Design Programming

DECO1012
Rob Saunders
Rob Saunders

web: http://web.arch.usyd.edu.au/~rob

e-mail: rob@arch.usyd.edu.au

office: Room 274, Wilkinson Building
Text & Transformations
Transform
Translation

• The `translate()` function moves the origin from the upper-left corner of the screen to another location.
  • It has two parameters. The first is the x-coordinate offset and the second is the y-coordinate offset: `translate(x, y)`
Translation

The same rectangle is drawn, but only the second is affected by `translate()` because it is drawn after

```plaintext
rect(0, 5, 70, 30);
translate(10, 30);
rect(0, 5, 70, 30);

rect(0, 5, 70, 30);
translate(10, -10);
rect(0, 5, 70, 30);
```
Translation

The `translate()` function is additive.

```python
rect(0, 5, 70, 30);
translate(10, 30);
rect(0, 5, 70, 30);
translate(10, 30);
rect(0, 5, 70, 30);
```
Transformation Matrices

- A transformation matrix is a set of numbers that defines how geometry is drawn to the screen.
- Transformation functions such as `translate()` alter the numbers in this matrix and cause the geometry to draw differently.
Transformation Matrices

- The `pushMatrix()` function records the current state of all transformations so that a program can return to it later.
- To return to the previous state, use `popMatrix()`.
Transformation Matrix Stack

- The `pushMatrix()` function is used to add a new coordinate matrix to the stack, and `popMatrix()` is used to remove one from the stack.
- Each `pushMatrix()` must have a corresponding `popMatrix()`.
Isolating Transformations

These sketches draw the same rectangles but the second one isolates the translation using `pushMatrix()` and `popMatrix()`.

```
translate(33, 0);
rect(0, 20, 66, 30);
rect(0, 50, 66, 30);
```

```
pushMatrix();
translate(33, 0);
rect(0, 20, 66, 30);
popMatrix();
rect(0, 50, 66, 30);
```
The \texttt{rotate()} function rotates the coordinate system so that shapes can be drawn to the screen at an angle.

- It has one parameter that sets the amount of the rotation as an angle: \texttt{rotate(angle)}
- The rotate function assumes that the angle is specified in units of radians.
Rotations

Shapes are always rotated relative to the origin (0,0), positive numbers rotate them in a clockwise direction.

smooth();
rect(55, 0, 30, 45);
rotate(PI/8);
rect(55, 0, 30, 45);

smooth();
rect(10, 60, 70, 20);
rotate(-PI/16);
rect(10, 60, 70, 20);
rotate(-PI/8);
rect(10, 60, 70, 20);
Scaling

- The `scale()` function magnifies the coordinate system so that shapes are drawn larger.
  - It has one or two parameters:
    - `scale(size)`
    - `scale(xsize, ysize)`
  - The parameters to scale are defined in terms of percentages expressed as decimals.
    - E.g., 2.0 for 200%, 1.5 for 150%, 0.5 for 50%
Scaling

Using just one parameter scales equally in x and y. Using two parameters allows the scaling to be different in x and y.

```plaintext
smooth();
ellipse(32, 32, 30, 30);
scale(1.8);
ellipse(32, 32, 30, 30);

smooth();
ellipse(32, 32, 30, 30);
scale(2.8, 1.8);
ellipse(32, 32, 30, 30);
```
Scaling

To keep the same stroke weight and scale a shape, divide the parameter of the `strokeWeight()` function by the scale value.

```plaintext
float s = 1.8;
smooth();
ellipse(32, 32, 30, 30);
scale(s);
strokeWeight(1.0 / s);
ellipse(32, 32, 30, 30);
```
Scaling

As with `translate()` and `rotate()`, the effects of each `scale()` accumulate each time the function is run.

```plaintext
rect(10, 20, 70, 20);
scale(1.7);
rect(10, 20, 70, 20);
scale(1.7);
rect(10, 20, 70, 20);
```
Combining Transformations

rect(50,20,40,20) → rotate(PI/12)

rect(0,0,40,20) → translate(50,20) → rotate(PI/12)
Combining Transformations

The order that transformations are applied can have a significant effect on the result.

```plaintext
smooth();
translate(width/2, height/2);
rotate(PI/8);
rect(-25, -25, 50, 50);
```

```plaintext
smooth();
rotate(PI/8);
translate(width/2, height/2);
rect(-25, -25, 50, 50);
```
Coordinate view
Reading the code from top to bottom

Shape view
Reading the code from bottom to top

Translate

Rotate

Draw rectangle

Draw rectangle

Rotate

Translate
Coordinate view
Reading the code from top to bottom

Shape view
Reading the code from bottom to top

Rotate

Translate

Draw rectangle

Draw rectangle

Translate

Rotate
Repeating Transformations

```java
background(0);
smooth();
stroke(255, 120);
translate(66, 33);
for (int i = 0; i < 18; i++) {
    strokeWeight(i);
    rotate(PI/12);
    line(0, 0, 55, 0);
}
```
Repeating Transformations

background(0);
smooth();
noStroke();
fill(255, 48);
translate(33, 66);
for (int i = 0; i < 12; i++) {
    scale(1.2);
    ellipse(4, 2, 20, 20);
}
Text & Typography
Characters

- The **char** data type stores typographic symbols such as A, d, 5, and $.
- The name **char** is short for character, chars are defined using single quotes.

```java
char a = 'n';    // Assign 'n' to variable a
char b = n;     // ERROR! Without quotes, n is a variable
char c = "n";   // ERROR! The "" defines n as a String
char d = 'not'; // ERROR! Chars can hold only one character
```
Characters as Numbers

char letter = 'A';
for (int i = 0; i < 26; i++) {
  print(letter);
  letter++;
}
println('.');
Characters as Numbers

```java
char letter = 'A';
for (int i = 0; i < 26; i++) {
    print(letter);
    letter++; 
}
println('.');
```

Output:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ.
```

```
Words and Sentences

- Use the **String** data type to store words and sentences.
  - Double quotation marks define a string.
- The **String** data type is an *object*.
  - String variables are declared similarly to `int`, `float`, etc., but the word `String` must be capitalised.
// The String data type can contain long and short text elements
String s1 = "Rakete bee bee?";
String s2 = "Rrrrrrrrrrrrrrrrrrrrrummmmmpfffff tillfffff toooooo?";
println(s1); // Prints "Rakete bee bee?"
println(s2); // Prints "Rrrrrrrrrrrrrrrrrrrrrummmmmpfffff tillfffff toooooo?"

// Strings can be combined with the + operator
String s3 = "Rakete";
String s4 = "rinnzekete";
String s5 = s3 + s4;
println(s5); // Prints "Rakete rinnzekete"
To use a font in Processing, it must first be converted into the VLW format.

To manually convert a font, select the “Create Font” option from the Tools menu. Select a font from the list and click “OK.” The font generated and is copied into the current sketch’s data folder.

To convert a font from within a Processing sketch we use the `createFont()` function.
Fonts

- To load a font use the `loadFont()` function
  - Processing has a unique data type called `PFont` to store font data.
  - Use `loadFont()` to load the font into a variable of type `PFont`.
  - Use the `textFont()` function with a loaded `PFont` object to set it as the current font.
To create a font from within a Processing sketch use the `createFont()` function

- Use the `createFont()` function to create the font in a variable of type `PFont`.
- The parameters passed to `createFont()` is a `String` containing the name of the font installed on the system to use, and the size of the font that we want to create.

```
createFont("Times New Roman", 18)
```
What Fonts Are Available?

› When using `createFont()` we often need to find out what fonts are available on a system from within our code.

› Processing provides a way to list all of the fonts installed on a system using the `PFont.list()` function:

```java
String[] fontNames = PFont.list();
```
```java
size(400, 1000);
background(0);

String[] fontList = PFont.list();
println(fontList);

fill(255);
for (int i = 0; i < height/20; i++) {
    textFont(createFont(fontList[i], 18));
    text(fontList[i], 20, (i+1)*20);
}
```
To load or not to load...

- Determining whether to manually create a font in Processing and then use `loadFont()` or to create a font programmatically using `createFont()` may depend on the types of font that you want to use.
- If you’re using a common font, then you can use `createFont()` and the system will likely have it.
- If you’re using an uncommon font, then you will probably be best creating it manually.
Drawing Text

- The `text()` function is used to draw characters to the screen:
  - `text(data, x, y)`
  - `text(stringdata, x, y, width, height)`
PFont font;
font = loadFont("Rockwell-ExtraBold-32.vlw");
textFont(font);
fill(0);
text("LAX", 0, 40);
text("AMS", 0, 70);
text("FRA", 0, 100);
Drawing Text with Word Wrapping

PFont font;
font = loadFont("Rockwell-ExtraBold-12.vlw");
textFont(font);
fill(0);
String s = "Response is the medium";
text(s,10, 20, 80, 50);

Response is the medium
Drawing Text with Colour

PFont font;
font = loadFont("Rockwell-ExtraBold-32.vlw");
textFont(font);
fill(255); // White
text("DAY", 0, 40);
fill(0); // Black
text("CVG", 0, 70);
fill(102); // Gray
text("ATL", 0, 100);
PFont font;
font = loadFont("Rockwell-ExtraBold-72.vlw");
textFont(font);
fill(0,160); // Black with low opacity
text("1", 0, 80);
text("2", 15, 80);
text("3", 30, 80);
text("4", 45, 80);
text("5", 60, 80);
Drawing Text with Multiple Fonts

PFont f1, f2;
f1 = loadFont("Rockwell-Bold-32.vlw");
f2 = loadFont("Rockwell-BoldItalic-32.vlw");
fill(0);
textFont(f1);
text("GNU", 6, 45);
textFont(f2);
text("GNU", 2, 80);
PFont font;
font = loadFont("Rockwell-ExtraBold-32.vlw");
textFont(font);
fill(0);
text("LNZ", 0, 40); // Large
textSize(18);
text("STN", 0, 75); // Medium
textSize(12);
text("BOS", 0, 100); // Small
PFont font;
font = loadFont("Rockwell-ExtraBold-12.vlw");
textFont(font);
textSize(32);
fill(0);
text("LNZ", 0, 40); // Large
textSize(18);
text("STN", 0, 75); // Medium
textSize(12);
text("BOS", 0, 100); // Small
Text Leading and Alignment

- The `textLeading()` function sets the spacing between lines of text:
  - `textLeading(dist)`
  - `dist` is the distance between lines in pixels

- The `textAlign()` function sets the text alignment:
  - `textAlign(MODE)`
  - `MODE` can be LEFT, CENTER, or RIGHT
Drawing Text with Leading

PFont font;
font = loadFont("Rockwell-ExtraBold-12.vlw");
textFont(font);
String lines = "L1 L2 L3";
textLeading(10);
fill(0);
text(lines, 5, 15, 25, 100);
textLeading(20);
text(lines, 36, 15, 25, 100);
textLeading(30);
text(lines, 68, 15, 25, 100);
Drawing Text with Alignment

PFont font;
font = loadFont("Rockwell-ExtraBold-12.vlw");
textFont(font);
line(50, 0, 50, 100);
fill(0);
textAlign(LEFT);
text("Left", 50, 20);
textAlign(RIGHT);
text("Right", 50, 40);
textAlign(CENTER);
text("Center", 50, 80);
Measuring Text with `textWidth()`

```java
PFont font;
font = loadFont("Rockwell-ExtraBold-32.vlw");
textFont(font);
fill(0);
char c = 'U';
float cw = textWidth(c);
text(c, 22, 40);
rect(22, 42, cw, 5);
String s = "UC";
float sw = textWidth(s);
text(s, 22, 76);
rect(22, 78, sw, 5);
```
Measuring Text with `textAscent()`

```cpp
PFont font;
font = loadFont("Rockwell-Bold-32.vlw");
textFont(font);

textSize(32);
float ascent = textAscent();
text("dp", 0, 70);
line(0, 70 - ascent, 100, 70 - ascent);

textSize(64);
ascent = textAscent();
text("dp", 35, 70);
line(35, 70 - ascent, 100, 70 - ascent);
```
Measuring Text with textDescent()

PFont font;
font = loadFont("Rockwell-Bold-32.vlw");
textFont(font);

textSize(32);
float descent = textDescent();
text("dp", 0, 70);
line(0, 70+descent, 100, 70+descent);

textSize(64);
descent = textDescent();
text("dp", 35, 70);
line(35, 70+descent, 100, 70+descent);
PFont font = loadFont("Rockwell-Bold-72.vlw");
textFont(font);
textAlign(CENTER, CENTER);

String s = "CODE";
float sw = textWidth(s);
float sh = textAscent() + textDescent();

size(400, 400);
smooth();
noStroke();
rectMode(CENTER);

background(255);
translate(width/2, height/2);
for (int i = 0; i < 16; i++) {
    fill(255, 32, 127, 128);
    rect(0, 0, sw, -sh);
    fill(255, 128);
    text(s, 0, 0);
    rotate(PI/8);
    scale(0.95);
}
Lab Exercises
Lab Exercises

- Use `translate()` to reposition a shape multiple times to create a composition.
  - Use `pushMatrix()` and `popMatrix()` to rearrange the composition.

- Use `scale()` with a for structure to scale a shape multiple times.

- Combine `translate()` and `rotate()` to rotate a shape around its own center.
Lab Exercises

› Draw a paragraph of text to the display window.

› Use two different typefaces to display the dialogue between two characters.
  ‣ Try using the `textWidth()`, `textAscent()` and `textDescent()` functions to calculate the size of your text and draw some decoration

› Use the text and transformation functions to create a complex composition.