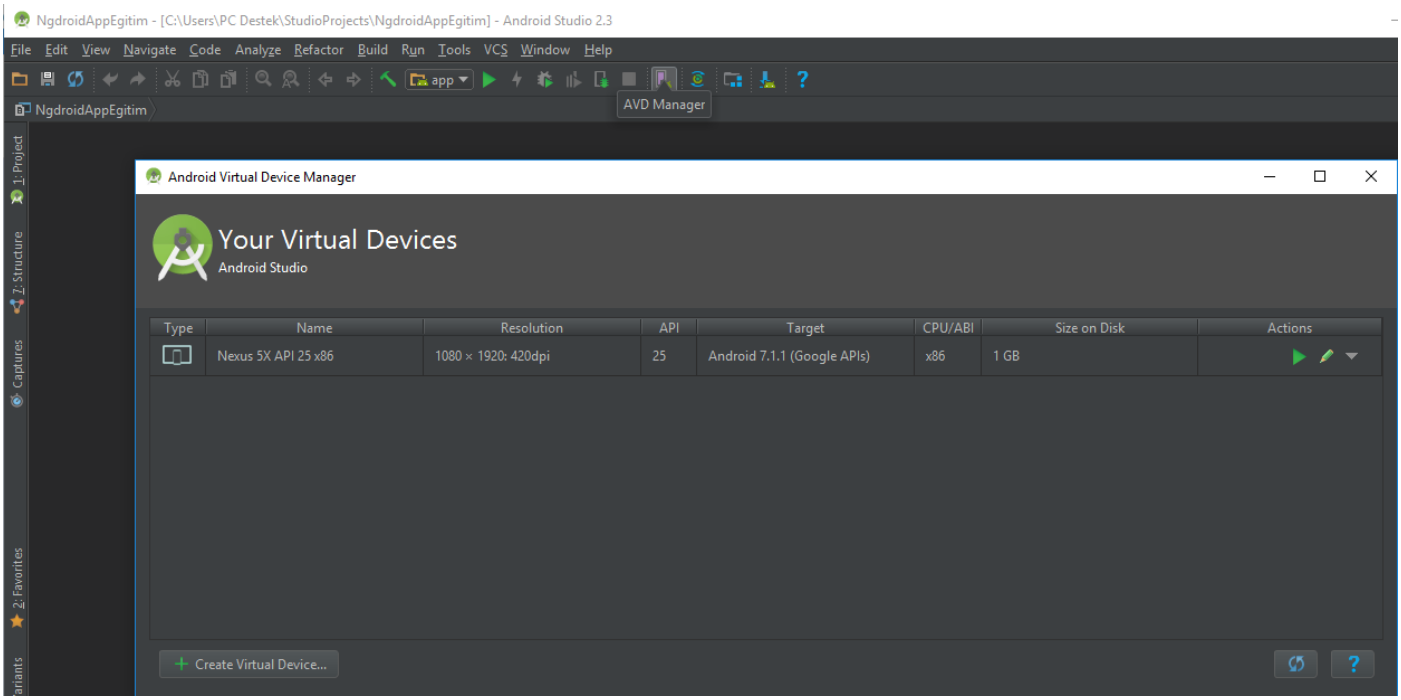


Android Studio'yu çalıştırdığınızda artık kendi git hesabınızı kullanarak projeyi indirebilir/klonlayabilirsiniz.

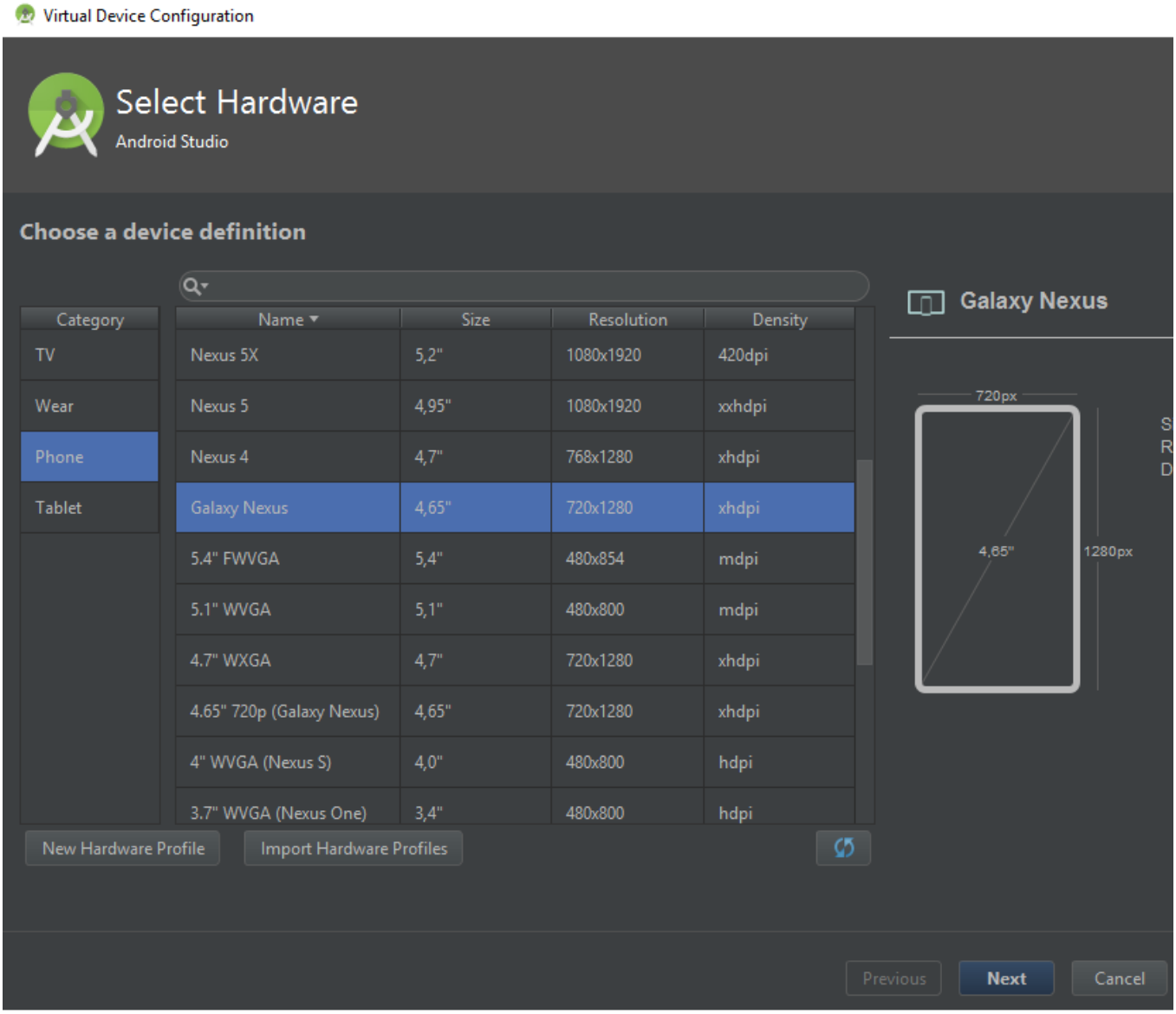


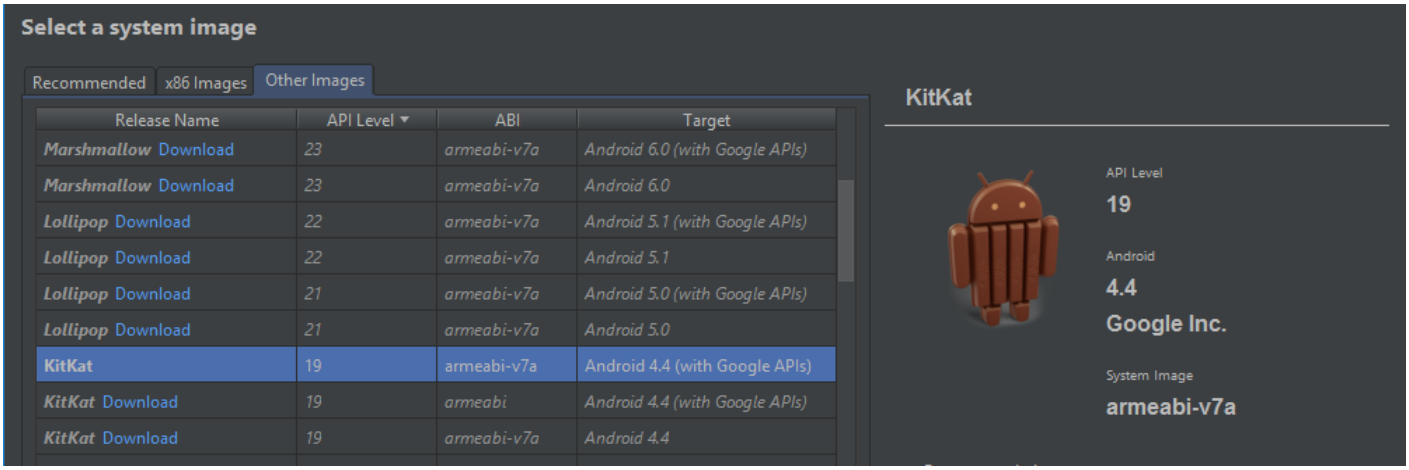


Projeyi açtığınızda AVD Manager'den (Android Virtual Device) kullanacağınız sanal aygıtı seçmelisiniz.

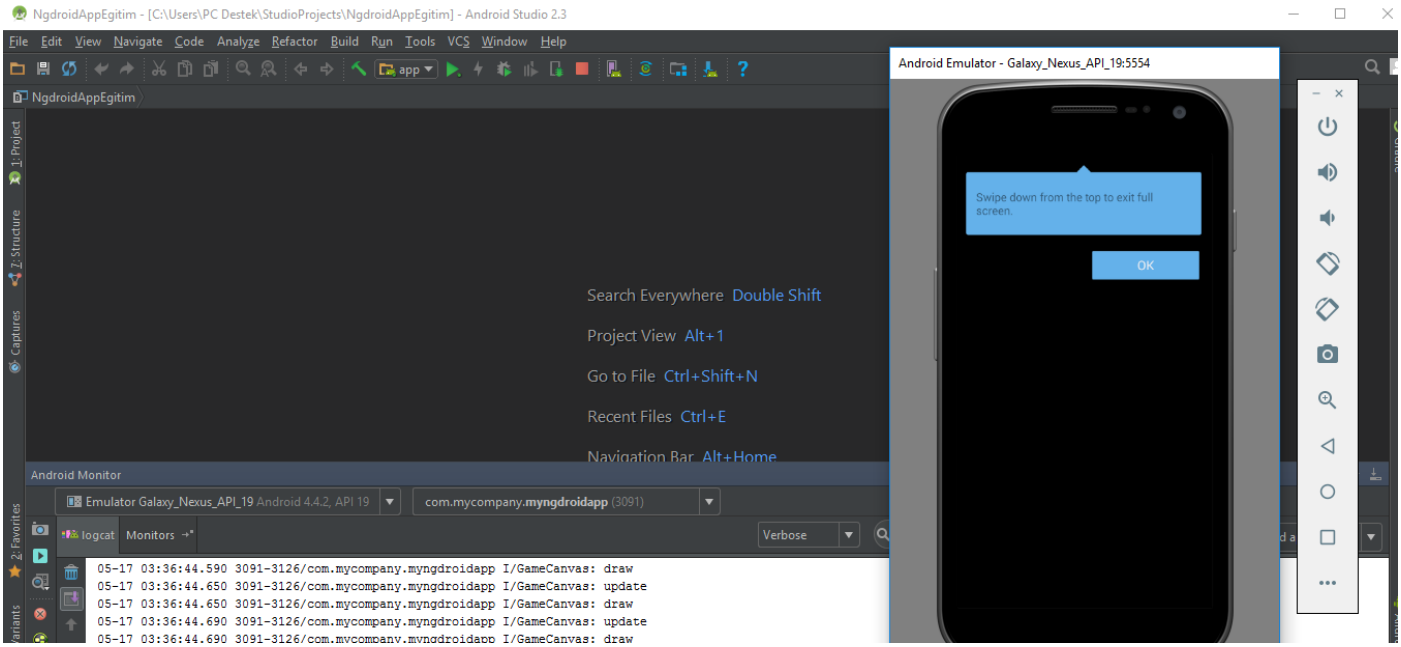


Genelde tüm bilgisayarlara uyumlu olduğundan aygıt tercihini Galaxy Nexus, System Images tercihini Kitkat 19 armeabi-v7a android 4.4 (with google APIs) yapıyoruz.



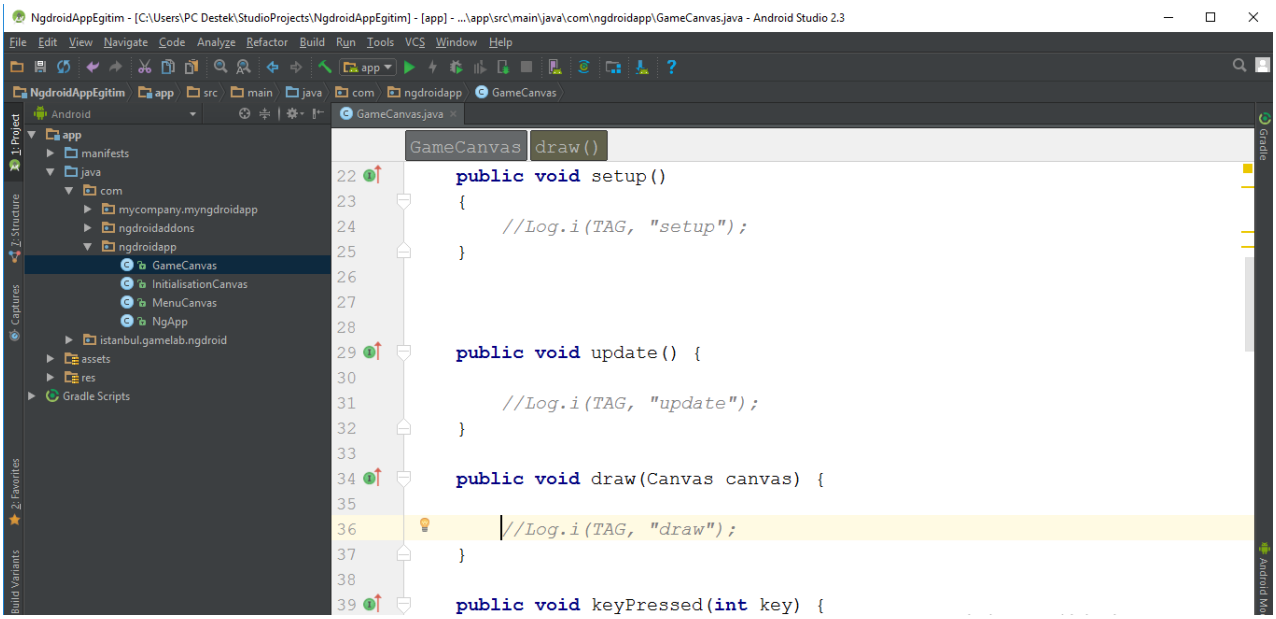


Projeyi çalıştırdığınızda Android Monitor'da ard arda update draw satırlarını görüyorsanız projeniz artık kullanıma hazır haldedir.



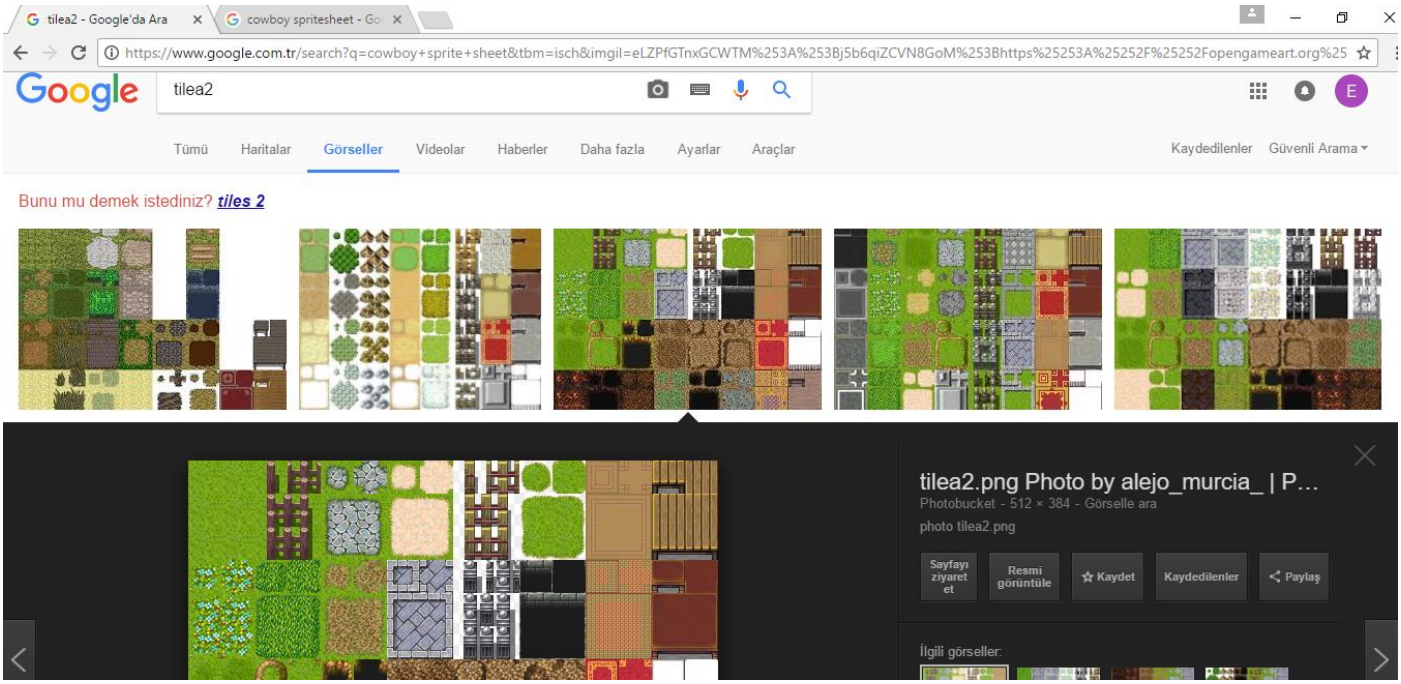
2 – Animasyonun Ekrana Gösterilmesi

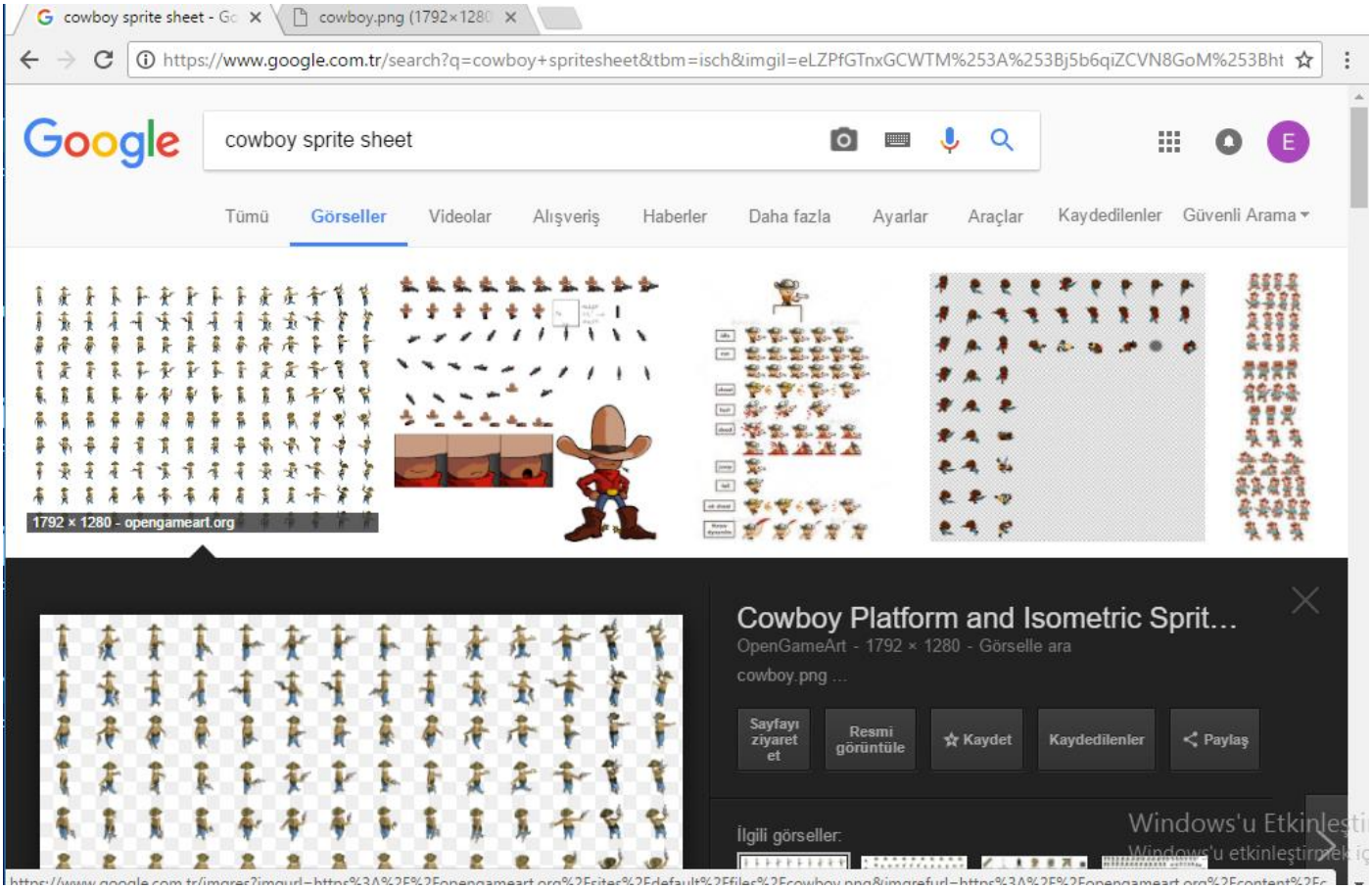
Projede GameCanvas.java sayfasını açalım.



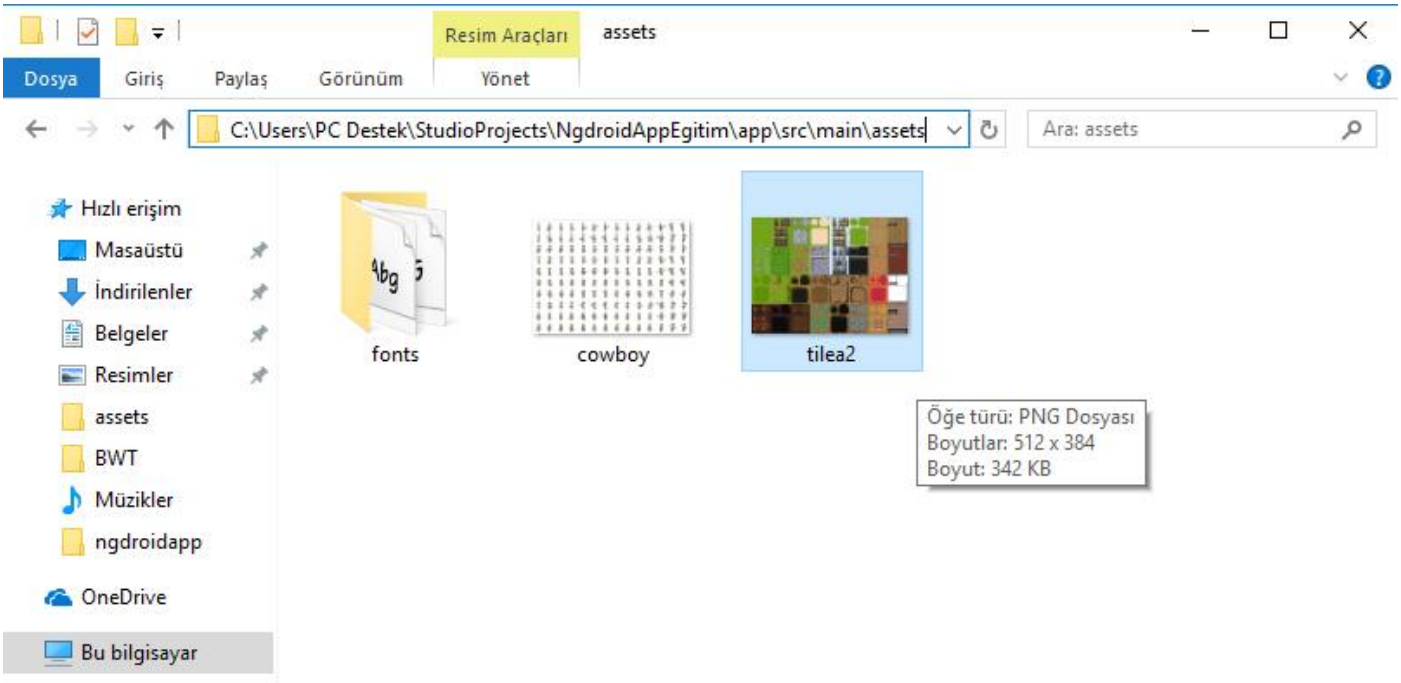
Burada çizdirme, güncelleme, dokunma gibi metotlar bulunur. Setup() metodu initialization işlemi için kullanılır ve yalnız oyun açıldığı an çalışır. Update() metodu oyun içerisindeki değişkenlerin güncellenmesi için, draw() metodu ise ekrana çizdirme işlemleri için kullanılır. Update ve draw metotları sürekli çalışırlar. Setup(), update() ve draw() metotlarındaki Log.i("TAG", ...) fonksiyonlarına şu anda ihtiyaç duyulmadığından comment out edebilir ve ya kaldırabilirsiniz.

Öncelikle ekrana çizdireceğimiz zemin için bir tileset, kontrol edeceğimiz karakter için de bir spritesheet indirmeliyiz. Tileset için "tilea2.png", karakterimiz için "cowboy.png" spritesheet'ini kullanabiliriz.





İndirdiğimiz dosyaları projemizin assets klasörüne yerleştirdikten sonra ekrana gösterme işlemlerine başlayabiliriz.



2.1 Zeminin Ekranda Gösterilmesi

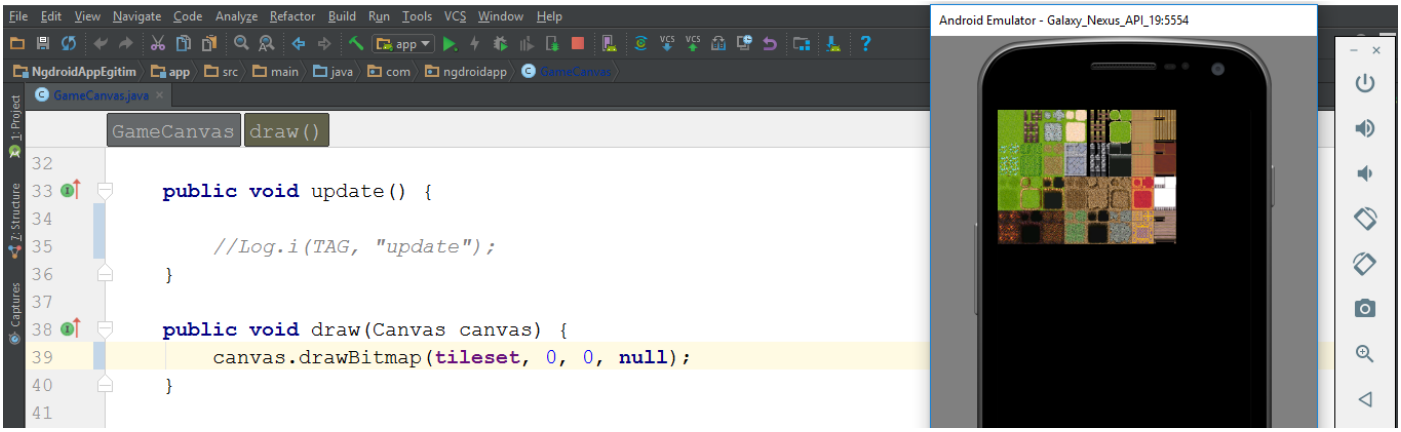
Gösterme işlemine ekrana zemin çizdirerek başlayalım. Sayfanın başına “tileset” isminde yeni bir Bitmap nesnesi oluşturalım.

```
GameCanvas.java
GameCanvas

14
15
16  public class GameCanvas extends BaseCanvas {
17
18      private Bitmap tileset;
19
20
21      public GameCanvas(NgApp ngApp) { super(ngApp); }
24
25      public void setup()
```

Ardından setup() metoduna gidip tileset’e kullanacağımız zemin dosyası tilea2.png yi atalım.

```
19      private Bitmap tileset;
20
21
22      public GameCanvas(NgApp ngApp) { super(ngApp); }
25
26      public void setup()
27      {
28          tileset = Utils.loadImage(root, "tilea2.png");
29      }
30
```

Artık canvas.drawBitmap() ile tilesetimizi ekrana çizdirebiliriz.



Görüldüğü gibi tilesetimizi herhangi bir kırpma ve ya yerleştirme işlemi yapmadan öylece ekrana vermiş olduk. Daha karmaşık çizdirme işlemlerine başlamadan önce yapmamız gereken ilk şey, üzerinde çalışacağımız görüntüyü olduğu gibi ekranda göstermektir. Çalışmalarımıza da bu ilkeyi benimseyerek, yani basitten karmaşığa doğru ilerleyerek devam edeceğiz.

Tileseti ekranda sorunsuz bir şekilde çizdirebildiğimize göre artık bir zemin görüntüsü belirleyip ekranda sadece bu görüntünün çizilmesini sağlayabiliriz. Önce projemizin assets klasöründe bulunan, tileset olarak kullandığımız “tilea2.png” dosyamızı inceleyelim.

Dikkat edersek her bir zemin tipinin iki tane 32 x 32 px lik kareden ve bir 64 x 64 px lik kareden oluştuğunu görebiliriz. Bu tarz bir tileset, zeminleri 2’nin kuvvetinden oluşan alanlarda tutulduğundan Bitmap adını alır. Şimdi bir zemin tercih edip bu zeminin sol üst köşesinin ve sağ alt köşesinin koordinatlarını alalım.

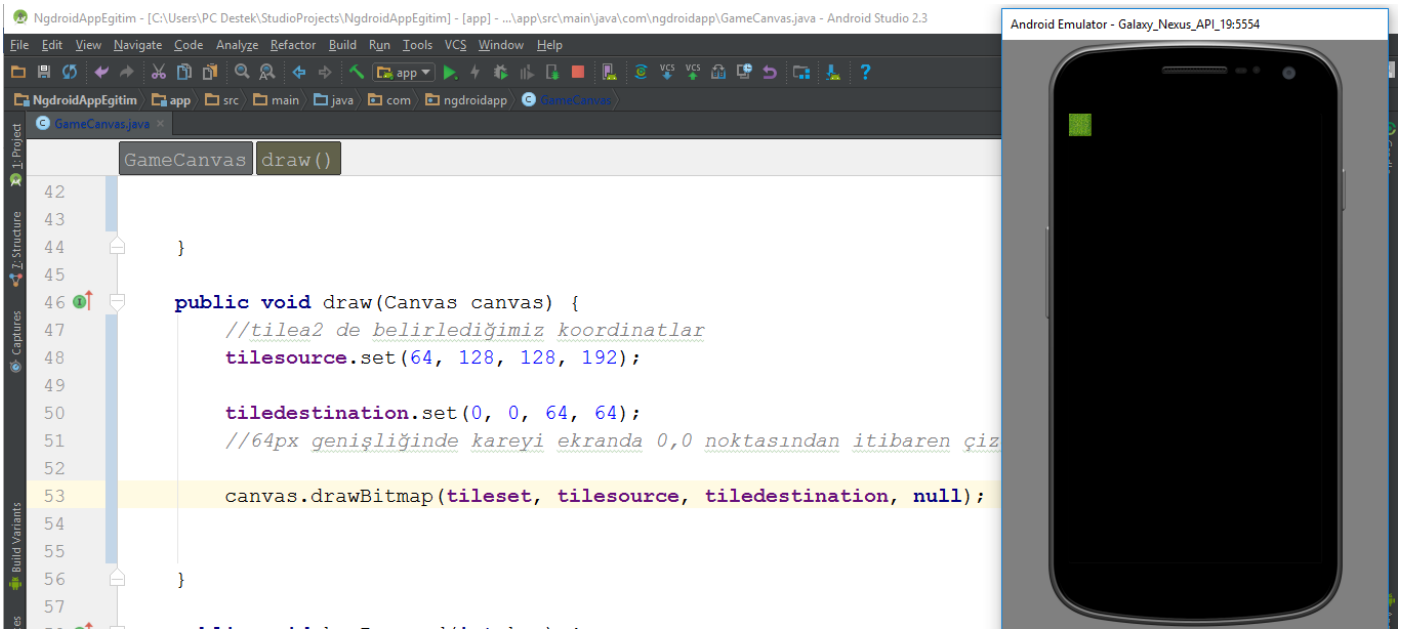
Başlangıç olarak 64x64px lik çimen zemini kullanalım. Bu kare 64, 128 noktasından başlayıp 127, 191 noktasında bitiyor. Bu koordinatları projemizde kare nesnesi oluşturmak için kullanacağız.

Sayfamızın başına gidip “tilesource” ve “tiledestination” isimli kareleri tanımlayalım. Sonra da setup() metodunda bunlardan yeni nesne oluşturalım.

```
public class GameCanvas extends BaseCanvas {

    private Bitmap tileset;
    private Rect tilesource, tiledestination;



    public GameCanvas(NgApp ngApp) { super(ngApp); }

    public void setup()
    {
        tileset = Utils.loadImage(root, "tilea2.png");
        tilesource = new Rect();
        tiledestination = new Rect();
```

tilesource ye kullanacağımız zeminin tilea2 deki koordinatlarını yerleştireceğiz (x1,y1 x2,y2). Tiledestination'ı da seçtiğimiz zeminin ekranda gösterileceği yeri ve boyutlarını belirlemek için kullanacağız. Şimdi draw() metoduna gidip bu karelerin değerlerini belirleyelim ve zemini ekranın sol üst köşesinde gösterelim

```
public void draw(Canvas canvas) {
    //tilea2 de belirlediğimiz koordinatlar
    tilesource.set(64, 128, 128, 192);

    tiledestination.set(0, 0, 64, 64);
    //64px genişliğinde kareyi ekranda 0,0 noktasından itibaren çizdiriyoruz

    canvas.drawBitmap(tileset, tilesource, tiledestination, null);
```



Zemini ekrana bu şekilde çizdirebildiğimize göre artık koordinatlarımızı değişkenlere atayıp bu değişkenler üzerinden çalışabiliriz. Şimdi tilesource ve tiledestination karelerinin başlangıç noktalarını, genişliklerini ve yüksekliklerini tanımlayalım.

```
20      private Bitmap tileset;
21      private Rect tilesource, tiledestination;
22
23      //tilesource genişlik, yükseklik, tiledestination genişlik, yükseklik
24      private int tilesrcw, tilesrch, tiledstw, tiledsth;
25
26      private int tilesrcx, tilesrcy, tiledstx, tiledsty;
27
```

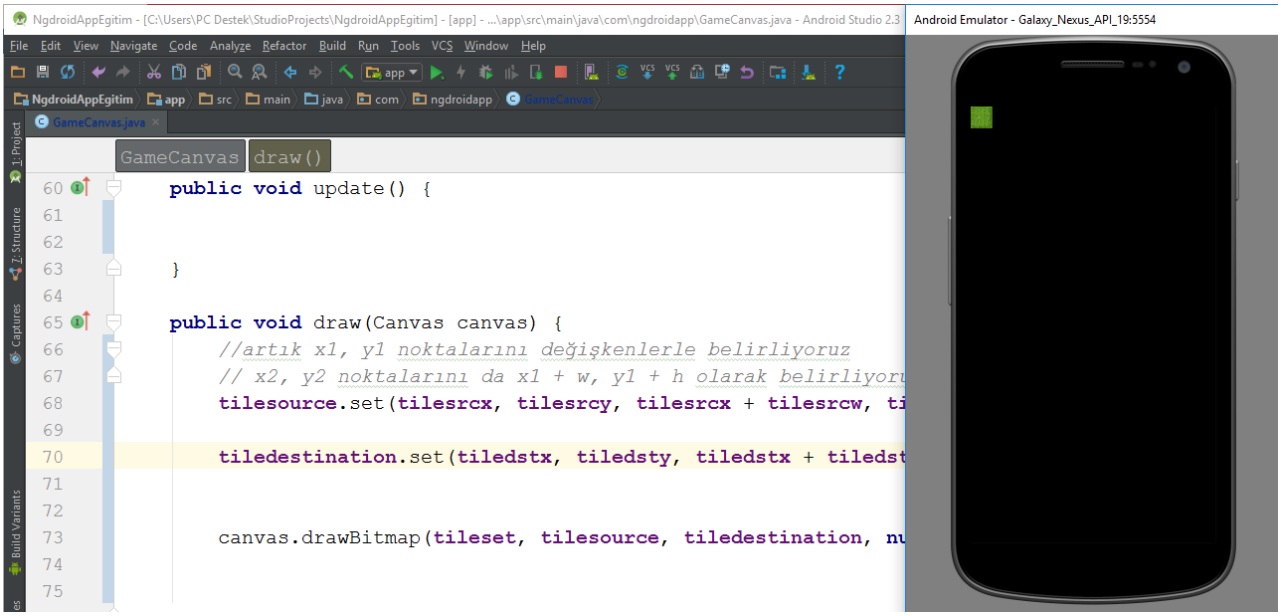
Sonra da setup() metodunun içerisinde değişkenlerimize başlangıç değerlerini girelim.

```
40          //zeminin genişliğini/yüksekliğini tilea2'de 64 bulmuştuk
41          tilesrcw = 64;
42          tilesrch = 64;
43          //zeminin başlangıç noktasını tilea2'de 64, 128 bulmuştuk
44          tilesrcx = 64;
45          tilesrcy = 128;
46
47          //ekranda zeminin yerleştirildiği karenin genişliği/yüksekliği
48          tiledstw = 64;
49          tiledsth = 64;
50          //karenin başlangıç noktaları, sol üst köşe
51          tiledstx = 0;
52          tiledsty = 0;
```

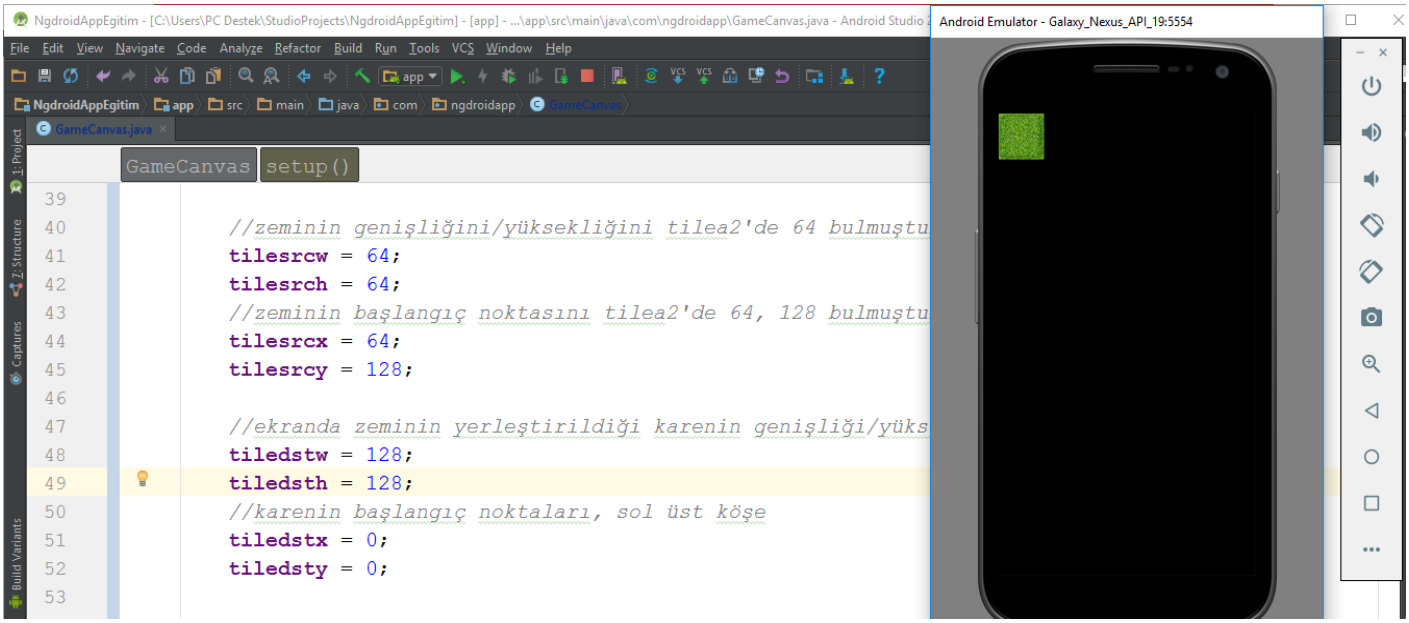
Ardından draw() metodunun içerisinde tilesource ve tiledestination karelerini set ettiğimiz yere gidip önceden kullandığımız koordinatları değişkenler cinsinden girelim.

```
65      public void draw(Canvas canvas) {
66          //artık x1, y1 noktalarını değişkenlerle belirliyoruz
67          // x2, y2 noktalarını da x1 + w, y1 + h olarak belirliyoruz
68          tilesource.set(tilesrcx, tilesrcy, tilesrcx + tilesrcw, tilesrcy + tilesrch);
69
70          tiledestination.set(tiledstx, tiledsty, tiledstx + tiledstw, tiledsty + tiledsth);
71
72
73          canvas.drawBitmap(tileset, tilesource, tiledestination, null);
```

Projeyi çalıştırdığımızda zemini yine aynı yerde göreceğiz.

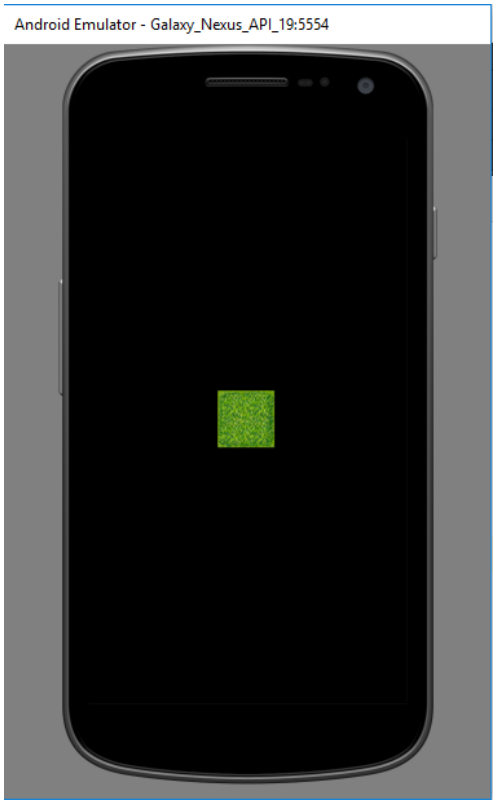


İstersek tiledstw ve tiledsth değerlerini değiştirerek zeminin çizildiği karenin boyutlarını artırabiliriz. Örnek olarak setup() metodunun içerisine gidip tiledstw ve tiledsth değişkenlerinin ilk değerlerini 128 yapalım



Şimdi de karenin çizildiği yeri değiştirelim. Örnek olarak zemini ekranın tam orta noktasına çizdirelim.

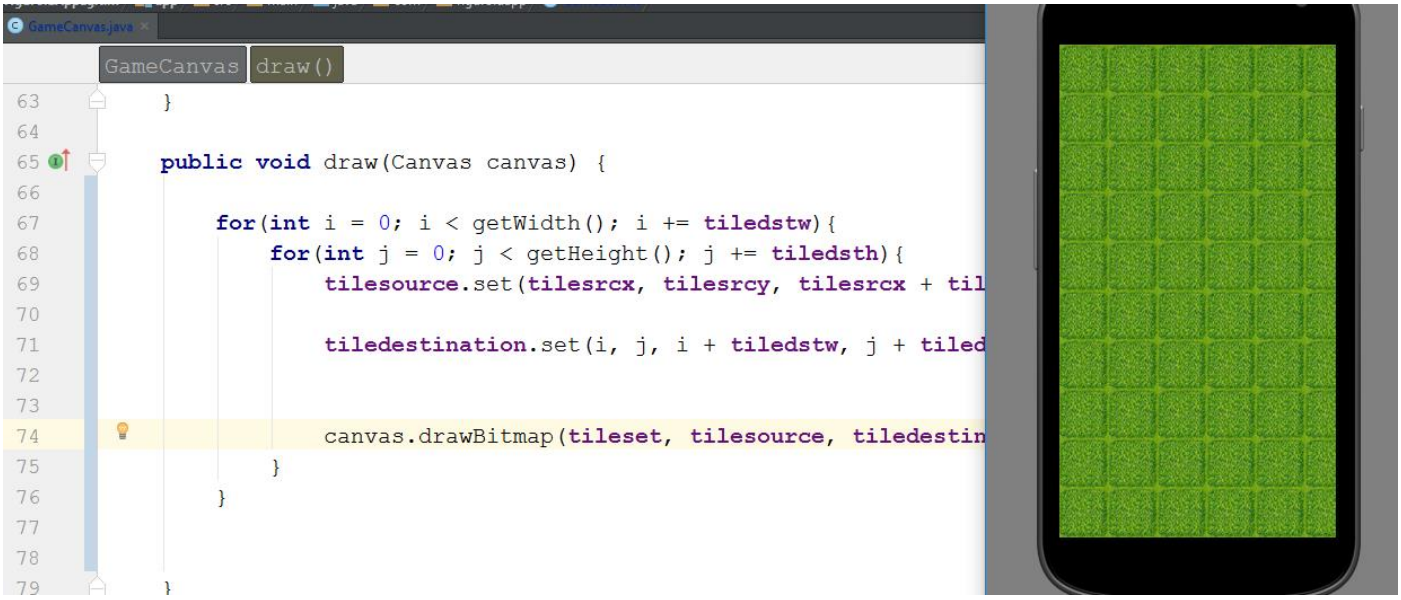
```
64
65    public void draw(Canvas canvas) {
66
67
68        tiledstx = (getWidth() - tiledstw) / 2;
69        tiledsty = (getHeight() - tiledsth) / 2;
```



getWidth() ekranın genişliğini, getHeight() ekranın yüksekliğini verir. Ekranın orta noktasının x koordinatını getWidth() / 2 ile, y koordinatını ise getHeight() / 2 ile elde ediyoruz. tiledstx ve tiledsty değerlerine doğrudan (getWidth() / 2) ve (getHeight() / 2) değerlerini verseydik karenin sol üst köşesi, ekranın tam ortasında kalacağından kare, orta noktadan biraz ileride çizilecekti. Kareyi tam ortalayabilmek için
tiledstx = (getWidth / 2) – (tiledstw / 2) ,
tiledsty = (getHeight / 2) – (tiledsth / 2) yaptık. Yukarıda ise bu işlemlerin sadeleşmiş halini yazdık.

Şimdi ekranın tamamını seçtiğimiz zeminle dolduralım. Bu işlemi kareleri uç uca ekleyerek yapacağız.

```
65    public void draw(Canvas canvas) {
66
67        for(int i = 0; i < getWidth(); i += tiledstw){
68            for(int j = 0; j < getHeight(); j += tiledsth){
69                tilesource.set(tilesrcx, tilesrcy, tilesrcx + tilesrcw, tilesrcy + tilesrch);
70
71                tiledestination.set(i, j, i + tiledstw, j + tiledsth);
72
73
74                canvas.drawBitmap(tileset, tilesource, tiledestination, null);
75            }
76        }
```



2.2 Karakterin Ekranda Gösterilmesi

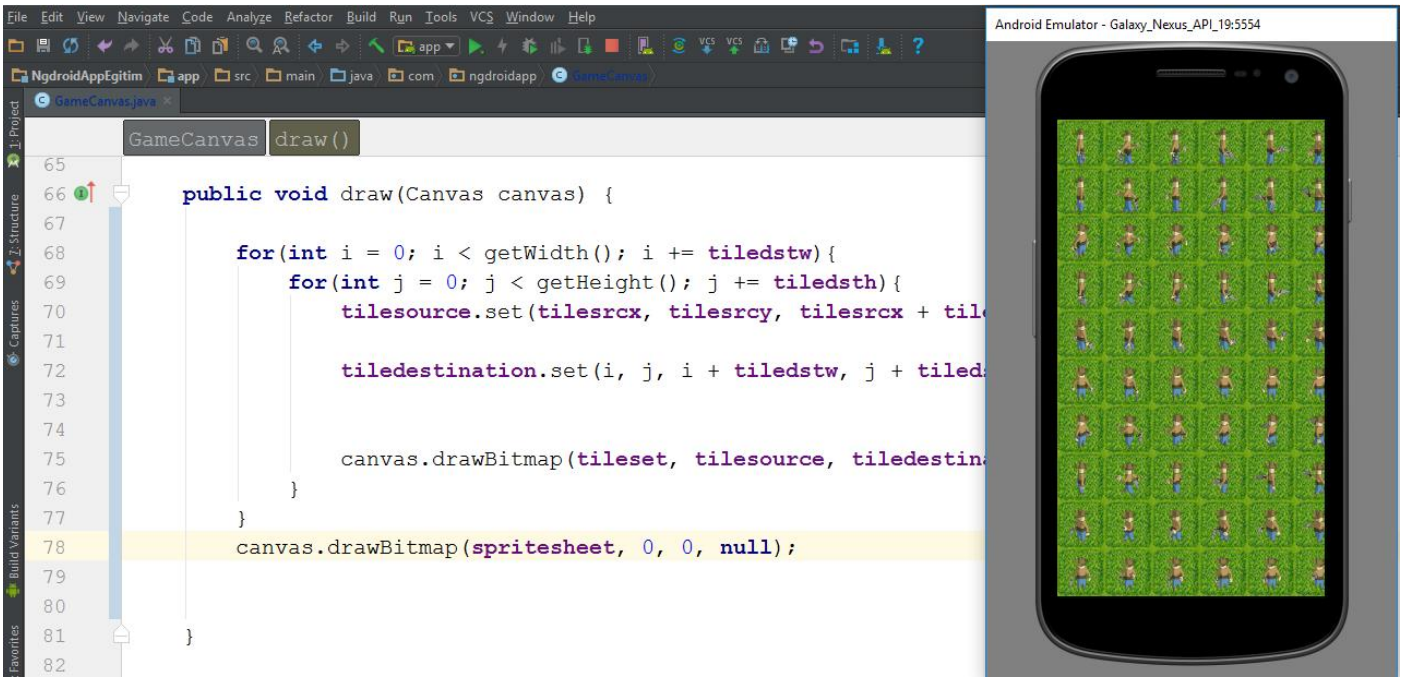
Artık oyun karakteri ile ilgili işlemlere başlayabiliriz. Önce “spritesheet” adında yeni bir bitmap daha oluşturup setup() metodunun içerisinde “cowboy.png” dosyasını spritesheet’e atayalım.

```
18  public class GameCanvas extends BaseCanvas {
19
20      private Bitmap tileset, spritesheet;
21      private Rect tilesource, tiledestination;
```

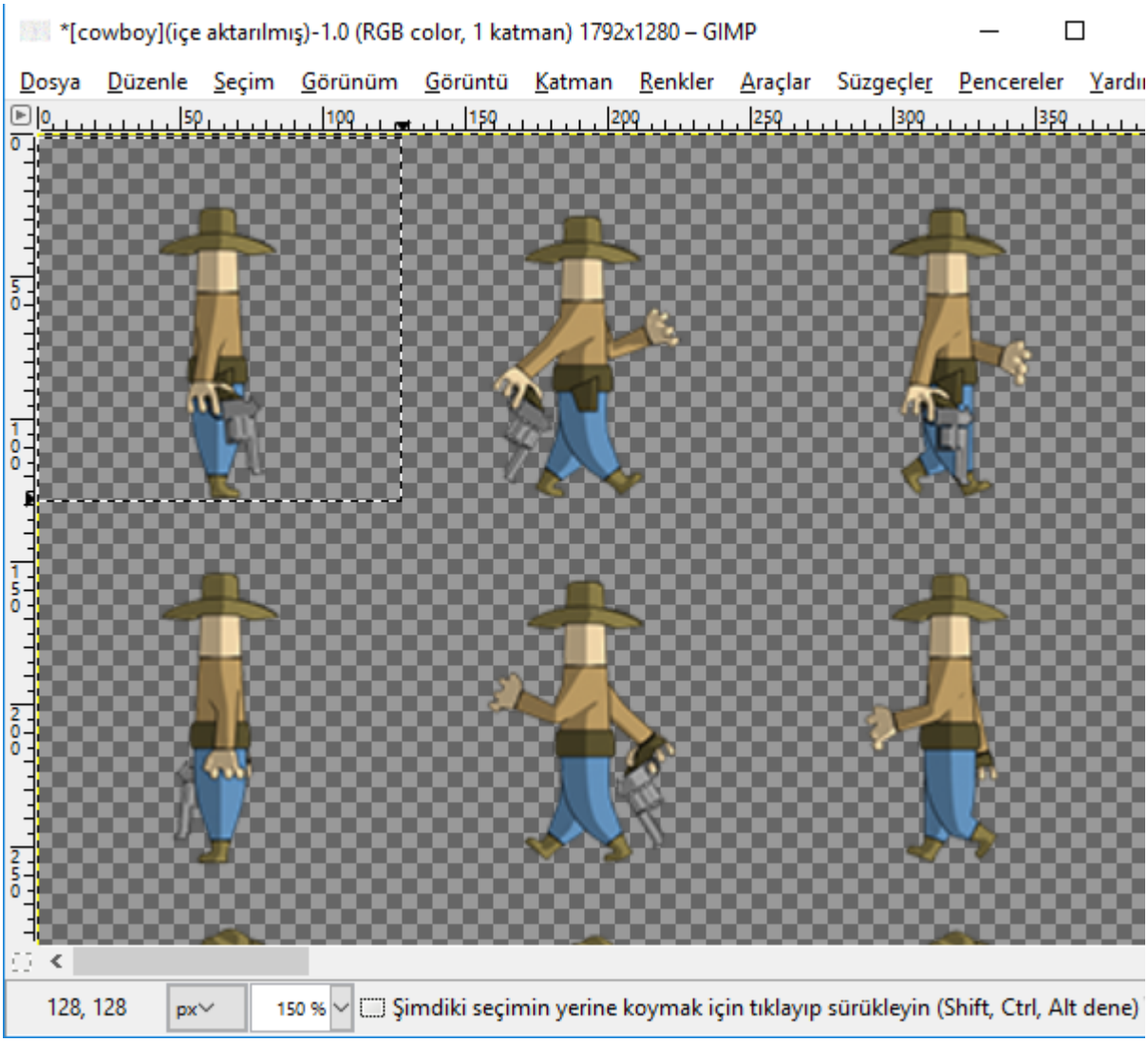
```
34      public void setup()
35      {
36          tileset = Utils.loadImage(root, "tilea2.png");
37          spritesheet = Utils.loadImage(root, "cowboy.png");
```

Spritesheet’in düzgün çalıştığını anlamak için önce ekranda gösterelim.

```
66  public void draw(Canvas canvas) {
67
68      for(int i = 0; i < getWidth(); i += tiledstw){
69          for(int j = 0; j < getHeight(); j += tiledsth){
70              tilesource.set(tilesrcx, tilesrcy, tilesrcx + tilesrcw, tilesrcy + tilesrch);
71
72              tiledestination.set(i, j, i + tiledstw, j + tiledsth);
73
74
75              canvas.drawBitmap(tileset, tilesource, tiledestination, null);
76          }
77      }
78      canvas.drawBitmap(spritesheet, 0, 0, null);
```



Spritesheet'i ekranda gösterdikten sonra kırpma ve yerleştirme işlemlerine başlayabiliriz. Önce cowboy.png dosyasını inceleyelim.



Dikkat edersek her bir animasyon karesinin 128x128px'den oluştuğunu görebiliriz. Karakterimizi spritesheet'ten alırken bu değeri kullanacağız.

Şimdi spritesheet'in sol üst köşesindeki duran karakter karesini ekranda gösterelim. Zemin çizdirirken yaptığımız gibi karakter için de kare nesneleri oluşturacağız. Sayfanın başında "spritesource" ve "spritedestination" isimli kareleri oluşturalım. Sonra da setup() metodunun içerisinde bunlardan nesne oluşturalım.

```
18  public class GameCanvas extends BaseCanvas {
19
20      private Bitmap tileset, spritesheet;
21      private Rect tilesource, tiledestination, spritesource, spritedestination;
22
```

```
36      public void setup()
37      {
38          tileset = Utils.loadImage(root, "tilea2.png");
39          spritesheet = Utils.loadImage(root, "cowboy.png");
40
41          tilesource = new Rect();
42          tiledestination = new Rect();
43
44          spritesource = new Rect();
45          spritedestination = new Rect();
46
```

Çizdirme işlemine başlamadan önce bu kareler için genişlik / yükseklik tanımlayalım.

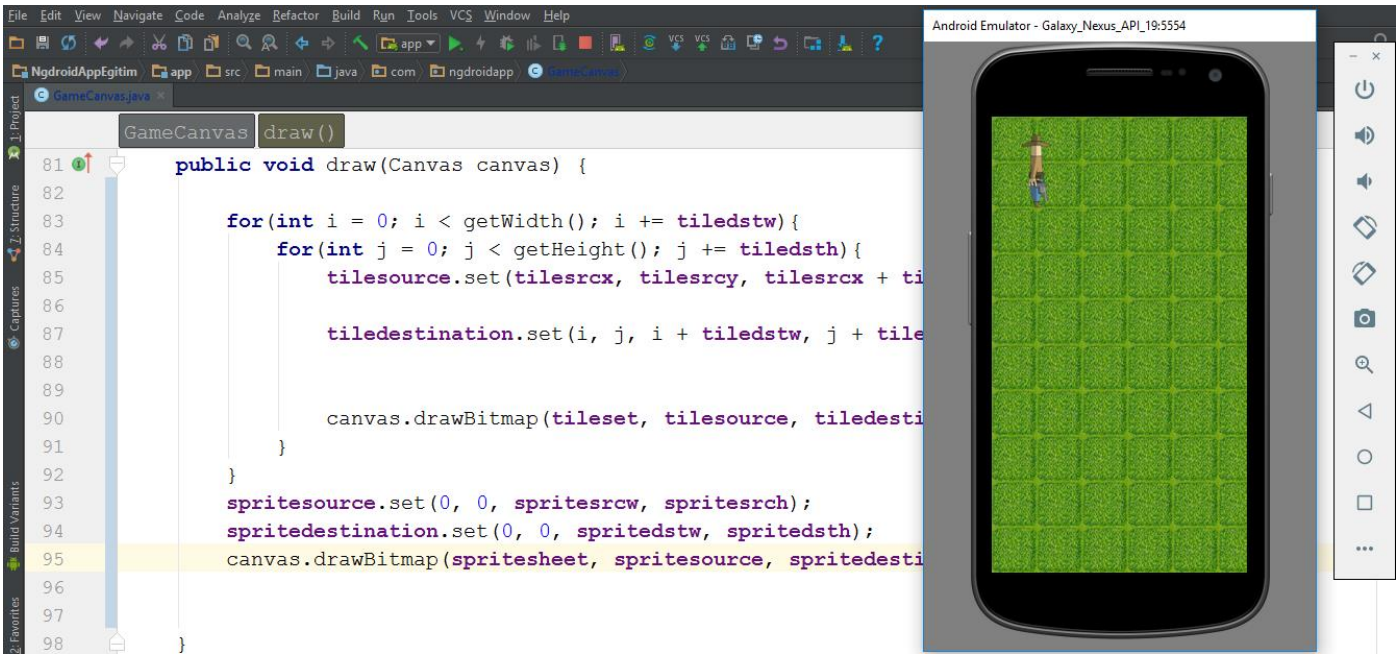
```
18  public class GameCanvas extends BaseCanvas {
19
20      private Bitmap tileset, spritesheet;
21      private Rect tilesource, tiledestination, spritesource, spritedestination;
22
23      //tilesource genişlik, yükseklik, tiledestination genişlik, yükseklik
24      private int tilesrcw, tilesrch, tiledstw, tiledsth;
25
26      private int tilesrcx, tilesrcy, tiledstx, tiledsty;
27
28      //spritesource genişlik / yükseklik, spritedestination genişlik / yükseklik
29      private int spritesrcw, spritesrch, spritedstw, spritedsth;
30
```

Tanımladığımız genişliklere setup() metodunda ilk değerlerini verelim.

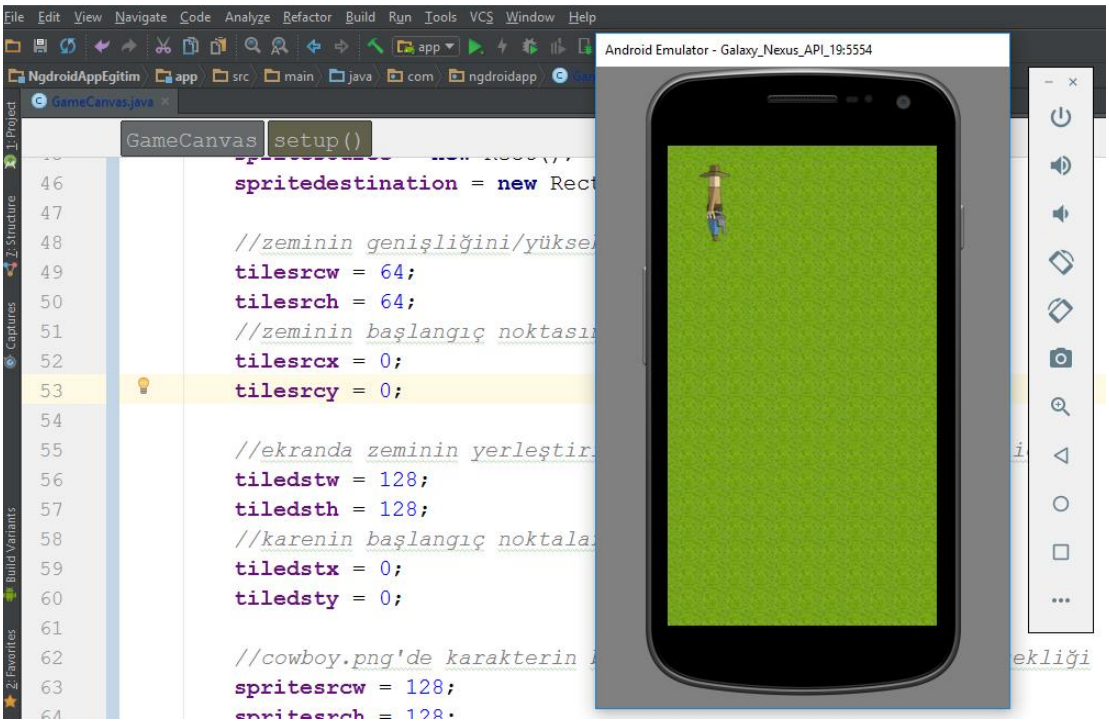
```
62          //cowboy.png'de karakterin bulunduğu karenin genişliği/yüksekliği
63          spritesrcw = 128;
64          spritesrch = 128;
65
66          //ekranda karakterin yerleştirildiği karenin genişliği/yüksekliği
67          spritedstw = 256;
68          spritedsth = 256;
```

Artık spritesource ve spritedestination karelerinin değerlerini set edip canvas.drawBitmap() ile çizdirme işlemine başlayabiliriz.

```
81  public void draw(Canvas canvas) {
82
83      for(int i = 0; i < getWidth(); i += tiledstw){
84          for(int j = 0; j < getHeight(); j += tiledsth){
85              tilesource.set(tilesrcx, tilesrcy, tilesrcx + tilesrcw, tilesrcy + tilesrch);
86
87              tiledestination.set(i, j, i + tiledstw, j + tiledsth);
88
89
90              canvas.drawBitmap(tileset, tilesource, tiledestination, null);
91          }
92      }
93      spritesource.set(0, 0, spritesrcw, spritesrch);
94      spritedestination.set(0, 0, spritedstw, spritedsth);
95      canvas.drawBitmap(spritesheet, spritesource, spritedestination, null);
96
97
98  }
```



Karakter bu zeminde zor göründüğünden daha iyi bir zemin seçelim. Setup() metodunda tilesrcx ve tilesrcy değerlerini 0, 0 yapalım.



2.3 Karakterin yer değiştirmesi

Karakterimizin çizildiği karenin sol üst köşesi için gereken değişkenleri oluşturalım.

```
30    private int spritedstx, spritedsty;
```

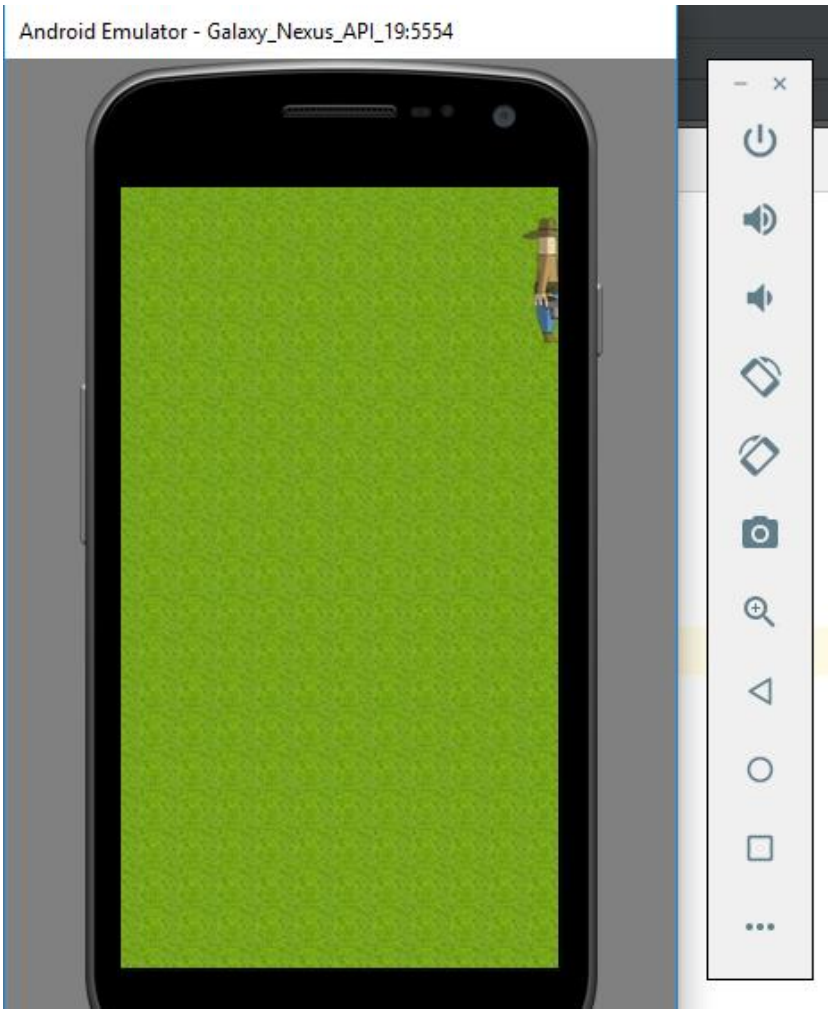
Setup() metodunun içerisinde bu değişkenlere ilk değerlerini verelim

```
75        //karakteri ekranda çizdirdiğimiz karenin sol üst köşesinin koordinatları
76        spritedstx = 0;
77        spritedsty = 0;
```

Animasyon, ard arda gelen görüntülerden oluşur. Her görüntüleme anında* spritedstx ve spritedsty değerlerini değiştirerek karakterimizin yer değiştirmesini sağlayabiliriz. update() ve draw() metotları her görüntü anında çalıştığından yaptığımız değişikliği update() metodunun içerisine yerleştirebiliriz. Şimdi karakterimizi sağa doğru ilerletmeyi deneyelim. * frame

```
85    public void update() {
86        spritedstx += 32;
```

Projeyi çalıştırdığımızda karakterin sağa doğru ilerleyip ekranın dışına çıktığını göreceğiz.

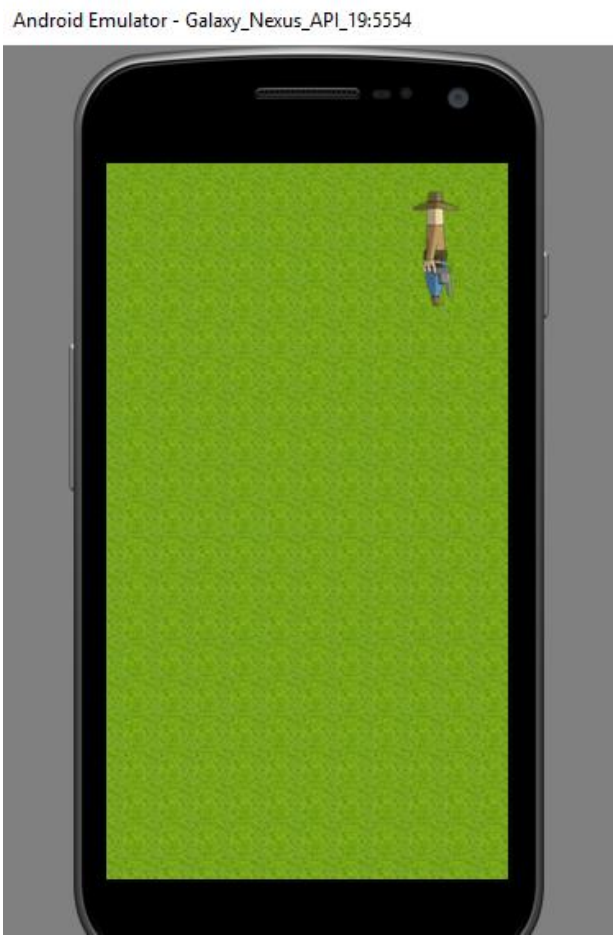


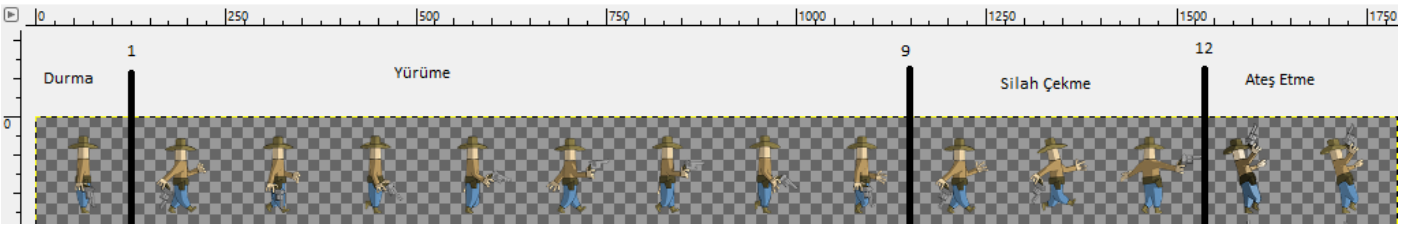
Karakterin ekran dışına çıkmasını engellemek için bir koşul oluşturalım.

```
85      public void update() {
86          spritedstx += 32;
87          if(spritedstx >= getWidth() - spritedstw){
88              spritedstx = getWidth() - spritedstw;
89          }
90      }
```

Böylece karakterimiz ekranın sonuna geldiğinde duracaktır.

Şimdiye kadar karakterimiz yürüme animasyonu yapmadan yer değiştiriyordu. Yürüme görüntüsünü oluşturmak için cowboy.png dosyasında ihtiyacımız olan animasyon karelerini belirledikten sonra spritesource'da bu karelerin sırayla girilmesini sağlayacağız.





Cowboy.png'ye baktığımızda soldan 0. karenin durma, 1' den 8'e kadar olan karelerin yürüme animasyonu olduğunu görebiliriz. Kaçıncı karede olduğumuzu "frameno" değişkeni ile belirleyelim. Frameno ile kare genişliğini çarparsak karenin cowboy.png'deki x koordinatını elde ederiz. Bu koordinatı spritesource.set() içerisinde kullanmadan önce gerekli değişkenleri oluşturalım.

```
29      private int spritesrcx, spritesrcy, spritedstx, spritedsty;
```

Artık spritesource'a dinamik değerler vereceğimiz için cowboy.png den aldığımız koordinatları değişkenlere atamalıyız.

```
31      //animasyon türleri: durma, yürüme, silah doğrultma, ateş etme
32      private int animationtypes = 4;
33
34      //cowboy.png'deki kare numarası, animasyon türü 0 1 2 3, kare numarası alt ve üst sınır.
35      private int framenum, animationtype, animationfirstframenum[], animationlastframenum[];
```

Setup() metodunda başlangıç değerlerini verelim

```
76          //cowboy.png'de seçtiğimiz karenin sol üst köşesinin koordinatları
77          spritesrcx = 0;
78          spritesrcy = 0;
```

```
87      //cowboy.png de soldan sağa kaçıncı karede olduğumuzu gösterir 0, 1, ..., 13
88      framenum = 0;
89
90      //animasyon türü: durma = 0, yürüme = 1, silah doğrultma = 2, ateş etme = 3
91      animationtype = 1;
```

```
93          //animasyon türleri için alt / üst sınır dizisi
94          //kare numarasını (frameno) bu sınırlar arasında artıracağız
95          animationfirstframenum = new int[animationtypes];
96          animationlastframenum = new int[animationtypes];
97
98          //durma (0)animasyonu 0. karede başlar & biter.
99          animationfirstframenum[0] = 0;
100         animationlastframenum[0] = 0;
101
102         //yürüme (1) animasyonu 1. karede başlar, 8. kerede biter.
103         animationfirstframenum[1] = 1;
104         animationlastframenum[1] = 8;
105
```

```
106         //silah çekme (2) animasyonu 9. karede başlar, 11. karede biter.
107         animationfirstframenum[2] = 9;
108         animationlastframenum[2] = 11;
109
110         //ateş etme animasyonu 12. karede başlar, 13. karede biter.
111         animationfirstframenum[3] = 12;
112         animationlastframenum[3] = 13;
```

Şimdi cowboy.png'deki yürüme animasyonunu update() metodunun içerisinde çalıştıralım

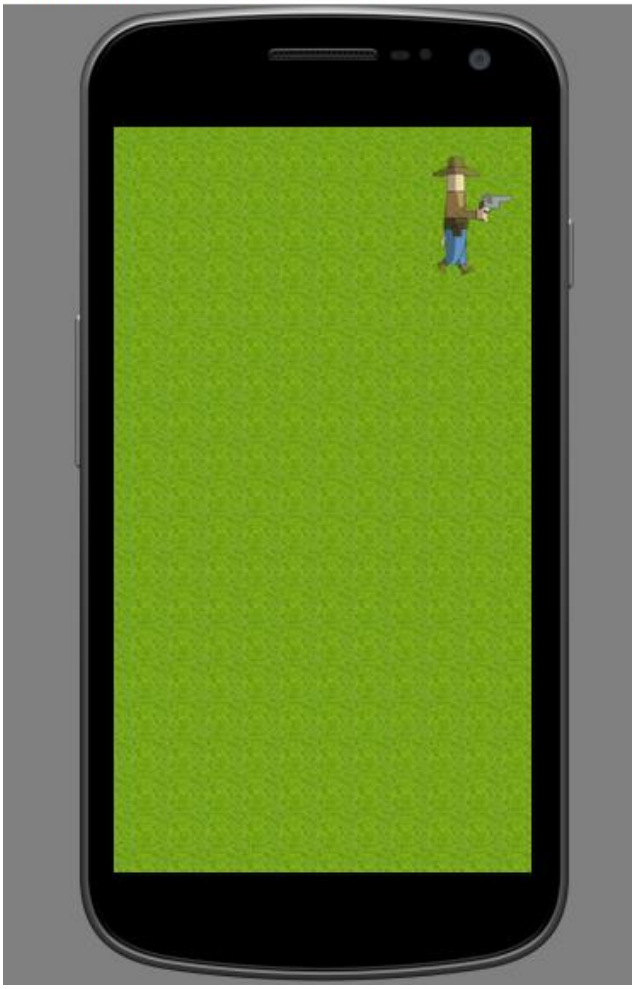
```
117     public void update() {
118         //update her çalıştığında animasyon bir sonraki kareye geçer
119         framenum++;
120
121         //animasyon son karesine ulaştığında framenum'u animasyonun ilk karesine getirir
122         if(framenum > animationlastframenum[animationtype]){
123             framenum = animationfirstframenum[animationtype];
124         }
125
126         spritedstx += 32;
127         if(spritedstx >= getWidth() - spritedstw){
128             spritedstx = getWidth() - spritedstw;
129
130         }
131         //cowboy.png'deki animasyon karesinin koordinatını framenum ve genişlik cinsinden girelim
132         spritesrcx = framenum * spritesrcw;
```

spritesrc değerlerini draw() metodunda spritesource.set() içerisine yerleştirelim.

Karakter ekranın sonuna ulaştığı halde yürümeye devam ettiğini göreceğiz. Update() metodunda karakterin ekranın dışına çıkmasını engellediğimiz yere gidip buraya durma animasyonunu da ekleyelim.

```
148         spritesource.set(spritesrcx, spritesrcy, spritesrcx +  spritesrcw, spritesrcy + spritesrch);
149         spritedestination.set(spritedstx, spritedsty, spritedstx + spritedstw, spritedsty + spriteds
150         canvas.drawBitmap(spritesheet, spritesource, spritedestination, null);
```

Android Emulator - Galaxy_Nexus_API_19:5554



```
126         spritedstx += 32;
127         if(spritedstx >= getWidth() - spritedstw){
128             spritedstx = getWidth() - spritedstw;
129             //karakter ekranın soluna ulaştığında durma animasyonuna geçilir
130             animationtype = 0;
131         }
```

3. Animasyona Kullanıcı Kontrollerinin Eklenmesi (User Interaction)

Karakterimizin gideceği yönü belirlemek için x ve y eksenlerinde ilerleme hızları tanımlayalım.

```
30      //sprite hız x/y, sprite işaret x/y
31      private int spritevx, spritevy, spriteix, spriteiy;
```

Setup()'ta başlangıç değerlerini girelim

```
86          spritevx = spritedstw / 8;
87          spritevy = spritedsth / 8;
88
89          spriteix = 1;
90          spriteiy = 0;
```

Update() metodunda spritedstx değerine ekleme yaparak ilerlemesini sağlıyorduk. Şimdi bu değeri hız ve işaret cinsine çevirelim.

```
135         spritedstx += spritevx * spriteix;
136         spritedsty += spritevy * spriteiy;
```

Ekrana dokunduğumuz noktadan elimizi kaldırdığımız noktaya yön göstererek karakterimizin gideceği yönü belirleyeceğiz. Önce ekrana dokunduğumuz nokta için değişkenler tanımlayalım.

```
33      //ekrana dokunmaya başadığımız noktanın x/y koordinatları
34      private int touchdownx, touchdowny;
```

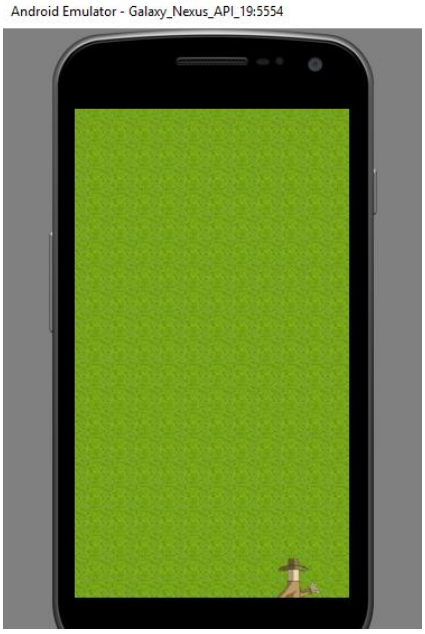
touchDown() metodunun içerisinde, ekrana dokunmaya başladığımız anın x/y değerleri kaydedilir. Buradaki x/y değerlerini değişkenlerimize verelim.

```
278     public void touchDown(int x, int y) {
279         touchdownx = x;
280         touchdowny = y;
281     }
```

touchUp() metodunda da elimizi ekrandan çektiğimiz anın x/y değerleri kaydedilir. Bu değerler ile touchdown değişkenleri arasındaki farkı bularak karakterimizin gideceği yönü belirleyeceğiz.

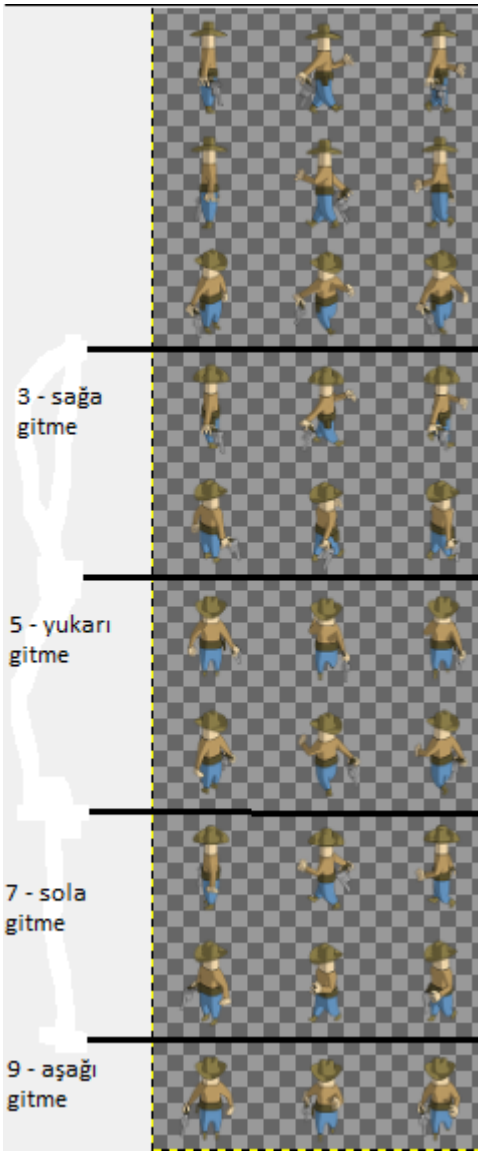
```
202     public void touchUp(int x, int y) {
203         int xfarki = x - touchdownx;
204         int yfarki = y - touchdowny;
205
206         //xfarkı büyükse sağa/sola, yfarki büyükse yukarı/aşağı gitsin
207         if(Math.abs(xfarki) >= Math.abs(yfarki)){
208             spriteix = 1;
209             if(xfarki < 0){
210                 spriteix = -1;
211             }
212             spriteiy = 0;
213         }
214         else{
215             spriteiy = 1;
216             if(yfarki < 0){
217                 spriteiy = -1;
218             }
219             spriteix = 0;
220         }
```

Ekranın diğer kenarlarında karakterimize durma koşulu oluşturmamıştık. Duvarlarda karakterin durmasını sağladıktan sonra diğer yönler için yürüme animasyonu oluşturalım.



```
141         if(spritedstx >= getWidth() - spritedstw){
142             spritedstx = getWidth() - spritedstw;
143             //karakter ekranın soluna ulaştığında durma animasyonuna geçilir
144             animationtype = 0;
145         } else if(spritedstx < 0){
146             //sol duvarda durur
147             spritedstx = 0;
148             animationtype = 0;
149         } else if(spritedsty >= getHeight() - spritedsth){
150             //alt duvarda durur
151             spritedsty = getHeight() - spritedsth;
152             animationtype = 0;
153         } else if(spritedsty < 0) {
154             //üst duvarda durur
155             spritedsty = 0;
156             animationtype = 0;
157         }
```

Şimdi cowboy.png'den diğer yönlerin animasyon karelerine bakalım.



Yukarıdan aşağıya baktığımızda 3. animasyon satırının sağa gitme, 5. satırın yukarı, 7. satırın sola, 9. satırdaki karelerin aşağı gitme animasyonları olduğunu görürüz. Yukarıdan aşağı kaçıncı karede olduğumuzu "yön" adında bir değişkene atayalım. Sonra da bu değişkeni kare genişliğiyle çarpıp spritesrcy değerine verelim.

```
36      private int yon;
```

```
124         yon = 3;
```

```
163         spritesrcx = framenum * spritesrcw;
164         spritesrcy = yon * spritesrch;
165
166     }
```

```
224         if(Math.abs(xfarki) >= Math.abs(yfarki)){
225             spriteix = 1;
226             yon = 3;
227             if(xfarki < 0){
228                 spriteix = -1;
229                 yon = 7;
230             }
231             spriteiy = 0;
232         }
233         else{
234             spriteiy = 1;
235             yon = 9;
236             if(yfarki < 0){
237                 spriteiy = -1;
238                 yon = 5;
239             }
240             spriteix = 0;
241         }
```