Processing Workshop

Till Nagel, IUAV, 10/2008

Assignment presentations

Directory: Create a directory with your name(s), e.g. “A1-Nagel”

Export: Load your sketch in Processing and go to menu “Tools” and select “Archive Sketch”. Copy the archive (ZIP file) into the directory.

Scribbles / Sketches: Copy existing images into the directory.

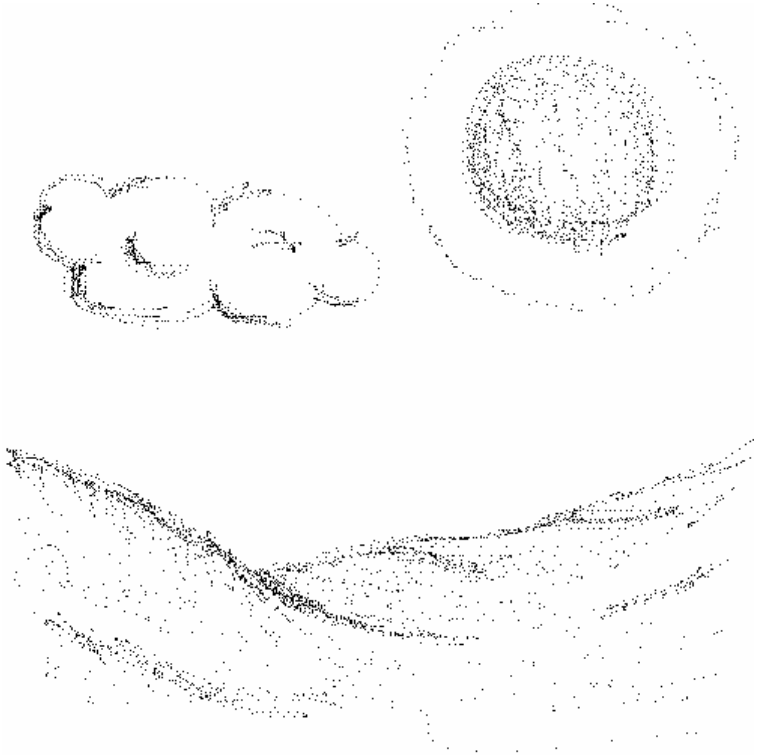
Hand-over: Transfer your directory to the USB stick into directory “Assignment1”

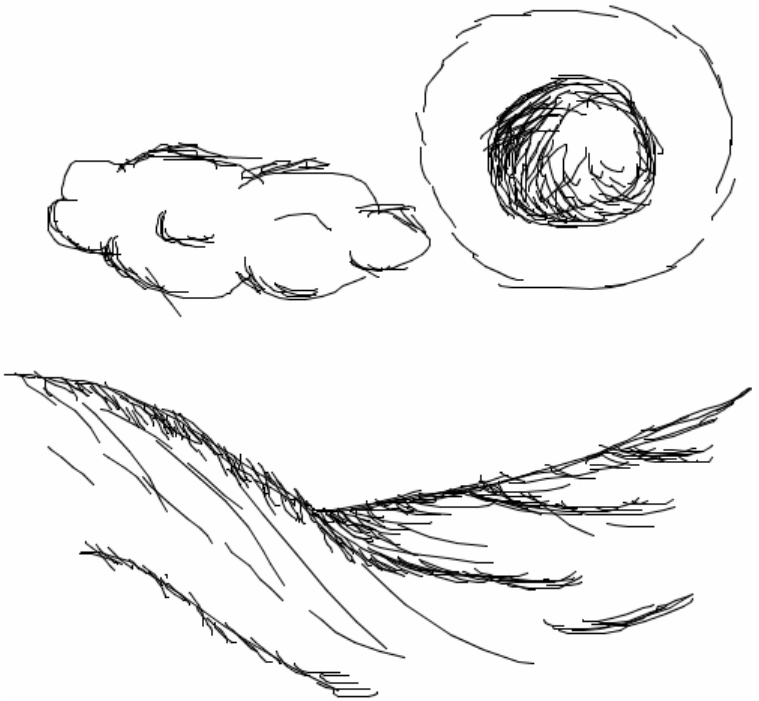
Interaction 2

Mouse buttons

```
if (mousePressed) {
  if (mouseButton == LEFT) {
    fill(255, 0, 0);
  }
}
rect(mouseX, mouseY, 10, 10);
```

Mouse tracking





Example: LineDrawing

Mouse dragging

Example: SimpleDrag

Mouse events

```
void draw() {
  if (mousePressed) {
    stroke(random(255));
  }
  line(width/2, height/2, random(width), random(height));
}
```

Mouse events: mousePressed()

```
void draw() {
  line(width/2, height/2, random(width), random(height));
}

void mousePressed() {
  stroke(random(255));
}
```

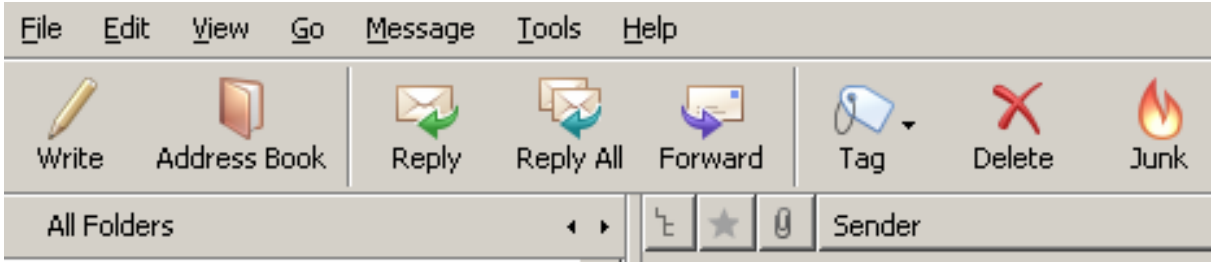
Mouse events: mouseReleased()

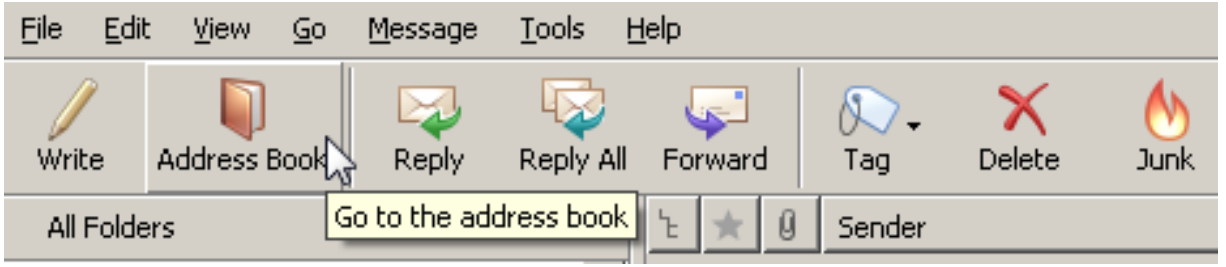
```
int s = 50;

void setup() {
  size(400, 400);
}

void draw() {
  background(0);
  ellipse(mouseX, mouseY, s, s);
}

void mouseReleased() {
  s = s + 10;
}
```

Example: MouseEventCircleRelease

Buttons & other simple interactive elements

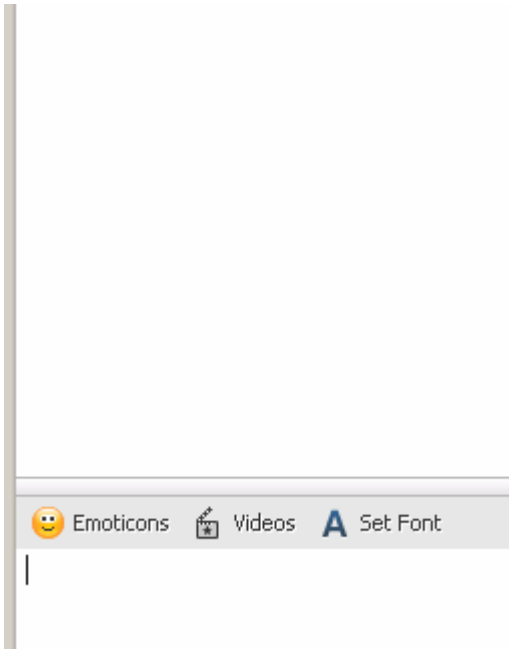
Buttons & other simple interactive elements

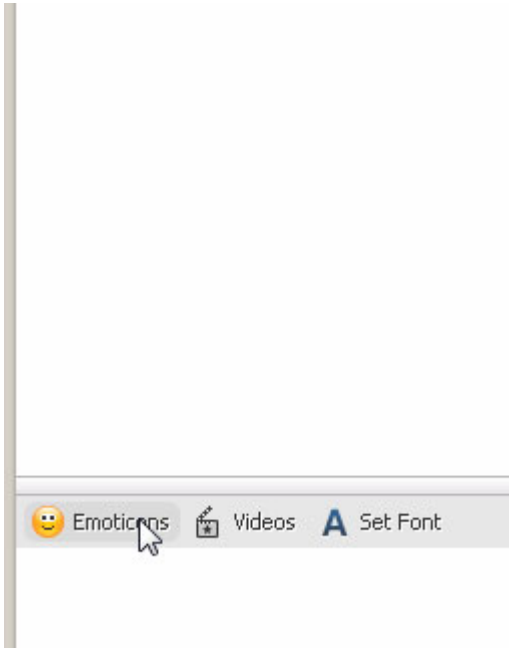


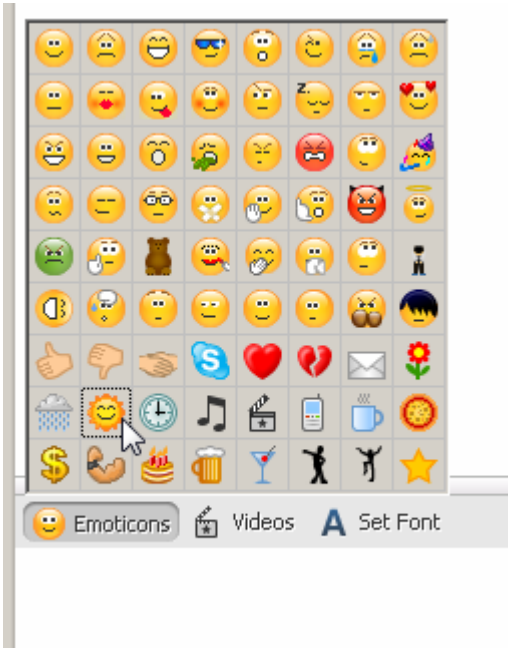


From: Mozilla Thunderbird 2.0.0

Buttons & other simple interactive elements







From: Skype 3.6.0

Keyboard

```
if (keyPressed) {
  fill(255, 0, 0);
}
rect(mouseX, mouseY, 10, 10);
```

Keyboard

```
if (keyPressed && key == 'a') {
  fill(255, 0, 0);
}
rect(mouseX, mouseY, 10, 10);
```

Keyboard

```
if (keyPressed) {
  if (key == 'r') {
     fill(255, 0, 0);
  }
  if (key == 'b') {
     fill(0, 0, 255);
  }
}
rect(mouseX, mouseY, 10, 10);
```

Function with return value

```
size(300, 500);
noStroke();

float x = 10.3;
```

Function with return value

```
size(300, 500);
noStroke();

float x = 10.3;

float y = sin(x);
```

Function with return value

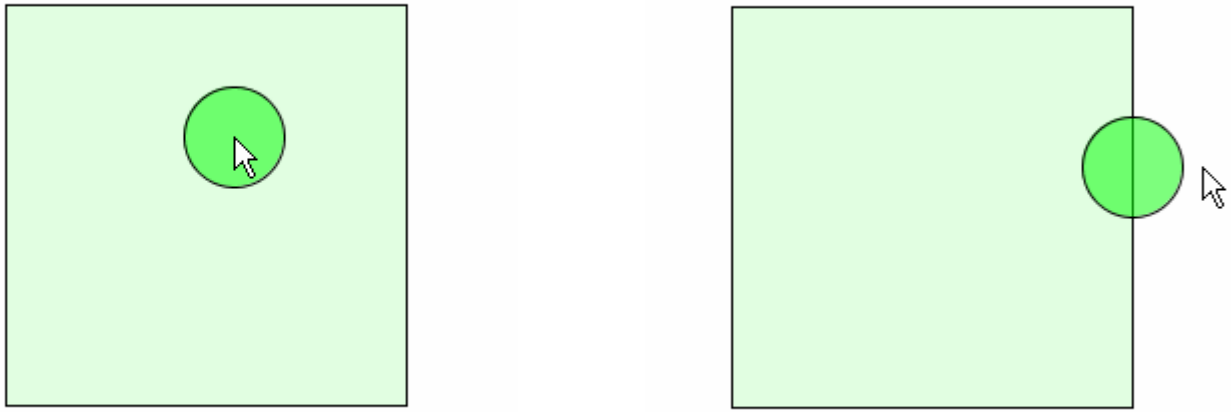
```
size(300, 500);
noStroke();

float x = 10.3;

float y = sin(x);
float c = cos(x);
float m = min(10, 100);
```

Function with return value

```
size(300, 500);
noStroke();

float x = 10.3;

float y = sin(x);
float c = cos(x);
float m = min(10, 100);

ellipse(x, sin(x), 20, 20);
```

constrain()

```
float x = constrain(mouseX, 100, 300);
float y = constrain(mouseY, 100, 300);
ellipse(x, y, 50, 50);
```



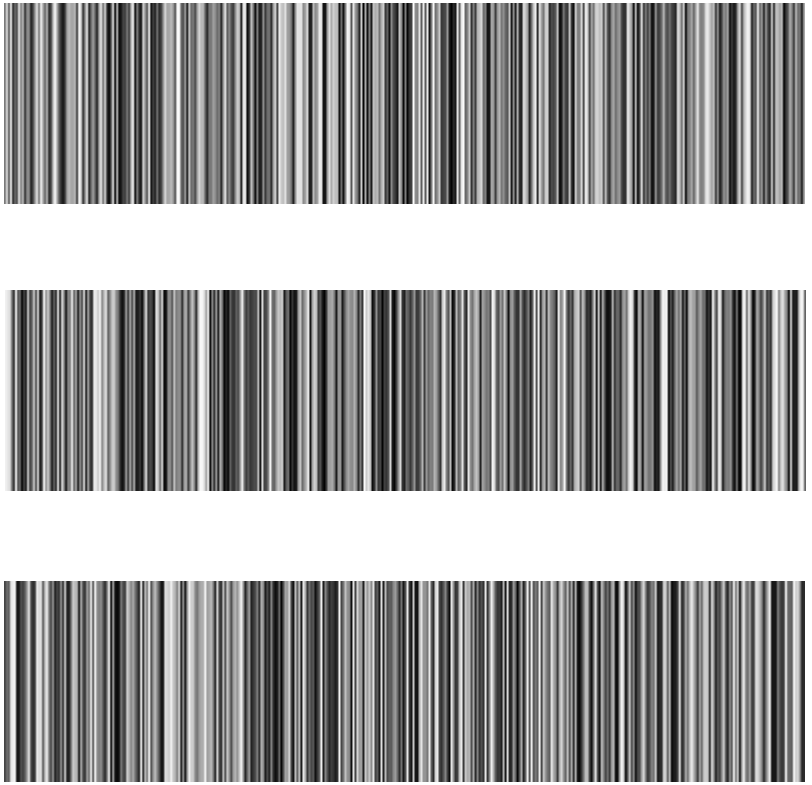
Example: Constrain

random()

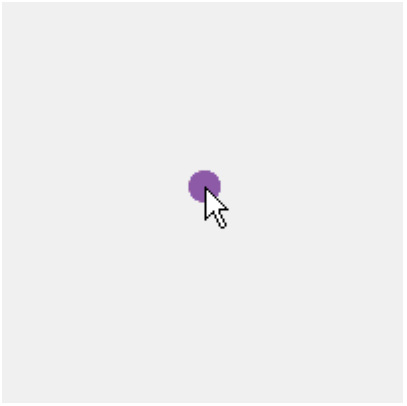
```
size(400, 100);
background(random(255));
```

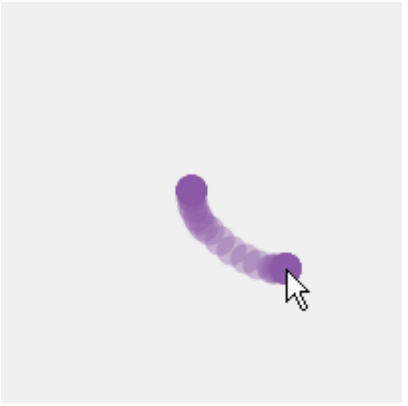
random(value)

```
int x = 0;

void setup() {
  size(400, 100);
  smooth();
}

void draw() {
  stroke(random(255));
  line(x, 0, x, height);
  x = x + 1;
}
```

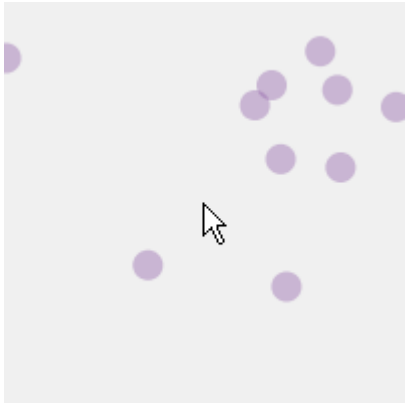


Draw at mouse position

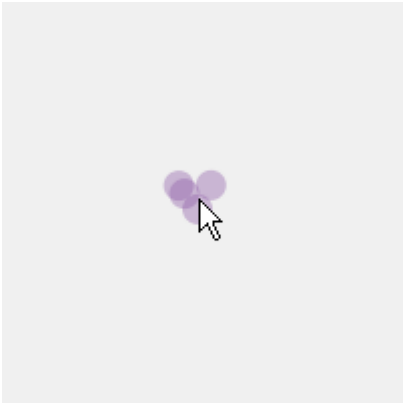


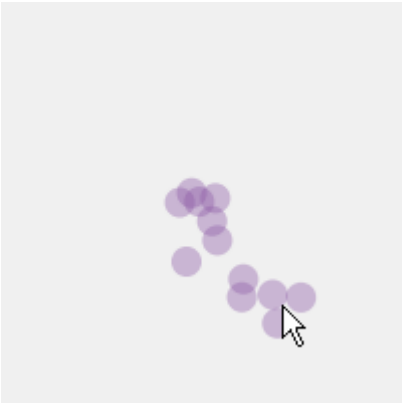


Draw at random positions

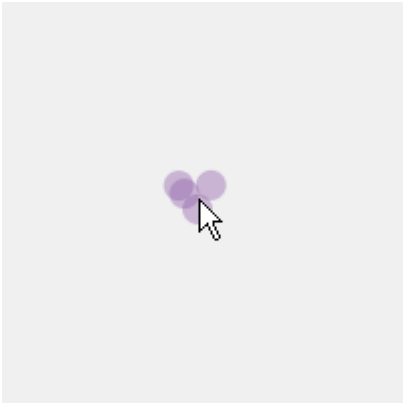


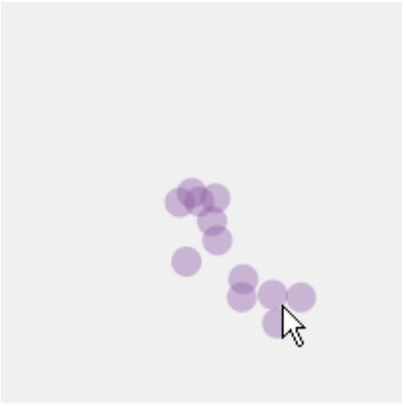
mouse + random = A dripping brush





mouse + random = A dripping brush





```
float x = mouseX + random(-10, 10);
float y = mouseY + random(-10, 10);
ellipse(x, y, 15, 15);
```

Example: DrippingBrush

Images in Processing

```
PImage foto;

void setup() {
  size(400, 200);
  foto = loadImage("btk.jpg");
}

void draw() {
  image(foto, 10, 20);
}
```

Images in Processing

loadImage(fileName)

```
PImage foto;
PImage foto2;

void setup() {
  size(400, 200);
  foto = loadImage("btk.jpg");
  foto2 = loadImage("meinFoto.png");
}
```

image(imageVar, x, y)

```
// Draws the photo at position 100, 200
image(photo, 100, 200);

// Draws the picture at mouse position
image(pic1, mouseX, mouseY);

// Draws the picture and resizes it to 200x300
image(pic2, 10, 10, 200, 300);
```

Exercises

E11: Draw transparent coloured circles at random positions.

Variation: Use random sizes, with each circle greater than 20 pixels.

E12: Load an image and let it move over the screen.

Variation: Load different images and animate them in various ways.

Exercises

E13: Create an interactive element, and use three different images or icons for the states normal, mouse-over and mouse-out.

Variation: Create an outer glow which responds to the proximity of the mouse pointer.
Lookup `dist()` in the Processing reference and use it.