BAB 8
SPRITE & COLLISION DETECTION

8.1 Pendahuluan

Sebuah game pasti tidak lepas dari karakter dan animasi. Bisa saja karakternya berupa seekor burung gelatik yang terbang kesana kemari memburu mangsanya. Bagaimana karakter burung gelatik berpindah tempat atau menangkap mangsanya? modul ini akan membahas pemrograman dasar pembentukan karakter serta bagaimana karakter tersebut berinteraksi terhadap karakter lainya.

8.2 Tujuan

Setelah mempelajari modul ini peserta diharapkan dapat:

- Mendefinisikan dan Mengembangkan Sprite
- Mengenali jenis-jenis pengujian tabrakan
- Mengimlpementasikan pengujian tabrakan

8.3 Sprite

Sprite adalah hasil enkapsulasi object dalam sebuah game. Sebagai contoh ketika menggunakan object shape atau gambar seperti pesawat, maka perlu dilakukan proses transformasi seperti translasi dan rotasi,(tranformasi pada java bekerja dengan cara menduplikasi object asli). Ketika proses translasi dilakukan pada sebuah object, maka object yang di gambarkan pada layar adalah duplikatnya sedangkan object asli tetap pada posisi awal oleh sebab itu maka mustahil melakukan deteksi tabrakan atau bahkan sekedar mengetahui lokasi object terkini.

Dengan metode enkapsulasi, maka setiap object sprite mempunyai atribut dan method-method yang dibutuhkan dalam game. Selain itu sprite juga tidak hanya menyediakan accesor dan mutator posisi object tetapi juga dapat menyediakan arah pandang object, arah gerak object dan method lain yang dibutuhkan. Class sprite nantinya juga dapat di gunakan sebagai parent dari actor-actor dalam game sehingga masing-masing actor dapat memiliki behavior tersendiri.

Sebenarnya mendefinisikan sebuah sprite untuk semua tipe game sangat sulit karena setiap jenis game mempunyai karakteristik tersendiri. Sebuah game arcade seperti GalaktikWar membutuhkan object yang dapat bergerak 360°, sedangkan game isometric hanya mendefinisikan 4, 8 atau 16 arah gerak dan game side scrolling seperti MarioBros hanya butuh 3 arah gerak. Selain itu juga pengujian benturan yang membutuhkan akurasi berbeda untuk masing-masing game.

8.4 Mengimplementasikan Class Sprite

Seperti yang telah di singgung, class Sprite yang dibuat akan mendefinisikan atribut-atribut dasar sebuah object dalam game. Class ini tidak akan membuat sebuah class Sprite yang sudah lengkap dengan efek suara, pengkondisian object dll. Sebaliknya hanya membuat class sprite sederhana yang nantinya dapat diturunkan oleh class lainya yang lebih spesifik.

8.5 Konstruktor Sprite

Class sprite diinisialisasi dengan posisi x dan y , object ImageManager serta nama gambar.

```
protected ImageManager imsMgr;  // Image pool
protected int locx, locy;       // Lokasi Sprite
```

```
protected int dx, dy;           // Langkah sprite (dalam pixel)



public Sprite(int x, int y, ImageManager imsMgr, String name)
{
        locx = x; locy = y;
        dx=dy=0;
        this.imsMgr = imsMgr;
        setImage(name);        // set default gambar
}
```

Variabel dx dan dy berfungsi menentukan nilai langkah sprite. Setiap perpindahan tidak dilakukan dengan mengubah langsung ke variabel x dan y namun dengan menentukan nilai lebar langkah dan selanjutnya penambahan dilakukan setiap kali updateSprite() dipanggil.

8.6 Gambar Sprite

Sebelumnya telah dibuat sebuah ImageManager yang menangani pemuatan gambar, jadi dalam class Sprite ini tinggal menggunakanya.

```
    private String imageName;       // nama gambar
    private BufferedImage image;    // Penampung gambar
    private int width, height;      // Dimensi gambar

    private ImagesAnimator animator;  // untuk memainkan strip dan
numbered
    private boolean isLooping;        // toogle image berulang



public void setImage(String name)
{
    imageName = name;
    image = imsMgr.getImage(imageName);
    if (image == null) {
      System.out.println("Tidak ada gambar dengan nama " +
imageName);
    }
```

```
    else {
      width = image.getWidth();
      height = image.getHeight();
    }
    // gambar diasumsikan gambar tunggal
    animator = null;
    isLooping = false;
}
```

Selain menyediakan akses ke gambar tunggal, sprite juga memungkinkan menggunakan gambar ber-nomor atau gambar strip. Penggunaan gambar strip dapat dilakukan dengan memanfaatkan class ImageAnimator melalui method loopImage().

```
public void loopImage(int animPeriod, double seqDuration,boolean
isRepeating)
{
    if (imsManager.numImages(imageName) > 1) {
      animator = null;   // untuk garbage collection
      animator = new ImageAnimator(imageName, animPeriod,
seqDuration,isRepeating, imsManager);
      isLooping = true;
    }
    else
      System.out.println(imageName + " bukan gambar berurutan");
}
```

8.7 Bounding Box

Meskipun pengujian trabrakan akan lebih di utamakan untuk di turunkan, kelas sprite akan menyediakan sebuah Rectangle dengan besar sesuai dengan besar gambar untuk pengujian tabrakan.

```
public Rectangle getBoundingBox() {
      return new Rectangle(locx, locy, width, height);
}
```

8.8 Update Sprite

Sprite memiliki atribut dx dan dy yang merupakan besar velocity sprite tersebut. Kedua atribut tersebut menentukan kecepatan dan arah gerak dari sprite tersebut. Atribut velocity akan digunakan sebagai increment posisi sprite setiap kali sprite di update. Selain perubahan posisi sprite, update pada sprite juga akan mengirimkan pesan ke object ImageAnimator jika sprite tersebut memiliki lebih dari satu gambar.

```
public void updateSprite() {
      if (isActive()) {
            locx += dx;
            locy += dy;
            if (isLooping)
               // update posisi gambar pada animator
                  animator.updateTick();
      }
}
```

Kecepatan gerak sprite akan sangat dipengaruhi banaknya pemanggilan method update. Sebuah sprite dengan dx=5 yang di-update sebanyak 40 kali dalam satu detik berarti sprite tersebut akan berpindah 5 x 40 pixel, nilai yang berbeda akan diperoleh jika kecepatan update berubah.

8.9 Menggambar Sprite

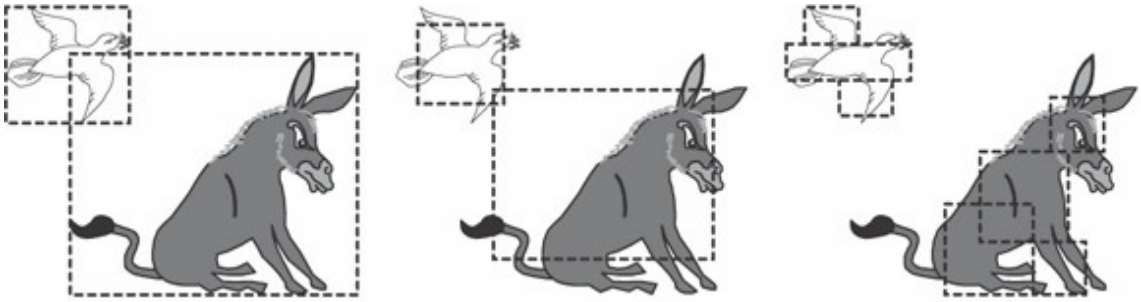
Method ini memiliki input berupa object Graphics untuk menampilkan gambar. Jika sprite memiliki lebih dari satu gambar, method ini akan meminta gambar terkini pada object ImageAnimator.

```
      public void drawSprite(Graphics g) {
            if (isActive()) {
                  if (isLooping)
                        image = animator.getCurrentImage();
                  g.drawImage(image, locx, locy, null);
```

```
            }
        }
```

8.10 Collision Detection

Pada bahasan sprite terdapat method getBound(), method tersebut menyediakan rectangle yang akan digunakan dalam proses pendeteksian tabrakan antar sprite. Perhatikan visualisasi beberapa cara pendeteksian berikut ini:



Gambar 8.1 Mendeteksi Tabrakan

Gambar pertama menunjukan visualisasi yang digunakan dalam class sprite. Cara ini adalah cara paling mudah dan tercepat, namun seperti yang terlihat diatas cara ini kurang bagus secara akurasi. Program akan mendeteksi terjadinya tabrakan ketika bound kedua sprite bersinggungan meskipun secara kasat mata sprite tersebut sama sekali tidak bersinggungan. Cara ini ideal untuk game yang tidak terlalu memperhatikan akurasi deteksi tabrakan atau game yang menggunakan sprite mendekati kotak.

Cara kedua adalah menggunakan apa yang disebut reduzed size bounding box (kotak batas yang diperkecil). Cara ini cukup mudah, hanya perlu mengurangi beberapa pixel dari ukuran sebenarnya dari image sprite sehingga pendeteksian masuk lebih dalam seperti terlihat pada gambar cara diatas lebih akurat dari pada cara pertama.

Cara ketiga bisa dikatakan cara paling akurat dari ketiga cara diatas. Idenya adalah menyediakan beberapa bounding box sekaligus untuk keperluan pendeteksian tabrakan. Karena menggunakan banyak bounding box, maka di munkinkan penyesuaian bounding box dengan bentuk gambar pada sprite. Dengan cara ini bisa menggunakan sebanyak mungkin bounding box agar lebih akurat dengan catatan semakin banyak bounding box maka akan semakin banyak waktu yang diperlukan untuk mendeteksi tabrakan antar sprite.