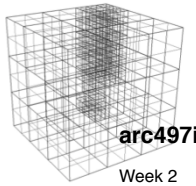


arc497i

Week 2

Assignment: State of the Art – Computational Design
Re-cap
Variables (What? How? For what?)
Data Types
Syntax
Control Flow
Conditionals
Loops
Next assignment Assignment 1.2
Logistics: Scoring and Submissions

Assignment 1.1

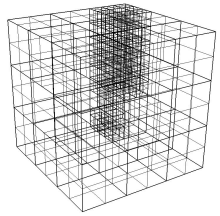
Review

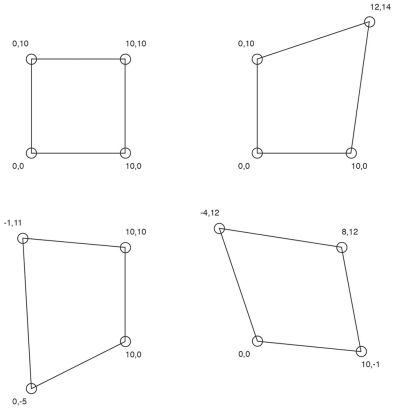
Recap

What we are doing
Basics, canvas.
Drawing with code: shapes, pixels, colors.
Processing basics: syntax, function calls
Processing reference
Interactivity basics: setup and draw, mouse and keyboard
Questions?

GEOMETRY AS DATA, DATA AS MATERIAL

What are designs made of?





Computer drawing as structured drawing

A BIT OF POETRY

A point is a pair (or a triad) of coordinates.
A line is a pair of points.
A surface is what results from interpolating a collection of lines.
A polygon is a closed collection of lines.
A solid is the projection of a polygon into the third dimension.
But all these objects can also be spaces of their own.
We can imagine we live in a point, a zero-dimensional space.
We can imagine we live in a line, a one-dimensional space.
We can imagine we live in a square, a two-dimensional space.
We can imagine we live in a box, a three-dimensional space.
And so on...

A BIT OF POETRY

An active line on a walk, moving freely, without goal. A walk for a walk's sake. The mobility agent, is a point, shifting its position forward (Fig. 1):

Fig. 1

The same line, accompanied by complementary forms (Figs. 2 and 3):

Fig. 2

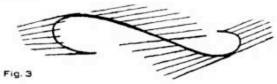


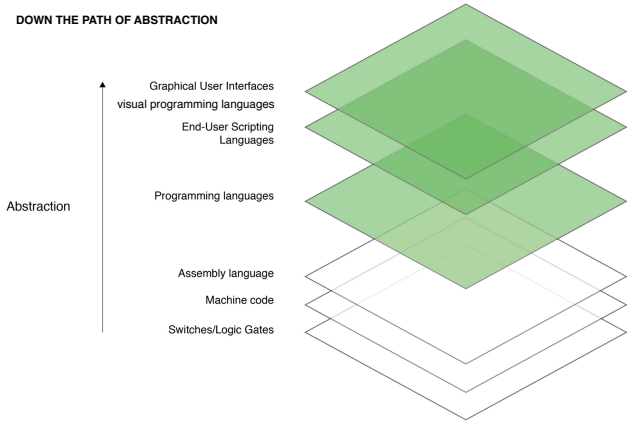
Fig. 3

Paul Klee. Pedagogical Sketchbook

$TDCREATE	9	$SPLFRAME
40	$USRTIMER	70
2454838.701285579	70	0
9	1	9
$TDUCREATE	9	$SPLINETYPE
40	$ANGBASE	70
2454838.909618912	50	6
9	0.0	9
$TDUPDATE	9	$SPLINESEGS
40	$ANGDIR	70
2454838.70128559	70	8
9	0	9
$TDUUPDATE	9	$HANDSEED
40	$PDMODE	5
2454838.909618923	70	2CB
9	0	9
$TDINDWG	9	$SURFTAB1
40	$PDSIZE	70
0.0000000116	40	6
9	0.0	9
$TDUSRTIMER	9	$SURFTAB2
40	$PLINEWID	70
0.0000000116	40	6

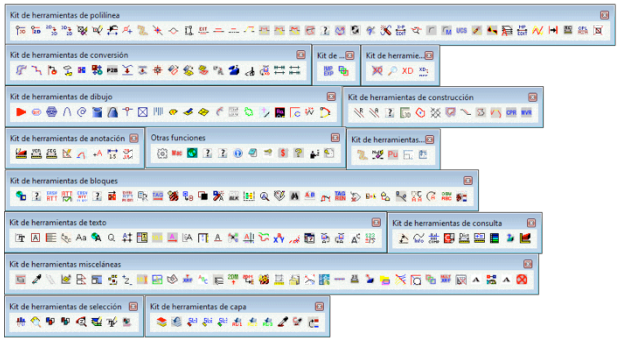
Geometry as data
Excerpt of the geometric information in a DXF computer file

DOWN THE PATH OF ABSTRACTION



Layers of Abstraction diagram

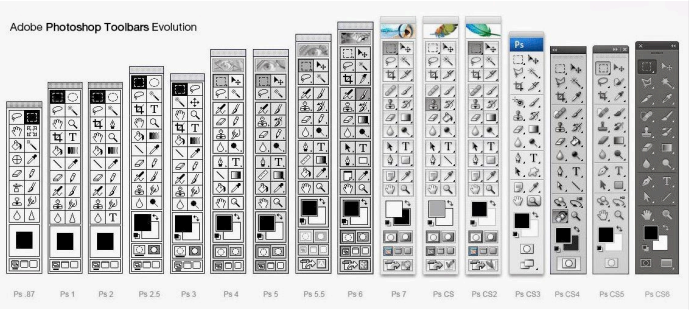
DOWN THE PATH OF ABSTRACTION: GRAPHICAL USER INTERFACES



The Graphic User Interfaces (GUIs) commonly resemble the tools. This is a metaphor allowing users to understand the unfamiliar based on the familiar.

Seeing these toolbars we can think… It is a manual world after all!

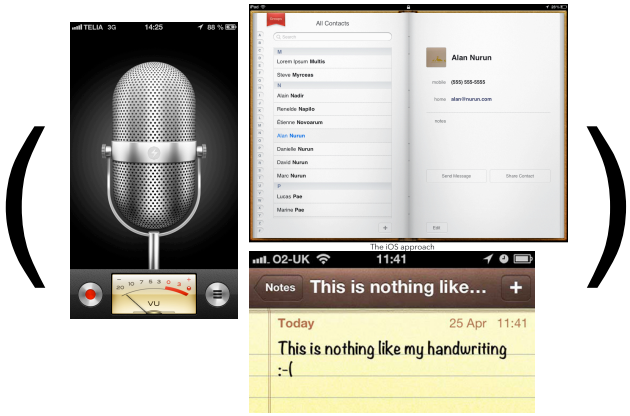
DOWN THE PATH OF ABSTRACTION: GRAPHICAL USER INTERFACES



The Graphic User Interfaces (GUIs) commonly resemble the tools. This is a metaphor allowing users to understand the unfamiliar based on the familiar (key word -> skeuomorphic).

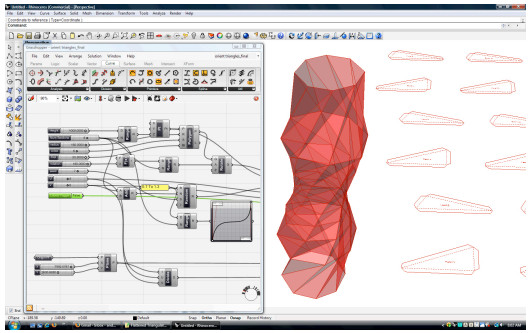
Seeing these toolbars we can think… It is a manual world after all!

DOWN THE PATH OF ABSTRACTION: SKEUOMORPHISM



Let's digress for a second to learn the word "Skeuomorphism."

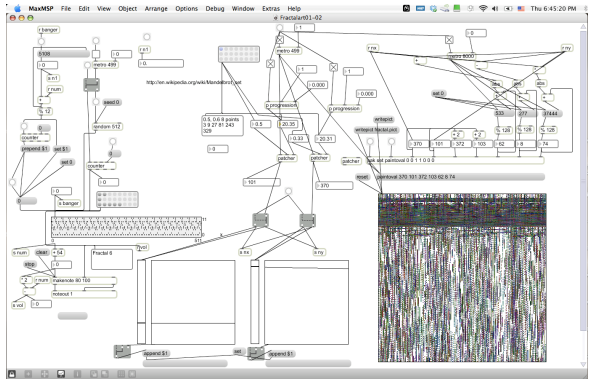
DOWN THE PATH OF ABSTRACTION: VISUAL PROGRAMMING LANGUAGES



GRASSHOPPER AND MAX

Visual programming languages provide users who do not know how to program with tools to automate software's functions.

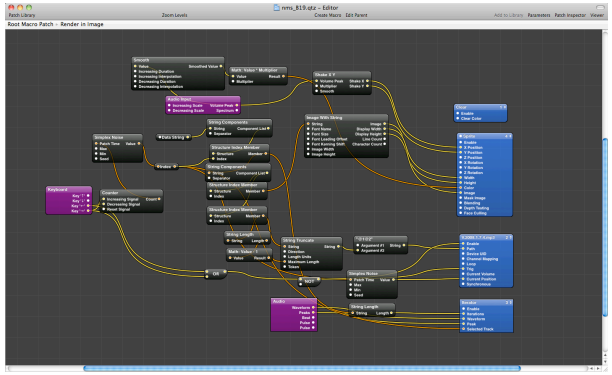
DOWN THE PATH OF ABSTRACTION: VISUAL PROGRAMMING LANGUAGES



Visual programming languages provide users who do not know how to program with tools to automate software's functions.

MAX MSP

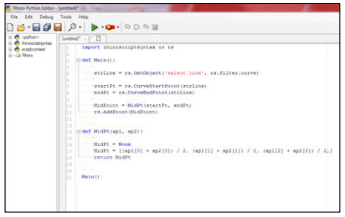
DOWN THE PATH OF ABSTRACTION: VISUAL PROGRAMMING LANGUAGES

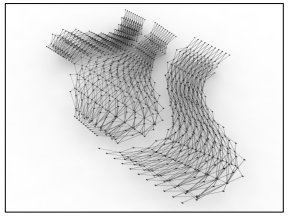


Visual programming languages provide users who do not know how to program with tools to automate software's functions.

Quartz Composer

DOWN THE PATH OF ABSTRACTION: END–USER SCRIPTING





RhinoScript, 3DSMax, Illustrator, AutoCAD

End User Programming Languages enable the automation of software functions. They work within software environments, and produce **scripts**.

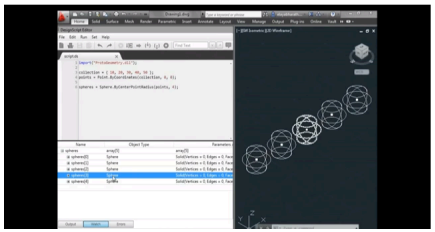
They are commonly based upon the syntax, or comprise subsets of, programming languages (RhinoScript-VB, AutoCAD-LISP, ActionScript(Flash)-Java/C).

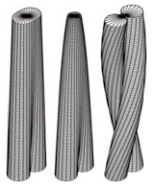
From Excel and Word to AutoCAD and CATIA, a wide range of software systems allow users to automate their functions using end-user scripting.

References

Robert Aish, Neil Katz

DOWN THE PATH OF ABSTRACTION: END–USER SCRIPTING





RhinoScript, 3DSMax, Illustrator, AutoCAD

End User Programming Languages enable the automation of software functions. They work within software environments, and produce **scripts**.

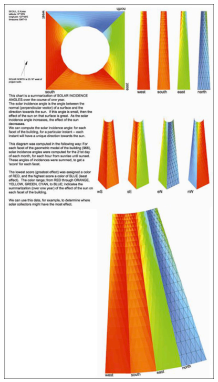
They are commonly based upon the syntax, or comprise subsets of, programming languages (RhinoScript-VB, AutoCAD-LISP, ActionScript(Flash)-Java/C).

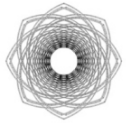
From Excel and Word to AutoCAD and CATIA, a wide range of software systems allow users to automate their functions using end-user scripting.

References

Robert Aish, Neil Katz

DOWN THE PATH OF ABSTRACTION: END–USER SCRIPTING



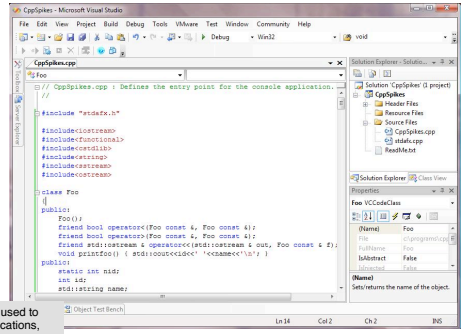


SOM Lotte Tower. Computational Work by Neil Katz

References

Robert Aish, Neil Katz

DOWN THE PATH OF ABSTRACTION: PROGRAMMING LANGUAGES



Programming Languages can be used to create stand-alone software applications, executable files we call programs.

They are commonly written within Interactive Development Environments (like Eclipse), or IDEs.

DOWN THE PATH OF ABSTRACTION: ASSEMBLY LANGUAGE

```
section   .text
    global _start   ;must be declared for linker (ld)
_start:             ;tells linker entry point
    mov   edx,len   ;message length
    mov   ecx,msg   ;message to write
    mov   ebx,1     ;file descriptor (stdout)
    mov   eax,4     ;system call number (sys_write)
    int   0x80      ;call kernel

    mov   eax,1     ;system call number (sys_exit)
    int   0x80      ;call kernel

section   .data
msg db 'Hello, world!', 0xa  ;our dear string
len equ $ - msg      ;length of our dear string
```

When the above code is compiled and executed, it produces following result:

```
Hello, world!
```

Assembly Language is pretty bad.

Who uses this?

People who design and maintain higher level programming languages (like Java, C, etc.).

People who write compilers (compiling means translating a high level program into assembly code).

People with a lot of time in their hands.

DOWN THE PATH OF ABSTRACTION: MACHINE CODE

```
8020   78
 8021  A9 80
 8023  8D 15 03
 8026  A9 2D
 8028  8D 14 03
 802B  58
 802C  60
 802D  EE 20 D0
 8030  4C 31 EA
```

Machine code is badass.

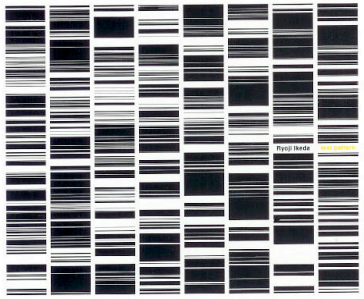
"may be regarded as the lowest-level representation of a compiled and/or assembled computer program or as a primitive and hardware-dependent programming language."

Who uses this?

Geeks / Super programmers

People with a lot of time in their hands.

DOWN THE PATH OF ABSTRACTION: SWITCHES



Processors work binary switches turning on and off (or rather: electrically charging/ discharging).

There are no good images of this. All I could find was super cheesy.

So here is a Ikeda Ryoji image.

Recap

What we are doing

Processing Basics

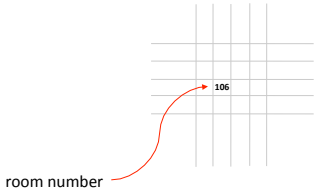
- Drawing with code: canvas, shapes, pixels, colors.
- Processing basics: syntax, function calls
- Processing reference
- Interactivity basics: setup and draw, mouse and keyboard

Variables

A variable is a named pointer to a location in the computer's memory where data is stored.

It can be thought of as a bucket, or box.

Crucial for the program to 'remember' information, in order to refer back to it. What kind of information? Any: color, positions, text, sizes… You name it.

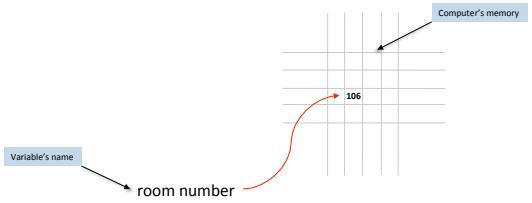


Variables

A variable is a named pointer to a location in the computer's memory where data is stored.

It can be thought of as a bucket, or box.

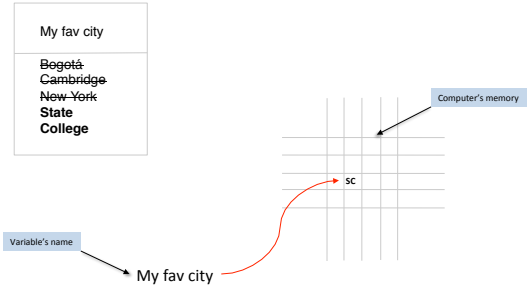
Crucial for the program to 'remember' information, in order to refer back to it. What kind of information? Any: color, positions, text, sizes… You name it.



Variables

Variables can, well, vary…

For example, a "My fav city" variable could change in time:



Variables

- Variables can hold different types of objects.
- There are *primitive* and *compound* objects.
- Some primitive types are: *int, float, char…*
- A simple variable declaration looks like this:

```
int course_room;
```

Declaring variables

(Primitive) Data types:

```
boolean   // true or false
char      // 'a' , 'b' , etc.
byte      // a small number (-128 to 127)
short     // a large number (-32768 to 32767)
int       // a big number (-2147 b to 2147 b)
long      // a huge number (...)
float     // a 32 bit decimal
double    // a 64 bit decimal
```

(Compound) Data types:

```
Arrays
Strings
…
```

Initializing variables

```
int course_room;
course_room = 106;
```

Declaration

Assignation

or

```
int course_room = 10;
```

Shortcut!

Examples

```
int room_number = 106;
char initial = 'd';
float a = 10.0;
float b;
b = a + 3.4;
float c = a/b +3;
```

For now, place your variables on top of your code.

Value assignation is always right to left.

Simple math with variables

+	add
-	subtract
*	multiply
/	divide

Basic math operators

For example

```
int posX = 15;
line(posX, 0, posX, height);
posX = posX + 40;
strokeWeight(4);
line(posX, 0, posX, height);
```

New system variable!

Naming Variables

- Don't use spaces or symbols (except underscore).
- Variable names can't start with a number.
- Avoid using system variables.
- Use meaningful names (avoid generic or unrelated names).
- Naming styles: "Camel-case" or "underscored."

Unrelated image

Example

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

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

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

Variable declaration and initialization

Use of variable in a function call

Example

```
int circleX = 0;
int circleY = 100;

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

void draw(){
  background(255);
  ellipse(circleX, circleY, 30, 30);
  circleX = circleX + 1;
}
```

Variable declaration and initialization

Introducing variation. This assignment increments the value of circleX at each iteration of the draw() method.

Exercise 2

Write code for a program that draws a circle that grows in size, following the mouse.

Example

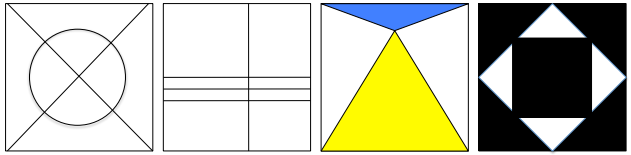
How can we change the speed, and rates of change, of the circle?

```
float rectangleX = 0;
float rectangleY = 0;
float rectangleW = 20;
float rectangleH = 20;
float backgroundColor = 255;
float change = 0.1;

void setup(){
  size(500, 500);
  fill(0);
  smooth();
}

void draw(){
  background(backgroundColor);
  rect(rectangleX, rectangleY, rectangleW, rectangleH);
  rectangleX = rectangleX + change;
  rectangleY = rectangleY + change;
  change +=0.2;
}
```

The variable "change" is used to increment and decrement the values.

Exercise 3 (pairs)

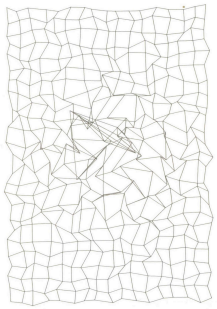
1. Pick one of the images below.
2. Write code that draws them with hard-coded values.
3. Replace the hard-coded values with variables
4. Write assignment operations in draw() that change the value of the variables (and the drawing). Consider interactivity.



Incremental Development

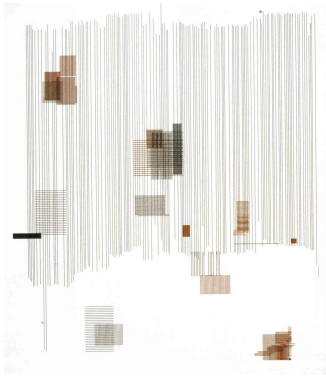
Complexity is often built of simple rules and elements.
So far: kind of "deterministic" ways of producing output.

Chance

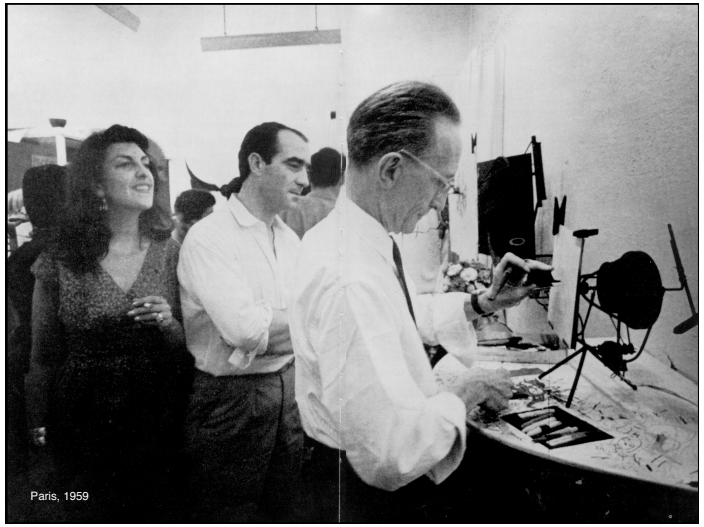


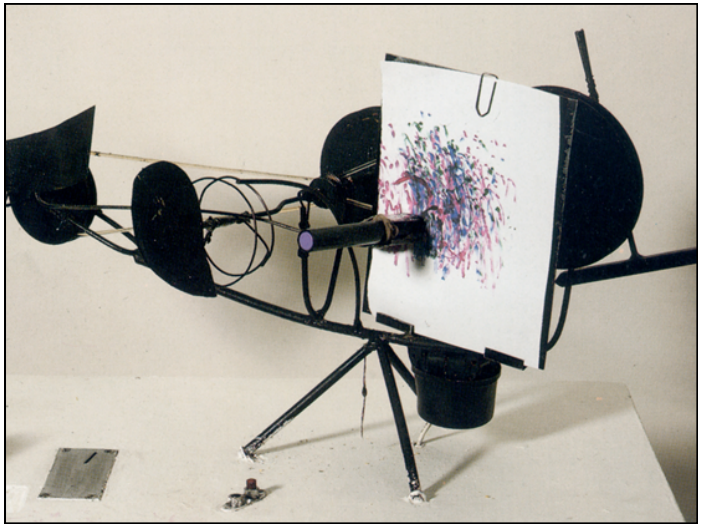
Georg Nees (1968)

Chance



Frieder Nake (1965)





Chance

```
float w = random (1, 100);
rect (10, 10, w, 50);
```

The function random "returns" a number

Chance

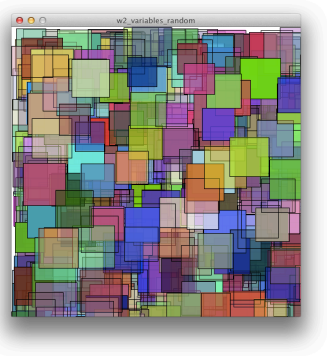
```
float w = random (1, 100);
rect (10, 10, w, 50);
```

The function random "returns" a number

if you need an integer, you need to "cast " it.

```
int w = (int) random (1, 100);
rect (10, 10, w, 50);
```

Chance



Write the code for creating this

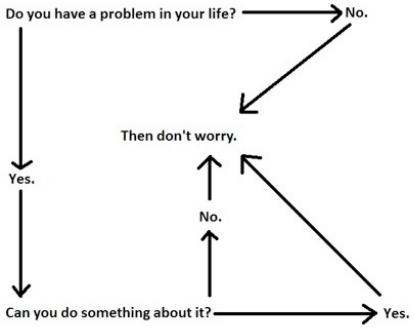
So Far:

Variables
- Named pointers to memory locations.
- Declare, initialize, use.
- There are system variables (width, mouseX, etc.), and user-defined.

Data Types
- Important for memory handling.
- Required for variable declaration.
- int, float, char…
-Primitive and compound.

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

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

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

boolean expression

code to execute if boolean is true

Conditionals

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

boolean expression

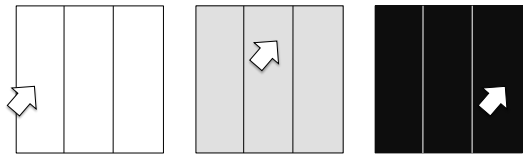
code to execute if boolean is true

code to execute if boolean is false

Conditionals

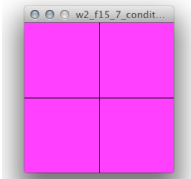
Exercise 6: Simple rollover

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



*Example from Shiffman

Conditionals



Logical Operators

Can help avoid nested ifs (which are inconvenient)

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

Logical Operators

Exercise:

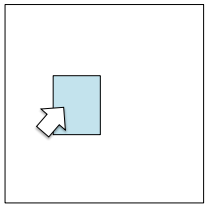
Are the following boolean expressions true or false when x = 5 and y = 6?

```
!(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

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?

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:

```
line(0, 0, 0, height);
line(10, 0, 10, height);
line(20, 0, 20, height);
line(30, 0, 30, height);
…
```

Iteration (loops)

In a way, loops are kinds of conditionals
They do something while a certain condition is met.

While loop:

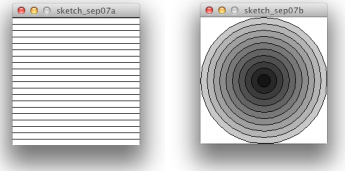
Boolean expression

```
int count = 0;
int posX = 0;
int spacing = 10;

while (count < 10){
  line(posX + (spacing * count), height, posX + (spacing * count), 0);
  count = count + 1;
}
```

Iteration (loops)

Exercise

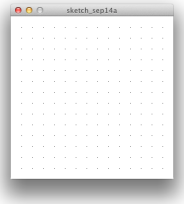
1. Write code that draws one of the following



Iteration (loops)


```
For (i = 0; i < 10; i = i+1){

}
```

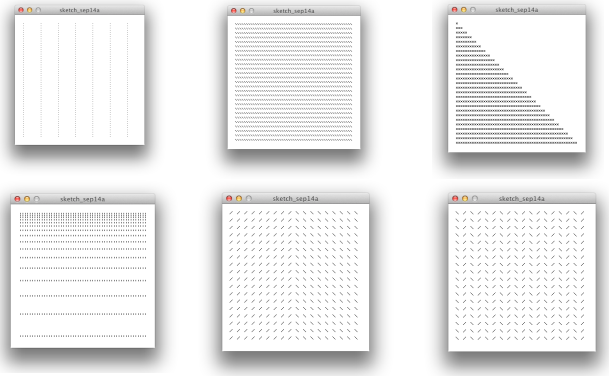
Iteration (loops)



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

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

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

Extra

- Shapes

Logistics

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