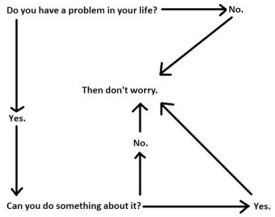
arc497i

Week 3

Review:
Control Flow
Conditionals
Loops
New:
Functions (Modularity, Abstraction)
Objects (OOP)
Arrays
Next assignment Assignment 1.2
Logistics: Scoring and Submissions

33

Conditionals



Conditionals

If I am hungry, order pizza.

Boolean expression — Execution code

Conditionals

If I am hungry, order pizza.

Boolean expression — Execution code

If I am hungry, order pizza. Otherwise, take a nap.

Boolean expression — Execution code — Execution code if the boolean expression evaluates to false

Conditionals

Boolean expressions always evaluate to either true or false.

- Learning how to program is fun -> **true**
- Le-Corbusier is an architect from the barroque -> **false**
- There are hundreds of restaurants in State College -> **false!**
- The result of adding 2 and 2 is 5 -> **false**
- 10 is less than 30 -> **true**

Conditionals

Boolean expressions always evaluate to either true or false.

x > 10 -> **depends on the value of x**
a == 1 -> **depends on the value of a**

Conditionals

Relational operators

>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
==	equality
!=	inequality

Conditionals

```
if (boolean expression){
   // code to execute if true
}
```

Conditionals

boolean expression

code to execute if boolean is true

```
if (mouseX < width/2){
   fill(255);
   rect(0, 0, width/2, height);
}
```

Conditionals

boolean expression

code to execute if boolean is true

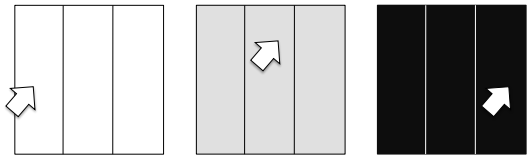
code to execute if boolean is false

```
if (mouseX < width/2){
   background(255);
}
else{
   background(0);
}
```

Conditionals

Exercise 6: Simple rollover

Use conditionals to change the colors of the background depending on the position of the cursor.



*Example from Shiffman

Logical Operators

Can help avoid nested ifs (which are inconvenient)

&&	(logical AND)
||	(logical OR)
!	(logical NOT)

Logical Operators

Exercise:

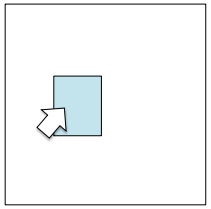
Are the following boolean expressions true or false when x = 5 and y = 6?

```
!(x>6)                  _____
(x == 6 && x == 5)      _____
(x == 6 || x == 5)      _____
(x > 1 && x < 10)       _____
```

Logical Operators

Exercise: A button

Create a button: when the mouse is pressed over a rectangle, the background should change color.
TIP: use logical operators.



Logical Operators

```
boolean button = false;

int x = 50;
int y = 50;
int w = 30;
int h = 30;

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

void draw(){

 if (button){
  background(255);
  stroke(0);
 } else{
   background(0);
   stroke(255);
 }
 fill(175);
 rect(x, y, w, h);
}

void mousePressed(){
 if (mouseX > x && mouseX < x+w && mouseY > y && mouseY < y+h)
{
   button = !button; //switch
 }

}
```

querying the Boolean variable

checking for rollover position

switching the Boolean variable

Logical Operators

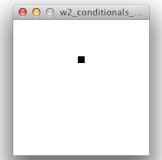
Exercise: Programming Interactivity

1. Take the code for the moving circle, make the circle start moving when the mouse is pressed.
2. Can you make it switch direction when it touches the edge of the window?

Programming Interaction

```
int circleX = 20;
int circleY = 100;

void setup(){
  stroke(0);
  fill(175);
  size(200,200);
}

void draw(){
  background(255);
  ellipse(circleX, circleY, 30, 30);
}
```

Simple Simulation



```
float x = 100;
float y = 0;

float speed = 0;
float gravity = 0.1;

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

void draw(){
  background(255);
  fill(0);
  noStroke();
  rectMode(CENTER);
  rect(x, y, 10, 10);
  y = y + speed;
  speed = speed + gravity;

  if (y> height){
    speed = speed * -0.95;
  }
}
```

Speed increment because of gravity

Reversing the speed when reaching the window's limit.

Programs as vehicles of design exploration.

A. Killian's CADenary project / Gaudí / Analog computation

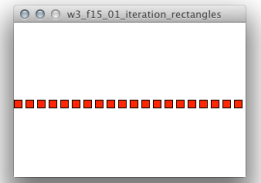
-https://vimeo.com/9662024
-http://designexplorer.net/projectpages/cadenary.html

-Can you think of other examples?

Iteration (loops)

This could take a lot of code:

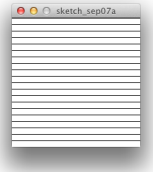
```
rect(0, width/2, 10, 10);
rect(15, width/2, 10, 10);
rect(30, width/2, 10, 10);
rect(45, width/2, 10, 10);
…
```



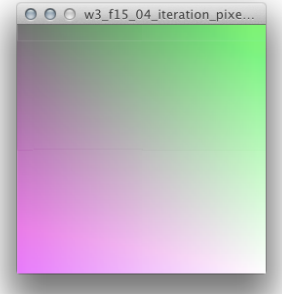
Iteration (loops)

Exercise

1. Write code that draws one of the following



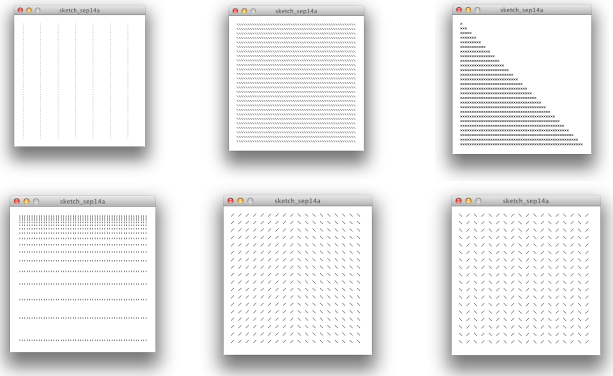
Iteration (loops)



Exercise: How to draw patterns in 2-D?

Try to imagine what code would the image above

Iteration (loops)



Iteration (loops)

-Shortcuts

```
x++;          x = x+1;
x--;          x = x-1;
x+=2;         x = x+2;
x*=3          x = x*3;
```

- Scope: Global vs. Local. Example.
- Careful with infinite loops, examples.

Review Last Week

Variables: named pointers to memory locations.

- Declare, initialize, use.
- There are system variables (width, mouseX, etc.), and user-defined variables.
- Both kinds are key.

Data Types:

- Important for memory handling.
- Required for variable declaration.
- int, float, char…

Chance (random) operations

- random(0, 100)
- some functions can return numbers

Review This Week

Conditionals.

- Boolean expressions.
- if, else, else if
- Logical operators.
- Simple math
- GUI basics (buttons, events, interactions)
- Speed, gravity (basic physics simulation)
- Variable scope

Iteration

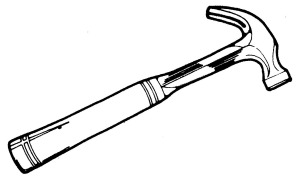
- While and for loops
- Nested iteration, pattern-making

Functions

Functions are blocks of code that can be re-called anytime by the program.

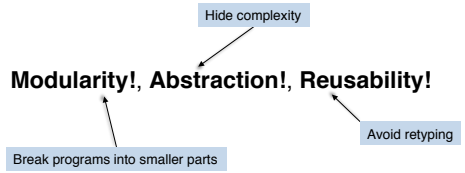
They are like tools you use to perform tasks

They are essential to make powerful, readable, debuggable programs



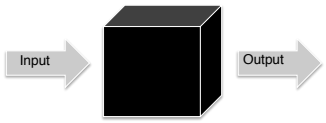
Functions

Functions represent 3 crucial characteristics of computation:



Functions

Functions are often referred to as “black boxes”



Functions

Defining functions.
3 things to remember:

- Return type
- Function name
- Arguments

Functions

Return Type | Function name | Arguments

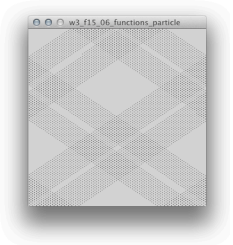
```
void draw_red_square(){
   // code for drawing red square
}
```

Note: Functions won’t work unless called!

Functions

Exercise: write a program that creates a a geometric shape of random size every time you click the mouse.

Functions




```
void draw(){
  background(200);

  if ((posx > width) || (posx < 0)){
    incrementx *= -1;
  }
  if ((posy > height)||(posy<0)){
    incrementy *= -1;
  }

  posx = posx + incrementx;
  posy = posy + incrementy;
  ellipse(posx, posy, 5, 5);


}
```

Exercise: Open w3_f15_06 and modularize it.

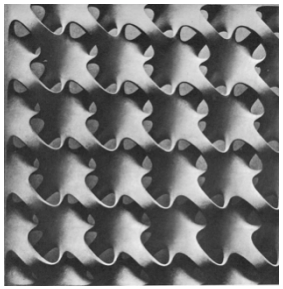
Functions

Modularization, variation, repetition, are concepts familiar to designers and architects.

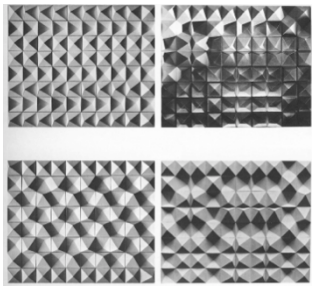
Bramante, Cancelleria, Rome, from Kepes, 1966. p. 107

Functions

Modularization, variation, repetition, are concepts familiar to designers and architects.

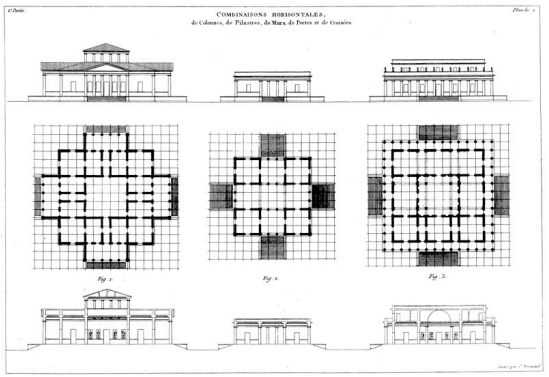


modular sunscreen, from Kepes, 1966. p. 91



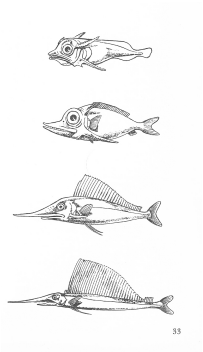
Surface combinatorial studies, from Kepes, 1966. p. 96

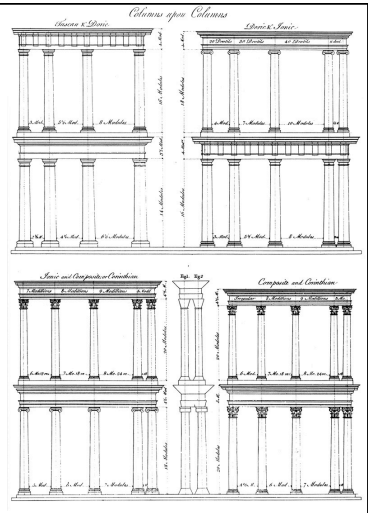
Functions



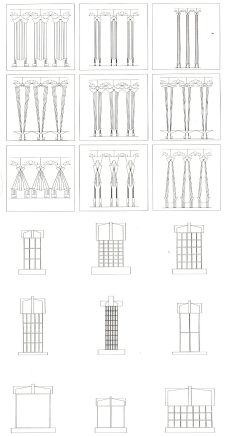
Taxis representada nos Précis des leçons d'architecture, J. N. L. Durand, 1823.http://daniloarquiteto.files.wordpress.com/2008/11/fig-03-taxis.jpg

Functions



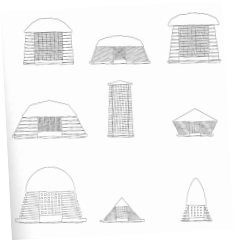


Functions



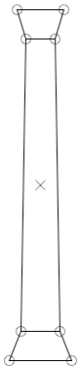
[Parametric] Variations on a window Motif by Charles Rennie Mackintosh. (Images by Eric Silver). From The Art of Computer Graphics Programming (Mitchell, Ligget, Kvan, 1987)

Functions



Variations on the elevation of the AEG Turbine Factory (Peter Behrens, Berlin, 1908). Images by Brooke Fabricant. From The Art of Computer Graphics Programming (Mitchell, Ligget, Kvan, 1987), p. 497

Functions
A Parametric Object



Capitel

Shaft

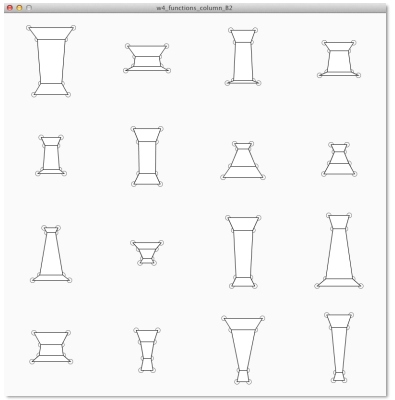
Base

Functions

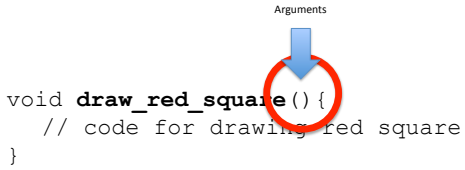
```
int x = 400;  // pos x
int y = 400;  // pos y

int bH = 60;  // base height
int sH = 200; // shaft height
int cH = 40;  // capital height
int dL = 50;  // diameter low
int dH = 45;  // diameter high

void setup(){
 size(800, 800);
 smooth();
}

void draw(){
  background(255);
  noFill();
  stroke(0);

  quad(x-(dH/2), y-(sH/2), x+(dH/2), y-(sH/2), x+(dH/2) + (dH*0.3), 1.0*(y-((sH/2)+cH)), x-((dH/2) + (dH*0.3)), y-((sH/2)+cH));
  quad(x-(dL/2), y + (sH/2), x +(dL/2), y + (sH/2), x + (dH/2), y - (sH/2), x - (dH/2), y - (sH/2));
  quad(x-(dL/2)-(dL*0.3), y + (sH/2) +bH, x+(dL/2)+(dL*0.3), y + (sH/2) + bH, x + (dL/2), y + (sH/2), x - (dL/2), y + (sH/2));
}
```

```
void keyPressed(){

 // change column dimensions

 if (key == CODED){
   if (keyCode == UP){
     shaftHeight += rate;
   }
   if (keyCode == DOWN){
     shaftHeight -= rate;
   }
   if (keyCode == LEFT){
     diameterLow -= rate;
   }
   if (keyCode == RIGHT){
     diameterLow += rate;
   }

 }
 }

void draw(){
      draw_column(width/2, height/2, baseHeight, shaftHeight, capitalHeight, diameterLow, diameterHigh);
}
```

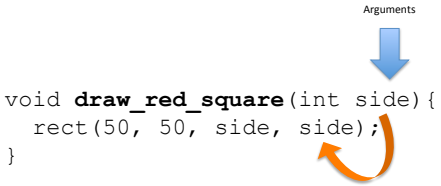
```
void draw_column(int x, int y, int bH, int sH, int cH,
int dL, int dH){

  background(255);

  noFill();

  strokeWeight(1);

  //draw capital
  quad(x-(dH/2), y-(sH/2), x+(dH/2), y-(sH/2), x+(dH/2) +
(dH*0.3), 1.0*(y - ((sH/2)+cH)), x-((dH/2) + (dH*0.3)), y
- ((sH/2)+cH));

  // shaft
  quad(x-(dL/2), y + (sH/2), x +(dL/2), y + (sH/2), x +
(dH/2), y - (sH/2), x - (dH/2), y - (sH/2));

  // base
  quad(x-(dL/2)-(dL*0.3), y + (sH/2) +bH, x+(dL/2)+
(dL*0.3), y + (sH/2) + bH, x + (dL/2), y + (sH/2), x -
(dL/2), y + (sH/2));

}
```



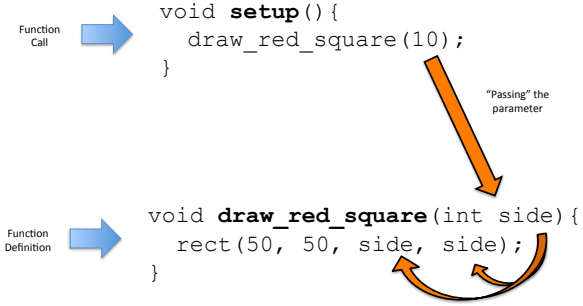
Parametric constructs:
the power of arguments



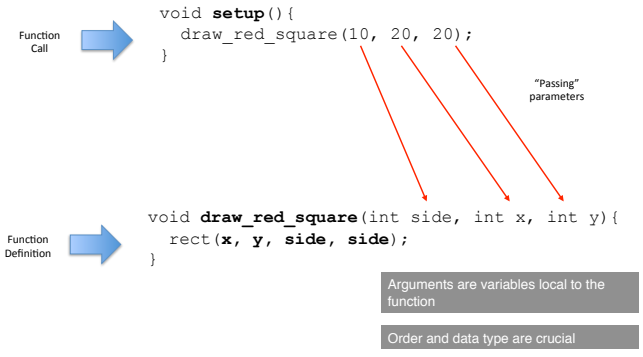
Parametric constructs:
the power of arguments



Parametric constructs:
the power of arguments



Parametric constructs:
the power of arguments



Exercise 3 (board):

consider the following function

```
void sum(int a, int b, int c){

  int total = a + b + c;
  println(total);

}
```

Write the code to call that function

Functions

Exercise 4 (board):

consider the following code

```
multiply(3.3, 2.9);
```

Write the function that performs the indicated operation and prints the result in the screen.

Exercise 5 (laptops):

```
int x = 0;
int y = 50;
int s = 1;

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

void draw(){
  background(255);
  _____________________________;
  _____________________________;
  _____________________________;
}

void move(){
 x = x + s;
}

void bounce(){
 if ((x >= width) || (x <= 0)){
  s *= -1;
 }
}
```

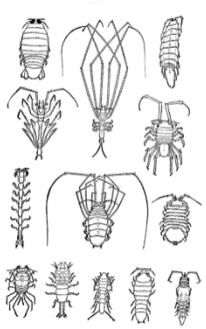
```
void drawCar(int x, int y, int size, color
c){

 int o = size/4;
 rectMode(CENTER);
 stroke(200);
 fill(c);
 rect(x, y, size, size/2);
 fill(200);

 rect(x-o, y-o, o, o/2);
 rect(x+o, y-o, o, o/2);
 rect(x-o, y+o, o, o/2);
 rect(x+o, y+o, o, o/2);

}
```

Exercise

Write a function that takes any value in inches and returns its value in mm.

Exercise 7: parametric form (dep. on time)

Take one of your previous sketches, modularize it creating functions for drawing the element(s). Now use a loop to call the function transforming the arguments.



Functions
Return Types

```
void sum(int a, int b){
  println(a + b);
}

int sum(int a, int b){
  int total = a + b;
  return total;
}
```

Now we can do things like

```
int x = sum(40, 30);
int z = sum(40, 30)/3;
line(10, 10, 20, sum(14, 12));
```

Object Oriented Programming (OOP)

```
int my_int();

someobject my_object();
```

When we create classes, we define new object types. They work the same as Processing's primitive data types (int, float, etc.)

Classes are also blocks of code.

Object Oriented Programming (OOP)
Anatomy of a Class

```
class Spot(){
  float x, y, diameter;
  Spot(float xpos, float ypos, float dia){
    x = xpos;
    y = ypos;
    diameter = dia;
  }
  void display(){
    ellipse(x, y, diameter, diameter);
  }
}
```

Class name

Data

Constructor

Functionality

All classes must contain name, data, constructor, and functions!

Object Oriented Programming (OOP)
How to use classes: Create Objects

```
Spot my_spot;

void setup(){
  size(100, 100);
  smooth();
  noStroke();
  sp = new Spot(33, 50, 30);
}

void draw(){
  background(0);
  sp.display();
}
```

Declaring the object

Object initialization

Call to object's function
Notice '.' syntax.

Object Oriented Programming (OOP)
How to use classes: Create Objects

```
Spot mySpotA, mySpotB, mySpotC;

void setup(){

  size(200, 200);
  smooth();
  noStroke();
  mySpotA = Spot(20, 50, 40);
  mySpotB = Spot(30, 10, 50);
  mySpotC = Spot(80, 50, 30);

}

void draw(){
  fill(0, 15);
  rect(0,0,width, height);
  fill(255);
  mySpotA.display();
  mySpotB.display();
  mySpotC.display();
}
```

Declaring objects

Initializing objects (using constructor)

Calling object's functions

OOP is just a way of organizing functionality around entities we define.

Object Oriented Programming (OOP)
Add Functionality

```
class Spot{
  float x, y, diameter, speed;
  Spot(float xpos, float ypos, float dia, float sp){
    x = xpos;
    y = ypos;
    diameter = dia;
    speed = sp;
  }
  void move(){
    y+= speed*direction;
    if ((y>(height-diameter/2))||(y<diameter/2)){
      direction*=-1;
    }
  }
  void display(){
    ellipse(x, y, diameter, diameter);
  }
}
```

How would we have to change the main code?

Object Oriented Programming (OOP)
Using Arrays to Store Multiple Objects

```
int numSpots = 10;
Spot[] spots = new Spot[numSpots];

Spot my_spot;

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

  for (int i = 0; i<spots.length; i++){
     float x = 10 + i*16;
     spots[i] = new Spot(x, 50, 16, random(0.3, 2.0));
  }
}

void draw(){
  background(0);
  for (int i = 0; i < spots.length; i++){
    spots[i].move();
    spots[i].display();
  }
}
```

Declaring array size

Declaring array

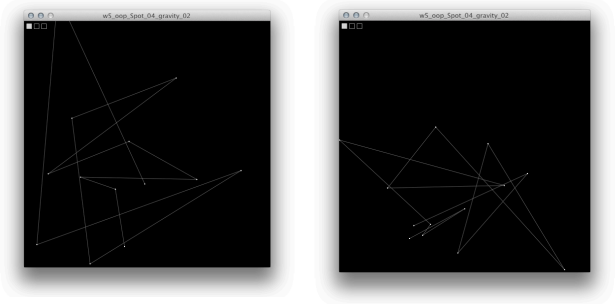
Arrays are compound data types. They extend a collection Java class.

In essence, they are collections of objects we can refer to individually.

"populating" the array

Iterating through the contents of the array

Object Oriented Programming (OOP)
Using Arrays to Store Multiple Objects



Review

Functions: modularity, abstraction

- Creating our own functions
- Return types
- Parametric forms

Object Oriented Programming: another kind of modularity. Functionality grouped around entity types.

-Classes
  -Data
  -Constructor
  -Functionality

-Objects
  - Initialization
  - Activation

-Arrays: collections of objects

A Useful IDE

Sublime Text 2 (http://www.sublimetext.com/2)

Yugo Nakamura

http://yugop.com/ver3/index.asp?id=12

http://yugop.com/ver3/index.asp?id=12

Next class:

We've already covered the fundamental topics of computer programming. What follows is a bit more exploratory, building on top of what we know.

-Specific algorithms
-Using libraries
-Good practices

Logistics

- All assignments due before class.
  - Blog Post with full documentation
  - Code Post/s in GitHub