Better Program Design

Creative Coding & Generative Art in Processing 2
Ira Greenberg, Dianna Xu, Deepak Kumar

Slides revised by Michael Goldwasser

Variables: Naming Values

- **Values**
  42, 3.14159, 2013, "Hi, my name is Joe!", true, false, etc.
  - **Numbers**
    - **Integers**
      ```
      int meaningOfLife = 42;
      int year = 2013;
      ```
    - **Floating point numbers**
      ```
      float pi = 3.14159;
      ```
  - **Strings**
    ```
    String greeting = "Hi, my name is Joe!";
    ```
  - **Boolean**
    ```
    boolean keyPressed = true;
    ```

Variables: Naming Values

Variables have a Type

- **Values**
  42, 3.14159, 2013, "Hi, my name is Joe!", true, false, etc.
  - **Numbers**
    - **Integers**
      ```
      int meaningOfLife = 42;
      int year = 2013;
      ```
    - **Floating point numbers**
      ```
      float pi = 3.14159;
      ```
  - **Strings**
    ```
    String greeting = "Hi, my name is Joe!";
    ```
  - **Boolean**
    ```
    boolean keyPressed = true;
    ```


Variables: Naming Values

Variables have a Name

- **Values**
  42, 3.14159, 2013, "Hi, my name is Joe!", true, false, etc.
  - **Numbers**
    - **Integers**
      ```
      int meaningOfLife = 42;
      int year = 2013;
      ```
    - **Floating point numbers**
      ```
      float pi = 3.14159;
      ```
  - **Strings**
    ```
    String greeting = "Hi, my name is Joe!";
    ```
  - **Boolean**
    ```
    boolean keyPressed = true;
    ```

Variables: Naming Rules & Conventions

- Names begin with a letter, an underscore (_), or a dollar sign ($)

  Examples: `weight`, `_meaningOfLife`, `$value`

- Names may include numbers, but only after the initial character

  Examples: `value1`, `score5`, ~~`5bestFriends`~~

- No spaces are permitted in names

  Examples: ~~`value 1`~~, ~~`dollar sign`~~

- Processing Conventions
  - Names begin with a lowercase letter

    Example: `meaningOfLife`, `highestScore`

  - Constants are written in all caps

    Example: `DAYS_IN_WEEK`, `PI`

Variables: Declarations & Initialization

- Declaring variables

```
int meaningOfLife;
int year;
float pi;
String greeting;
boolean keyPressed;
```

- Initializing values in declarations

```
int meaningOfLife = 42;
int year = 2013;
float pi = 3.14159;
String greeting = "Hi, my name is Joe!";
boolean keyPressed = true;
```

The color type

- Processing has a type called color

```
color firebrick = color(178, 34, 34);
color chartreuse = color(127, 255, 0);
color fuchsia = color(255, 0, 255);
```

```
fill(firebrick);
rect(50, 100, 75, 125);
```

Expressions: Doing Arithmetic

- **Assignment statement**

  <variable> = <expression>;

  Examples:

```
meaningOfLife = 42;
area = length * height;
perc =statePop/totalPop*100.0;
```

- **Operators**

  \+ (addition)
  \- (subtraction)
  * (multiplication)
  / (division)
  % (modulus)

  Example:

```
mouth_x = ( (leftIris_x + irisDiam)/2 + eyeWidth )/4;
```

Arithmetic with int and float values

```
int x = 42;          vs     int x = 42.0;

float x = 42.0       vs     float x = 42;

float x = 7/2;       vs     float x = 7.0/2.0;
```


Arithmetic with int and float values

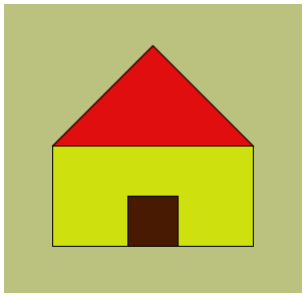
```
int x = 42;          vs     int x = 42.0;       // error

float x = 42.0       vs     float x = 42;       // same 42.0

float x = 7/2;       vs     float x = 7.0/2.0; // 3.0 vs 3.5
```

- Type of variable is important and determines the value that can be assigned to it.
- Result of division depends upon operands

`int/int`	yields an integer result
`float/int`	yields a float result
`int/float`	yields a float result
`float/float`	yields a float result

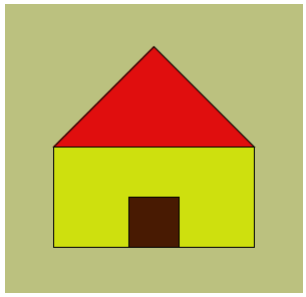
Using Variables

```
// Draw a simple house
// Create and set canvas

size(300, 300);
smooth();
background(187, 193, 127);

// wall
fill(206, 224, 14);
rect(50, 150, 200, 100);

// Draw Door
fill(72, 26, 2);
rect(125, 200, 50, 50);

// Draw roof
fill(224, 14, 14);
triangle(50, 150, 150, 50, 250, 150);
```



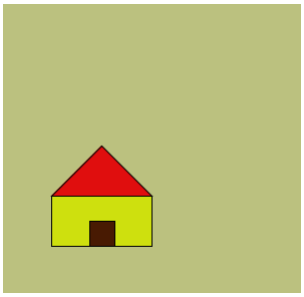
A Better House Sketch

```
// Draw a simple house
int houseX = 50;                // bottom left corner of house
int houseY = 250;

int houseHeight = 200;          // overall width and height of house
int houseWidth = 200;

int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
int roofHeight = houseHeight/2;
int doorHeight = houseHeight/4;
int doorWidth = houseWidth/4;

// Create and set canvas

size(300, 300);
smooth();
background(187, 193, 127);

// wall
fill(206, 224, 14);
rect(houseX, houseY - wallHeight,
     houseWidth, wallHeight);

// Draw Door
fill(72, 26, 2);
rect(houseX + houseWidth/2 - doorWidth/2, houseY-doorHeight,
     doorWidth, doorHeight);

// Draw roof
fill(224, 14, 14);
triangle(houseX, houseY - wallHeight,
         houseX+houseWidth/2, houseY-houseHeight,
         houseX+houseWidth, houseY-wallHeight);
```



A Better House Sketch

```
// Draw a simple house
int houseX = 50;                 // bottom left corner of house
int houseY = 250;

int houseHeight = 100;           // overall width and height of house
int houseWidth = 100;

int wallHeight = houseHeight/2;  // height of wall is 1/2 of house height
int roofHeight = houseHeight/2;
int doorHeight = houseHeight/4;
int doorWidth = houseWidth/4;

// Create and set canvas

size(300, 300);
smooth();
background(187, 193, 127);

// wall
fill(206, 224, 14);
rect(houseX, houseY - wallHeight,
     houseWidth, wallHeight);

// Draw Door
fill(72, 26, 2);
rect(houseX + houseWidth/2 - doorWidth/2, houseY-doorHeight,
     doorWidth, doorHeight);

// Draw roof
fill(224, 14, 14);
triangle(houseX, houseY - wallHeight,
         houseX+houseWidth/2, houseY-houseHeight,
         houseX+houseWidth, houseY-wallHeight);
```



Processing: Predefined Variables

- **width**, **height**
  The width & height of the canvas used in the sketch

- **PI**, **HALF_PI**, **TWO_PI**
  For different values of π. Note that

```
HALF_PI = PI/2
TWO_PI = 2*PI
```

- **displayWidth**, **displayHeight**
  The width and height of the monitor being used. This is useful in running fullscreen sketches using:

```
size(displayWidth, displayHeight);
```

- **mouseX**, **mouseY**
  The current mouse location in sketch (…coming soon!)

Additional Bells and Whistles

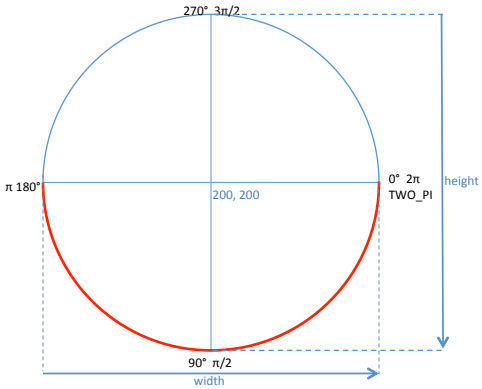
Basic Shapes: Arcs

- What is an arc?

Basic Shapes: Arcs

```
arc(x, y, width, height, startAngle, endAngle);
```

- degrees *vs* radians

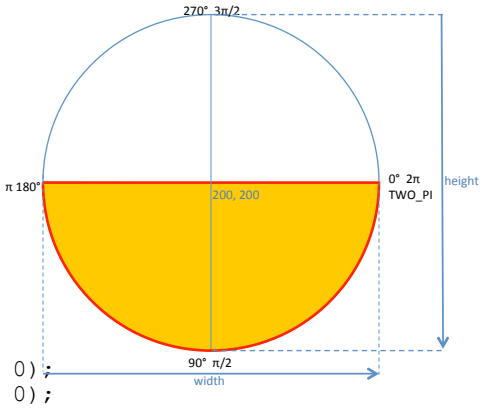



```
noFill();
stroke(255, 0, 0);
arc(200, 200, 150, 150, 0, PI);
```


Basic Shapes: Arcs

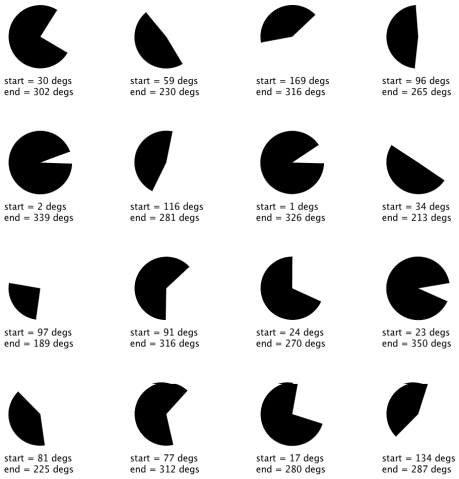
```
arc(x, y, width, height, startAngle, endAngle);
```

- degrees *vs* radians




```
fill(255, 255, 0);
stroke(255, 0, 0);
arc(200, 200, 150, 150, 0, PI);
```

Basic Shapes: Arcs



Basic Shapes: Quadrilaterals

```
quad(x1, y1, x2, y2, x3, y3, x4, y4);
```

```
noStroke();
fill(12, 37, 80);
quad(100, 50, 150, 100, 100, 150, 50, 100);
```

```
fill(240, 127, 71);
quad(100, 50, 200, 50, 250, 100, 50, 100);
```

```
noStroke();
fill(163, 208, 193);
quad(100, 50, 150, 100, 100, 150, 250, 100);
```

Drawing Text

`text(string, x, y);`
Draws string with bottom left corner at x, y

`textSize(fontSize);`
Can be used to specify font size

`fill(…)` used to specify color

`textAlign(…)` used to control vertical, horizontal alignment (see Processing reference)


```
size(300, 300);
background(185, 216, 153);

textSize(32);
text("Processing", 25, 100);
textSize(40);
fill(40, 62, 17);
text("Processing", 25, 150);
textSize(50);
fill(160, 20, 5);
text("Processing", 25, 200);
```