COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

Active processing

Active Processing

So far – *static processing*

one run through, shows the result at the end

Active processing

the program starts, and keeps running!!

things can animate

respond to keyboard, mouse!

New command: random

Generate a number [0..high)

from 0 to high but excluding high.

random(high);

e.g.,

random(5); // generate a random number 0..4

random *returns* a number that you can use anywhere you can use a literal

```
println(random(10));
```

random 10 gives you a number

That number is placed in the println

```
println(random(10));
```

→`println(3);`

Boring static program:

Line from center to random spot

```
line(250,250,random(500),random(500));
```

Left to right, two times it calls random:

```
line(250,250,random(500),random(500));
line(250,250,124,random(500));
line(250,250,124,492);
```

Moving to active..

Active programs have two regions, or blocks.

- setup – run once
- draw – run over and over!!

60 times a second!

Processing has special syntax to specify blocks

Processing *block* syntax

```
{
    // this is a code block
}
```

How to setup your active blocks

You can give blocks names – these are called functions (later). Kind of like making your own commands

Some commands take parameters and some give you data. Some do both.

line(1,2,3,4);

random(500)

We setup our regions in a similar way

How to setup your active blocks

resultType blockName(parameters)

{

}

for now, don’t worry too much about the result or parameters…

No result? Put “void”

Active blocks – setup and draw

```
void setup()
{
   // run once
}

void draw()
{
   // run 60 times every second
}
```

NO COMMANDS OUTSIDE THESE BLOCKS

(only create variables.. Later)

First active program:

```
void setup()
{
  size(500,500);
  line(250,250,random(500),random(500));
}
void draw()
{
}
```

Make it active – move to the draw

```
void setup()
{
   size(500,500);
}
void draw()
{
 line(250,250,random(500),random(500));
}
```

Variables and blocks **RULE #1**

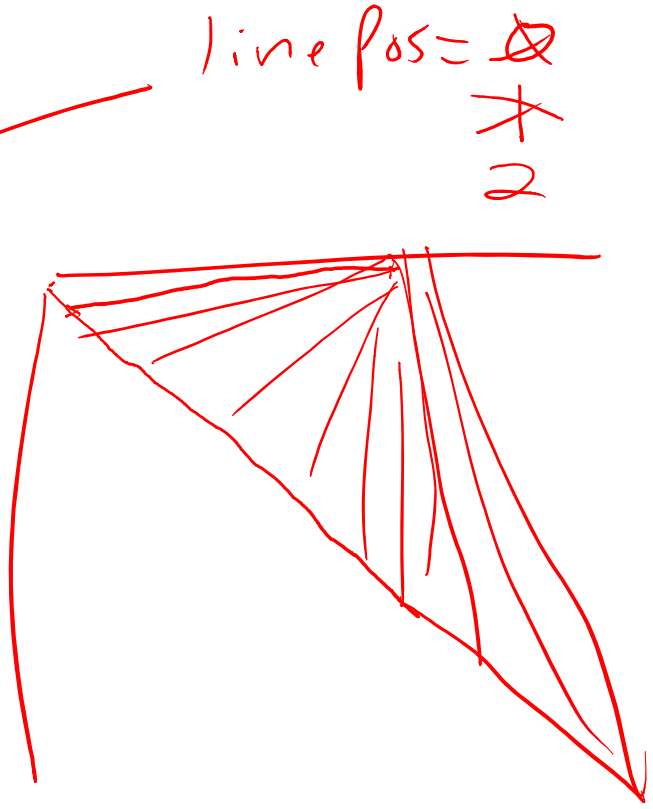
If you define a variable inside a block, it only exists until the block is finished. You can do it, but once the block is over, the variable gets destroyed.

The next time into the block, the variable gets re-created and re-initialized

Called a **local variable**

What do you expect to happen in this program

```
void setup()
{
  size(500,500);
}
void draw()
{
  int linePosition = 0;
  line(250,0,linePosition,linePosition);
  linePosition = linePosition + 1;
}
```



Variables and blocks **RULE #2**

A variable created inside one block is only visible within that block. Other blocks cannot see them.

This is called a scope rule – it defines the area where a variable is visible

Maybe move the declaration to the startup?

```
void setup()
{
  size(500,500);
  int linePosition = 0;
}
void draw()
{
  line(250,0,linePosition,linePosition);
  linePosition = linePosition + 1;
}
```

Solution: global variables

Put variables at the top of the program, outside the blocks

- visible in all blocks
- only created once and not destroyed on every redraw!

What happens?

1 → globals get created

2 → setup BlockRuns
1 time.

3 → Draw Block Repeats

result

```
int linePosition = 0;
void setup()
{
  size(500,500);
}
void draw()
{
  line(250,0,linePosition,linePosition);
  linePosition = linePosition + 1;
}
```

MOUSE!!! YAY!!

Processing globals

// current mouse position at beginning of DRAW

mouseX

mouseY

// previous mouse position, at last draw

pmouseX

pmouseY

Example 1: line to mouse position

Example 1: line from last position to current

Example 3: don’t erase background

New commands:

New commands!
max(a,b) – gives the bigger of the two
min(a,b) – gives the smaller of the two

e.g.,

int bigger = max(100,200);
→ int bigger = 200;

int smaller = min(max(3, 5), 4);
→ int smaller = min(5, 4);
→ int smaller = 4;

Hold the roof up!

Make a line that is falling down, like a roof, and hold it up with the mouse.

First, make a falling line

- Variable for current line position
- Draw across the screen at that y
- Increase y each time it draws