Bab 4
Low Level User Interface

4.1 Tujuan

Setelah mempelajari bab ini, Pelajar diharapkan mampu untuk :

- Memahami event handling level rendah dalam MIDP
- Menggambar dan menampilkan teks, gambar, garis, kotak, dan sudut
- Menentukan warna, huruf, dan coretan untuk operasi menggambar
- Memahami dan menggunakan class Canvas dan Graphic
- Mengetahui bagaimana menggunakan GAME API
- Menggambar grafik berskala

4.2 Pengenalan

Pada bab sebelumnya, kita telah membahas tentang bagaimana cara membuat user interface level tinggi seperti list, form, dan field input. Mereka bersifat user interface level tinggi dan programmer tidak perlu khawatir tentang menggambar pixel layar atau mengatur posisi teks pada layar. Semua program telah menetapkan jenis komponen dan label elemen. Sistem tersebut akan menangani gambar pada layar, scrolling dan layout.

Satu kelemahan ketika hanya menggunakan komponen user interface level tinggi adalah program tidak memiliki kendali penuh sebuah layar. Ada saat dimana kita ingin menggambar sebuah garis, gambar beranimasi dan mempunyai kendali untuk mengatur pixel pada layar.

Pada bab ini, kita akan berhadapan langsung dengan layar. Kita akan mempelajari class Canvas, dimana akan menjadi pendukung dari proses menggambar kita. Kita juga akan menyelidiki ke dalam class Graphic, dimana memiliki metode untuk menggambar garis, kotak, sudut, dan teks. Kita juga akan membahas huruf, warna dan gambar.

4.3 Canvas

Canvas adalah subclass dari Displayable. Itu adalah sebuah class abstrak yang harus di-extend sebelum sebuah aplikasi dapat menggunakan fungsi-fungsi yang ada.

Canvas dapat digabungkan dengan subclass Displayable level tinggi yaitu Screen. Program dapat pindah ke dan dari Canvas dan Screen.

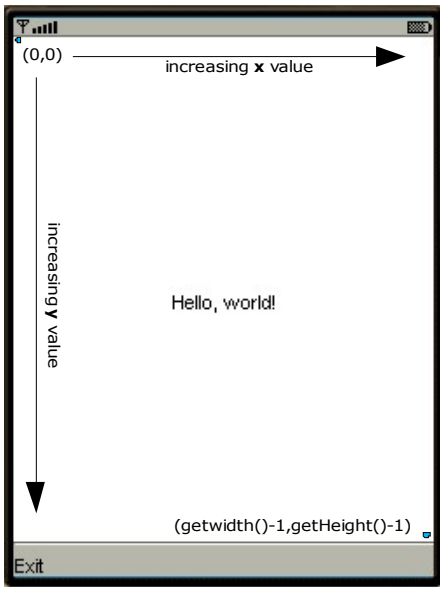
Canvas menggambarkan metode-metode event handling kosong. Aplikasi harus mengesampingkan mereka untuk handle event.

Class Canvas menggambarkan sebuah metode abstrak yang disebut paint(). Aplikasi menggunakan class Canvas harus menyediakan sebuah implementasi untuk metode paint().

4.3.1 Sistem Koordinat

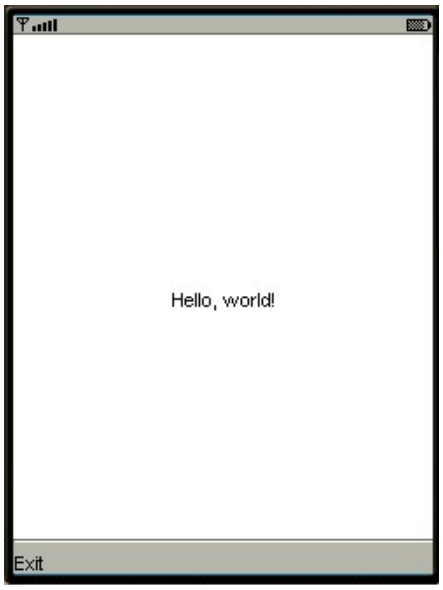
Sistem koordinat dari Canvas adalah berbasis nol. Koordinat x dan y dimulai dengan nol. Pojok kiri atas dari Canvas berkoordinat (0,0). Koordinat x bertambah dari kiri ke kanan. Sedangkan koordinat y bertambah dari atas ke bawah. Metode getWidth() dan getHeight() mengembalikan nilai lebar dan tinggi berturut-turut.

Pojok kanan bawah pada layar memiliki koordinat (getWidth()-1,getWidth()-1). Setiap perubahan yang terjadi pada ukuran yang diberikan untuk area menggambar pada Canvas dilaporkan kepada aplikasi oleh metode sizeChanged(). Ukuran yang tersedia pada Canvas mungkin saja berubah jika ada pergantian antara mode layar full dan normal atau penambahan dan pengurangan sebuah komponen seperti Command.



Gambar 1: Sistem Koordinat

4.3.2 "Hello,World!"



Gambar 2: Hello World MIDlet menggunakan canvas

```
import javax.microedition.midlet.*;

import javax.microedition.lcdui.*;


public class HelloCanvasMIDlet extends MIDlet {

private Display display;

HelloCanvas canvas;

Command exitCommand = new Command("Exit", Command.EXIT, 0);


public void startApp() {

if (display == null){

canvas = new HelloCanvas(this, "Hello, world!");

display = Display.getDisplay(this);
```

```
}
display.setCurrent(canvas);
}
public void pauseApp() {
}
public void destroyApp(boolean unconditional) {
}
protected void Quit(){
destroyApp(true);
notifyDestroyed();
}
}

class HelloCanvas extends Canvas implements CommandListener {
private Command exitCommand = new Command("Exit", Command.EXIT, 0);
private HelloCanvasMIDlet midlet;
private String text;
public HelloCanvas(HelloCanvasMIDlet midlet, String text) {
this.midlet = midlet;
this.text = text;
addCommand(exitCommand);
setCommandListener(this);
}
```

```
protected void paint(Graphics g) {

// membersihkan layar dengan mengisi semua layar dengan warna putih

g.setColor(255, 255, 255 );

g.fillRect(0, 0, getWidth(), getHeight());

// mengatur warna tulisan dengan warna hitam

g.setColor(0, 0, 0);

// dan menulis sebuah text

g.drawString(text,

getWidth()/2, getHeight()/2,

Graphics.TOP | Graphics.HCENTER);

}

public void commandAction(Command c, Displayable d) {

if (c == exitCommand){

midlet.Quit();

}

}

}
```

Dengan midlet “Hello,World!”, kita membuat sebuah class yang ber-extend Canvas

```
class HelloCanvas extends Canvas implements CommandListener {
```

Kemudian kita menambahkan perintah (“Exit”) dan mengatur command listener nya.

```
addCommand(exitCommand);

setCommandListener(this);
```

Kita menciptakan command listener dengan mengimplementasikan class CommandListener. Ini berarti membuat class yang memiliki metode commandAction.

```
class HelloCanvas extends Canvas implements CommandListener {

...


public void commandAction(Command c, Displayable d) {

...
```

Inti dari program ini adalah metode paint(). Set pertama dari pemanggilan metode adalah membersihkan layar.

```
g.setColor(255, 255, 255 );

g.fillRect(0, 0, getWidth(), getHeight());
```

Dan kemudian grafik memanggil metode drawString() untuk menampilkan "Hello,World!" pada layar :

```
// mengatur warna tulisan dengan warna hitam

g.setColor(0, 0, 0);

// dan menulis sebuah text

g.drawString(text, getWidth()/2, getHeight()/2,

Graphics.TOP | Graphics.HCENTER);
```

4.3.3 Perintah

Seperti halnya list, textBox, dan form, Canvas juga mempunyai Command yang disediakan dan dapat merespon untuk event Command. Langkah-langkah untuk menambahkan Command ke dalam Canvas adalah sama :

1. Buatlah objek Command.

```
private Command exitCommand = new Command("Exit", Command.EXIT, 0);
```

2. Gunakan useCommand() untuk menambahkan perintah ke dalam canvas(atau Form, list, text box)

```
addCommand(exitCommand);
```

3. Gunakan setCommandListener() untuk mendaftarkan class mana yang akan mendapat event command untuk perintah dalam Displayable.

```
setCommandListener(this);
```

4. Buatlah sebuah commandListener dengan mengimplementasikan class commandListener dan menyediakan sebuah metode commandAction().

```
class HelloCanvas extends Canvas implements CommandListener {
...
public void commandAction(Command c, Displayable d) {
if (c == exitCommand){
// do something
}
}
```

4.3.4 Event key

Subclass dari Canvas dapat merespon sebuah event tombol dengan metode-metode sebagai berikut :

keyPressed(int keyCode)	Dipanggil ketika kunci ditekan
keyRepeated(int keyCode)	Dipanggil ketika kunci diulang
keyReleased(int keyCode)	Dipanggil ketika kunci dilepas

Canvas mendefinisikan kode tombol ini : KEY_NUM0, KEY_NUM1, KEY_NUM2, KEY_NUM3, KEY_NUM4, KEY_NUM5, KEY_NUM6, KEY_NUM7, KEY_NUM8, KEY_NUM9, KEY_STAR, and KEY_POUND.

Untuk mendapatkan "String" nama sebuah kunci, gunakan metode getKeyName(int keyCode).

4.3.5 Aksi Permainan

Masing-masing kode tombol bisa dipetakan menjadi sebuah aksi game. Sebuah key code bisa dipetakan kepada aksi sebuah game. Class Canvas mendefinisikan aksi game ini : UP, DOWN, LEFT, RIGHT, FIRE, GAME_A, GAME_B, GAME_C, GAME_D. Sebuah program dapat menerjemahkan sebuah key code ke dalam aksi game menggunakan metode getGameAction(keyCode).

Metode getKeyCode(int gameAction) mengembalikan key code yang berkaitan dengan suatu aksi game. Sebuah aksi game dapat memiliki lebih dari satu key code yang berkaitan dengannya. Jika terdapat lebih dari satu key code yang berkaitan dengan aksi game, hanya satu key code yang akan dikembalikan.

Sebuah aplikasi perlu menggunakan getGameAction(int keyCode) daripada langsung menggunakan kode tombol yang telah didefinisikan. Secara normal, jika suatu program ingin merespon kunci "UP", sebaiknya menggunakan kunci KEY_NUM2 atau key code yang spesifik untuk tombol UP. Program menggunakan metode ini tidaklah portable sejak sebuah perangkat memiliki layout kunci yang berbeda dan key code yang berbeda pula. KEY_NUM2 mungkin menjadi kunci "UP" untuk sebuah

perangkat, tetapi mungkin juga menjadi kunci "LEFT" untuk perangkat lainnya. GetGameAction() akan selalu mengembalikan "UP", tidak terikat pada kunci mana yang ditekan selama dia adalah kunci "UP" di dalam konteks dari layout kunci sebuah perangkat.

4.3.6 Event Pointer

Disamping dari event key, program MIDP juga dapat mengatasi event pointer. Hal ini bersifat benar jika sebuah perangkat memiliki sebuah pointer dan hal tersebut diimplementasikan di dalam sistem JAVA pada sebuah perangkat.

Metode hasPointerEvents() mengembalikan nilai true jika sebuah perangkat mendukung pointer yang bersifat ditekan dan dilepaskan. Metode hasPointerMotionEvents() mengembalikan nilai true jika sebuah perangkat mendukung event gerakan dari pointer.

```
public boolean hasPointerEvents()
public boolean hasPointerMotionEvents()
```

Sebuah event dapat di-generate oleh aktivitas pointer sebagai berikut : pointerPressed, pointerReleased dan pointerDragged. Sebuah aplikasi mengesampingkan metode-metode yang untuk diperhatikan ketika event ini terjadi. Koordinat (x,y) dari event (ketika pointer ditekan, dilepas atau digeser) ditetapkan didalam metode-metode callback ini.

```
protected void pointerPressed(int x, int y)
protected void pointerReleased(int x, int y)
protected void pointerDragged(int x, int y)
```

4.4 Grafik

Class Graphic adalah class utama untuk menulis teks, menggambar, garis, kotak dan sudut. Dia memiliki metode untuk menentukan warna, huruf, dan coretan.

4.4.1 Warna

Class Display memiliki metode untuk menentukan apakah sebuah perangkat memiliki fasilitas yang mendukung layar berwarna atau layar monochrome pada sebuah perangkat.

`public boolean` **`isColor`**`()`	Mengembalikan nilai true jika mendukung layar berwarna dan sebaliknya.
`public int` **`numColors`**`()`	Mengembalikan nomor warna(atau level abu-abu jika sebuah perangkat tidak mendukung warna) yang didukung sebuah perangkat

Untuk mengatur warna yang digunakan untuk metode grafik berikutnya, gunakan metode setColor(). SetColor() memiliki dua bentuk:

```
public void setColor(int red, int green, int blue)

public void setColor(int RGB)
```

Pada bentuk pertama, Anda menentukan komponen warna merah, hijau, dan biru. Pada bentuk kedua komponen warna ditentukan dalam bentuk 0x00RRGGBB. Pemanggilan setColor(int) pada contoh berikut akan melakukan hal yang sama:

```
int red, green, blue;

...

setColor(red, green, blue);

setColor( (red<<16) | (green<<8) | blue );
```

Metode lainnya untuk memanipulasi warna adalah :

`public int` **`getColor`**`()`	**Mengembalikan warna terbaru dalam bentuk integer.**
`public int` **`getRedComponent`**`()`	Mengembalikan komponen merah sebagai warna terbaru
`public int` **`getGreenComponent`**`()`	Mengembalikan komponen hijau sebagai warna terbaru
`public int` **`getBlueComponent`**`()`	Mengembalikan komponen biru sebagai

public int **getColor**()	**Mengembalikan warna terbaru dalam bentuk integer.**
	warna terbaru
public int **getGrayScale**()	Mengembalikan nilai abu-abu sebagai warna terbaru
public void **setGrayScale**(int value)	Memilih nilai abu-abu untuk mengganti operasi menggambar

4.4.2 Huruf

Sebuah huruf memiliki tiga atribut yaitu bentuk, type, dan ukuran. Huruf tidak diciptakan oleh aplikasi. Sebagai gantinya, sebuah aplikasi meminta sistem untuk memilih model atribut huruf dan sistem mengembalikan huruf yang sesuai dengan model atribut yang diminta. Sistem tidak menjamin akan mengembalikan semua atribut huruf yang dipilih. Jika sistem tidak memiliki huruf yang sesuai dengan permintaan, dia akan mengembalikan sebuah huruf hampir mirip dengan atribut yang diminta.

Huruf adalah sebuah class yang terpisah. Seperti yang dinyatakan di atas, aplikasi tidak menciptakan objek huruf. Sebagai gantinya, metode-metode statis getFont() dan getDefaultFont() digunakan untuk meminta sebuah huruf dari sistem.

public static Font **getFont**(int face, int style, int size)	**Mengembalikan sebuah huruf dari sistem yang sesuai dengan atribut**
public static Font **getDefaultFont**()	Mengembalikan huruf default yang digunakan oleh sistem
public static Font **getFont**(int fontSpecifier)	Mengembalikan huruf yang digunakan untuk komponen UI level tinggi. FontSpecifier bisa jadi : FONT_INPUT_TEXT atau FONT_STATIC_TEXT

Face adalah salah satu dari `FACE_SYSTEM`, `FACE_MONOSPACE`, atau `FACE_PROPORTIONAL`.

Style bisa jadi STYLE_PLAIN atau kombinasi STYLE_BOLD, STYLE_ITALIC dan STYLE_UNDERLINED. Kombinasi style ditentukan oleh penggunaan bitwise operator OR (|). Sebuah style huruf tebal(bold) dan garis miring(italic) dideklarasikan sebagai :

STYLE_BOLD | STYLE_ITALIC

Ukuran huruf bisa jadi : SIZE_SMALL, SIZE_MEDIUM, SIZE_LARGE

Metode ini mengembalikan atribut huruf tertentu:

```
public int getStyle()
public int getFace()
public int getSize()
public boolean isPlain()
public boolean isBold()
public boolean isItalic()
public boolean isUnderlined()
```

4.4.3 Style Coretan

Metode setStrokeStyle(int style) menetapkan style coretan bahwa akan digunakan untuk menggambar garis, sudut, dan kotak. Style coretan tidak mempengaruhi teks, gambar, dan operasi mewarnai.

`public void setStrokeStyle(int style)`	Mengatur style coretan untuk menggambar garis, sudut, kotak, dll
`public int getStrokeStyle()`	Mengembalikan style coretan terbaru

Nilai valid untuk style adalah SOLID dan DOTTED.

4.4.4 Clipping

Suatu bidang clipping adalah suatu kotak di dalam objek Graphics yang ada. Setiap operasi grafik hanya akan mempengaruhi pixel-pixel didalam area clip. Pixel yang berada diluar clipping tidak akan dipengaruhi oleh setiap operasi grafik.

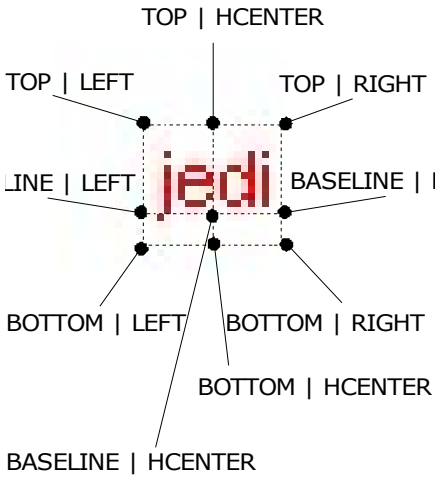
`public void` **`setClip`**`(int x, int y, int width, int height)`	**Mengatur area clip yang tersedia menjadi kotak, ditentukan oleh koordinat**
`public int` **`getClipX`**`()`	Mengembalikan offset X dari area clip yang tersedia, sehubungan dengan awal mula dari konteks grafik ini
`public int` **`getClipY`**`()`	Mengembalikan offset Y dari area clip yang tersedia
`public int` **`getClipWidth`**`()`	Mengembalikan lebar dari area clip yang tersedia
`public int` **`getClipHeight`**`()`	Mengembalikan tinggi dari area clip yang tersedia

4.4.5 Anchor Points

Teks digambar sesuai dengan sebuah anchor points. Metode drawString() mengharap sebuah koordinat (x,y) sesuai dengan anchor points.

public void **drawString**(String str, int x, int y, int anchor)

Anchor harus suatu kombinasi horisontal yang konstan (LEFT,HCENTER, atau RIGHT) dan vertikal yang konstan (TOP, BASELINE, atau BOTTOM). Horisontal dan vertikal yang konstan harus dikombinasikan mengunakan operator bitwise OR (|). Ini berarti menggambar teks berhubungan dengan baseline dan horisontal tengah akan membutuhkan sebuah nilai anchor BASELINE | HCENTER.



Gambar 3: titik anchor teks

4.4.6 Menggambar Teks

Metode untuk menggambar teks dan karakter adalah :

`public void`
`  drawString(String str,`
`  int x,`
`  int y,`
`  int anchor)`
	Menggambar teks dalam str menggunakan warna dan huruf yang tersedia. (x,y) adalah koordinat titik anchor

`public void`
`  drawSubstring(String str,`
`  int offset,`
`  int len,`
`  int x,`
`  int y,`
`  int anchor)`
	Sama seperti drawString, kecuali ini hanya akan menggambar substring dari offset (berbasis nol) dengan panjang length.

`public void`
`  drawChar(char character,`
`  int x,`
`  int y,`
`  int anchor)`
	Menggambar karakter dengan warna dan huruf yang tersedia

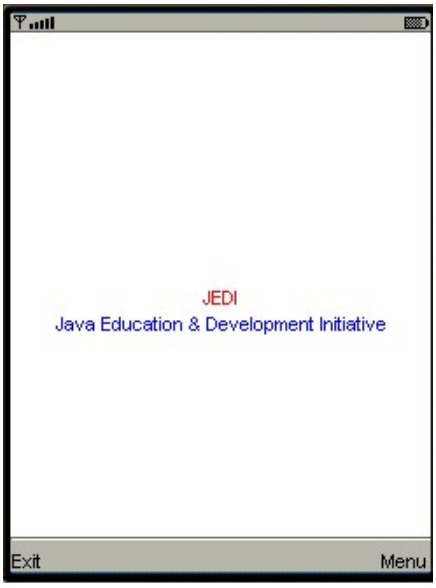
`public void`
`  drawChars(char[] data,`
`  int offset,`
`  int length,`
`  int x,`
`  int y,`
`  int anchor)`
	Menggambar karakter dalam data array karakter, dimulai dari indeks offset dengan panjang length

Berikut adalah beberapa metode dari Font yang berguna dalam menggambar teks:

`public int getHeight()`	Mengembalikan tinggi teks dalam huruf ini. Tinggi dikembalikan termasuk spasi ekstra.

	Hal ini memastikan bahwa dua teks digambar dengan jarak ini dari titik anchor ke titik anchor lainnya akan berisi cukup ruang antara dua baris teks.
`public int` **`stringWidth`**`(String str)`	Mengembalikan lebar total dalam pixel dari ruang yang ditempati oleh string ini jika digambar menggunakan huruf ini
`public int` **`charWidth`**`(char ch)`	Mengembalikan lebar total dalam pixel dari ruang yang ditempati oleh karakter ini jika digambar menggunaka huruf ini
`public int` **`getBaselinePosition`**`()`	Mengembalikan jarak dalam pixel antara TOP dan BASELINE pada teks, berdasarkan pada huruf ini

```
g.setColor(255, 0, 0); // merah

g.drawString("JEDI",

getWidth()/2, getHeight()/2,

Graphics.TOP | Graphics.HCENTER);

g.setColor(0, 0, 255); // biru

Font font = g.getFont();

g.drawString("Java Education & Development Initiative",

getWidth()/2, getHeight()/2+font.getHeight(),

Graphics.TOP | Graphics.HCENTER);
```



Gambar 4: Hasil operasi drawString()

4.4.7 Menggambar garis

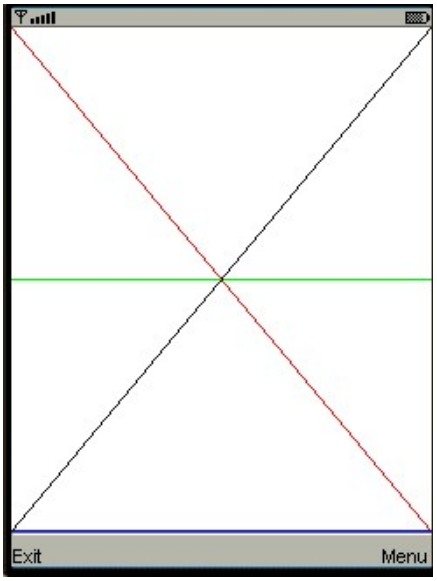
Satu-satunya metode grafik untuk menggambar garis didefinisikan sebagai :

public void **drawLine**(int x1, int y1, int x2, int y2)

Metode ini menggambar sebuah garis menggunakan warna yang tersedia dan coretan antara koordinat (x1,y1) dan (x2,y2).

```
g.setColor(255, 0, 0); // red

// garis dari pojok kiri atas ke pojok kanan bawah layar

g.drawLine(0, 0, getWidth()-1, getHeight()-1);

g.setColor(0, 255, 0); // green

// garis horisontal pada tengah layar

g.drawLine(0, getHeight()/2, getWidth()-1, getHeight()/2);

g.setColor(0, 0, 255); // blue
```

```
// garis horisontal pada bawah layar

g.drawLine(0, getHeight()-1, getWidth()-1, getHeight()-1);

g.setColor(0, 0, 0); // black

// garis dari pojok kiri bawah ke pojok kanan atas layar

g.drawLine(0, getHeight()-1, getWidth()-1, 0);
```



Gambar 5: hasil pemanggilan metode drawLine()

4.4.8 Menggambar kotak

Metode grafik untuk menggambar kotak adalah :

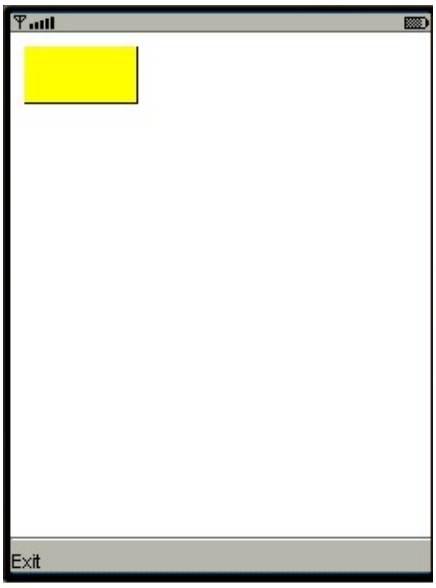
```
public void drawRect(int x, int y, int width, int height)
public void drawRoundRect(int x, int y,
                int width, int height,
                int arcWidth, int arcHeight)
public void fillRect(int x, int y, int width, int height)
public void fillRoundRect(int x, int y,
                int width, int height,
                int arcWidth, int arcHeight)
```

Metode drawRect() menggambar sebuah kotak dengan pojok kiri atas pada koordinat (x,y) dan luas area (width+1 x height+1). Parameter yang sama ada bersama

drawRoundRect(). Parameter tambahan arcWidth dan arcHeight adalah diameter horisontal dan vertikal dari busur dari keempat sudut.

Jika Anda akan mengenali, definisi drawRect dan drawRoundRect menetapkan lebar dari kotak yang digambar pada layar adalah dengan width+1 dan tingginya dengan height+1. Hal ini sangat tidak intuitif, tetapi seperti itulah spesifikasi MIDP menggambarkan metode ini. Untuk meng-agravate tidak konsistensi dari "off-by-one" ini, metode fillRect dan fillRoundRect hanya mengisi sebuah area kotak (width x height). Anda akan melihat ketidakcocokan ini jika anda memasukkan parameter yang sama untuk drawRect dan fillRect (dan drawRoundRect vs fillRoundRect). Sisi kanan dan bawah dari kotak digambar oleh kepalsuan drawRect di luar area yang diisi oleh fillRect.

```
// menggunakan tinta hitam untuk drawRect
g.setColor(0, 0, 0);
g.drawRect(8, 8, 64, 32);

// menggunakan tinta kuning untuk fillrect
// untuk menampilkan perbedaan antara drawRect dan fillrect
g.setColor(255, 255, 0);
g.fillRect(8, 8, 64, 32);
```

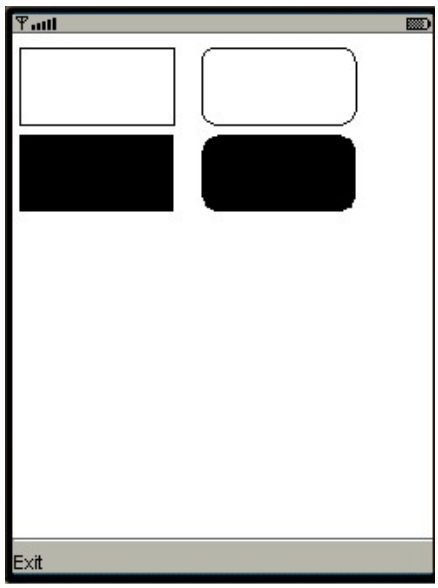


Gambar 6: hasil dari penggunaan parameter yang sama untuk drawRect dan fillRect

// mewarnai warna pena dengan warna hitam

```
g.setColor(0, 0, 0);

// menggambar kotak pada (4,8) dengan lebar 88 dan tinggi 44
```

```
// kotak pada kiri atas

g.drawRect(4,8,88,44);


// elips pada kanan atas

g.drawRoundRect(108,8,88,44,18,18);


// kotak pada kiri bawah

g.fillRect(4,58,88,44);


// elips pada kanan bawah

g.fillRoundRect(108,58,88,44,18,18);
```



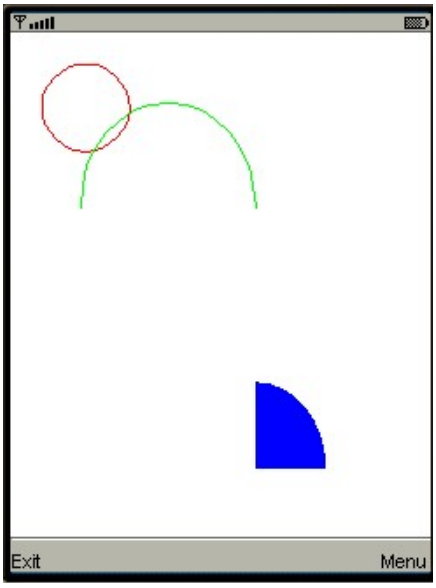
Gambar 7:drawRect(), fillRect(), drawRoundRect(), dan fillRoundRect()

4.4.9 Menggambar Sudut

Metode untuk menggambar bundar atau eclips adalah :

`public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)`	**Menggambar arc dengan pusat pada (x,y) dan dimensi (width+1 x height+1). Arc digambar mulai dari startAngle dan extend untuk derajat arcAngle. 0 derajat terletak pada jarum jam 3.**
`public void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)`	Mewarnai bidang bundar dan eclips yang telah dibuat dengan warna yang tersedia.

```
g.setColor(255, 0, 0);

g.drawArc(18, 18, 50, 50, 0, 360); // menggambar sebuah lingkaran

g.setColor(0, 255, 0);

g.drawArc(40, 40, 100, 120, 0, 180);

g.setColor(0, 0, 255);

g.fillArc(100, 200, 80, 100, 0, 90);
```



Gambar 8: Hasil pemanggilan metode drawArc dan fillArc

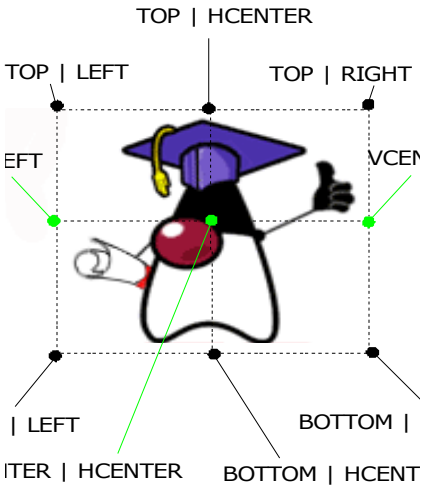
4.4.10 Melukis gambar

Gambar digambar dengan metode drawImage()

public void **drawImage**(Image img, int x, int y, int anchor)

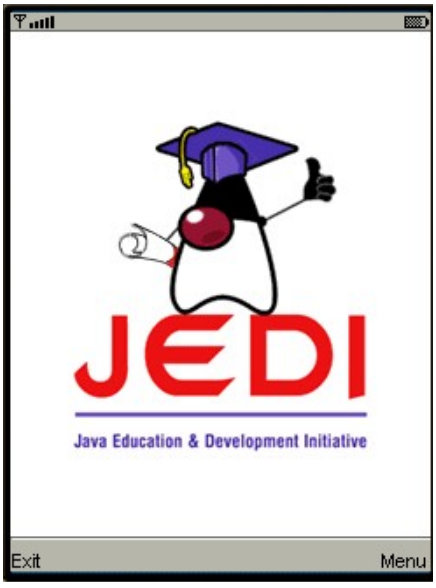
Selama dengan drawString, x dan y adalah koordinat titik anchor. Perbedaannya adalah bahwa vertikal anchor tetap adalah VCENTER yang berdasar BASELINE.

Anchor harus berupa kombinasi dari horizontal constant(LEFT, HCENTER atau RIGHT) dan vertical constant (TOP,VCENTER atau BOTTOM). Horizontal dan Vertical Constants dikombinasikan dengan menggunakan operator bitwise OR(|).



Gambar : Image Anchor Points

```
try {
       Image image = Image.createImage("/jedi.png");
       g.drawImage(image,
       getWidth()/2, getHeight()/2,
       Graphics.VCENTER | Graphics.HCENTER);
} catch (Exception e){}
```



Gambar : Output dari drawImage()

4.5 Game API

4.5.1 Game API

Aplikasi games memiliki peranan utama pada aplikasi mobile. Sebagian besar aplikasi dibuat pada pangsa pasar mobile adalah games. Action, strategy, board and card games dan sebagainya, seluruhnya terdapat pada aplikasi mobile.

Sebagian besar produsen game telah membuat API tersendiri untuk berbagai fungsi bermain game yang hanya akan bekerja pada handset yang dibuat oleh perusahaan tersebut. Hal ini berarti bahwa sebuah game yang dibangun menggunakan API dari salah satu produsen tidak akan berjalan pada device hasil produksi dari produsen lain.

Untuk menjembatani perbedaan ini, MIDP versi 2 telah memiliki fungsionalitas dasar yang secara spesifik ditujukan aplikasi game.

Class utama Game API dari MIDP adalah class GameCanvas. Class GameCanvas merupakan perluasan dari class Canvas yang kita gunakan dalam pembuatan low – level user interface. Dua kelemahan utama dari class Canvas dalam pemrograman game adalah tidak memadainya kemampuan untuk mengatur proses repaint dan ketidakmampuan untuk mengatur bagaimana pointer events serta quick keys diteruskan pada canvas.

Komponen user interface dari MIDP umumnya berupa event driven. Events berupa antrian berurutan dan diteruskan terhadap aplikasi satu persatu, beserta tunda waktu antar waktu dimana event dibuat (key press).

GameCanvas memungkinkan aplikasi mengumpulkan events yang terbuat dan melakukan proses repaint pada canvas dengan cepat. Struktur program menjadi lebih bersih karena terdapat rangkaian perulangan utama dimana proses painting dan pengumpulan events dilakukan.

GameCanvas menggunakan teknik double buffering. Seluruh proses pembuatan interface dilakukan di off-screen buffer, kemudian di transfer dari area buffer tersebut menuju area yang terlihat pada canvas. Aplikasi anda harus menggunakan instance method dari class Graphics berupa method getGraphics(). Setiap pemanggilan terhadap method ini mengembalikan sebuah instance baru dari off-screen buffer yang anda gunakan dalam proses pembuatan user interface.

Untuk memperbaharui screen tersebut, anda harus memanggil flushGraphics() untuk melakukan proses repaint secara cepat dengan isi dari off-screen buffer. Perhatikan bahwa anda hanya perlu memanggil method getGraphics() sekali saja, karena sebuah buffer teralokasi setiap kali anda memanggil method ini.

MyCanvas.java :

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class MyCanvas extends GameCanvas implements Runnable {
	private boolean running;
	private long delay;
	private int currentX, currentY;
	private int screenWidth;
	private int screenHeight;

	public MyCanvas() {
		super(true);
		screenWidth = getWidth();
		screenHeight = getHeight();
		currentX = screenWidth / 2;
		currentY = screenHeight / 2;
		delay = 20;
	}

	public void start() {
		running = true;
		Thread thread = new Thread(this);
		thread.start();
	}
	public void stop() { running = false; }
		// The Game Loop
		public void run() {
		Graphics g = getGraphics();
		while (running == true) {
		getInput();
		drawScreen(g);
		try { Thread.sleep(delay); } catch (InterruptedException ie) {}
		}
	}

	private void getInput() {
		int keyStates = getKeyStates();
		if ((keyStates & LEFT_PRESSED) != 0) {
			currentX = Math.max(0, currentX - 1);
		}
		if ((keyStates & RIGHT_PRESSED) != 0) {
			currentX = Math.min(screenWidth, currentX + 1);
		}
		if ((keyStates & UP_PRESSED) != 0) {
			currentY = Math.max(0, currentY - 1);
		}
		if ((keyStates & DOWN_PRESSED) != 0) {
			currentY = Math.min(screenHeight, currentY + 1);
		}
	}

	private void drawScreen(Graphics g) {
		g.setColor(0xffffff);
		g.fillRect(0, 0, getWidth(), getHeight());
		g.setColor(0x0000ff);
		g.drawString("JEDI", currentX, currentY,
		Graphics.TOP|Graphics.LEFT);
		flushGraphics();
	}
}
```

GameMidlet.java:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class GameMidlet extends MIDlet {
	private Display display;
	public void startApp() {
		display = Display.getDisplay(this);
		MyCanvas gameCanvas = new MyCanvas();
		gameCanvas.start();
		display.setCurrent(gameCanvas);
	}
	public Display getDisplay() {
		return display;
	}
	public void pauseApp() {
	}
	public void destroyApp(boolean unconditional) {
		exit();
	}
	public void exit() {
		System.gc();
		destroyApp(false);
		notifyDestroyed();
	}
}
```

4.5.2 Layers

Layers adalah elemen visual dari sebuah screen. Layer adalah abstract class yang merepresentasikan objects pada screen. Sprite dan TiledLayer adalah subclasses dari class Layer.

Tiled Layer adalah rangkaian dari beberapa persegi empat yang berukuran sama dan gambar – gambar yang memadai untuk ditempatkan pada persegi empat tersebut. Layer ini dibangun dengan menempatkan gambar – gambar dan elemen – elemen visual dalam area ini. Sebuah gambar dapat digunakan oleh lebih dari satu persegi empat sehingga dapat menghemat ruang dan memory. Tiled Layers umumnya digunakan sebagai background pada game.

4.5.3 Sprites

Sprites adalah objects grafis yang anda lihat pada game. Object ini dapat berupa character, kunci, tombol, pintu ataupun peluru. Sebuah sprite bersifat statis ataupun animasi.

Animasi sprite terbuat dari beberapa elemen sprite dengan perbedaan – perbedaan kecil dan tersusun sedemikian hingga membentuk kesan bergerak. Rangkaian sprite ini disebut sebagai frame.

Contoh kode berikut ini (dari Project Game2) mendemonstrasikan cara penggunaan sprites. Program ini menggunakan sprite sederhana dengan dua frame. Frame ditampilkan menurut penekanan tombol oleh user (UP, DOWN, LEFT atau RIGHT).

GameCanvas dengan Sprite :

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class MyCanvas extends GameCanvas implements Runnable {
	private boolean running;
	private long delay;
	private int currentX, currentY;
	private int screenWidth;
	private int screenHeight;
	private Sprite sprite;

	public MyCanvas() throws Exception {
		super(true);
		screenWidth = getWidth();
		screenHeight = getHeight();
		currentX = screenWidth / 2;
		currentY = screenHeight / 2;
		delay = 20;
		Image image = Image.createImage("/jedi.png");
		sprite = new Sprite(image, 32, 32);
	}

	public void start() {
		running = true;
		Thread thread = new Thread(this);
		thread.start();
	}
	public void stop() { running = false; }
	// The Game Loop
	public void run() {
	Graphics g = getGraphics();
	while (running == true) {
		getInput();
		drawScreen(g);
		try { Thread.sleep(delay); } catch (InterruptedException ie) {}
		}
	}

	private void getInput() {
		int keyStates = getKeyStates();
		sprite.setFrame(0);
		if ((keyStates & LEFT_PRESSED) != 0) {
			currentX = Math.max(0, currentX - 1);
			sprite.setFrame(0);
		}
```

```
            if ((keyStates & RIGHT_PRESSED) != 0) {
                currentX = Math.min(screenWidth, currentX + 1);
                sprite.setFrame(1);
            }
            if ((keyStates & UP_PRESSED) != 0) {
                currentY = Math.max(0, currentY - 1);
                sprite.setFrame(1);
            }
            if ((keyStates & DOWN_PRESSED) != 0) {
                currentY = Math.min(screenHeight, currentY + 1);
                sprite.setFrame(0);
            }
        }

        private void drawScreen(Graphics g) {
            g.setColor(0xffffff);
            g.fillRect(0, 0, getWidth(), getHeight());
            g.setColor(0x0000ff);
            sprite.setPosition(currentX, currentY);
            sprite.paint(g);

            flushGraphics();
        }
}
```

4.5.4 LayerManager

Class LayerManager akan memberikan kemudahan dalam pengaturan keseluruhan Sprites dan TiledLayers. LayerManager mengatur seluruh Sprites dan TiledLayers yang anda buat. Dan anda tidak perlu untuk membuat seluruh object tersebut satu persatu. LayerManager yang akan melakukannya untuk anda. LayerManager juga mengatur pengurutan objek dari dasar hingga paling atas.

4.6 Scalable 2D Graphics

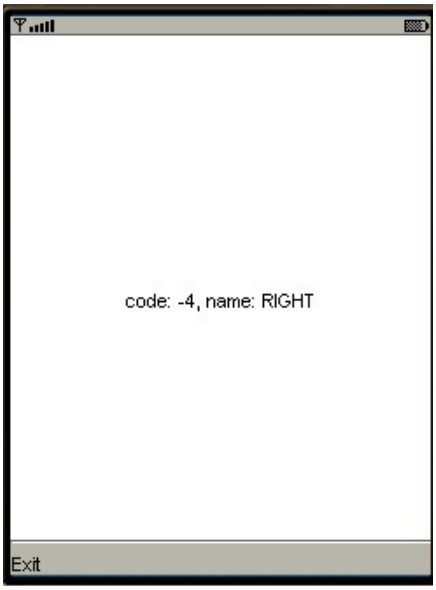
JSR 226 menyediakan method untuk proses rendering dan transforming atas grafis vector-based 2D.

Format gambar raster-based seperti GIF melakukan proses encode terhadap pewarnaan pada tiap-tiap pixel pada area persegi empat yang menunjukkan bentuk gambar. Gambar dengan tipe vector-based hanya memiliki instruksi penggambaran yang menentukan bagaimana pixel-pixel dari gambar tersebut harus diwarnai. Vector tersebut merepresentasikan sebuah gambar yang berukuran jauh lebih kecil, sebuah nilai lebih dalam penggunaan resource pada mobile devices.

4.7 Latihan

4.7.1 Key Codes

Buatlah sebuah MIDlet yang akan menampilkan kode dan nama dari tombol yang ditekan oleh user. Gunakan sebuah Canvas dan tempatkan keterangan kode dan nama tepat di tengah dari tampilan pada layer.


code: -4, name: RIGHT
Exit