Game Canvas, Thread, Sprite

Southeast Asian Ministers of Education Organization
Regional Open Learning Center (SEAMEO SEAMOLEC)

Game Canvas

MOBILE GAME
J2ME

Latar Belakang

Masalah yang dihadapi ketika menggunakan class Canvas :

- Sulit mengatur layer
- Sulit meramalkan kapan system akan memanggil method paint()
- Bisa saja terjadi delay pada saat menangkap key event.

GameCanvas digunakan untuk memperluas fungsi yang dimiliki oleh class Canvas

GameCanvas

GameCanvas merupakan sebuah class yg terdapat pada package **javax.microedition.lcdui.game.***

GameCanvas menyediakan method getKeystate() untuk membaca key-key yang ditekan.

GameCanvas VS Canvas

Ada beberapa keuntungan yg ditawarkan oleh class GameCanvas dibandingkan dengan class Canvas:

1. Memiliki kontrol terhadap aplikasi, saat display mengalami update. Menunggu AMS memanggil paint().
2. Dapat mengontrol region/bagian dari screen yang akan di update.
3. Menyediakan suatu mekanisme *easy-to-use(*polling) untuk melakukan query current key yang sedang ditekan oleh user.
4. Menggunakan teknik double buffering untuk menghindari kedipan(flicker) pada animasi.

GameCanvas VS Canvas

Perbedaan antara Canvas dan GameCanvas :

- Untuk menggunakan Canvas, buat subclassnya, definisikan method **paint()**.
- Di dalam method paint(), gunakan Graphics untuk me-render gambar pada screen.
- Ketika ingin mengubah sesuatu dan mengupdate-nya pada screen, maka harus memanggil method **repaint()**, kemudian system memanggil paint() sekali lagi untuk penggambaran.

GameCanvas VS Canvas

Perbedaan antara Canvas dan GameCanvas :

- Untuk menggunakan GameCanvas, buat subclassnya. Untuk menggambar pada screen, gunakan Graphics melalui method **getGraphics()**.
- Untuk mengupdate gambar muncul pada screen, panggil **flushGraphics()**.
- Untuk update yang lebih pesifik, gunakan method **flushGraphics(int x, int y, int width, int height)**, yang hanya meng-update region/bagian tertentu dari screen.

Game Loop

Sebuah game atau animasi dibangun dengan cara mengulang berjalannya sekumpulan program.

Pada kondisi loop, value dari instant variable dicek dan kemudian meng-update game state.

Berdasarkan game state, code kemudian digambar/paint/repaint pada game screen.

Game loop merupakan suatu loop yg tidak terhingga.

Code pada loop harus mengikuti eksekusi dari current thread, yaitu memenuhi kondisi sleep setiap beberapa millisecond untuk mengontrol rate permainan.

Game Loop

```
Graphics g = getGraphics();
while(true){
	//cek input dari user
	//update game state
	//menggambar sesuatu menggunakan g
	flushGraphics();
}
```

Keystate Polling

GameCanvas menawarkan sebuah method alternatif untuk merespon key press, yang diharapkan menjadi cara bagi user untuk dapat mengontrol game.

GameCanvas menyediakan sebuah method yang mengembalikan nilai dari *current state* dengan menggunakan method berikut:

public int getKeyStates()

Keystate Polling

Daripada menunggu system untuk memanggil key callback method pada Canvas, anda dapat langsung mencari tahu state dari device key.

Integer yg direturn menggunakan satu bit untuk merepresentasikan masing-masing dari 9 game action. Sebuah bit satu mengindikasikan sebuah key press, sementara sebuah bit nol mengindikasikan tidak ada key press. Masing-masing bit direpresentasikan oleh sebuah constant/konstanta di dalam GameCanvas.

Dengan mengambil current state dari key(sebuah teknik yg dinamakan **Polling**), anda dapat merespon user action di dalam game loop.

GameCanvas Bit Constant

GameCanvas Bit Constant	Corresponding GameCanvas Action Constant
UP_PRESSED	UP
DOWN_PRESSED	DOWN
LEFT_PRESSED	LEFT
RIGHT_PRESSED	RIGHT
FIRE_PRESSED	FIRE
GAME_A_PRESSED	GAME_A
GAME_B_PRESSED	GAME_B
GAME_C_PRESSED	GAME_C
GAME_D_PRESSED	GAME_D

Game Loop(2)

```
Graphics g = getGraphics();
while(true) {
        // Cek user input.
        int ks = getKeyStates();
        if ((ks & UP_PRESSED) != 0)
                moveUp();
        else if ((ks & DOWN_PRESSED) != 0)
                moveDown();
        // ...
        // Update game state.
        // menggambar sesuatu menggunakan g.
        flushGraphics();
}
```

End of slides

Southeast Asian Ministers of Education Organization
Regional Open Learning Center (SEAMEO SEAMOLEC)

THREAD

MOBILE GAME
J2ME

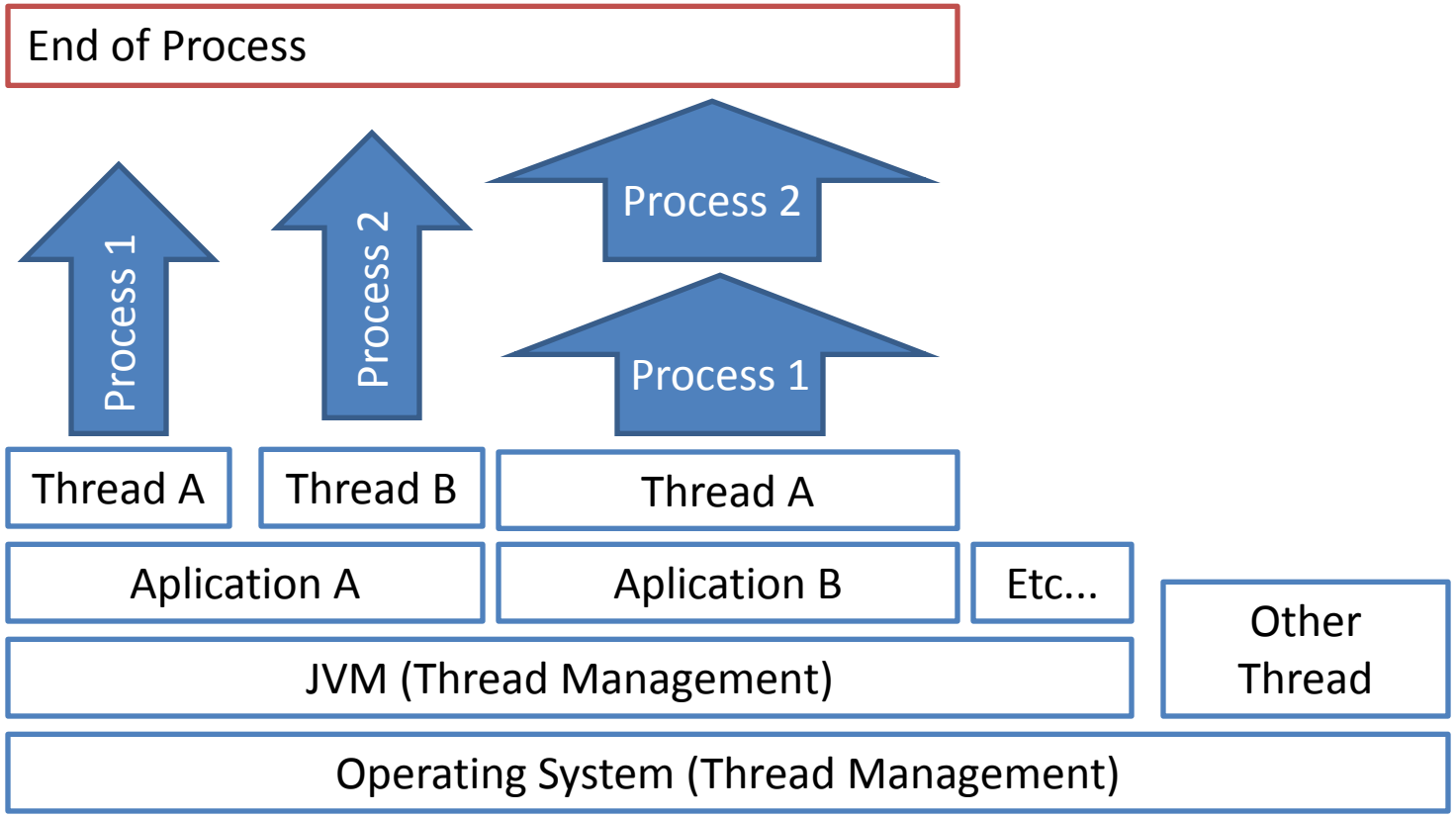
Tujuan

- Mengenal Class Thread
- Mengetahui Fungsi-Fungsi Pada Class Thread
- Mengetahui State Pada Thread
- Menggunakan Class Thread dalam Aplikasi
- Mengetahui dan memanfaatkan Class Counter sebagai timer

Mengenal class *Thread*

- *Thread* memungkinkan sebuah aplikasi *Java* untuk melakukan banyak aktivitas/operasi secara simultan (serentak).
- *Thread* adalah unit fundamental dari eksekusi program. Setiap aplikasi minimal memiliki sebuah *thread* untuk menjalankan kode.
- Aplikasi yang memiliki dua *thread* atau lebih, biasa disebut dengan *multithread application*.

Mengenal class *Thread*



Thread State

- *Running*, saat *thread* sedang menjalankan kode. (*live thread*).
- *Ready*, saat *thread* siap untuk mengeksekusi kode. (*live thread*).
- *Suspended*, saat *thread* sedang menunggu *external event.* Contoh: menunggu data yang datang dari *device* lain. Begitu data datang(terjadi *event)*, maka *thread* kembali ke *state ready*. (*live thread*).
- *Terminated*, saat *thread* selesai mengeksekusi kode. (*dead thread*).

Method Pada Thread

- **run()**

Class harus mengimplementasi Runnable dan mengoverride method run supaya dapat menjalankan method ini.

- **setPriority(...)**

Merubah prioritas pada sebuah object thread yang dibuat.

- **sleep(...)**

Menghentikan sementara proses dalam aplikasi dimana satuan yang digunakan adalah mili detik.

- **start()**

Menjalankan object Thread.

- **yield()**

Menunda dan mengijinkan thread lain untuk melakukan eksekusi

Inherit Class Thread

Membuat sebuah class menjadi turunan dari class Thread, dimana sub class yang dibuat harus meng-override method run.

```
class MyThread extends Thread{
     public static void main(String args[]){
          // Membuat object dan menjalankannya:
          MyThread p = new MyThread();
          p.start();
     }
     public MyThread(){ }
     public void run(){
          // operasi menghitung skor game
     }
}
```

Implement Runnable

Mengimplementasikan interface Runnable pada sebuah class yang dibuat dan harus meng-override *method run* dari interface *tersebut.*

```
class MyThread implements Runnable{
     public static void main(String args[]){
          // Membuat object dan menjalankannya:
          MyThread p = new MyThread();
     }
     private Thread t;
     MyThread() {
          t=new Thread(this);
          t.start();
     }
     public void run() {
               // operasi menghitung skor game
     }
}
```

Menghentikan Thread

Memberikan flag untuk menghentikan thread yang sedang berjalan.

```
public class MyThread implements Runnable{
    private boolean selesai = false;
    public void run(){
        while( !selesai ){
            // lakukan operasi yang diperlukan
        }
    }
    public void stop(){
          selesai = true;
    }
}
```

Menggunakan Class Counter

Method	Keterangan
setCount(int count)	Setting nilai awal conter, nilai defaultnya adalah 0
setInc(int inc)	Setting jumlah yang akan ditambahkan setiap kali update counter nilai defaultnya 1
setActive(boolean active)	Mengaktifkan counter dafaultnya false
getInc()	Mengambil nilai increament
getCount()	Mengambil nilai hasil update counter
String convertToTime()	Mengkonversi menjadi time hh:mm:ss
int getHours()	Mengambil jumlah jam
int getMinutes()	Mengambil jumlah menit
int getSecond()	Mengambil jumlah detik

Source Class Counter

```
import java.util.TimerTask;
public class Counter extends TimerTask {

   private int count;
   private int inc = 1;
   private boolean active;

   public void run() {
      if (active) {
         count += inc;
      }
   }

   public void setCount(int count) {
      this.count = count;
   }
```

Source Class Counter

```
private int hours = 0, minutes = 0, second = 0;

   public String convertToTime() {
      tmp = count;
      hours = tmp / 3600;
      tmp = tmp % 3600;
      minutes = tmp / 60;
      second = tmp % 60;
      return "" + hours + ":" + minutes + ":" + second;
   }
```

Source Class Counter

```
    public void setInc(int inc) {
        this.inc = inc;
    }

    public int getInc() {
        return inc;
    }

    public int getCount() {
        return count;
    }

    public void setActive(boolean active) {
        this.active = active;
    }
}
```

Scheduling Class Counter

```
private Timer t;
private Counter count;
-----------------------------------------------------------------------------------
    timer = new Timer();
    count = new Counter();
    count.setActive(true);
    count.setCount(10); // inisialisasi counter
    // nilai positif untuk increament dan negatif untuk decreament
    count.setInc(-1);
    timer.schedule(count, 1000, 1000);
-----------------------------------------------------------------------------------
System.out.println("Time : "+count.getCount());
-----------------------------------------------------------------------------------
g.drawString("Time : "+count.getCount(),50,50,0);
```

End of slides

Southeast Asian Ministers of Education Organization
Regional Open Learning Center (SEAMEO SEAMOLEC)

SPRITE

MOBILE GAME
J2ME

Tujuan

- Mengenal dan menggunakan class Sprite
- Memanipulasi graphics dengan menggunakan method yang ada pada class *sprite*
- Membuat animasi dengan *class Sprite.*

Apa itu Sprite?

- *Sprite* adalah istilah yang sering muncul untuk pengolahan grafis animasi pada sebuah Game
- Sebagian besar objek pada sebuah game dikategorikan sebagai grafis khusus yang disebut sprites.
- Sprite dapat berupa karakter utama, *peluru, monster*, musuh, kekuatan spesial, kunci, pintu, dst.

Frame For Sprite

Frame dapat dibuat secara terurut ataupun tidak terurut, namun pada umumnya *sprites* ditampilkan secara terurut untuk memudahkan pengkodean.

Contoh Sebuah *Sprite*

Contoh Kumpulan beberapa *Sprite*

image diatas adalah ilustrasi dari kumpulan *sprite* yang menampilkan frame tunggu dan 4 *frame* dasar lainnya.

Constructor Sprite

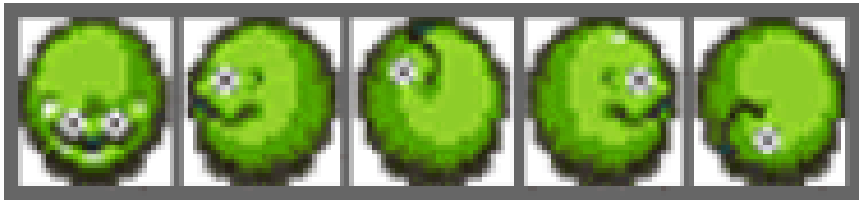
Sprite (Image image) – membuat frame sprite tunggal, tidak dianimasikan.

Sprite (Sprite sprite) – membuat sprite baru dari sprite lainnya

Sprite (Image image, int frameWidth, int frameHeight)– membuat animasi *sprite* lebih dari 2 frame, *frameWidth* adalah lebar dari sebuah *sprite* dan *frame Height* adalah tinggi dari sebuah *sprite*.

Sprite

- Kita dapat memecah kumpulan sprite menjadi frame secara tersendiri.
- Pada contoh ini total lebar kumpulan sprite adalah 160 pixel
- Dibagi dengan 5 sehingga menjadi 5 buah frame dengan lebar masing-masing 32 pixel. Tinggi masing-masing frame adalah 32 pixel.
- Tinggi dan lebar tidaklah selalu sama, tetapi lebar dan tinggi harus konstan untuk semua sprite pada kumpulan sprite.

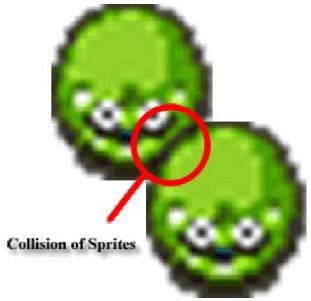


Mengapa 8 Bit, 16 Bit, 32 Bit?

- Kebanyakan grafis termasuk sprite didalamnya, biasanya akan mengalami ketidaksesuaian tampilan antara batas lebar dan tinggi layar pada device mobile.
- Dikarenakan jumlah pixel yang digunakan berhubungan dengan jumlah warna yang digunakan.
- Mengacu pada kedalaman bit atau kedalaman warna.
- Banyak bit per pixel berarti banyak pula warna yang terdapat didalamnya.

Sprite Collision

Deteksi tabrakan sprite bertujuan sebagai salahsatu indikasi penambahan level (level ups), kekuatan (power ups), kehilangan tenaga, membuka sebuah pintu, atau indikator yang mengijinkan pemain untuk menaiki tangga.



collidesWith(Image image, int x, int y, boolean pixelLevel)

atau

collidesWith(Sprite sprite, boolean pixelLevel)

Sprite Collision

- ✓Parameter yang melewatkan tipe data Image perlu menentukan posisi image tersebut, hal ini mengacu pada x dan y pada sudut kiri atas image.
- ✓*pixelLevel* adalah sebuah nilai *boolean* dimana nilai true mengindikasikan deteksi level dari *pixel* dan *false* mengindikasikan perpotongan persegi.

```
collidesWidth(TiledLayer tiledLayer, Boolean pixelLevel)
```

Sprite Sequence

getFrameSequenceLength()

Jumlah elemen pada sebuah rangkan frame

getRawFrameCount()

Untuk mengambil urutan pada gambar sprite.

getFrame()

mendapatkan kembali indeks angka pada rangkaian frame, tetapi ini tidak berlaku untuk frame yang saat ini telah ditampilkan

nextFrame()

menset rangkaian frame pada frame selanjutnya, jika rangkaian berada pada frame terakhir maka akan diset menjadi frame pertama.

prevFrame()

menset rangkaian frame pada frame sebelumnya, jika rangkaian berada pada frame pertama maka akan diset menjadi frame terakhir.

setFrame(int sequenceIndex)

untuk menset rangkaian frame secara manual.

setFrameSequence(int[] sequence)

untuk menset frame yang belum ditetapkan secara manual.

Transparan VS Non Transparan Sprite

Berhati-hatilah pada penggunaan image transparan dan non transparan.

Game sederhana seperti *TicTacToe*, image transparan mungkin saja tidak penting atau tidak diperlukan.

Pada game interaktif tingkat tinggi dimana ada banyak potensi untuk tabrakan antar *sprite* dan image background yang bervariasi,

Transformasi Sprite

Method *setTransform(transform) melewatkan parameter* bertype *integer.*

Static Fields	Integer Value
TRANS_NONE	0
TRANS_MIRROR_ROT180	1
TRANS_MIRROR	2
TRANS_ROT180	3
TRANS_MIRROR_ROT270	4
TRANS_ROT90	5
TRANS_ROT270	6
TRANS_MIRROR_ROT90	7

End of slides