Changing Control

Testing and Repetition

Lawrence Snyder
University of Washington, Seattle
Let’s Begin W/ Idea From Last Lab

- We saw how to change the color of the ball and its direction with a mouse & key clicks
- Recall

```cpp
void keyPressed() {
    incDec = -incDec;
}
void mousePressed() {
    int temp;
    temp = bPos;
    bPos = gPos;
    gPos = rPos;
    rPos = temp;
}
```
Rule: Assignment always moves information from right to left, as in

```c
void keyPressed ( ) {
  incDec = - incDec;
}
```

Rule: Always evaluate (compute) the right side, then assign the result to the name on the left side ...
Expressions

- Facts about expressions
  - Expressions are formulas using:
    + - * / % || ! && == < <= >= > !=
  - Operators can only be used with certain data types and their result is a certain data type
  - Putting in parentheses is OK, and it’s smart

- Rules about expressions
  - Expressions can usually go where variables can go
Expressions, the Picture

- **Facts**
  - Expressions are formulas: $a+b$  
    $points*wgt$  
    $(year\%4==0)$  
    $7!=4$  
    $(age>12) \&\& (age<20)$
  - “Need & give data types”
    - $+ - * / \% < <= => >$  
      want numbers;
    - $\&\& ! ||$  
      want logical (Boolean) values
    - $== and !=$  
      want operands to be same type
  - “Parentheses are good”: $(a * b) + c$ is the same as  
    $a*b+c$, but easier to read
mod (%) is what’s left after divide

- $a \% b$ (read, “a mod b”) is the amount left after “b divides into a evenly”

- **Examples:**
  - $0 \% 3$ is 0
  - $1 \% 3$ is 1
  - $2 \% 3$ is 2
  - $3 \% 3$ is 0
  - $4 \% 3$ is 1
  - $5 \% 3$ is 2
  - $6 \% 3$ is 0

  **Even:** a number $n$ is even if $n \% 2 == 0$

  **Leap Year:** $\text{year}$ is a leap year if $\text{year} \% 4 == 0$

  **Asian Zodiac:** $\text{year1}$ and $\text{year2}$ are the same sign if $\text{year1} \% 12 == \text{year2} \% 12$
As numbers get larger, mod will cause them to “drop to 0” ... this is a Ninja move

```c
int ra = 0;

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

void draw() {
    background(255, 245, 220);
    raff();
    ra = (ra + 1)%150;
}

void raff() {
    fill(0,100,0);
    rect(240,260+ra, 40, 45);
    ...
```
Repetition (or looping)

- Repeating commands is a powerful way to use a computer ... we could repeat them, but all programming systems have a way to loop:
  - Lightbot 2.0 used recursion, a function calling itself
  - Symbolic Lightbot prefixed a number, 2:Step

- Processing (and other modern languages) use a **for** loop: count 0,1,2,3,4

```plaintext
for (i = 0; i < 5; i = i + 1) {
    rect(10+20*i,10,10, 10);
}
```
A for loop has several parts, all required ...

```
for (int j = 0; j < 10; j = j + 1) {
   <stuff to be repeated>
}
```

The result of this statement is 10 copies of the stuff to be repeated
Tests, A/K/A If statements

- The instructions of a program are executed sequentially, one after another ... sometimes we want to skip some: Say “Hello” to the If

- If also has a required form

```cpp
if (year%4 == 0) {
    <stuff to do if condition true>;
}
```

```cpp
if (chosen_tint != color(0,0,255)) {  //No TRUE blue!
    fill(chosen_tint);
}
```
An **if**-statement has a standard form:

```java
if ( bmi > 18.5 && bmi <= 24.9 ) {
    fill(0, 255, 0);
}
```

The result is that if bmi is in range the fill color is green (indicating OK).
What happens if we want to do something else if the condition is false? What else? `else`!

The `else` statement must follow an `if` ...

```java
if (year%4 == 0) {
    //stuff to do if condition true;
} else {
    //stuff to do if condition false;
}
```
The standard form may now be obvious

```java
if (year%4 == 0) {
    feb_days = 29;
} else {
    feb_days = 28;
}
```

Else must follow if because it does the test

open brace, immediately after “else”

The result is sets the number of days in February based on leap year

finally, close brace
Let’s go to processing for an example

```java
void draw(){
    ellipse(mouseX,mouseY,3,3);
    if(mouseX<10 && mouseY<10) {
        background(255);
    }
    if(mousePressed) {
        fill(0,0,255);
    } else {
        fill(255,0,0);
    }
}
```
Writing Programs

- Naturally, programs are given sequentially, the declarations at the top
- Braces {} are statement groupers ... they make a sequence of statements into one thing, like the “true clause of an If-statement”
- All statements must end with a semicolon EXCEPT the grouping braces ... they don’t end with a semicolon (OK, it’s a rare inconsistency about computer languages!)
- Generally white space doesn’t matter; be neat!