Drawing with Code

Dr Kaz Grace

kazjon@arch.usyd.edu.au
Why draw?

• It’s easy to see what the code is doing.
• You can make interesting things!
Processing

http://processing.org

```java
size(600,600); //Size the window.
background(0); //Set the background to black.
smooth(); //Make curved lines draw nicely.
noStroke(); //Turn off borders around things.

for(int i = 255; i > 0; i-=5) 
  if(random(1) < 0.75)
    fill(i);
  else
    fill(255-i);
  ellipse(width/2,height/2,i*(1.75-random(0.5)),i*2);
```

Kaz Grace | kazjon@arch.usyd.edu.au
Monday, 24 September 2012
Sketching with the basics
Emphasise the Exploration

• Assess by 50/50 art and code.

• Set constrained but open-ended tasks:
  • Beginners: “Draw a pleasing arrangement of five different shapes.”
  • Intermediate: “Make an interactive sketch that lets you draw unexpected shapes with the mouse.”
  • Advanced: “Visually represent tomorrow’s weather forecast in a novel way.”
LEARNING PROCESSING

A Beginner’s Guide to Programming Images, Animation, and Interaction

Daniel Shiffman

http://learningprocessing.com

Kaz Grace | kazjon@arch.usyd.edu.au

Monday, 24 September 2012
Context Free

• Drawing shapes with rules.
• [http://www.contextfreeart.org/](http://www.contextfreeart.org/)
Drawing by the rules

• Formally, a “rule” converts a non-terminal symbol into a set of other symbols.

• A rule can call other rules and transform how those rules are applied.

• Rules don’t know anything about how, when or where they were called, just the current transformation.
Getting Started

• **Rules**: tell ContextFree what to draw.

• **Adjustments**: change how rules are drawn.

• The three base rules: SQUARE, TRIANGLE, CIRCLE.

```
startshape SHAPES

rule SHAPES {
    SQUARE{}
    TRIANGLE{x 1.5}
    CIRCLE{x 3}
}
```
Rules can call other rules

```
startshape SHAPEGROUPS

rule SHAPEGROUPS {
    SHAPES{}
    SHAPES{y -2}
    SHAPES{y -4}
}

rule SHAPES {
    SQUARE{}
    TRIANGLE{x 1.5}
    CIRCLE{x 3}
}
```

“SHAPEGROUPS” is a rule that calls the rule “SHAPES” three times, but SHAPEGROUPS itself does not do any drawing.

What would happen if the “y -2” and “y -4” adjustments were taken out?
Stacking up adjustments

• Adjustments can be applied any time a rule is drawn.
• Adjustments always stack. That means each rule is affected by all the adjustments applied to the rule that called it. Adjustments even affect other adjustments.

startshape ADJUSTEDSHAPES

rule ADJUSTEDSHAPES {
  SHAPES {rotate 10}
  SHAPES {y -2 x -3}
  SHAPES {y -4 size 2}
}

Notice how the size adjustment scales not just the shapes, but the space between them!
Looping with adjustments

• Use “x * {adjustments} RULE{}” to call a rule multiple times applying the adjustment incrementally.

• Note the difference between these two loops:

```plaintext
startshape SQUARES

rule SQUARES {
    10 * {x 1.5} SQUARE{rotate 5}
    10 * {x 1.5 rotate 5} SQUARE{y 2}
}
```
List of adjustments

• “x <num>” moves on the x axis by num.
• “y <num>” moves on the y axis by num.
• “size <num>” (or “s”) scales in both axes by num.
• “size <num1> <num2>” (or “s”) scales in the x axis by num1 and the y axis by num2.
• “rotate <num>” (or “r”) rotates by num degrees.
• “flip <num>” (or “f”) reflects about a line at num degrees.
List of adjustments (cont)

• “hue <num>” (or “h”) changes the hue by num degrees. Hue is colour, and ranges from 0 to 360. It starts at 0.

• “saturation <num>” (or “sat”) changes the saturation by num. Saturation is the intensity of the colour, and ranges from 0 to 1, but starts at 0.

• “brightness <num>” (or “b”) changes the brightness by num. Brightness is the lightness of the colour, and ranges from 0 to 1, but starts at 0.

• “alpha <num>” (or “a”) changes the opacity by num. Alpha ranges from 0 to 1, but starts at 1.

• saturation, brightness and alpha increase the current value by num% if you give them a positive number, or decrease it by num% if you give them a negative one.
Time to draw cool things!

• You now know all the different commands in ContextFree. You can now draw millions of shapes!
• I want you to draw me 100 different triangles. Every triangle must be unique in some way, and the more different they all are, the better!
• Start with something like this and add adjustments:

```plaintext
rule TRIANGLEGRID {
    10 * {y 1.5} TRIANGLEROW{}
}

rule TRIANGLEROW {
    10 * {x 1.5} TRIANGLE{}
}
```
end of part 1
Rules may call themselves

• What happens if you call a rule inside itself?

```latex
startshape FADING_CIRCLE

rule FADING_CIRCLE {
    CIRCLE{brightness 0}
    FADING_CIRCLE{brightness 0.25 size 0.75}
}

• ContextFree will keep going unless the shapes it draws are smaller than 1 pixel across.

• When you do this, you really should include a size command, or it won’t ever stop!
```
This is Recursion

• Recursion is when part of a program calls itself over and over until there’s some reason to stop.

• Recursion is very cool, both for drawing things and for some quite complicated programming tasks.

• Recursion is hard. It’s one of the most common failure points in first-year university programming.

• Here’s another example:

```plaintext
rule SPIRAL {
  20 * {y 0.5 s 0.975 sat 0.2 b 0.1} SQUARE{}
  SPIRAL {x 1 rotate 3 size 0.99 hue 3}
}
```
Randomly chosen rules

• If there are two copies of a rule, ContextFree picks one to run at random each time it’s called.

```plaintext
startshape RING
rule RING {12 * {x 1 r 30 hue 30} FADING{sat 0.5}}
rule FADING 2 {
    CIRCLE{b 0}
    FADING{b 0.2 s 0.9}
}
rule FADING 1 {
    SQUARE{b 0}
    FADING{b 0.2 size 0.9}
}
```

These two numbers are the chance each duplicate version of the rule will be chosen. In this case the first “FADING” is twice as likely to run as the second.
Randomness and recursion

- A rule can be used to sometimes add adjustments.

```plaintext
startshape SPIRAL
  background{b -1}
rule SPIRAL {
   10 * {s 0.75 sat 0.25 b 0.25} SQUARE{}
   SPIRAL {x 1 rotate 5 size 0.95 hue 10}
}
rule SPIRAL 0.25 {
   SPIRAL {flip 0}
}
```

This just makes the background black.

Kaz Grace | kazjon@arch.usyd.edu.au

Monday, 24 September 2012
Splitting while recursing

• What if a recursive rule variant calls itself twice?

startshape SPIRAL
background{b -1}
rule SPIRAL 0.99 {
    10 * {s 0.75 sat 0.25 b 0.25} SQUARE{}
    SPIRAL {x 1 rotate 3 size 0.99 hue 3}
}
rule SPIRAL 0.01 {
    SPIRAL{}
    SPIRAL {flip 0}
}

Note how rarely this rule will get called. The splitting rule is called just 1% of the time!
Time to draw cool things!

• Recursion and randomness allow you to make a single grammar that draws many different shapes.
• I want you to draw me a plant. Any kind of plant is OK - flower, moss, bush, tree, etc. But not just an ordinary plant, an ALIEN plant. One like nothing on Earth.
• As crazy as possible, if you please!
• Every time you render it should draw a different plant of the same species!

• Hint: Pretty much anything can be a plant if you come up with a cool enough description of it.
end of part 2