CS110 Introduction to Computing

Professor Blank

Bryn Mawr College

Spring 2015

http://cs.brynmawr.edu/cs110db

Overview

- Random numbers
- mouseX, mouseY
- setup() & draw()
- frameRate(), loop(), noLoop()
- Mouse and Keyboard interaction
- Arcs, curves, bézier curves, custom shapes
- Hue-Saturation-Brightness vs. Red-Green-Blue color
- Example Sketches
- OpenProcessing website

Syntax

- Function *call*
  - `line( 10, 10, 50, 80 );`
  - Name
  - The commas
  - The parens ()
  - The semicolon

- Code block
  - The curly braces {}

- Comments
  - //
  - /* and */

Variables

- A name to which data can be assigned
- A variable name is declared as a specific data type
- Names must begin with a letter, “_” or “$” and can container letters, digits, “_” and “$”

```
boolean bReady = true;
int i;
int j = 12;
float fSize = 10.0;
color _red = color(255,0,0);
String name123 = “Fred”;
PImage img;
```

Variable Uses

- Use a value throughout your program,
  - but allow it to be changed
- As temporary storage for a intermediate computed result
- To parameterize – instead of hardcoding coordinates
- Special variables (preset variables)
  - **`width, height`**
  - **`screen.width, screen.height`**
  - **`mouseX, mouseY`**
  - **`pmouseX, pmouseY`**

Primitive Data Types

Type	Range	Default	Bytes
boolean	{ true, false }	false	?
byte	{ 0..255 }	0	1
int	{ -2,147,483,648 .. 2,147,483,647 }	0	4
long	{ -9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807 }	0	8
float	{ -3.40282347E+38 .. 3.40282347E+38 }	0.0	4
double	*much larger/smaller*	0.0	8
color	{ #00000000 .. #FFFFFFFF }	*black*	4
char	*a single character* 'a', 'b', ...	'\u0000'	2

Other "things" ...

Type	Range	Default	Bytes
String	a series of chars in quotes “abc”	null	?
PImage	an image	null	?
PFont	a font for rendering text	null	?

...

```
String message = "Hello World!";
```

Conditionals: if-statement

Programmatic branching …

```
if ( boolean_expression ) {
    statements;
}

// What does this do?
void draw() {
    if ( mouseX > 50 && mouseY > 50 ) {
        ellipse( mouseX, mouseY, 10, 10 );
    }
}
```

Relational Expressions

<	less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to
==	is equivalent
!=	is not equivalent

Relational Expressions: Examples

```
1.  if ( true ) { … }
2.  if ( 10 > 10 ) { … }
3.  if ( 10 >= 10 ) { … }
4.  if ( 'a' == 'a' ) { … }
5.  if ( 'a' != 'a' ) { … }
6.  if ( "Bryn Mawr" != "bryn mawr" ) { … }
```

Logical Expressions

&& logical conjunction (and)

- both expressions must be true for conjunction to be true

|| logical disjunction (or)

- either expression must be true for disjunction to be true

! logical negation (not)

- true $\rightarrow$ false, false $\rightarrow$ true

Logical Expression Examples

1. `if ( (2 > 1) && (3 > 4) ) { … }`
2. `if ( ("blah" == "blah") && (1 + 2 == 3) ) { … }`
3. `if ( !false ) { … }`
4. `if ( !(1 < -1) ) { … }`
5. `if ( !(10 < 20) || false ) { … }`
6. `if ( !(10 > 20) && (10 < 20) ) { … }`
7. `if ( (true || false) && true ) { … }`
8. `if ( (true && false) || true ) ) { … }`
9. …

Conditionals: if-else-statement

```
if ( boolean_expression ) {
   statements executed when boolean_expression is true;
} else {
   statements executed when boolean_expression is false;
}

// What does this do?
void draw() {
   if ( mouseY < 50 ) {
       println("the sky");
   } else {
       println("the ground");
   }
}
```

Conditionals: if-else-if-statement

```
if ( boolean_expression_1 ) {
        statements;
} else if ( boolean_expression_2 ) {
        statements;
} else if ( boolean_expression_3 ) {
        statements;
} else {
        statements;
}
```

What will this do?

```
void setup() {
  size(500,500);
  smooth();
  ellipseMode(CENTER);
}

void draw() {
  if (mouseX < 250) {
    stroke(255, 0, 0);

    if (mouseY < 250) {
      fill(0, 255, 0);
    } else {
      fill(0, 0, 255);
    }

  } else {
    stroke(0, 0, 255);

    if (mouseY < 250) {
      fill(255, 0, 0);
    } else {
      fill(255);
    }
  }
  ellipse(mouseX, mouseY, 50, 30);
}
```

```
void setup() {
  size( 500, 500 );
  smooth();
}

void draw() {
  if ( mouseX > 100 ) {
    background( 255, 0, 0 );
  } else if ( mouseX > 200 ) {
    background( 0, 0, 255 );
  }
}
```

What does this do?

```
void setup() {
  size( 500, 500 );
  smooth();
}

void draw() {
  if ( mouseX > 200 ) {
    background( 255, 0, 0 );
  }

  if ( mouseX > 100 ) {
    background( 0, 0, 255 );
  }
}
```

What does this do?

Conditionals: switch-statement

- Works like a if-else statement.
- Convenient for large numbers of value tests.

```
switch( expression ) {
  case label1:          // label1 equals expression
     statements;
     break;
  case label2:          // label2 equals expression
     statements;
     break;
  default:              // Nothing matches
     statements;
}
```

```
void setup() {
  size(500, 500);
  smooth();
}

void draw() {
}

void keyPressed() {
  switch(key) {
    case 'l':
    case 'L':
      println("Turning left");
      break;
    case 'r':
    case 'R':
      println("Turning right");
      break;
  }
}
```

What does this do?

```
int positionX = 250;
int positionY = 250;
int deltaX = 0;
int deltaY = 0;

void setup() {
  size(500, 500);
  smooth();
}

void draw() {
  background(255);

  positionX = positionX + deltaX;
  positionY = positionY + deltaY;

  if (positionX < 0)
    positionX = 0;
  if (positionX > width)
    positionX = width;
  if (positionY < 0)
    positionY = 0;
  if (positionY > height)
    positionY = height;

  ellipse(positionX, positionY, 50, 50);
}
```

```
void keyPressed() {
    switch (keyCode) {
    case 37:
      deltaX = -2;
      deltaY = 0;
      break;
    case 39:
      deltaX = 2;
      deltaY = 0;
      break;
    case 38:
      deltaY = -2;
      deltaX = 0;
      break;
    case 40:
      deltaY = 2;
      deltaX = 0;
      break;
    case 32:
      deltaX = 0;
      deltaY = 0;
      break;
  }
}
```