

# Agate Mobile Game Developer Camp

---

## Programmer Handout - Day #3

Pada kesempatan kali ini kita akan mempelajari tentang penggunaan Image dan Sprite. Ada berbagai cara untuk memanggil Sprite, yaitu:

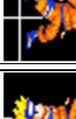
- `Sprite(Image image)`: Membuat non-animasi sprite
- `Sprite(Image image, int frameWidth, int frameHeight)`: Membuat animasi sprite
- `Sprite(Sprite s)`: Membuat sprite dari sprite yang sudah ada

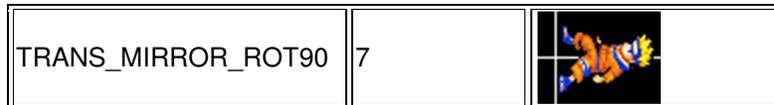
Sprite mempresentasikan visual dari sebuah object (image). Sebagai contoh teman-teman dapat membuat image sebagai Sprite.

Beberapa yang dapat dilakukan oleh Sprite yaitu:

### 1. Sprite Transformation

Untuk men-transform sebuah sprite, bisa kita lakukan dengan `setTransform()`.

Static Fields	Integer Value	Transformed Image
<code>TRANS_NONE</code>	0	
<code>TRANS_MIRROR_ROT180</code>	1	
<code>TRANS_MIRROR</code>	2	
<code>TRANS_ROT180</code>	3	
<code>TRANS_MIRROR_ROT270</code>	4	
<code>TRANS_ROT90</code>	5	
<code>TRANS_ROT270</code>	6	



Titik transform tergantung pada ReferencePixel.

## 2. Mengganti referensi pixel

ReferencePixel adalah pixel yang menjadi titik tumpu pada sebuah sprite. Bagian itulah yang menentukan titik transform dan lokasi dimana gambar akan ditaruh pada canvas. Misalkan pada source code kita `defineReferencePixel(lebarFrame/2, tinggiFrame)` maka artinya titik reference aka nada di tengah bawah dari sprite kita.

## 3. Animasi Sprite



Gambar di atas berukuran 144 x 51 px, dengan ukuran masing – masing frame 36 x 51 px.

Sehingga akan terbentuk 4 buah gambar. Urutan sequence dari sprite secara default, dihitung dari kiri ke kanan, mulai dari 0. Sehingga urutan default dari gambar di atas adalah 0, 1, 2, 3.

Jika kita gunakan `nextFrame()` maka urutan sequence framenya adalah 0, 1, 2, 3, 0, 1, 2, 3, dan seterusnya. Apabila kita ingin menentukan urutan sequence, maka kita bisa menggunakan `setFrameSequence(int[] sequence)`.

Animasi sprite dapat dilakukan dengan berbagai cara, yaitu:

- o `setFrame()`. Menentukan secara manual frame mana yang akan ditampilkan
- o `nextFrame()` dan `previousFrame()`. Secara otomatis akan bergerak framenya.

## 4. Collision Detection.

Pada kelas Sprite terdapat fungsi collision detection. Pertama kita bisa menentukan daerah collision kita dengan fungsi `defineCollisionRectangle(int x, int y, int width, int height)`. Lalu kita bisa melakukan pengecekan collision terhadap Sprite lain, TiledLayer, atau Image, dengan menggunakan syntax `collidesWith(Sprite s, Boolean boolean)`, `collidesWith(TiledLayer tile, Boolean boolean)`, atau `collidesWith(Image image, int x, int y, Boolean boolean)`.

## 5. Menggerakkan Sprite

Dengan menggunakan fungsi `setRefPixelPosition(int x, int y)` maka kita dapat menggerakkan Sprite kita sesuai dengan arah yang kita inginkan.

Penjelasan File:

### 1. Player.java

Buat sebuah class bernama Player.java

```
public class Player {
```

Inisialisasi variable

```
private Image image, image2;  
public Sprite naruto, narutoWalk;  
  
//default arah naruto, yaitu ke kanan  
public int hadap = 2;  
  
//Frame narutoIdle.png  
public static final int lebarFrameIdle = 36;  
public static final int tinggiFrameIdle = 51;  
  
//Frame narutoWalk.png  
public static final int lebarFrameWalk = 32;  
public static final int tinggiFrameWalk = 52;
```

Buat konstruktor

```
public Player(){  
    try {  
        //Create image  
        image = Image.createImage("/narutoIdle.png");  
        image2 = Image.createImage("/narutoWalk.png");  
  
        //inisialisasi Sprite ke sprite  
        naruto = new Sprite(image,lebarFrameIdle,tinggiFrameIdle);  
        narutoWalk = new Sprite(image2,lebarFrameWalk,tinggiFrameWalk);  
  
        //set referensi pixel  
        //ini opsional, defaultnya adalah 0,0  
        //kita ubah ke lebarFrame/2 dan tinggiFrame  
        naruto.defineReferencePixel(lebarFrameIdle/2, tinggiFrameIdle);  
        narutoWalk.defineReferencePixel(lebarFrameWalk/2, tinggiFrameWalk);  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

Buat kelas animasi(), untuk mengecek state apakah yang sedang berjalan

```
public void animasi(Graphics g, int x, int y, int hadap, int state){  
    //cek state, apakah bernilai 1 (idle) atau 2 (walk)  
    //state == 1 terjadi ketika kita tidak menekan tombol apapun  
    //state == 2 terjadi ketika kita menekan tombol kiri atau kanan  
    if (state == 1) {  
        idle(g, x, y, hadap);  
    }else if (state == 2){  
        walk(g, x, y, hadap);  
    }  
}
```

Buat kelas idle(), untuk menggerakkan naruto ketika state dalam keadaan idle

```
public void idle(Graphics g, int x, int y, int hadap){
    //cek naruto akan menghadap ke arah mana
    //jika hadap = 1, maka transform sprite secara mirror
    if (hadap == 1) {
        naruto.setTransform(2);
    }
    //jika hadap = 2, maka transform sprite ke arah default
    else if(hadap == 2)
        naruto.setTransform(0);

    //set posisi dari gambar berdasarkan referensi pixel
    naruto.setRefPixelPosition(x, y);

    //gambar ke frame selanjutnya
    naruto.nextFrame();

    //paint sprite ke canvas
    naruto.paint(g);
}
```

Buat kelas walk(), untuk menjalankan naruto ketika state dalam keadaan walk

```
public void walk(Graphics g, int x, int y, int hadap){
    //cek naruto akan menghadap ke arah mana
    //jika hadap = 1, maka transform sprite secara mirror
    if (hadap == 1) {
        narutoWalk.setTransform(2);
    }
    //jika hadap = 2, maka transform sprite ke arah default
    else if(hadap == 2)
        narutoWalk.setTransform(0);

    //set posisi dari gambar berdasarkan referensi pixel
    narutoWalk.setRefPixelPosition(x, y);

    //gambar ke frame selanjutnya
    narutoWalk.nextFrame();

    //paint sprite ke canvas
    narutoWalk.paint(g);
}
```

## 2. gameCanvas.java

Buat sebuah class dengan nama file gameCanvas.java, extends GameCanvas dan implements Runnable.

```
public class gameCanvas extends GameCanvas implements Runnable {
```

Inisialisasikan variable yang akan dibutuhkan

```
Thread thread;
Graphics g;

//Inisialisasi kelas Player
private Player player;
int posisiPlayerX;
int posisiPlayerY;
//default arah naruto, yaitu ke kanan
private int hadap = 2;
//default gerakan naruto, 1 = idle, 2 = walk
private int state = 1;

//inisialisasi lokasi gambar background
private int gambarX = 0;
private int gambarY = 2;//karena ukuran bg 240x318
```

Buat **konstruktor** dari kelas ini

```
protected gameCanvas() {
    super(true);

    //Membuat canvas menjadi FullScreenMode
    this.setFullScreenMode(true);

    //Inisialisasi g
    g = getGraphics();

    //Inisialisasi Thread
    thread = new Thread(this);
}
```

Buat fungsi **start()**, untuk menjalankan thread

```
public void start() {
    thread.start();
}
```

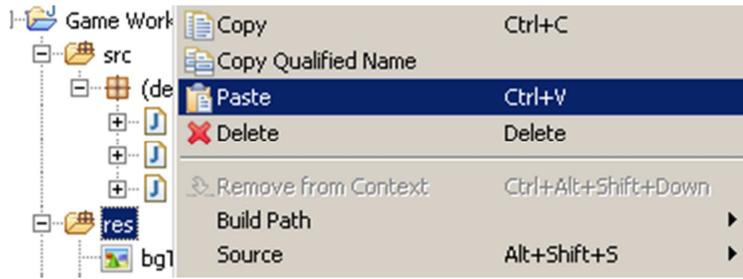
Buat fungsi **initialize()**, untuk inisialisasi variable yang perlu dibutuhkan deklarasi hanya sekali (di luar gameloop)

```
private void initialize() {
    //inisialisasi posisi pemain
    player = new Player();
    posisiPlayerX = getWidth()/2;
    posisiPlayerY = getHeight();
}
```

Buat background dari sebuah gambar.

```
private void createBG() {  
    try {  
        g.drawImage(Image.createImage("/bgTanah.png"), gambarX, gambarY, 0);  
    } catch (IOException e) {  
        // TODO Auto-generated catch block  
        e.printStackTrace();  
    }  
}
```

Ada 2 cara untuk memasukkan asset ke dalam project, pertama copy gambar kemudian klik kanan pada folder res, lalu paste di eclipse langsung.



Atau bisa langsung ke folder projectnya. Lalu paste di folder res. Setelah itu jangan lupa di-refresh, sampai muncul gambar kita di eclipse.



Buat kelas animasiNaruto(), untuk memberikan parameter masukkan hadap dan state ke player.animasi()

```
private void animasiNaruto(int hadap, int state) {  
    player.animasi(g, posisiPlayerX, posisiPlayerY, hadap, state);  
}
```

Buat kelas getinput(), untuk mengubah variable hadap dan state, sesuai dengan inputan kita.

```
private void getinput(){
    int keystate = getKeyStates();
    //cek apakah tombol kiri ditekan
    //jika ya, hadap = 1 (menghadap kiri)
    //state = 2 (naruto berjalan)
    //posisi naruto thd X dikurangi (bergerak ke kiri)
    if((keystate & LEFT_PRESSED) != 0){
        hadap = 1;
        state = 2;
        posisiPlayerX -= 5;
    }
    //cek apakah tombol kiri ditekan
    //jika ya, hadap = 2 (menghadap kanan)
    //state = 2 (naruto berjalan)
    //posisi naruto thd X ditambah (bergerak ke kanan)
    else if((keystate & RIGHT_PRESSED) != 0){
        hadap = 2;
        state = 2;
        posisiPlayerX +=5;
    }else
        state = 1;
}
```

Dan yang terakhir, buat game loop kita, run(),

```
public void run() {
    //inisialisasi variabel yang hanya digunakan sekali
    initialize();
    while (true) {
        //Membuat background berwarna hitam
        g.setColor(17,242,130);
        g.fillRect(0, 0, getWidth(), getHeight());

        //Membuat background dari image
        createBG();

        //Memanggil fungsi getinput()
        getinput();

        //Memanggil fungsi draw()
        animasiNaruto(hadap, state);

        flushGraphics();
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

TIPS:

1. Mohon diperhatikan, bahwa urutan pemanggilan fungsi berpengaruh terhadap level canvas. Contohnya kode di atas, kita membuat `g.fillRect`, memanggil `createBG()`, dan `animasiNaruto()`. Maka yang paling pertama dibuat, yaitu `g.fillRect()`, akan terdapat di paling bawah canvas, sedangkan yang terakhir dipanggil, yaitu `animasiNaruto()`, akan terdapat di paling atas canvas.
2. Manfaatkan fungsi transform seefektif mungkin, sehingga kita tidak perlu membuat `narutoIdle` versi hadap kiri ataupun `narutoWalk` versi hadap kiri.
3. Folder **res** merupakan folder teratas dari asset kita, jadi apabila kita membuat `image` `Image.createImage("/narutoIdle.png")` maka artinya gambar tersebut berada di folder **res**.
4. Pastikan ketika create image, lokasikan file gambar dengan menambah "/", karena terkadang error jika kita memanggil langsung "narutoIdle.png" ketika menjalankannya di device aslinya.
5. Jangan lupa ingatkan artist untuk memberi catatan ukuran gambar dan ukuran per framenya.
6. Pahami alur program di atas, jika sudah mengerti maka tidaklah susah ketika kita ingin membuat gerakan berlari atau melompat.

**Tantangan.**

Buat sprite naruto berlari dan melompat!