BAB 5
SPRITE

5.1 TUJUAN

Setelah mempelajari bab ini, siswa diharapkan mampu untuk :

- Memahami dan menggunakan *class sprite*
- Menggambar dan menyusun *sprite*
- Memanipulasi graphics dengan menggunakan method yang ada pada class *sprite*
- Dapat membuat animasi dengan *class sprite.*

5.2. PENGENALAN SPRITE

Salah satu faktor yang membuat keberhasilan game adalah grafis yang unggul. Sebagian besar objek pada sebuah game dikategorikan sebagai grafis khusus yang disebut sprites. Sebuah sprite dapat berupa *bullet, monster*, karakter utama, musuh, kekuatan spesial, kunci dan pintu.

Contoh sebuah Sprite

Pada umumnya *Sprite* adalah grafis animasi. Grafis animasi dibuat dari *sprite* yang sama namun berbeda penampakannya. Kumpulan sprite biasanya mengacu sebagai sebuah kumpulan *frame. Frame* ini dapat dibuat secara terurut ataupun tidak terurut. Umumnya *sprites* ditampilkan secara terurut untuk memudahkan pengkodean.

Contoh kumpulan beberapa sprite

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

5.3. SPRITE CONSTRUCTOR

Ada 3 konstruktor yang disediakan oleh kelas sprite

1. *Sprite (Image image)* – membuat frame sprite tunggal, tidak dianimasikan.
2. *Sprite (Sprite sprite)* – membuat sprite baru dari sprite lainnya
3. *Sprite (Image image, int frameWidth, int frameHeight)* – membuat animasi *sprite* dengan lebih dari 2 frame, *frameWidth* adalah lebar dari sebuah *sprite* dan *frameHeight* adalah tinggi dari sebuah *sprite*.

Contoh kumpulan Sprite dengan bingkai di sekelilingnya

Sesuai dengan gambar di atas, kita dapat memecah kumpulan sprite menjadi frame secara tersendiri. Pada contoh ini total lebar kumpulan sprite adalah 160 pixel, yang 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. Dengan kata lain, Anda tidak bisa memiliki sebuah frame dengan lebar 32 pixel dan sprite yang tersisa hanya memiliki lebar 16 pixel, semua fame harus memiliki lebar yang sama. Di dalam konstruktor Sprite (Image image, int frameWidth, int frameHeight) Anda akan diberitahu bahwa Anda tidak dapat menetapkan angka pada frame, hal ini secara otomatis dihitung oleh kelas Sprite.

5.4. MENGAPA 8 BIT, 16 BIT, 32 BIT?

Anda akan diberitahu bahwa kebanyakan grafis termasuk sprite didalamnya, biasanya akan mengalami ketidaksesuaian tampilan antara batas lebar dan tinggi layar. Hal ini dikarenakan jumlah pixel yang digunakan berhubungan dengan jumlah warna yang digunakan. Ini mengacu pada kedalaman bit atau kedalaman warna. Banyak bit per pixel berarti banyak pula warna yang terdapat didalamnya.

Rumusnya :

$$2^{\#\,bit} = \text{jumlah warna}$$

Gunakan rumus di atas untuk mengukur kedalaman 8, 16, 24 dan 32 bit, sehingga hasilnya :

2^8 = 256 warna

2^{16} = 65.536 warna

2^{24} = 16,7 juta warna

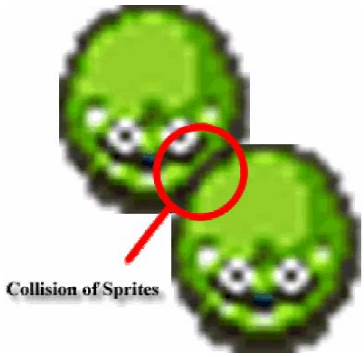
2^{32} = 16,7 juta warna ditambah 8 bit channel alpha, sebuah channel alpha merupakan sebuah mask, ini menentukan bagaimana sebuah warna pada satu pixel ketika digabungkan dengan pixel yang lain. Penggabungan terjadi ketika satu pixel berada di atas pixel lainnya.

Walaupun demikian, Anda dapat memiliki sprite dengan ukuran lebih besar dari 8 pixel dan 256 warna. Idealnya, ini memungkinkan Anda untuk mencari pilihan terbaik ketika berhubungan dengan grafis untuk perangkat mobile. Semakin banyak warna yang digunakan maka semakin banyak pula proses yang diperlukan untuk me-render grafis.

5.5. SPRITE COLLISION (TUBRUKAN SPRITE)

Tubrukan sprite adalah satu dari fungsi dasar pada berbagai aksi interaktif seperti pada game. Diantaranya adalah sprite yang berlari di dinding, sprite yang menembakkan sebuah peluru atau dua sprite saling mendahului satu sama lain. Deteksi tubrukan tidak diperlukan ketika suatu game berahir. Jadi tubrukan sprite jumlahnya tidak terbatas seperti tambahan level (level ups), tambahan kekuatan (power ups), kehilangan tenaga pada game balapan, membuka sebuah pintu, atau indikator yang mengijinkan pemain untuk menaiki tangga.

Jadi bagaimana caranya mendeteksi tubrukan sprite? Tentu saja kita akan deteksi jika pixel dari sebuah sprite saling melengkapi/bertubrukan dengan pixel lain dari sprite lainnya.



Tubrukan Sprite

Untungnya, ada kelas sprite yang dapat menangani hal tersebut.

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

atau

```
collidesWith(Sprite sprite, boolean pixelLevel)
```

Seperti yang anda lihat Anda dapat mendeteksi tubrukan dengan *sprite* lainnya atau dengan sebuah gambar. Metode dengan gambar sebagai sebuah masukan, Anda perlu menentukan lokasi gambar tersebut, hal ini mengacu pada x dan y pada sudut kiri atas gambar. *pixelLevel* adalah sebuah nilai *boolean* dimana nilai true (benar) mengindikasikan deteksi level dari *pixel* dan *false* (salah) mengindikasikan perpotongan persegi. Anda dapat menetapkan ukuran dari perpotongan persegi, Anda juga boleh menetapkan bahwa perpotongan persegi lebih kecil daripada gambar itu sendiri. Hal ini akan menghilangkan area yang berlebih pada sprite dimana tubrukan terjadi dikarenakan tidak terlihatnya tubrukan persegi tetapi disana tidak ada *pixel* buram yang telah bertubrukan.

Ini adalah metode yang ketiga :

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

Seperti pada sprite bertemu dengan background TiledLayer yang miring maka sprite akan bergerak naik dengan mengikuti arah kemiringan background TiledLayer

Ini serupa dengan dua metode sebelumnya. *TiledLayer* akan dijelaskan secara detil pada seksi selanjutnya.

5.6. TAMPILAN SPRITE

Untuk menampilkan atau membuat sprite secara sederhana dengan memanggil metode paint, Anda perlu menandai objek *Graphics* adalah parameter yang diperlukan. Mengingat bahwa Anda harus menggunakan metode *setVisible(boolean)* yang pertama dipanggil pada metode paint, Anda perlu menentukan agar nilai boolean didalamnya adalah true.

5.7. URUTAN TAMPILAN SPRITE

Ada beberapa metode yang tersedia :

- ✓ *getFrameSequenceLength()*
  - o mengembalikan jumlah elemen pada sebuah rangkan frame
- ✓ *getRawFrameCount()*
  - o Untuk mengambil urutan pada gambar sprite.
- ✓ *getFrame()*
  - o mendapatkan kembali indeks angka pada rangkaian frame, tetapi ini tidak berlaku untuk frame yang saat ini telah ditampilkan
- ✓ *nextFrame()*
  - o menset rangkaian frame pada frame selanjutnya, jika rangkaian berada pada frame terakhir maka akan diset menjadi frame pertama.
- ✓ *prevFrame()*
  - o menset rangkaian frame pada frame sebelumnya, jika rangkaian berada pada frame pertama maka akan diset menjadi frame terakhir.
- ✓ *setFrame(int sequenceIndex)*
  - o untuk menset rangkaian frame secara manual.
- ✓ *setFrameSequence(int[] sequence)*
  - o untuk menset frame yang belum ditetapkan secara manual.

Untuk lebih jelasnya lihat J2ME API.

5.8. KELEMAHAN SPRITE

Anda perlu berhati-hati pada penggunaan gambar transparan dan tidak transparan. Game sederhana seperti *TicTacToe*, gambar transparan mungkin saja tidak penting atau diperlukan. Pada game interaktif tingkat tinggi dimana ada banyak potensi untuk tubrukan antar *sprite* dan gambar latar belakang yang bervariasi, Anda mungkin akan

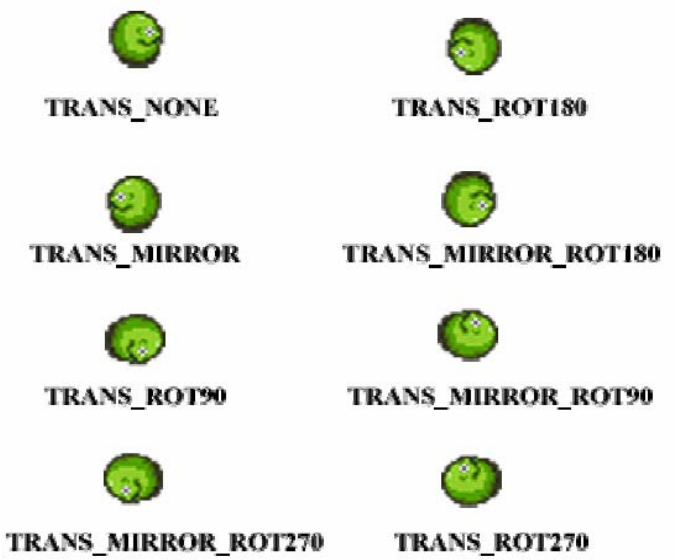
memepertimbangkan untuk menggunakan gambar transparan. Sebaiknya Anda mencek apakah perangkat mobile yang akan Anda gunakan mendukung gambar transparan jika Anda ingin menggunakannya.

Non Transparan Sprite vs Transparan Sprite

5.9. TRANSFORMASI SPRITE

Ada metode yang dapat memanipulasi rotasi *Sprite*. Transformasi diset dengan memakai metode *setTransform(transform).* Parameternya bernilai *integer*; walaupun demikian, ada nilai statis yang tersedia yang dapat digunakan:

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



5.10. OPTIMALISASI SPRITE

Optimalisasi gambar biasanya merupakan proses menghilangkan warna yang tidak dapt dilihat oleh mata manusia. Keuntungan dari optimalisasi adalah untuk mereduksi ukuran gambar, yang dikemas dalam file *jar*.

Contoh sederhana pada *emulator Sprite*

5.11. PERLUASAN KELAS SPRITE

Ada beberapa alasan mengapa anda boleh mewariskan kelas *sprite*, yaitu :

1. Dapat menambah kegunaan *sprite*
2. Kelas *sprite* dapat digunakan seperti kelas *sprite* dasar pada semua game *sprite*
3. Meringkas fungsi *sprite*

Mengacu pada point pertama, Anda mungkin menemukan kelas *sprite* telah disediakan oleh sun, tidak semua fitur yang ada akan Anda gunakan, sebagai contoh ketika Anda membuat game balap mobil. Pada game balapan kebanyakan kecepatan

adalah sebuah faktor, Jadi Anda boleh mewariskan kelas *sprite* dan menambahkan variabel yang diperlukan dan metode yang perlu digunakan.

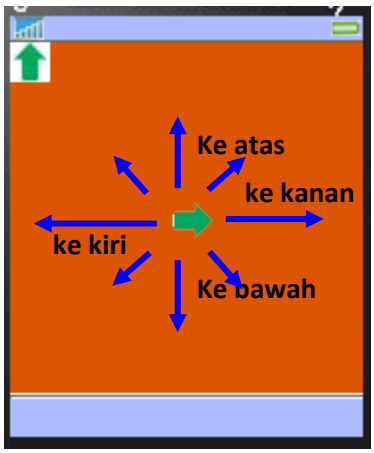
Contoh lainnya yaitu dimana Anda ingin mewariskan kelas *Sprite* adalah *Say* misalnya, anda memiliki beberapa jenis *sprite* yang berbeda tetapi setiap jenis memiliki karakteristik yang telah ditentukan. Sebagai contoh, say adalah sprite yang bertindak sebagai pemain memiliki atribut seperti tenaga, ketangkasan, kecepatan, kecerdasan dan sprite yang bertindak sebagai musuh memiliki atribut yang mengindikasikan ragam musuh yang ada dan bar kehidupan. Kedua *sprite* tersebut masih memberitahukan karakteristik yang sama seperti posisi dan nama mereka. Anda sekarang dapat membuat sebuah kelas *sprite* dasar yang berisi fitur umum untuk semua *sprite*, sebagaimana membuat *sprite child* yang berisi kedua karakteristik mereka dan sifat yang diwariskan masih disimpan pada karakteristik umum.

Anda mungkin telah diberitahu pada contoh sederhana sprite pergerakan *sprite* hanyalah kiri, kanan, atas dan bawah. Jika Anda mencoba untuk menekan tombol 2 sekali sepertimengarah pada kanan dan atas sehingga menciptakan sebuah arah sudut, ini adalah salah satu contoh dari dasar arah lainnya. Untuk menciptakan gerakan ke arah barat daya, tenggara, barat laut atau timur laut, Anda perlu mendefinisikan sprite yang tepat dan kalkulasi yang tepat. Ini hanyalah beberapa dasar manipulasi menggunakan perhitungan matematika, tetapi yang lebih penting manakah yang harus dikerjakan? Untuk menajaga kode Anda menjadi sedikit lebih teratur dan bersih mungkin Anda perlu mempertimbangkan untuk meng-enkapsulasi pekerjaan ini ke dalam sebuah Sprite sesuai selera Anda.

5.12. Membuat Sprite pada GameCanvas

```
class SpriteMoveCanvas extends GameCanvas implements Runnable {
// Constructor dan initialisasi
public SpriteMoveCanvas () throws Exception {
.....
        try{
        // Load Images untuk Sprites
                image = Image.createImage("/panah.png");
                imageTemp = Image.createImage("/non_trans_panah.jpg");
        }catch (Exception ioex) { System.out.println(ioex); }
        sprite = new Sprite (image,20,20);
        nonTransparentSprite = new Sprite (imageTemp,20,20);
}
.....
 // Method untuk menghandle inputan dari keypad
private void input() {
//input keyboard
.....
 }

 // Method untuk Display Graphics
private void drawScreen(Graphics g) {
// display untuk sprites
.....
}
}
```



5.13. DIMANA MENCARI SPRITE

Kebanyakan *developer* (pengembang) seperti Anda kemungkinan tidak memiliki keterampilan untuk menciptakan sprite yang indah.

Beberapa link yang menyediakan sprite antara lain :

1. http://www.arifeldman.com
2. http://www.idevgames.com
3. http://www.spriteworks.com

Pilihan Anda yang lain adalah menyewa desainer professional *sprite* yang terkadang disebut *pixel pusher*.

Akhirnya Anda akan selalu dapat belajar untuk membuat sprite Anda sendiri. Ini mungkin bukanlah ide yang buruk jika Anda berencana untuk membuat sebuah game 2D sederhana dan jika Anda tidak ingin menghabiskan waktu berpikir untuk mempelajari lebih banyak tentang menggambar sprite. Silakan jika Anda tidak memiliki kemampuan artistik Anda mungkin lebih baik memilih 2 saran yang pertama. Tetapi ingatlah grafis penting tetapi mengerjakan pembuatan game juga lebih penting. Jadi pada awalnya Anda mungkin hanya ingin menggunakan grafis yang sederhana tetapi tidak terlalu baik sampai Anda memiliki sebuah game yang berfungsi penuh. Kemudian Anda dapat memperhatikan perkembangan grafis.