BAB 6
GRAFIK BITMAP

6.1 Pendahuluan

Inti dari game 2 dimensi sebenarnya terletak pada image, dimana image yang telah dibuat ataupun diedit dari program seperti adobe photoshop, corel draw dll, kemudian diurutkan menjadi satu bentuk. Sebagai contoh sebuah pesawat yang untuh sampai pada pesawat yang hancur atau meledak dikumpulkan menjadi satu kesatuan dan akan dipanggil saat dibutuhkan (apabila terkena bom maka gambar pesawat yang ditampilkan meledak) ilustrasinya seperti itu dan masih banyak lagi yang lain sesuai dengan game yang akan dibuat.

6.2 Tujuan

Setelah mempelajari modul ini peserta diharapkan dapat:

- Mengetahui jenis-jenis format image
- Mengetahui cara meload file image
- Memuat gambar strip atau gambar bernomor

6.3 Pengenalan Image(Gambar)

Game sebenarnya tentu saja tidak hanya menggunakan primitive object seperti grafik vektor melainkan juga menggunakan bitmap. Sebuah game biasanya menggunakan format gambar GIF, JPEG, dan PNG.

Graphics Interchange Format atau gambar GIF sangat baik untuk tampilan grafis cartoon yang menggunakan beberapa warna. Gambar GIF menggunakan 256 elemen tabel warna untuk menyimpan informasi. Selah

satu dari table warna digunakan untuk representasi warna “Transparan” yang mana tidak digambar oleh Program JAVA.

File Joint Photographic Experts Group (JPEG) menggunakan 3 byte atau 24 bit per pixel, dimana 1 byte untuk merah, hijau dan biru (komponen RGB). File JPEG sangat bagus untuk gambar foto yang berukuran besar seperti background game. File JPEG tidak menyediakan transparansi.

Format Portable Network Graphics (PNG) dimaksudkan untuk menggantikan file GIF, dimana format ini menyertakan sebuah chanel Alpha yang memungkinkan adanya area tembus cahaya(Transluccency) pada sebuah gambar. Tembus cahaya (Transluccency) adalah bagian yang sangat berguna untuk efek game seperti tembakan laser, asap, dll. Beberapa pengembang lebih suka menggunakan PNG karena standarnya open source.

6.4 Memuat Gambar

Aplikasi java dapat dengan mudah memuat sebuah gambar menggunakan class Toolkit.

```
Toolkit tk = new Toolkit.getDefaultTollkit();
Image = tk.getImage(“test.jpg”);
```

Ketika menggunakan Toolkit, class tersebut menggunakan thread untuk memuat gambar secara terpisah. Karena class Toolkit menggunakan thread maka program kita tidak akan pernah tahu kapan pemuatan gambar selesai dilakukan, hasilnya gambar yang akan kita tampilkan bisa saja ditampilkan sebagian karena belum selesai di muat atau bahkan tidak sama sekali.

Untuk menutupi kelamahan Toolkit, Java menyediakan class MediaTracker, kita dapat menggunakan class ini agar aplikasi kita menunggu sampai sebuah gambar dimuat.

```
Image logo = getToolkit().getImage("test.jpg");

MediaTracker mediaTracker = new MediaTracker(this);
mediaTracker.addImage(logo, 0);
try
{
        mediaTracker.waitForID(0);
}
catch(InterruptedException e)
{
        System.out.println(e);
}
```

Meskipun menggunakan MediaTracker mudah, namun akan sangat membosankan jika harus melakukanya untuk setiap gambar yang akan dimuat (bayangkan jika memuat 100 gambar). Ada cara yang lebih mudah yaitu dengan menggunakan class ImageIO. Class ImageIO menyediakan cara tercepat memuat image.

```
Image logo=null;
try
{
        logo = ImageIO.read(new File("test.jpg"));
}
catch(IOException e)
{
        System.out.println(e);
}
```

6.5 Menggunakan BufferedImage

BufferedImage merupakan turunan dari class Image. Dengan BufferedImage maka gambar yang akan di tampilkan akan dimuat sampai selesai sehingga menghindari terjadinya penggambaran secara partial. Class BufferredImage juga memiliki keunggulan dalam hal manupulasi gambar melalui method-methodnya dan kemampuan untuk secara otomatis di ubah ke dalam Managed Image oleh JVM. Managed Image memungkinkan adanya akselerasi secara hardware.

```
BufferedImage im;

public void init( )
{
    try {
        im =ImageIO.read( getClass( ).getResource("ball.gif") );
        }
    catch(IOException e) {
        System.out.println("Error Loading Image : "e.toString());
    }
}
```

Cara termudah dan tercepat memuat image kedalam BufferedImage adalah menggunakan class ImageIO. Menurut beberapa pengujian class ImageIO lebih cepat 10% dari class ImageIcon, dan mungkin bisa semakin besar jaraknya jika yang di muat adalah gambar beresolusi besar.Untuk kepentingan lebih lanjut, BufferedImage yang akan kita buat akan di sesuaikan dengan konfigurasi device yang kita miliki.

```
try {
       BufferedImage im =
ImageIO.read(getClass( ).getResource(fnm));

       int transparency = im.getColorModel( ).getTransparency( );
       BufferedImage copy =  gc.createCompatibleImage(
```

```
                    im.getWidth( ),im.getHeight( ),transparency );

        Graphics2D g2d = copy.createGraphics( );

        g2d.drawImage(im,0,0,null);
        g2d.dispose( );
        return copy;
}
catch(IOException e) {
        System.out.println("Load Gambar " + fnm + " eror :\n" + e);
        return null;
}
```

Tiga argumen dalam method createCompatibleImage adalah lebar, tinggi dan nilai transparansi. Terdapat tiga nilai transparansi yaitu Transparency.OPAQUE, Transparency.BITMASK, dan Transparency. TRANSLUCENT . Opaque digunakan untuk mewakili gambar tanpa kemampuan transparansi, contohnya gambar berekstensi JPG. Bitmask digunakan untuk merepresentasikan area kosong (tidak digambar) yang dimiliki gambar berformat GIF. Translucent digunakan PNG sebagai area tembus cahaya. Tanpa adanya parameter transparansi kemungkinan besar gambar akan di gambar tidak sesuai dengan yang diharapkan.

6.6 Membuat Image Manager

Sebelumnya telah dibahas cara memuat image kedalam aplikasi, tetapi dalam game sebuah object nantinya tidak hanya terdiri dari gambar statis yang bergerak hanya karena transformasi. Pada game 2 dimensi setiap object bisa saja merupakan animasi bahkan ketika ia tidak mengalami transformasi. Untuk mengakomodasi keperluan tersebut maka akan digunakan 2 teknik. Teknik pertama adalah menyediakan sebuah gambar yang isinya terdiri dari sequencial animasi suatu karakter. Dengan teknik ini gambar tidak akan di

tampilkan seluruhnya melainkan per frame (setiap bagian dalam sequence akan kita anggap sebagai frame). Teknik kedua adalah menggunakan gambar yang berbeda untuk setiap frame namun dengan nama sama plus identitas iterasi (contohnya : player1.png, player2.png, player3.png dst).

6.7 Mendefinisikan Aturan Pemuatan Gambar

Untuk membuat sebuah class yang akan menerima pemuatan semua tipe gambar maka perlu mendefinisikan peraturan-peraturanya. Class ImageManager akan memiliki method yang berbeda untuk setiap penganganan jenis gambar yang berbeda.

Class Image Manager yang kita buat akan menggunakan sebuah object HashMap untuk memetakan setiap object gambar dengan sebuah kunci unik. Setiap gambar dapat diakses melalui method accessor dengan melewatkan nama sebagai parameter. Setiap nilai yang di simpan dalam HashMap adalah sebuah ArrayList. ArrayList digunakan karena anggota dari sebuah record dalam HashMap bisa berisi lebih dari satu gambar. Mungkin class ini sedikit rumit dan membingungkan, tetapi class ini akan sangat berguna untuk keperluan game berikutnya.

Untuk keperluan pemuatan gambar kedalam class ini, kita akan membuat 3 method. Method pertama digunakan untuk memuat gambar tunggal, method kedua digunakan untuk memuat gambar berurutan dengan postfiks urutan gambar, dan method terakhir digunakan untuk memuat gambar strip (frame). Setiap method akan menggunakan sepasang method yang berfungsi memisahkan prefix dan postfix parameter method bersangkutan dan sebuah method untuk me-muat image (jangan bingung, sebenarnya ketiga

method yang disebutkan sebelumnya tidak berfungsi memuat gambar secara langsung melainkan berfungsi me-metakan gambar yang di muat kedalam HashMap).

```
private String getPrefix(String fnm)
// mengambil String sebelum '.' dari nama file
{
      int posn;
      if ((posn = fnm.lastIndexOf(".")) == -1) {
            System.out.println("Prefix tidak ditemukan untuk nama
file: " + fnm);
            return fnm;
      }
      else
            return fnm.substring(0, posn);
}

private String getPosfix(String fnm)
// mengambil String sesudah '.' dari nama file
{
      int posn;
      if ((posn = fnm.lastIndexOf(".")) == -1) {
            System.out.println("Posfix tidak ditemukan untuk nama
file:" + fnm);
            return fnm;
      }
      else
            return fnm.substring(posn);
}
```

6.8 Gambar Tunggal

Untuk meload gambar tunggal, method yang digunakan sama dengan method pemuatan gambar yang telah dibahas. Method ini mengembalikan nilai boolean sehingga kita bisa menggunakannya untuk memastikan gambar berhasil dimuat.

```
public boolean loadSingleImage(String fnm)
    {
```

```
      String name = getPrefix(fnm);

      if (imagesMap.containsKey(name)) {
        System.out.println( "Error: " + name + " sudah digunakan");
        return false;
      }

      BufferedImage bi = loadImage(fnm);
      if (bi != null) {
        ArrayList imsList = new ArrayList( );
        imsList.add(bi);
        imagesMap.put(name, imsList);
        System.out.println("  disimpan " + name + "/" + fnm);
        return true;
      }
      else
        return false;
  }
```

6.9 Gambar Ber-Nomor

Method yang menangani gambar ber-nomor bekerja dengan memisahkan prefix dan postfix lalu menambahkan urutan gambar setelah prefix dan menggabungkan kemabali dengan postfix menjadi argumen method LoadImage(). Seperti halnya pemuatan gambar tunggal, nama unik yang dugunakan sebagai kunci HashMap adalah prefix dari parameter method ini.

Dig0.gif Dig1.gif Dig2.gif Dig3.gif Dig4.gif Dig5.gif

Gambar 6.1 Gambar Bernomor

```
public int loadNumImages(String fnm, int number)
{
      String prefix = getPrefix(fnm);
      String postfix = getPosfix(fnm);
```

```
        String imFnm;
        BufferedImage bi;
        ArrayList imsList = new ArrayList();
        int loadCount = 0;
        for(int i=0; i < number; i++) {
                imFnm = prefix + i + postfix;
                if ((bi = loadImage(imFnm)) != null) {
                        loadCount++;
                        imsList.add(bi);
                }
        }

        if (loadCount == 0)
                System.out.println("Tidak ada images dengan nama " +
prefix);
        else
                imagesMap.put(prefix, imsList);

        return loadCount;
}
```

6.10 Gambar Strip

Seperti yang telah di singgung, gambar strip sebenarnya adalah susunan gambar berurutan yang membentuk sebuah animasi dalam sebuah file citra. Biasanya gambar strip digunakan untuk animasi dengan pixel yang kecil. Gambar strip unggul dalam penggunaan waktu proses dibandingkan gambar ber-nomor. Karena merupakan potongan gambar, maka setiap potongan harus di pisahkan dari gambar asli sebelum di render.

Gambar 6.2 Gambar Strip 6x1

Gambar 6.3 Gambar Strip 8x4

Cara kerja dari method yang akan kita buat adalah memisahkan setiap potongan gambar kedalam satu record pada ArrayList. Pemisahan akan dilakukan oleh method loadStripImageArray().

```
public int loadStripImages(String fnm, int column,int row)
{
	String name = getPrefix(fnm);

	BufferedImage[] strip = loadStripImageArray(fnm, column, row);
	if (strip == null)
		return 0;

	ArrayList imsList = new ArrayList();
	int loadCount = 0;

	for (int i=0; i < strip.length; i++) {
		loadCount++;
		imsList.add(strip[i]);
	}

	if (loadCount == 0)
		System.out.println("Tidak ada image dengan prefix " +
name);
	else
		imagesMap.put(name, imsList);

	return loadCount;
}

public BufferedImage[] loadStripImageArray(String fnm, int
column,int row)
{
	if (column <= 0) {
		System.out.println("column <= 0; return null");
		return null;
	}

	BufferedImage stripIm;
	if ((stripIm = loadImage(fnm)) == null) {
		System.out.println("Return null");
		return null;
	}
```

```
      int imWidth = (int) stripIm.getWidth() / column;
      int imHeight =(int) stripIm.getHeight()/row;

      int transparency = stripIm.getColorModel().getTransparency();

      BufferedImage[] strip = new BufferedImage[column*row];
      Graphics2D stripGC;


      for (int i=0;i<row;i++)
      {
            for(int j=i*column;j<(i+1)*column;j++)
            {
                  int fx=(j%column)*imWidth;
                  int fy=i*imHeight;
                  strip[j] =  gc.createCompatibleImage(imWidth,
imHeight, transparency);

                  stripGC = strip[j].createGraphics();
                  stripGC.setComposite(AlphaComposite.Src);

                  stripGC.drawImage(stripIm,0,0, imWidth,imHeight,
                        fx,fy, fx+imWidth,fy+imHeight,null);
                  stripGC.dispose();
            }
      }
      return strip;
}
```

Bagian kode yang di cetak tebal adalah fungsi utama dari method pemuatan gambar strip. Untuk menentukan parameter pemotongan gambar pada method drawImage digunakan dua buah variabel yaitu fx dan fy. Kedua buah variabel tersebut menjadi titik awal pemotongan gambar.

```
int fx=(j%column)*imWidth;
int fy=i*imHeight;
```

Variabel fx menentukan titik x yang didapat dengan mendapatkan hasil modulo nomor kolom dengan jumlah kolom di kali besar gambar per potong. Sedangkan variabel fy menentukan titik y dengan cara mengalikan nomor baris dengan tinggi gambar.

6.11 Mengakses Gambar

Setelah berhasil memuat gambar, selanjutnya perlu mengakses gambar tersebut. Karena menggunakan HashMap untuk menyimpan gambar maka

untuk mengaksesnya baik gambar tunggal, gambar ber-nomor ataupun gambar strip menggunakan prefix dari gambar tersebut. Ada tiga versi untuk mengakses gambar, versi pertama menerima satu parameter berupa nama gambar, versi kedua menerima dua parameter yaitu nama gambar dan posisi gambar dalam ArrayList dan versi ketiga mengembalikan semua gambar yang terdapat dalam ArrayList dalam nama tertentu.

```
public BufferedImage getImage(String name)
{
      ArrayList imsList = (ArrayList) imagesMap.get(name);
      if (imsList == null) {
            System.out.println("Tidak ada gambar dengan nama " +
name);
            return null;
      }

      return (BufferedImage) imsList.get(0);
}

public BufferedImage getImage(String name, int posn)
{
      ArrayList imsList = (ArrayList) imagesMap.get(name);
      if (imsList == null) {
            System.out.println("Tidak ada gambar dengan nama " +
name);
            return null;
      }

      int size = imsList.size();
      if (posn < 0) {
            return (BufferedImage) imsList.get(0);   // return first
image
      }
      else if (posn >= size) {
            int newPosn = posn % size;   // modulo
            return (BufferedImage) imsList.get(newPosn);
      }
      return (BufferedImage) imsList.get(posn);
}

public ArrayList getImages(String name)
{
      ArrayList imsList = (ArrayList) imagesMap.get(name);
      if (imsList == null) {
            System.out.println("Tidak ada gambar dengan nama " +
name);
            return null;
      }

      return imsList;
}
```

6.12 Memainkan Gambar Bernomor dan Gambar Strip

Setelah berhasil memuat dan mengakses gambar masih perlu mengatur bagaimana gambar animasi (Bernomor dan Gambar Strip: selanjutnya disebut gambar animasi) di tampilkan ke layar, akan dibuat sebuah class lagi yang khusus menangani gambar animasi. Class ini akan mengatur urutan gambar keberapa yang dapat di akses melalui getCurrentImage(). Konstruktor class akan menerima parameter berupa nama gambar, periode animasi, panjang animasi secara keseluruhan, boolean pengatur pengulangan dan object ImageManager.

```
ImagesAnimator animator = new ImagesAnimator (imagesName,
animPeriod, seqDuration, isRepeating, imsLoader);
```

Proses update gambar bertumpu pada method updateTicks() yang dipanggil pada pengendali game utama. Harus diperhatikan bahwa dalam pemberian nilai parameter adalah besarnya nilai animPeriod dan seqDuration, kedua parameter ini akan sangat berpengaruh jika animasi yang dijalankan diset non-repeat. Ketika animasi di set no-repeat, maka harus menghitung jumlah gambar yang ada dalam sebuah ArrayList. Nilai seqDuration harus cukup besar hingga hasil bagi (showPeriod) dengan jumlah gambar masih dibawah nilai animPeriod. Jika showPeriod kurang dari animPeriod maka animasi tidak pernah berhenti pada saat semua urutan gambar telah ditampilkan.

Untuk mengetahui kondisi selesai tidaknya sebuah urutan animasi perlu mendesign sebuah interface yang berfungsi layaknya Listener seperti ActionListener dll. Class listener ini memiliki sebuah method abstract yang

harus diimplementasikan. Class Listener menerima kejadian yang dikirimkan oleh class ImageAnimator.

```
numbersPlayer =  new ImageAnimator ("numbers", PERIOD, 1, false,
imsLoader);
numbersPlayer.addImageSequenceListener(this);
```

Method updateTicks() berfungsi menentukan posisi gambar dalam ArrayList sekaligus mengirim kejadian selesainya sebuah urutan animasi kepada ImageSequenceListener (hanya jika isReapeating tidak di set false).

```
public void updateTick( )
{
        if (!ticksIgnored) {
        // update total waktu animasi, modulo seq duration
                animTotalTime = (animTotalTime + animPeriod) %
                                        (long)(1000 * seqDuration);

                // menentukan posisi gambar terkini
                imPosition = (int) (animTotalTime / showPeriod);

                if ((imPosition == numImages-1) && (!isRepeating)){
                        // menghentikan update tick pada gambar ini
                        ticksIgnored = true;
                        if (watcher != null)
                                watcher.sequenceEnded(imName);
                }
        }
}
```

Pengaksesan gambar terkini di lakukan melalui method getCurrentImage() yang mengembalikan dalam gambar ArrayList pada posisi yang ditentukan oleh imPosition.

```
public BufferedImage getCurrentImage( )
{
        if (numImages != 0)
                return imsLoader.getImage(imName, imPosition);
        else
                return null;
}
```