

Processing

Jedná se o programovací jazyk, který vznikl v roce 2001. Původně vznikl za účelem výuky základů programování s ohledem na vizuální kontext. Později se stal vývojářským nástrojem pro profesionály v mnoha oblastech – umělci, designéři, výzkumníci, ...

Je to open source jazyk, multiplatformní. Aktuálně existuje přes 100 rozšiřujících knihoven, vznikají o něm knihy, ...

Základy jazyka:

Processing má dvě hlavní funkce:

```
void setup() {  
}  
void draw() {  
}
```

Podle názvu je zřejmé, co budou funkce obsahovat. Setup slouží pro globální nastavení scény, proto se veškeré příkazy provedou pouze **jednou**. Oproti tomu funkce draw slouží k samotnému vykreslování do scény, proto se volá **opakovaně**, pro každý frame.

Ukažme si to na jednoduchém příkladě.

```
void setup() {  
    size(600, 600); // nastavení velikosti vykreslovaného okna  
    smooth(); //antialiasing  
    // background(0);  
}  
  
void draw() {  
    background(0);  
    ellipse(200, 200, 20, 30);  
}
```

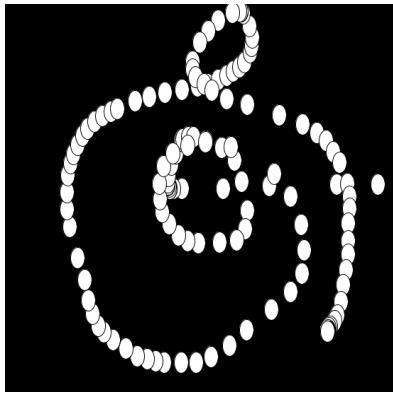
Ve funkci setup nejdříve nastavíme velikost vykreslovaného okna – pomocí příkazu size. V metodě draw pak nastavíme pozadí (příkaz background) a co budeme vykreslovat. V tomto případě pomocí příkazu ellipse s parametry pozice v x a y, šířka a výška elipsy, nastavíme parametry vykreslování. Funkce background může nabývat šedotónních odstínů – podle rozsahu 0 – 255, nebo podporuje rovněž RGB režim.

Pokud chceme, aby se pozice elipsy transformovala společně s pohybem myši, použijeme příkaz:

```
ellipse(mouseX, mouseY, 20, 30);
```

Pokud chceme, aby se při renderování elipsy uplatnil antialiasing, přidáme do setup volání funkce smooth().

Když chceme vidět rozdíl mezi voláním funkce v setup a v draw, stačí zkusit přesunout volání background z draw do setup:



Proměnné

Processing podporuje různé typy proměnných – integer, float, string, ... Obecně je lze klasifikovat do dvou skupin – čísla a písmena. Zadávat se stejně jako v Javě:

```
int x = 5;  
float y = 2.5;
```

V našem příkladě můžeme použít např. proměnné x a y s hodnotou 200 a použít je při určení pozice elipsy:

```
int x = 200;  
int y = 200;  
  
void draw() {  
    background(0);  
    ellipse(x, y, 20, 30);  
}
```

Pokud v draw nastavíme inkrementování některé proměnné, pak vytvoříme jednoduchou animaci.

```
void draw() {  
    background(0);  
    x = x + 5;  
    ellipse(x, y, 20, 30);  
}
```

Hodnotu 5 můžeme opět vložit do proměnné, např. speed.

Funkce

Opět podobně jako v Javě. Příklad:

```
void draw() {  
    drawEllipse(200, 200, 20);  
    drawEllipse(400, 200, 255);  
    drawEllipse(200, 400, 180);  
}
```

```
void drawEllipse(float x, float y, float red) {  
    fill(red, 0, 0);  
    ellipse(x, y, 50, 80);  
}
```

Reference

Na stránce www.processing.org v záložce References najdete seznam všech dostupných funkcí, včetně jejich dokumentace a ukázkového kódu.

Ctrl + T = zarovnání kódu

Cykly

Opět funguje stejně.

```
void draw() {  
    background(255, 128, 0);  
  
    for (int i = 0; i < 20; i++) {  
        ellipse(i*50, i*50, 50, 80);  
    }  
}
```

```
void draw() {  
    background(255, 128, 0);  
  
    for (int i = 0; i < 20; i++) {  
        for (int j = 0; j < 20; j++) {  
            ellipse(i*50, j*50, 50, 80);  
        }  
    }  
}
```

Podmínky

```
void draw() {  
    background(255, 128, 0);  
  
    for (int i = 0; i < 20; i++) {  
        if (i < 10) {  
            fill(255, 0, 0);  
        } else {  
            fill(0, 255, 0);  
        }  
        ellipse(i*50, 200, 50, 80);  
    }  
}
```