Motion

Animating Processing

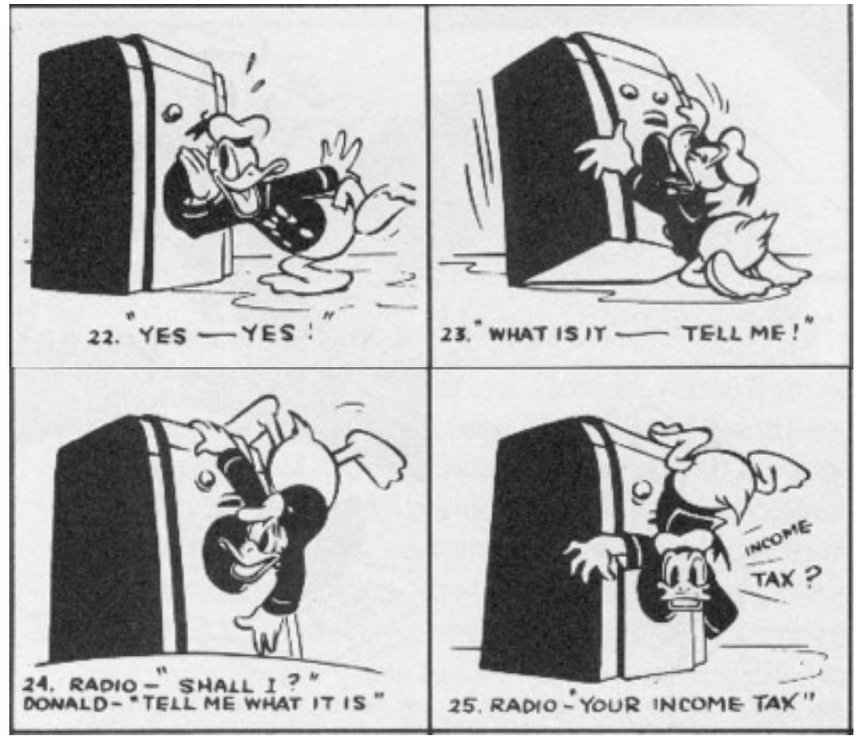
Motion

Storyboarding

storyboard

Developed early 1930s at the Walt Disney studio for animation sequences.

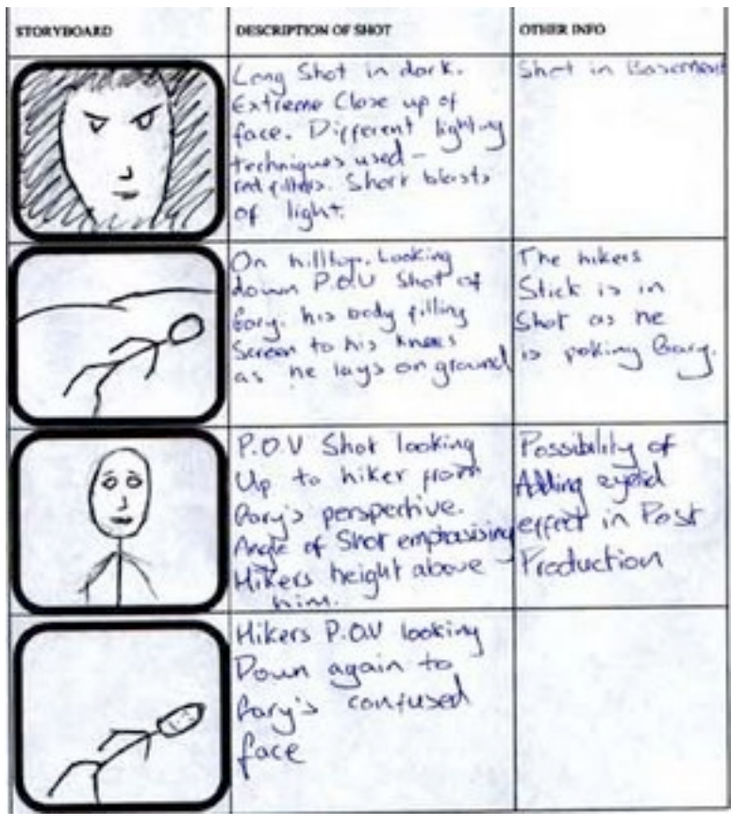
Adopted for use in film, theatre, television, comics and interactive media



storyboard

starting point for any motion based project

combination of sketches and notes describing setting, camera instruction and actions of characters



Motion

Storyboard

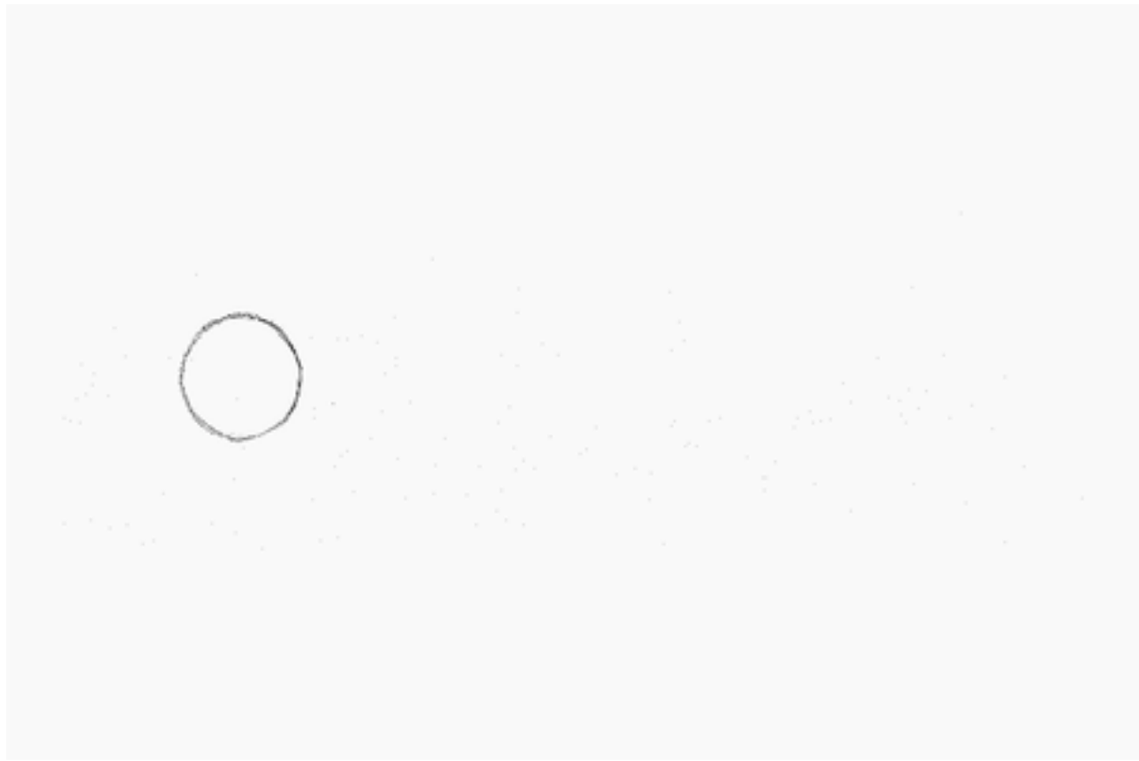
Motion

Animation Basics

anticipation

primary animation principle

preparation to action
and the reaction is the
recovery from the action



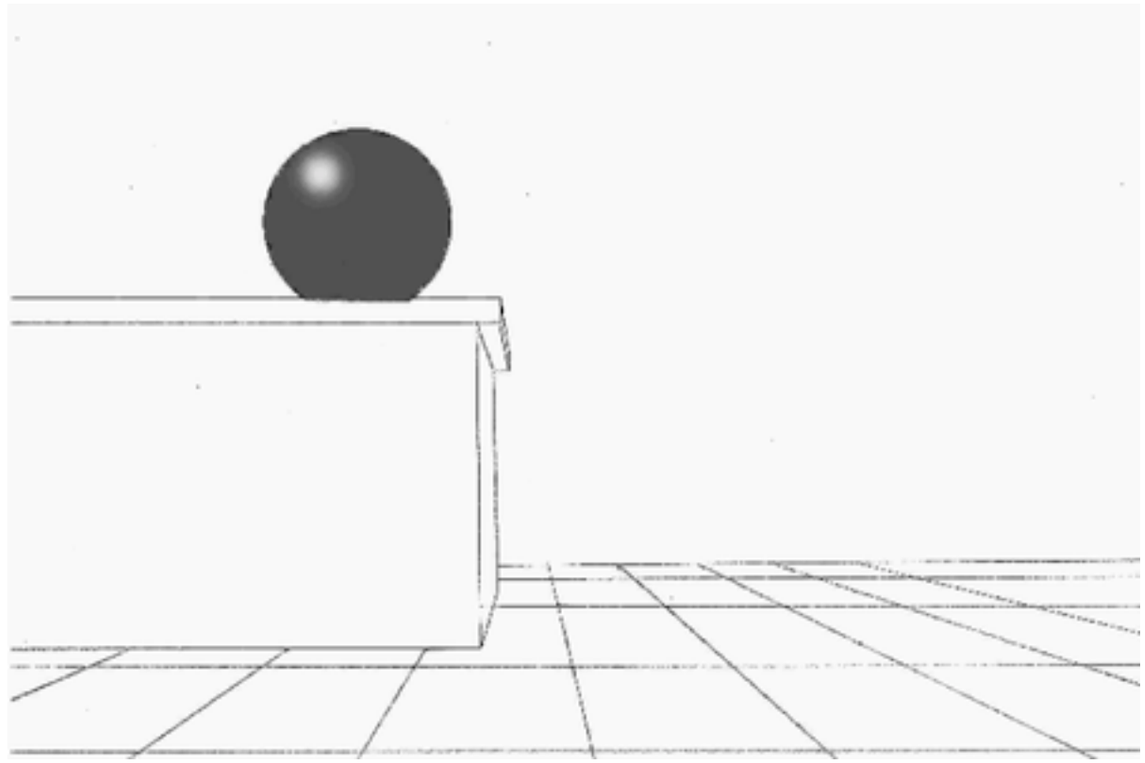
bounce

impacts on the surface,
it squashes a bit to indicate
visually that it has indeed
hit something

ease in/easeout
soften the end of an action

gravity

indicates a different type of weight



atmospheric

items seem blurry
the further they are

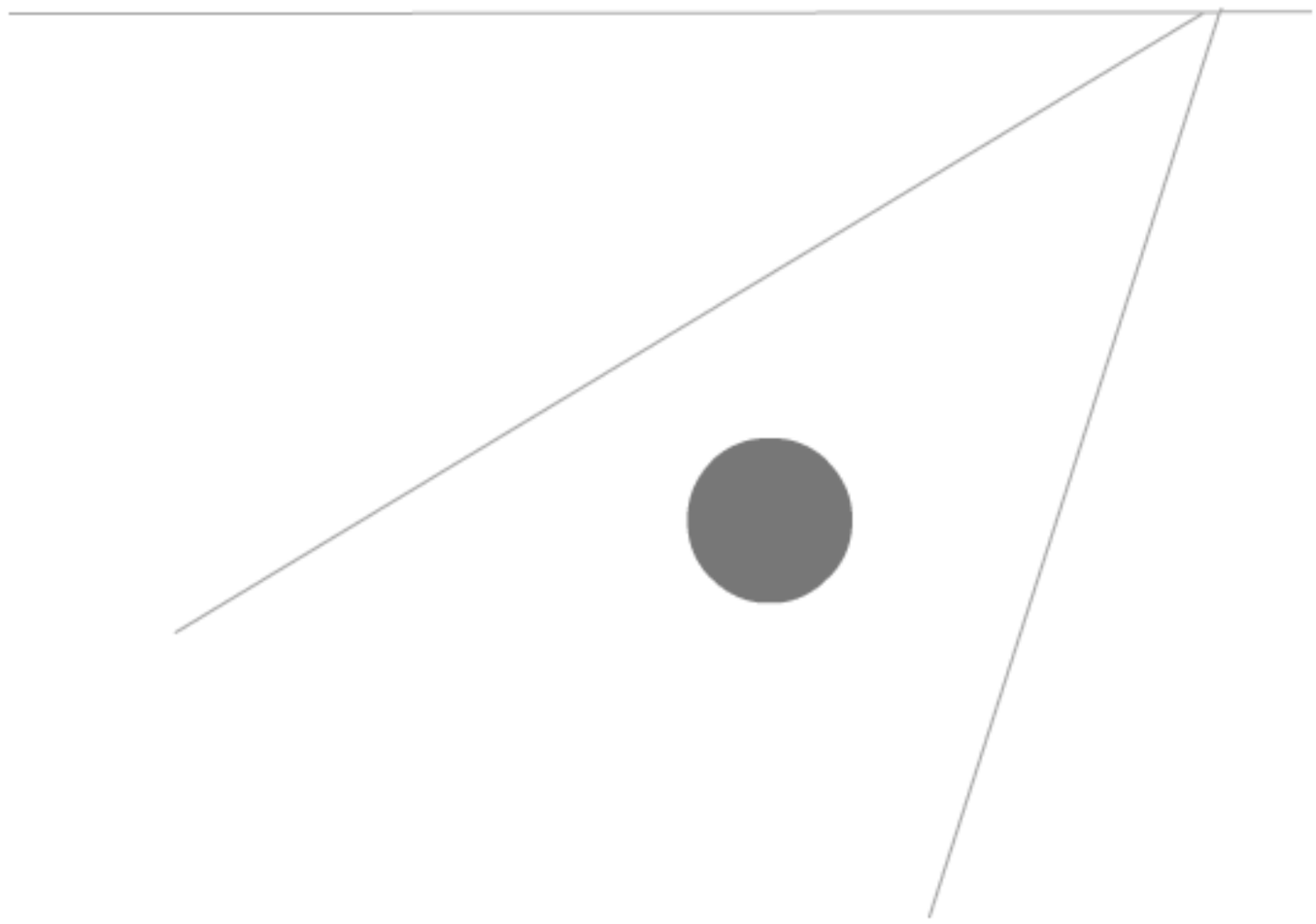
speed

near objects move fast,

objects further away seem to move slower

perspective

item diminishes in size as it travels to vanishing point



z axis

object speeds up as
it comes toward you

Motion

Coding Motion

frameRate()

number of frames displayed every second
default is 60 frames per second

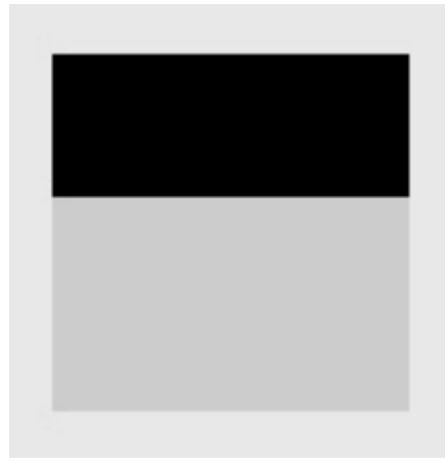
the lower / the slower

the higher / the faster

Coding Motion

continuous

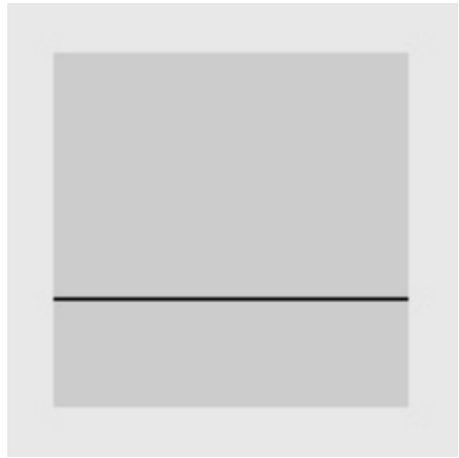
```
float y = 0.0;
void draw(){
   frameRate(20);
   line(0,y,100,y);
   y += 0.5;
}
```



fills screen by
drawing a line
at rate of 20 fps
(frames per second)

continuous

```
float y = 0.0;
void draw(){
   frameRate(60);
   background(204);
   line(0,y,100,y);
   y += 0.5;
}
```



continuously
draws background
and a line at 60 fps

Coding Motion

try it

```
float y = 0.0;    // y position variable

void setup() {
  size(100, 100);
  smooth();
  fill(0);
}

void draw() {
  background(204);              // draw background
  ellipse(50, y, 70, 70);       // get y position
  y += 0.5;                     // add 0.5 to y
}
```

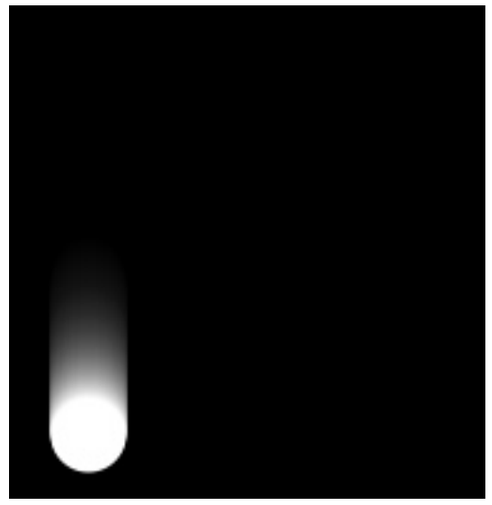
add a loop

```
float y = 0.0;    // y position variable

void setup() {
  size(100, 100);
  smooth();
  fill(0);
}

void draw() {
  background(204);                // draw background
  ellipse(50, y, 70, 70);  // get y position
  y += 0.5;                       // add 0.5 to y
  if (y > 150){    // if position is more than 150
    y = -50.0;     // set it to -50
  }
}
```

Motion

Controlling Motion

shapes in motion

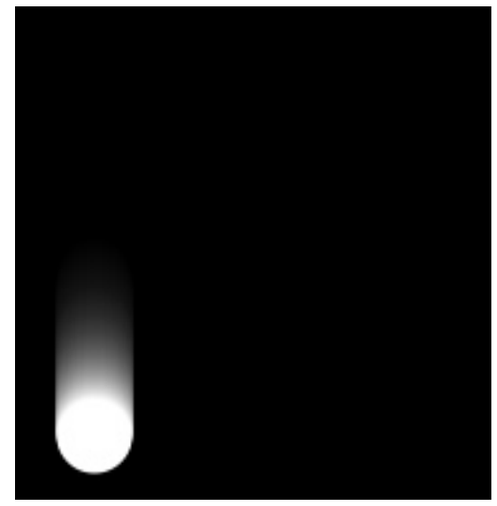


use a variable to change its attributes

change position relative to the left and right edges

change background value, stroke and fill

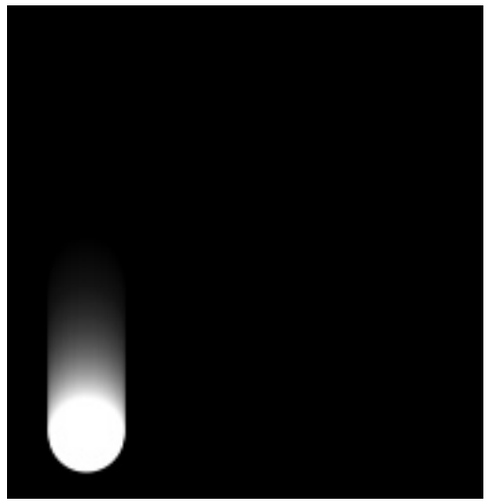
Coding Motion



```
float y = 50.0;       // y position variable
float speed = 1.0;    // value for speed
float radius = 15.0; // value for circle size
```

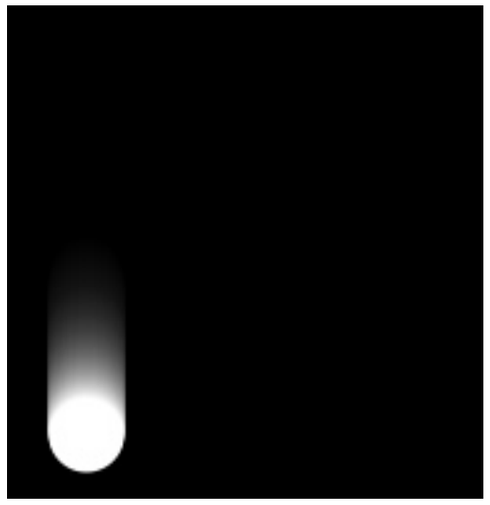
Coding Motion

```
float y = 50.0;       // y position variable
float speed = 1.0;    // value for speed
float radius = 15.0; // value for circle size


void setup(){
  size(200,200);
  smooth();
  noStroke();
  ellipseMode(RADIUS);
 // from centre point x,y,half width,half height
}
```



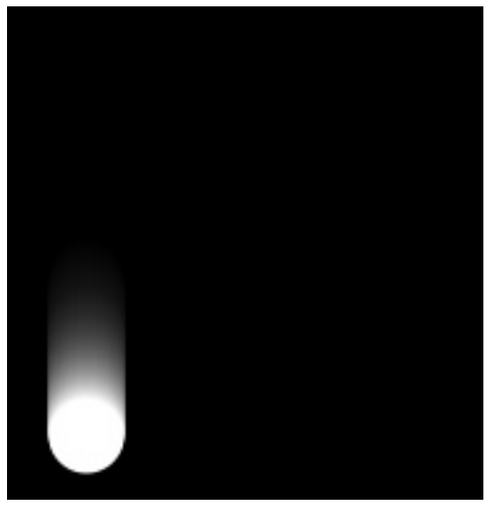
Coding Motion

```
float y = 50.0;       // y position variable
float speed = 1.0;    // value for speed
float radius = 15.0; // value for circle size

void setup(){
  size(200,200);
  smooth();
  noStroke();
  ellipseMode(RADIUS);
}

void draw(){
 fill(0, 12);   // background colour, alpha for motion blur
 rect(0,0,width,height); // background shape size of window
 fill(255);   // circle fill colour shape size of window
 ellipse(33, y, radius, radius);
 // draw circle based on variables
 y += speed;    // increase value to move circle
}
```



Coding Motion

```
float y = 50.0;       // y position variable
float speed = 1.0;    // value for speed
float radius = 15.0; // value for circle size

void setup(){
  size(200,200);
  smooth();
  noStroke();
  ellipseMode(RADIUS);
}
void draw(){
 fill(0, 12);  // background colour, alpha for motion blur
 rect(0,0,width,height); // background shape size of window
 fill(255);  // circle fill colour shape size of window
 ellipse(33, y, radius, radius);
 // draw circle based on variables
 y += speed;   // increase value to move circle

  if(y > height+radius){
  // if y is greater than height of window and half circle size
     y = - radius;
  // move it to above top of window with a negative position
   }
}
```



translate()

allows objects to be moved to any location within the window.

Coding Motion

translate()

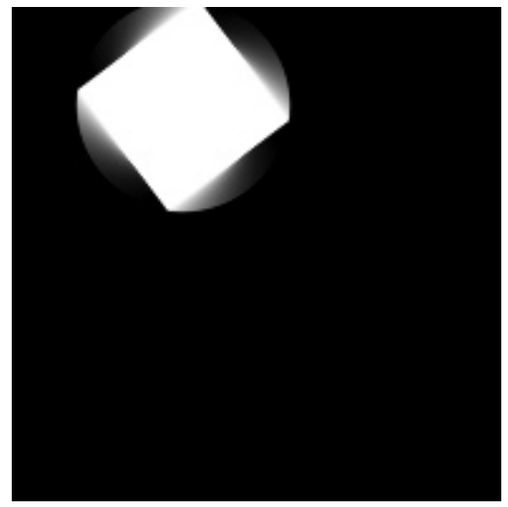
```
float x;                    // x position variable

void setup()
{
  size(200,200);
  noStroke();
  frameRate(60);
}
```

Coding Motion

translate()

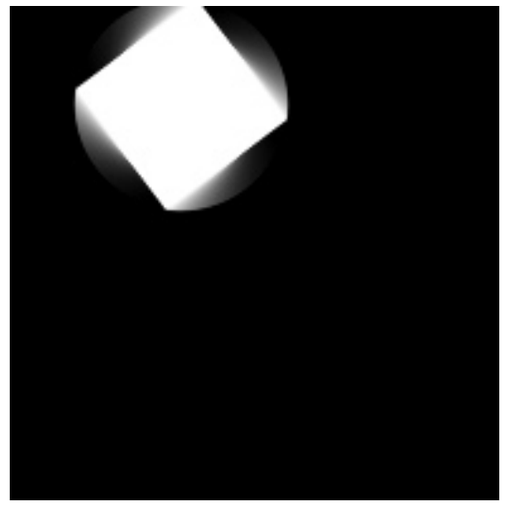
```
float x;                // x position variable
void setup()
{
  size(200,200);
  noStroke();
  frameRate(60);
}
void draw()
{
  background(102);
  x = x + 0.8;
  translate(x, 50);     // offset value
  fill(255);
  rect(50, 50, 40, 40);
}
```

rotate()



```
float angle = 0.0; // start variable for angle

void setup(){
  size(200, 200);
  smooth();
  noStroke();
}
```

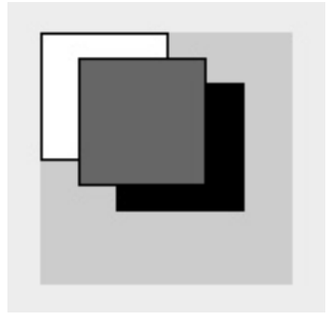
Coding Motion



```
float angle = 0.0; // start variable for angle

void setup(){
  size(200, 200);
  smooth();
  noStroke();
}

void draw(){
  fill(0, 12);      // bkg colour, alpha for motion blur
  rect(0, 0, width, height); // background shape
  fill(255);    // shape's colour
  angle = angle + 0.02;   // rate of rotation
  translate(100, 100);    // offset from orig position
  rotate(angle);          // rotate shape at angle value
  rect(-30, -30, 60, 60); // draw shape

}
```

Coding Motion

pushMatrix();
popMatrix();



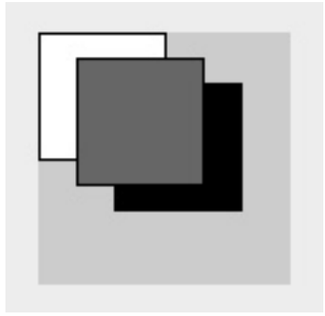
pushMatrix()
saves the current coordinate system to the stack

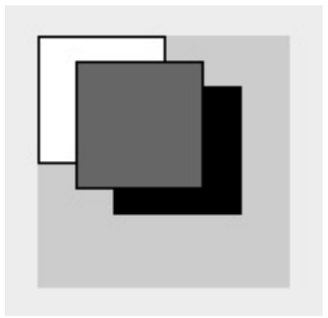
popMatrix()
restores the prior coordinate system

used together in conjunction with the methods like translate()

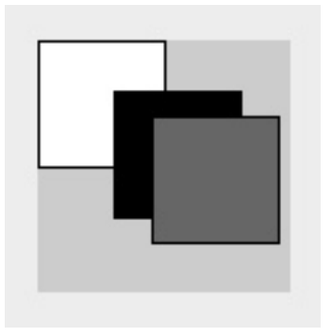
Coding Motion

pushMatrix();
popMatrix();





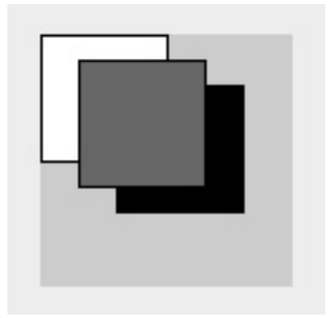
with **pushMatrix() / popMatrix()**



without **pushMatrix() / popMatrix()**

Coding Motion

pushMatrix();
popMatrix();



```
rect(0, 0, 50, 50);     //White
pushMatrix(); //set new coordinates
translate(30, 20); //offset from last shape
fill(0);
rect(0, 0, 50, 50);   //Black
popMatrix(); //restore original coordinates
fill(102);
rect(15, 10, 50, 50); //Grey
```

Typography

Generate Fonts and Text

Typography

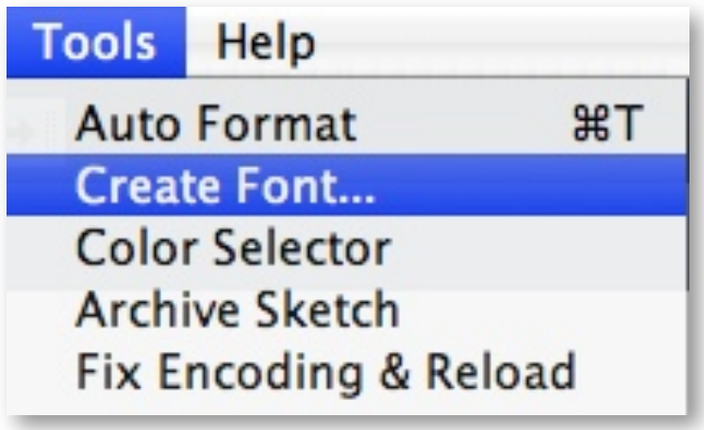
Create a Processing friendly font.

Create and Save a Processing File

Tools > Create Font

Pick a font and size from the list

Must do this for every font / size variation.

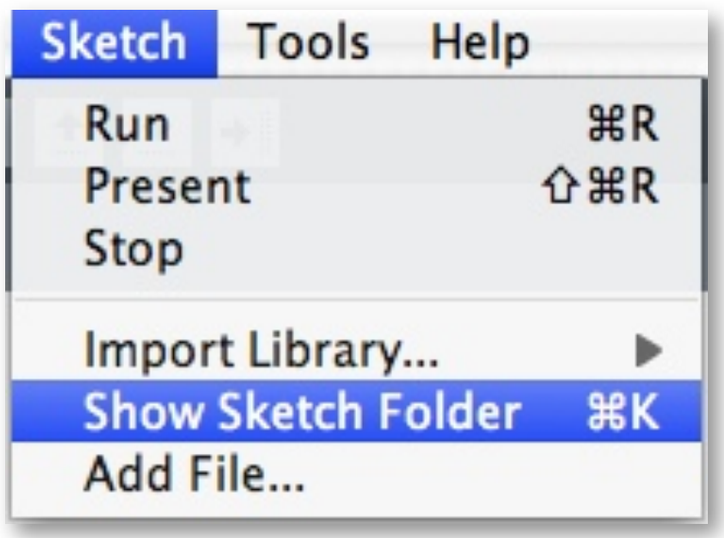


Processing creates a data folder with your font

Check if it is there

Sketch > Show Sketch Folder ⌘ K

Processing stores each letter as an image
in VLW file format.



Typography

To use this special font, it has to be loaded


```
PFont font;
font = loadFont("Avenir-Black-48.vlw");
// enter the exact file name of your VLW font
```

Typography

To use this special font, it has to be loaded


```
PFont font;
font = loadFont("Avenir-Black-48.vlw");
// enter the exact file name of your VLW font
```

```
textFont(font); // set the current text font

fill(0);
text("ABC", 0, 40); // put ABC at position 0, 40
```

Typography

To use this special font, it has to be loaded


```
PFont font;
font = loadFont("Avenir-Black-48.vlw");
// enter the exact file name of your VLW font

textFont(font); // set the current text font

fill(0);
text("A", 0, 40); // put A at position 0, 40

fill(0, 90);
text("B", 20, 40); // put B at position 20, 40

fill(0, 50);
text("C", 40, 40); // put B at position 40, 40
```

Make it move

```
PFont font;
float x1 = 0;

void setup(){
  size(100, 100);
  font = loadFont("Avenir-Black-48.vlw");
  // enter the exact file name of your VLW font
  textFont(font);
  fill(0);
}

void draw(){
  background(204);           // updates background
  text("Right", x1, 50);     // write this at specified position
  x1 += 1.0;                 // add 1 to x value
  if (x1 > width) {          // if x1 value greater than window
      x1 = -150;             // move it left, out of window
  }
}
```

Try adding a word moving right to left

```
PFont font;
float x1 = 0;

void setup(){
  size(100, 100);
  font = loadFont("Avenir-Black-48.vlw");
  // enter the exact file name of your VLW font
  textFont(font);
  fill(0);
}

void draw(){
  background(204);           // updates background
  text("Right", x1, 50);     // write this at specified position
  x1 += 1.0;                 // add 1 to x value
  if (x1 > width) {          // if x1 value greater than window
      x1 = -150;             // move it left, out of window
  }
}
```