Controlling Chaos :

Harnessing the Power of Random Number Generators as Creative Assistants

Don Bosley
don@bosleymusic.com / www.bosleymusic.com

Today’s Axiom

Nothing is random, simply underlying complexities that are not understood...

...except maybe in Quantum mechanics.

WHAT WE WILL COVER:

- I. Introduction
- II. RNG Basics
- III. Basic Control Applications
- IV. Some More Complex Decisions
- V. Compounding...

I. Introduction

- Benefits of using RNG
- Variables
- Power vs. Control
- Deterministic VS Nondeterministic
- All examples available at :

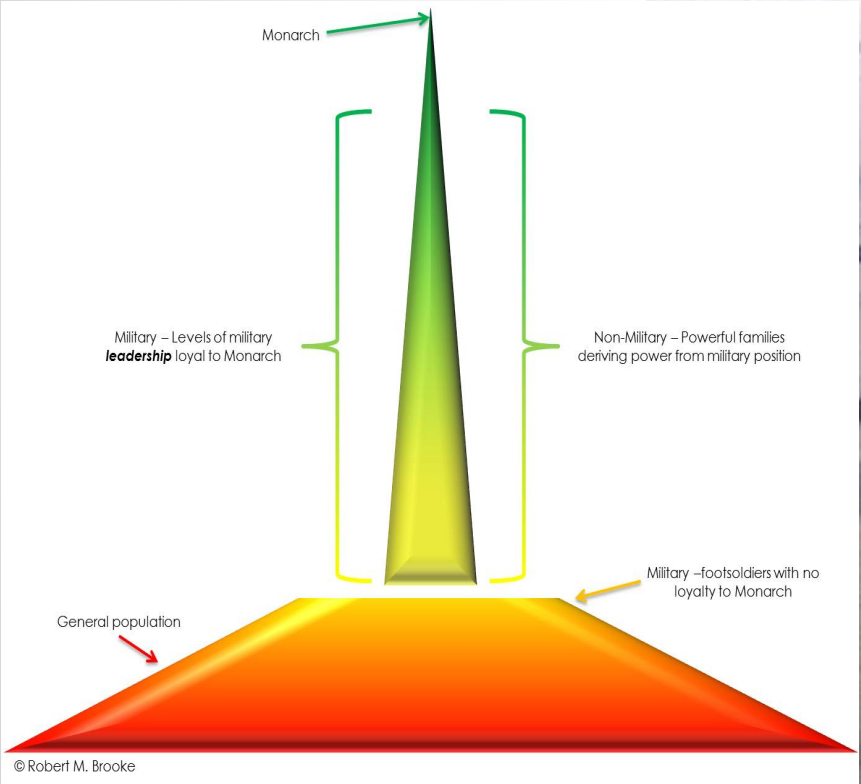
github.com/DonaldSBosley/20150605_Controlling_Chaos

John Cage

Benefits…

- Increase the speed of creative output
- Generate original possibilities
- Add extra control & complexity over details

Power vs. Control



Variables

What is a variable?

- A variable is something that can change

What does a variable have?

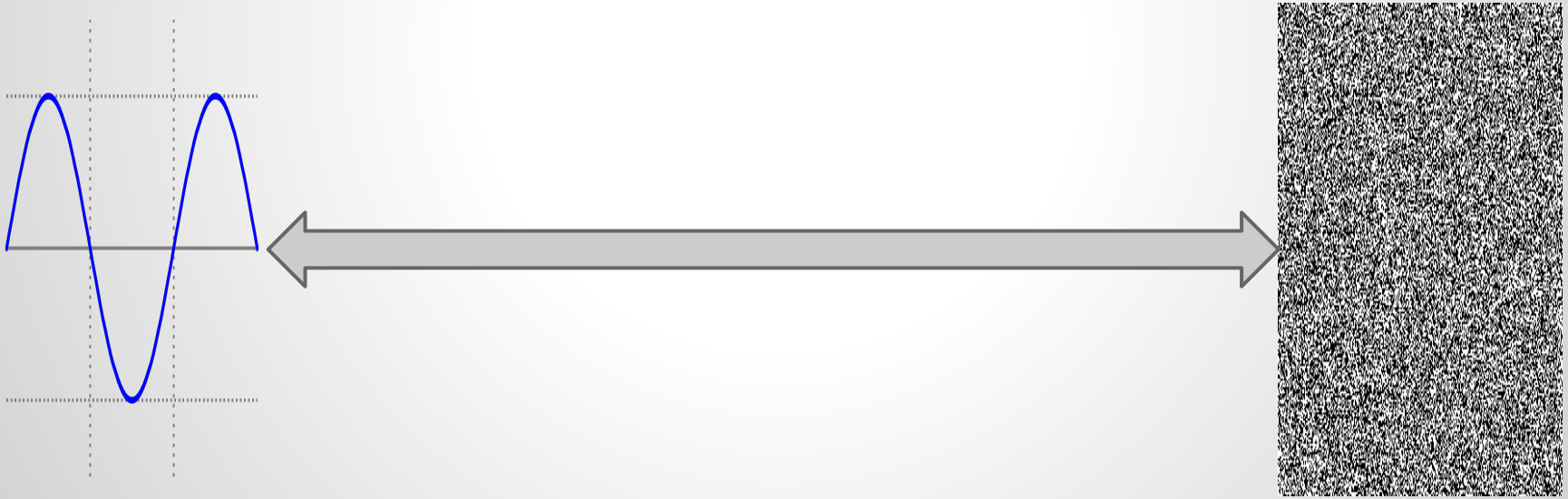
- A variable has a value

How is a variable changed?

- A variable is changed using a control

Deterministic VS Nondeterministic

Let’s use sound as an example...



II. RNG Basics

- Order in Chaos (Nothing is Random…)
- Seeding
- A Note on Distributions & Algorithms
- Getting Application Appropriate Values
- Fixed Sets/Outcomes

Let’s play a game...

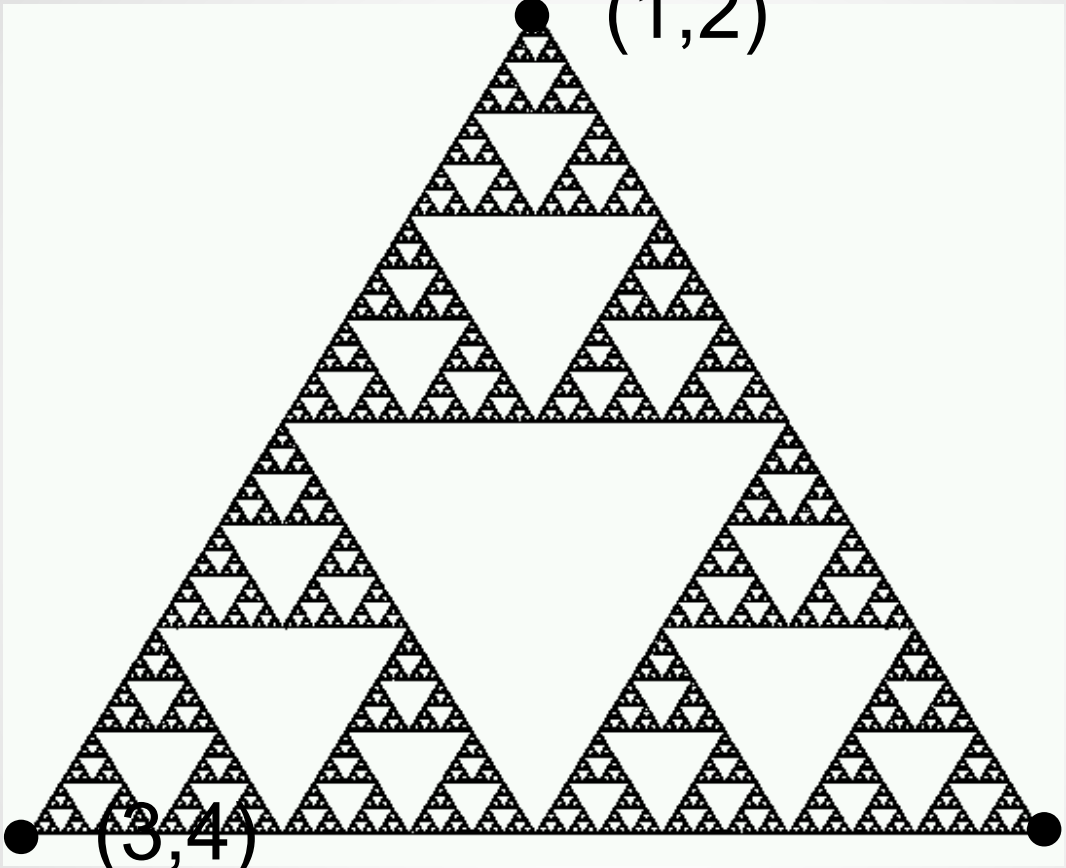
Mark these dots on paper, note the number values.

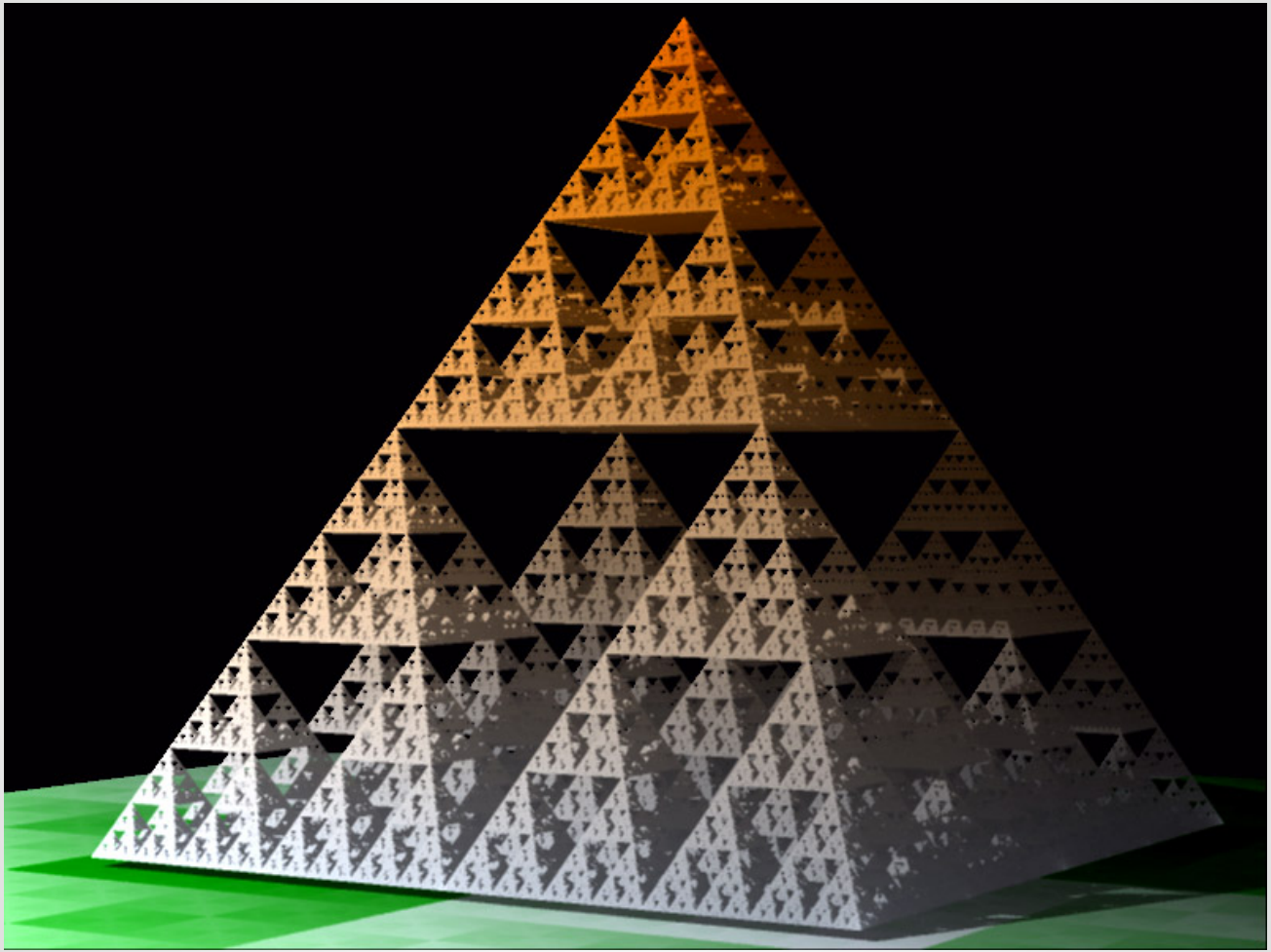
- (1,2)
- (3,4)
- (5,6)

1. **Close your eyes and pick a starting point**
2. **Roll the die in front of you,**
3. **Draw a point ½ the distance between your last point and the number you rolled**
4. **Repeat steps 2+3…(until I tell you stop)**

- (1,2)
- (3,4)
- (5,6)

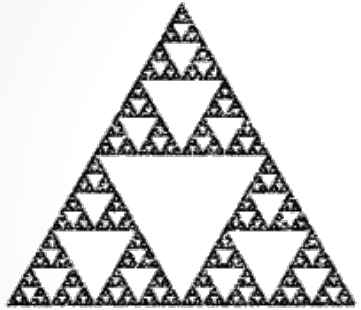
Just wait for it...


(1,2)
(3,4)
(5,6)



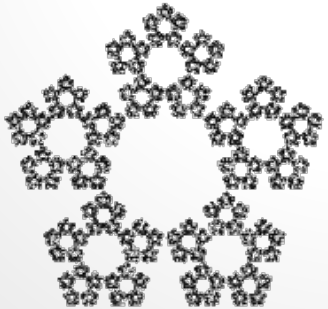
Note : Rolf asked me about different shapes:

$n = 3, r = 1/2$

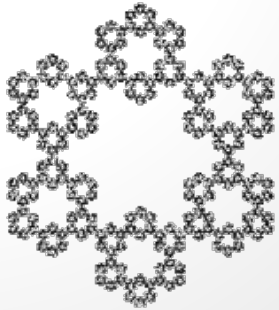


$n = 5, r = 1/3$

$n = 5, r = 3/8$



$n = 6, r = 1/3$



Seeding

- Your initial dot was the seed...
- All random number generators will begin with the same number unless seeded
- In general computing : Use Time and Date, combinations of serial numbers, etc...
- In electronics : sampled line noise
- Ties into initial conditions or state
- If it the algorithm is the same, and it always starts the same, the trajectory will always be the same...

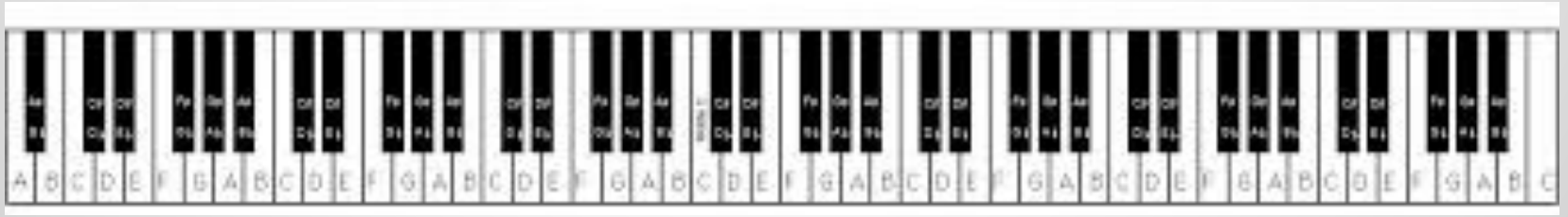
A Note on Distributions & Algorithms...

- Not all algorithms are created equally.
- Many do provide uniform or equal distributions, but not necessarily.
- Be aware of what your machine/application requires, and what you have available

Getting Application Appropriate Values

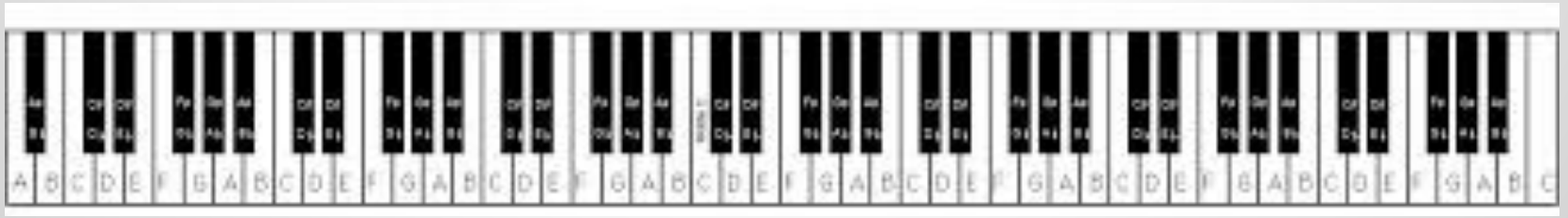
- Example:

  Selecting Random MIDI Note Values
- RNG Produces 0.0-1.0f
- We need 0-127 (integers)...



Getting Application Appropriate Values

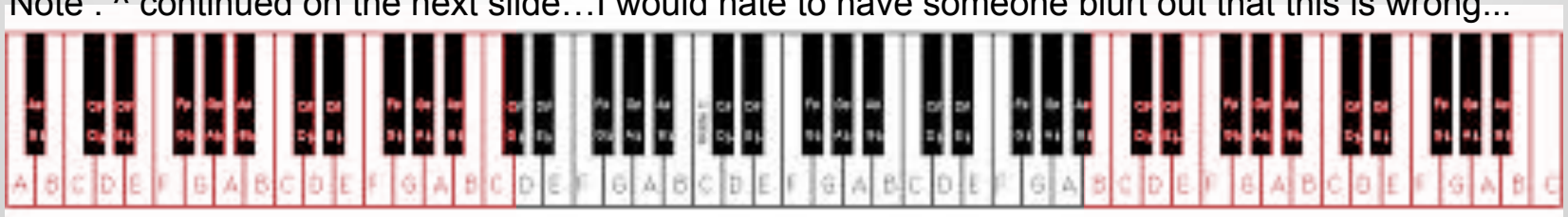
- Scaling, which is multiplication
- Get 0:127, from 0:1f,
- → ***round(random(0:1) * 127)***



What if we only want a specific range?

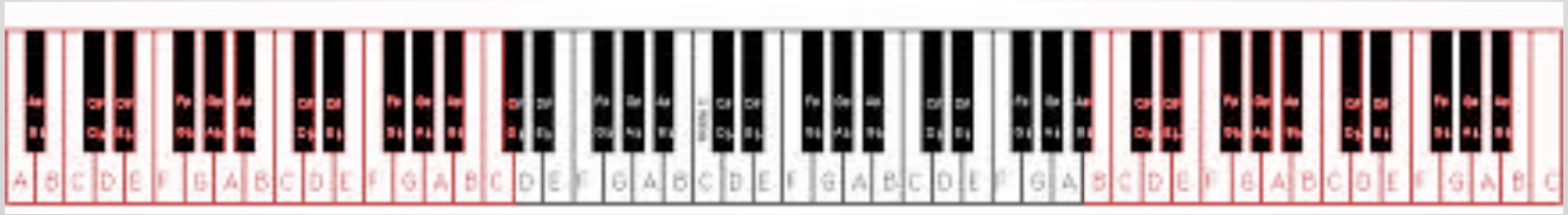
- Let’s say we’re controlling a mechanical instrument from servos...
- It can only produce notes 64:96 (inclusive)
- Our range is 33 notes wide…
- Scale → ***round(random(0:1) * 32)***

Note : ^ continued on the next slide…I would hate to have someone blurt out that this is wrong...



What if we only want a specific range?

- Scale → ***round(random(0:1) * 32)***
- Still produces 0 through 32
- Add **offset** to our equation
- *round(random(0:1) * 32)* ***+ 64***
- Now produces 64:96



Fixed Sets / Outcomes

- Index an array of known values…
- Let’s say you only want outputs :
- **vals = [0,5,210,473,600,9500]**
- Randomly generate the index :

  index = round(random(0:1) * 4)
- vals[index] → Gives desired output

Note : ^ This assumes logical indexing starting at 0.

III. Basic Control Applications

- Initial Conditions
- A basic evolution...
- Decisions, decisions to make

A simple repeating process...

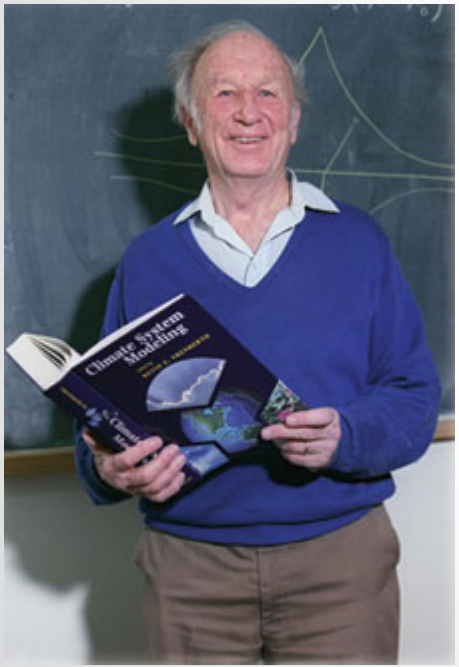
- Sketch : A_Rotation
- Circular rotation of a single color, at a fixed speed....
- What are the variables present?

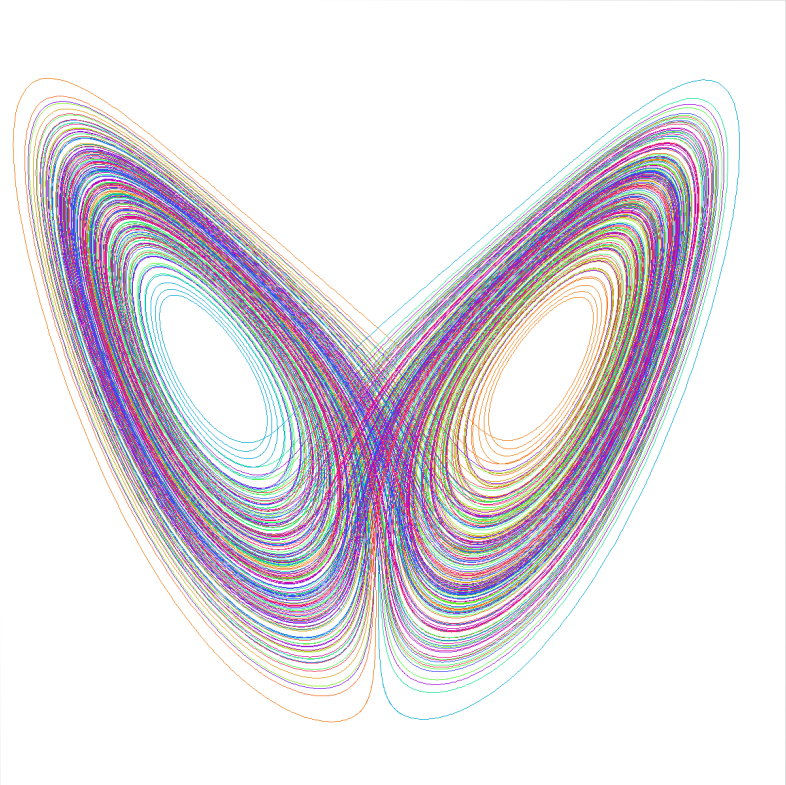
Changing the Initial Conditions

- Sketch : B_Random_Initial_Values
- Each time we start the sketch, there will be slight variations…

Note : The Initialization was happening via the Reset button on the arduino, which clears RAM, sets the program counter back to 0...etc...

Ed Lorenz





Randomness during operation...

- Sketch : C_Simple_Evolution
- We are now randomizing the color values and speed each iteration...

Decision Tree #1

- Sketch : D_Evolution_NumColors
- Let’s eliminate the “whiteness” and try to get some pure colors…
- Randomize the number of colors
- Randomly select which from each group
- Weight the All 3 color combinations...

Note : Big thank you to E.R. for the HSB suggestion for color changes.

Decision Tree #2

- Sketch : E_Evolution_NumColorsV2
- Let’s change the temporal elements…
- When the color changes
- When the speed changes

It’s Alive...

- Sketch : F_Let_It_Breathe
- Add a scaling factor to the color intensity

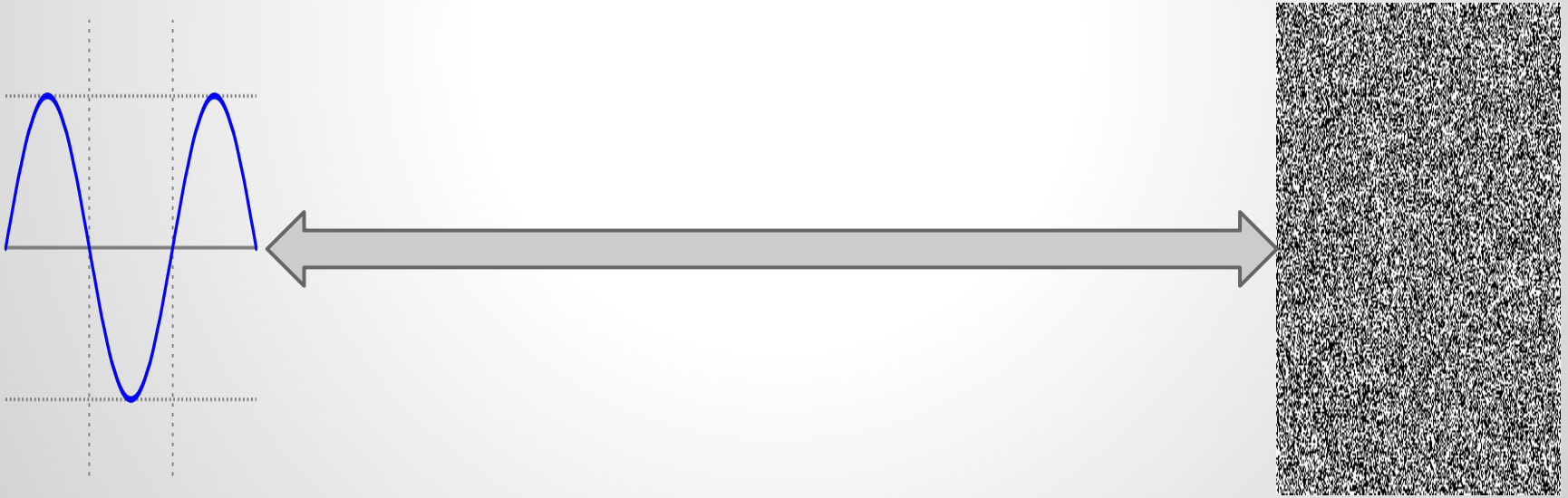
The Direction We're Going...

- I'm getting pretty bored with the direction being consistent and predictable
- Sketch : G_Changing_Direction
- Add a decision to change direction...

And now for something completely different....

Remember this little graphic?

NOTE : This is the ChromaRing test in the github repository. Keep in mind the simplicity of the code, but the utter lack of control that results.
PWR V. CTRL



IV. Some More Complex Decisions

- Adding Rules & Recursion
- Appearance of Randomness

If there is time...

- More Weighting Decisions
- Markov Chains

NOTE : 2 more examples were dumped into Git to cover these.

- Choice_On_Weighted_Probability
- Apparent_Random_With_Sequences

Rules and Recursion...

Conway's Game of Life :

John Conway (1970)

1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

http://pmav.eu/stuff/javascript-game-of-life-v3.1.1/

The appearance of randomness : What is the pattern?

15, 15, 14, 9, 12, 14, 13, 4, 2, 3, 6,10...

NOTE : I added this code to the repository. You can run this on arduino and view the sequence using serial print - just note the values are 8 and not 4 bit, so the sequence will not match.

Nothing is random...

1111 = 15

(1111 << 0) = (1111 | 1 << 2) = 1111 = 15

(1111 << 1) = (1110 | 1 << 1) = 1110 = 14

(1110 << 2) = (1000 | 1 << 0) = 1001 = 9

(1001 << 3) = (1000 | 1 << 2) = 1100 = 12

(1100 << 0) = (1100 | 1 << 1) = 1110 = 14

(1110 << 1) = (1100 | 1 << 0) = 1101 = 13

(1101 << 2) = (0100 | 1 << 2) = 0100 = 4

(0100 << 3) = (0000 | 1 << 1) = 0010 = 2

(0010 << 0) = (0010 | 1 << 0) = 0011 = 3

(0011 << 1) = (0110 | 1 << 2) = 0110 = 6

(0110 << 2) = (1000 | 1 << 1) = 1010 = 10...

...simply underlying complexities that are not understood.

Just a cycle of bit moves and masks with binary numbers.

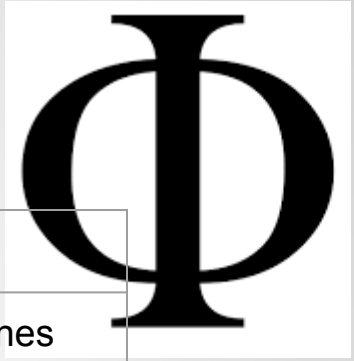
V. Compounding

- Examples from Audio/Music Work :
- System Studies
- Puzzle Variations

System Studies

- All decisions made by computer
- Weighting based on length of time, proportions between decisions
- Tracks decision times
- Stores localized “ideas” or thematic material

System Studies



Initial Decisions	Structural	Timbres / Notes
• Max number of instruments (polyphony) • Recursively selects each instruments' range • Starting pitch class set • Tempo • Beat Emphasis	• Running tab on How Often Each Instrument Changes Timbre • Permutations of Current Idea • Cross-transfer of thematic information from instrument to instrument • Add or subtract notes from the pitch class set • Modulate the pitch class set	• Number of Overtones • Amplitude of Overtones • - Made the mistake of ignoring the phase of overtones... • Individual ADSR for each overtone • AM / FM : Depth, Speed • Global Dynamic • Local dynamic variation on every note

Puzzle Variations (2012)

Questions?