

Podstawy tworzenia gier w J2ME, dla początkujących adeptów sztuki programowania

Marcin Stachniuk

mstachniuk@gmail.com

Blog: mstachniuk.blogspot.com

Twitter: [@MarcinStachniuk](https://twitter.com/MarcinStachniuk)

27 maja 2012

1 Podstawy rysowania

2 Obsługa klawiszy

3 Sprite

4 TiledLayer

5 Wykrywanie kolizji

6 Animacja

1 Podstawy rysowania

2 Obsługa klawiszy

3 Sprite

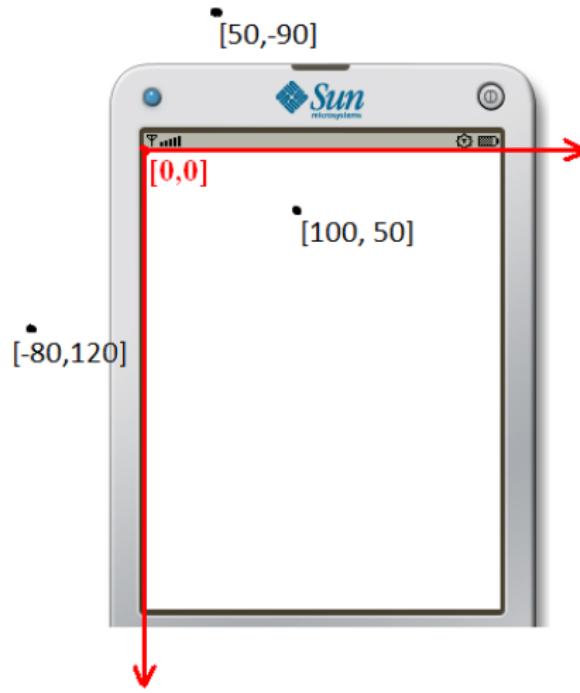
4 TiledLayer

5 Wykrywanie kolizji

6 Animacja

GameCanvas

Przestrzeń po której będziemy malować.



Metody:

- `setFullScreenMode(true);` – ustawienie pełnego ekranu.
- `Graphics g = getGraphics();` – pobranie buforu graficznego.
- `flushGraphics();` – namalowanie grafiki z bufora

Graphics

Klasa służąca do rysowania prymitywów, obrazków i tekstów.

Metody:

- `g.setColor(0xFFFFFFFF) / g.setColor(255, 255, 255)`
– ustawienie aktualnego koloru na biały
- `g.drawRect(10, 20, 50, 60)` – malowanie prostokąta w punkcie [10,20] i szerokości 50px i wysokości 60px
- `g.fillRect(10, 20, 50, 60)` – malowanie wypełnionego prostokąta

Image

Klasa służąca do wczytywania obrazków.

Metody:

- `Image tankImage = createImage("tank.png");`
– wczytywanie obrazka z pliku tank.png
- `tankImage.getHeight()` – pobranie wysokości obrazka
- `tankImage.getWidth()` – pobranie szerokości obrazka

Główna pętla programu *Tank2012.java*

```
1 public void run() {  
2     while (!canvas.isEndGame()) { // dopoki gra sie nie skonczyła  
3         try {  
4             canvas.inputKeys(); // obsługa wcisniętych klawiszy  
5             canvas.think(); // pozostała logika gry  
6             canvas.paint(); // odmalowanie ekranu  
7             Thread.sleep(100); // uspelenie wątku na 100 ms  
8         } catch (Exception ex) {  
9             ex.printStackTrace();  
10        }  
11    }  
12    destroyApp(false); // sprzątanie przed zamknięciem aplikacji  
13    notifyDestroyed(); // zamkniecie aplikacji  
14 }
```

Tu będzie nasza implementacja *TankGameCanvas.java*

```
1 public class TankGameCanvas extends AbstractTankCanvas {  
2     /** Przygotowanie aplikacji */  
3     public void init() {  
4     }  
5  
6     /** Metoda odpowiedzialna za malowanie po ekranie */  
7     public void paint() {  
8     }  
9  
10    /** Obsługa naciskanych klawiszy. */  
11    public void inputKeys() {  
12    }  
13  
14    /** Miejsce na logice aplikacji. */  
15    public void think() {  
16    }  
17  
18    /** Metoda informujaca, czy gra sie juz zakończyła czy nie. */  
19    public boolean isEndGame() {  
20    }  
21 }
```

1 Podstawy rysowania

2 Obsługa klawiszy

3 Sprite

4 TiledLayer

5 Wykrywanie kolizji

6 Animacja

Metody klasy GameCanvas

- `int keyStates = getKeyStates();` – pobranie stanu wciskniętych klawiszy
- `(keyStates & LEFT_PRESSED) != 0` – sprawdzenie czy naciśnięto w lewo

Klawisze akcji gry

DOWN_PRESSED, LEFT_PRESSED, RIGHT_PRESSED,
UP_PRESSED, FIRE_PRESSED, GAME_A_PRESSED,
GAME_B_PRESSED, GAME_C_PRESSED, GAME_D_PRESSED

Obsługa klawiszy – przykład

```
1  if ((keyStates & LEFT_PRESSED) != 0) {  
2      // dalsza logika  
3 }
```

1 Podstawy rysowania

2 Obsługa klawiszy

3 Sprite

4 TiledLayer

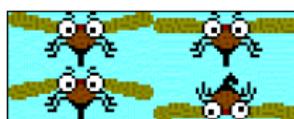
5 Wykrywanie kolizji

6 Animacja

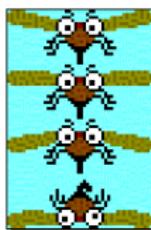
Sprite



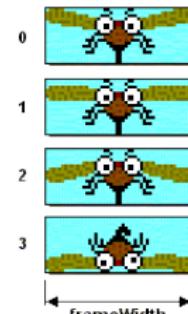
OR



OR



Frames



frameWidth
frameHeight

[http://java.sun.com/javame/reference/apis/jsr118/javax/microedition/lcdui/game\(Sprite.html](http://java.sun.com/javame/reference/apis/jsr118/javax/microedition/lcdui/game(Sprite.html)

Utworzenie Sprite'a:

- `tank = new Sprite(tankImage, 30, 30);` – utworzenie Sprite dla czołgu

Metody:

- `tank.paint(Graphics g)` – namalowanie czołgu
- `tank.setVisible(true)` – czołg staje się widoczny
- `tank.setVisible(false)` – czołg staje się niewidoczny
- `tank.isVisible()` – sprawdzenie czy czołg jest widoczny

Metody:

- `tank.getX()` – pobranie współrzędnej X dla czołgu
- `tank.getY()` – pobranie współrzędnej Y dla czołgu
- `tank.getHeight()` – pobranie wysokości dla czołgu
- `tank.getWidth()` – pobranie szerokości dla czołgu
- `tank.setPosition(2, 3)` – ustawienie pozycji czołgu w punkcie [2, 3]
- `tank.move(5, 8)` – przesunięcie czołgu o wektor [5, 8]
- `tank.defineReferencePixel(1, 5)` – definicja piksela referencyjnego dla czołgu
- `tank.setRefPixelPosition(4, 6)` – ustawienie pozycji piksela referencyjnego czołgu w pozycji [4, 6]

Zadanie 1

Dopisać poruszanie się czołgu w pionie i poziomie, za pomocą strzałek na klawiaturze telefonu.

Zadanie 2

Dopisać ograniczenie aby czołg nie wyjeżdżała poza ekran.

Zadanie 1

Dopisać poruszanie się czołgu w pionie i poziomie, za pomocą strzałek na klawiaturze telefonu.

Zadanie 2

Dopisać ograniczenie aby czołg nie wyjeżdżał poza ekran.

Metody:

- `tank.setFrameSequence(new int [] 2,3,0,1)`
– ustawienie sekwencji animacji na klatki (2,3,0,1)
- `tank.nextFrame()` – ustawienie następnej klatki
- `tank.prevFrame()` – ustawienie poprzedniej klatki
- `tank.getFrame()` – pobranie numeru aktualnej klatki
- `tank.setFrame(3)` – wybranie 3ciej klatki animacji

Zadanie 3

Dopisać obracanie czołgu lufą w kierunku jazdy.

1 Podstawy rysowania

2 Obsługa klawiszy

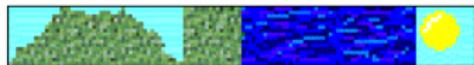
3 Sprite

4 TiledLayer

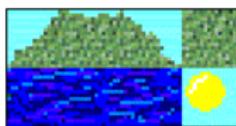
5 Wykrywanie kolizji

6 Animacja

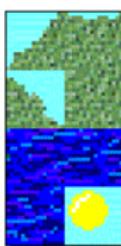
TiledLayer



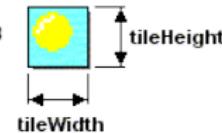
OR



OR



Tiles



<http://java.sun.com/javame/reference/apis/jsr118/javax/microedition/lcdui/game/TiledLayer.html>

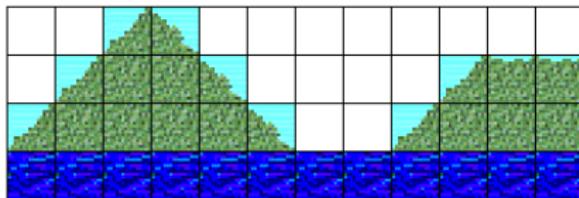
TiledLayer

Cells

0	0	1	3	0	0	0	0	0	0	0	0
0	1	4	4	3	0	0	0	0	1	2	2
1	4	4	4	4	3	0	0	1	4	4	4
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Animated Tiles

=



<http://java.sun.com/javame/reference/apis/jsr118/javax/microedition/lcdui/game/TiledLayer.html>

Konstruktor:

- `tiles = new TiledLayer(4, 6, image, 20, 20);`
– utworzenie nowego TiledLayer'a o 4ch kolumnach, 6ciu wierszach i rozmiarze pojedynczego kafelka 20x20.

Metody:

- `tiles.setCell(2, 3, 1)` - ustawienie kafelka nr. 1 na pozycji [2,3]

Zadanie 4

Wyświetlić TiledLayer'a z trawą na ekranie. Czołg może się „schować” w trawie.

1 Podstawy rysowania

2 Obsługa klawiszy

3 Sprite

4 TiledLayer

5 Wykrywanie kolizji

6 Animacja

Metody w Sprite:

- `tank.collidesWith(missile, false)` – test kolizji czołgu z innym Sprite'm (`missile`)
- `tank.collidesWith(tiles, false)` – sprawdzanie kolizji czołgu z kafelkiem TiledLayer'a

Zadanie 5

Wyświetlić TiledLayer'a z białymi cegłami i wodą na ekranie.

Przez białe cegły i wodę nie można przejechać.

Podpowiedź: Użyj nowego TiledLayer'a.

1 Podstawy rysowania

2 Obsługa klawiszy

3 Sprite

4 TiledLayer

5 Wykrywanie kolizji

6 Animacja

Metody:

- int animatedTile = tiles.createAnimatedTile(0); - utworzenie animowanego kafelka
- tiles.setAnimatedTile(animatedTile, 2) - ustawienie animowanego kafelka o numerze animatedTile na kafelek nr. 2.

Zadanie 6

Spraw, aby woda się animowała.

Zadanie 7

Wyświetl na ekranie mór z czerwonych cegieł i spraw, aby można było zniszczyć cegły za pomocą wystrzeliwanych pocisków.

Podpowiedź 1: Dla każdego murku utwórz pojedynczego Sprite'a.

Podpowiedź 2: Dla Sprite'a będącego pociskiem zdefiniuj pixel referencyjny w jego centrum.

Zadanie 6

Spraw, aby woda się animowała.

Zadanie 7

Wyświetl na ekranie mór z czerwonych cegieł i spraw, aby można było zniszczyć cegły za pomocą wystrzeliwanych pocisków.

Podpowiedź 1: Dla każdego murku utwórz pojedynczego Sprite'a.

Podpowiedź 2: Dla Sprite'a będącego pociskiem zdefiniuj pixel referencyjny w jego centrum.

Powinniśmy osiągnąć coś takiego:

?

- Kontakt z autorem:
e-mail: mstachniuk@gmail.com
blog: mstachniuk.blogspot.com
twitter: [@MarcinStachniuk](https://twitter.com/@MarcinStachniuk)
- Kod źródłowy:
<https://github.com/mstachniuk/tank2012j2me>

Podstawy tworzenia gier w J2ME, dla początkujących adeptów sztuki programowania

Marcin Stachniuk
mstachniuk@gmail.com

Dzięki za uwagę!

27 maja 2012