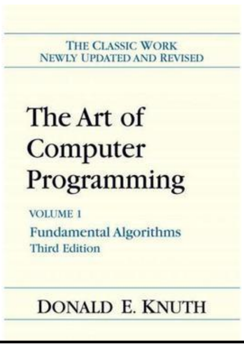
Procedural Content Generation for Computer Games

Lecture 3 – Procedural Art

Vojtěch Černý
cerny@gamedev.cuni.cz

Disclaimer: I am not an artist

(unless you count this)



THE CLASSIC WORK
NEWLY UPDATED AND REVISED

The Art of Computer Programming

VOLUME 1
Fundamental Algorithms
Third Edition

DONALD E. KNUTH

Can computers make art?

- What is art?

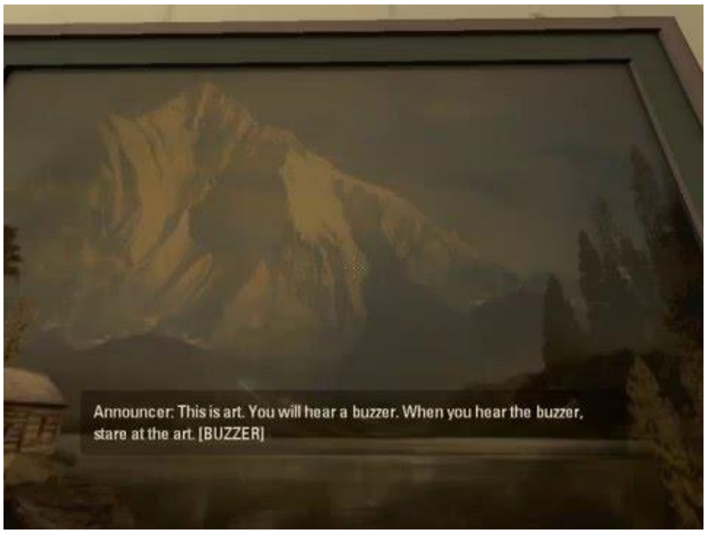

Announcer: This is art. You will hear a buzzer. When you hear the buzzer, stare at the art. [BUZZER]

Image from Portal 2.

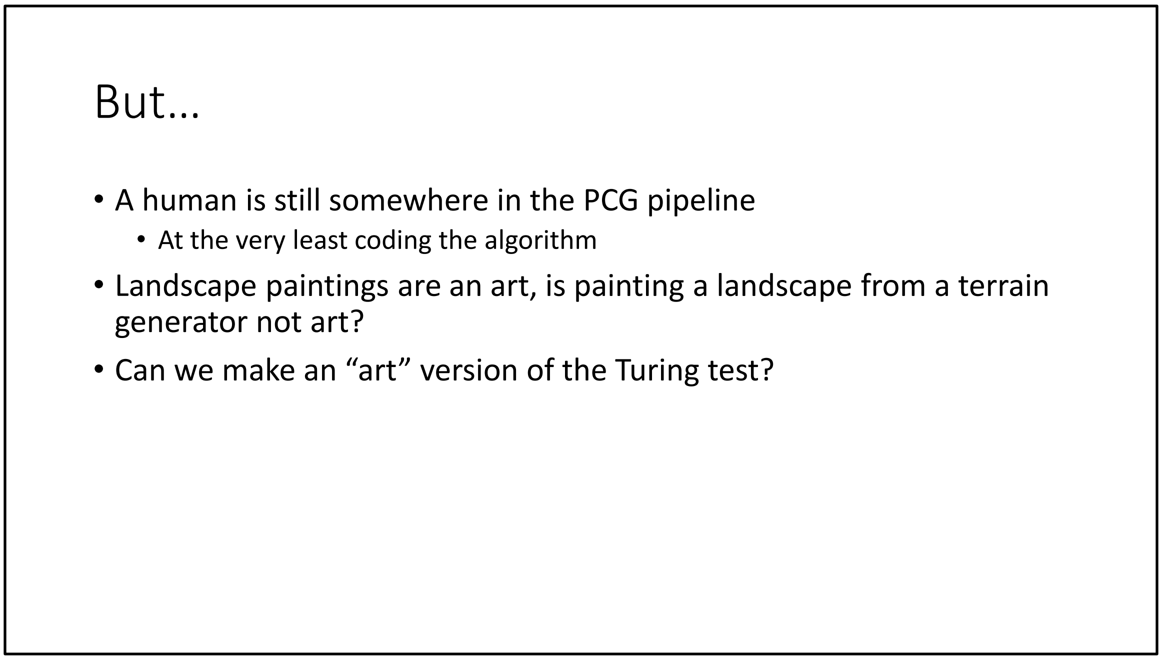
What is art?

- Merriam-Webster

the conscious use of skill and creative imagination especially in the production of aesthetic objects

So, computers are out of luck (unless you want to make artificial consciousness)

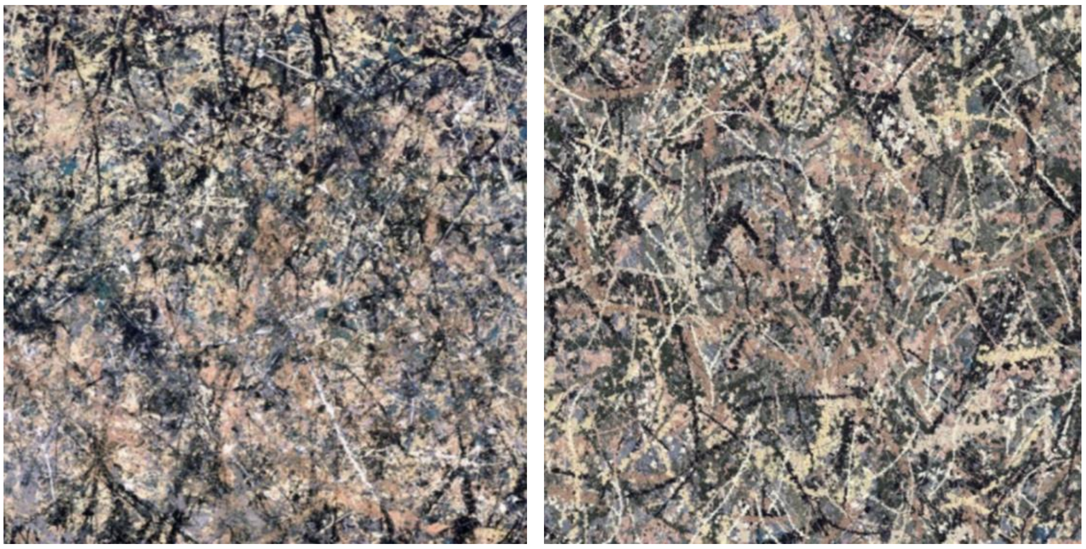
End of lecture.



But…

- A human is still somewhere in the PCG pipeline
  - At the very least coding the algorithm
- Landscape paintings are an art, is painting a landscape from a terrain generator not art?
- Can we make an "art" version of the Turing test?

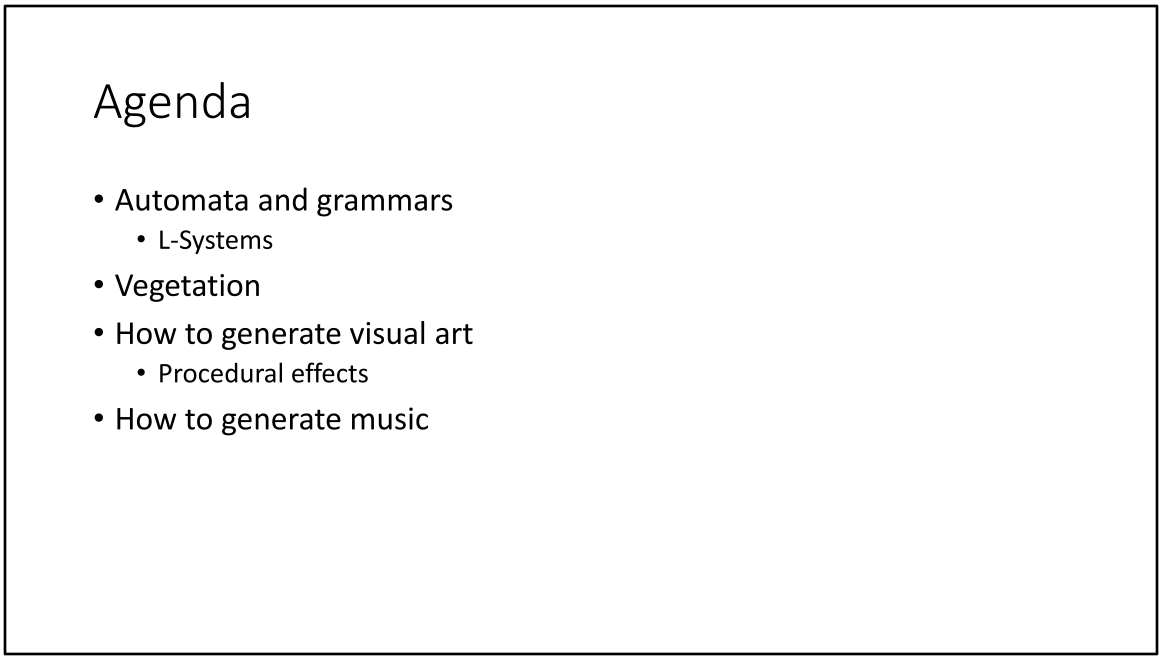
Also, more questions on wikipedia: https://en.wikipedia.org/wiki/Generative_art


One is generated, one is an actual painting by Jackson Pollock.
Where is the difference?

Layered modeling and generation of Pollock's drip style by Zheng et al. (2014)
Take a quiz – PCG or Human https://www.abc.net.au/news/2017-08-12/quiz-was-this-art-made-by-a-human-or-a-robot

So… can computers make art?

- A philosophical argument
- Let’s just ignore the nay-sayers and create something



Agenda

- Automata and grammars
  - L-Systems
- Vegetation
- How to generate visual art
  - Procedural effects
- How to generate music

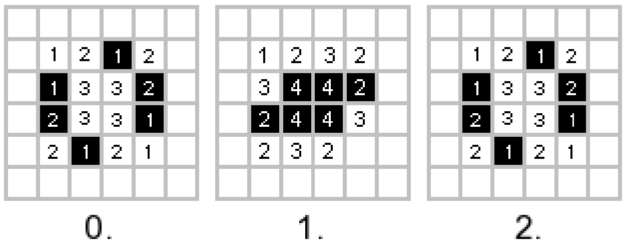
Automata and Grammars

(important algorithms)

Useful PCG techniques we will need in this and further lectures.
Automata = Cellular automata
Don’t worry, this is still a lecture on PCG

Cellular automata

- "Conway's Game of Life"-like
- Rules on how many neighbors a cell needs to:
    - survive (S)
    - be born (B)
- Original is B3/S23
- Can make "living" systems



- Also can make more coherent spaces (useful for e.g. cave generation)

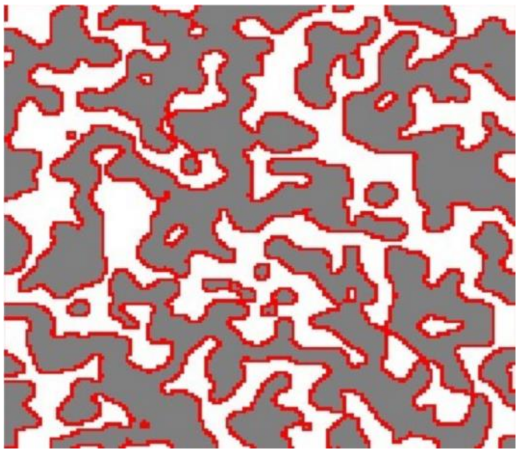
B3/S23

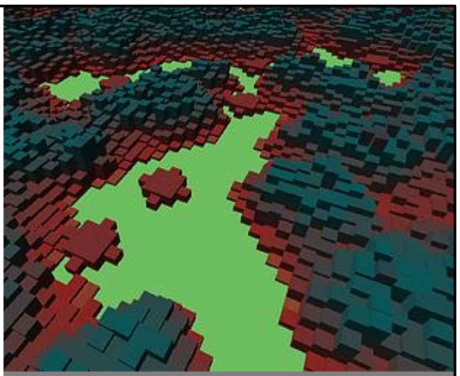
- survive (black stays black) on 2-3 neighbors (otherwise die – black turns white), born (white turns black) if there are exactly 3 neighbors

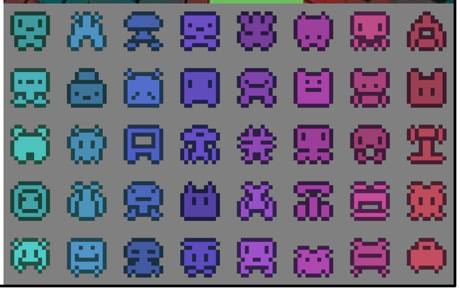
Nice selections of life-like rules: https://en.wikipedia.org/wiki/Life-like_cellular_automaton

Side-note:
This is also a rewriting system (some kind of grammar/L-system?)… it can be combined with other rewriting systems, e.g. graph automata (from graph grammars) also exist.

Cellular automata examples

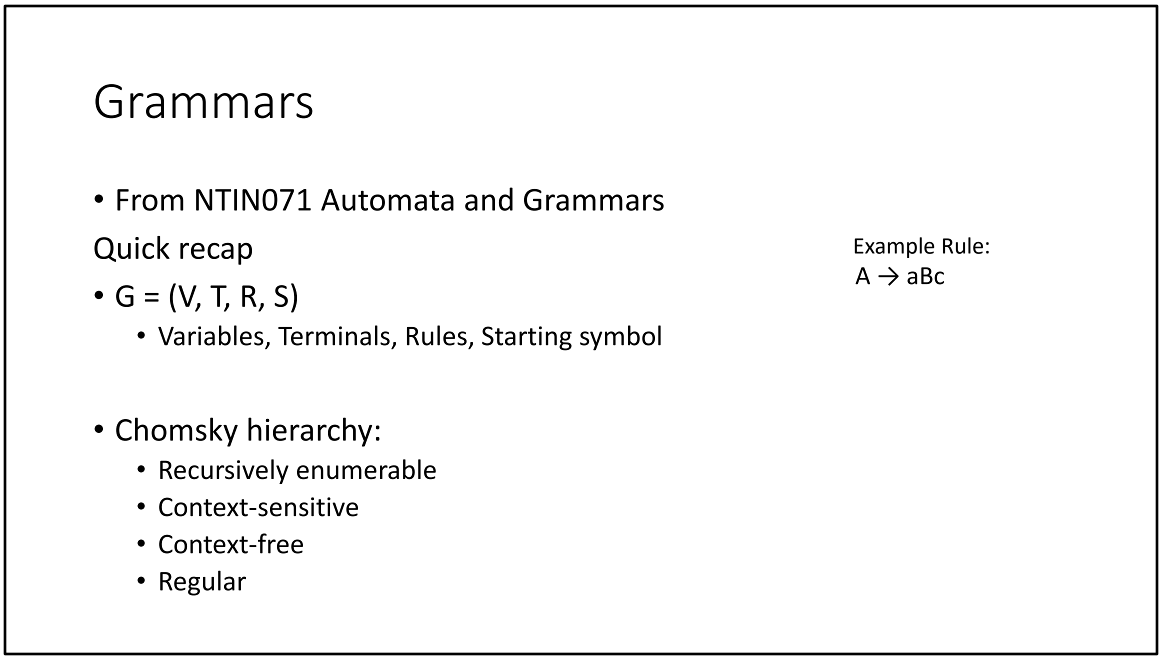






Traditional cellular automata (game-of-life) want to create long-time oscilating and interesting - not too exploding neither dying out systems.
In PCG, we usually don’t care about that as much. Often we just take (white)noise and run just a few steps to generate good results.

Caves are perhaps the most typical usage of Cellular automata in PCG.
Pixel art: https://github.com/yurkth/sprator



Grammars

- From NTIN071 Automata and Grammars

Quick recap

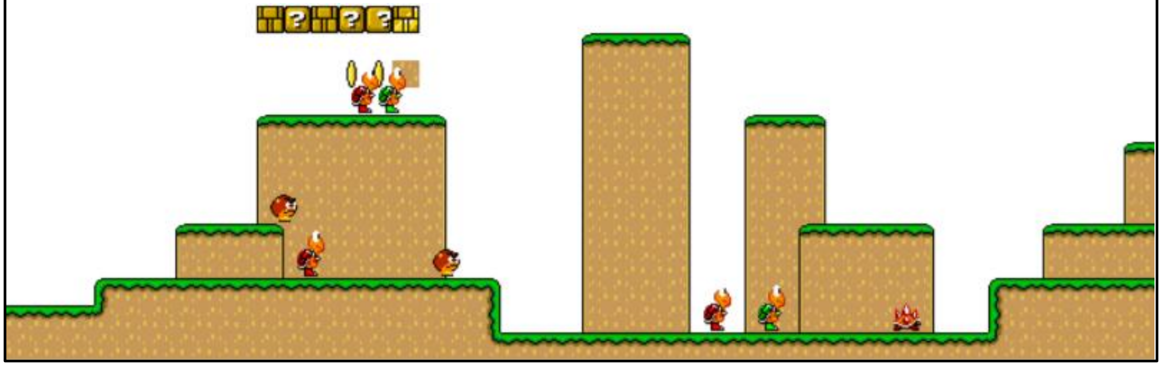
- G = (V, T, R, S)
  - Variables, Terminals, Rules, Starting symbol

Example Rule:
A → aBc

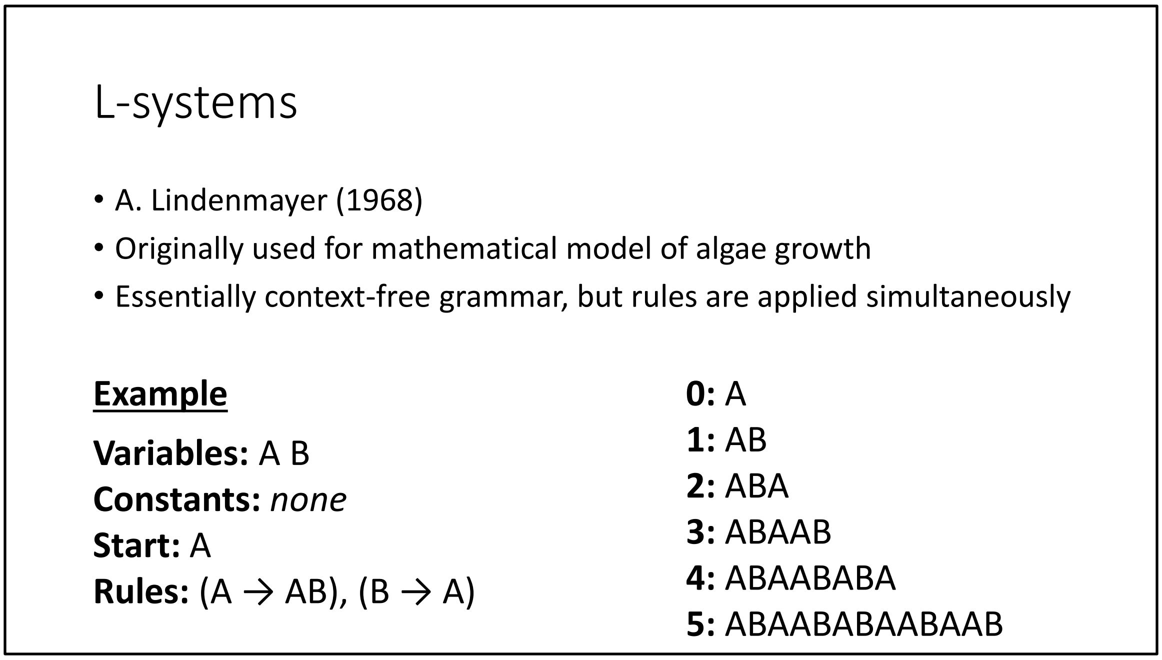
- Chomsky hierarchy:
  - Recursively enumerable
  - Context-sensitive
  - Context-free
  - Regular

Generating level with grammars

- Represent your levels as a series of terminal symbols
- Grammar can be a good representation for search methods
  - (Grammatical evolution)



Noor Shaker et al. - Evolving Levels for Super Mario Bros Using Grammatical Evolution (2012)



L-systems

- A. Lindenmayer (1968)
- Originally used for mathematical model of algae growth
- Essentially context-free grammar, but rules are applied simultaneously

<u>Example</u>

Variables: A B
Constants: *none*
Start: A
Rules: (A $\rightarrow$ AB), (B $\rightarrow$ A)

0: A
1: AB
2: ABA
3: ABAAB
4: ABAABABA
5: ABAABABAABAAB

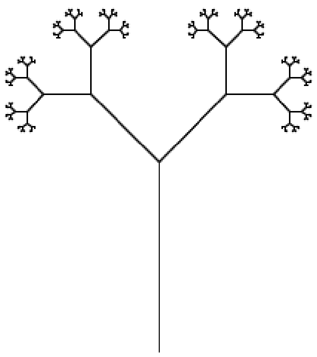
The Iterations form a Fibonacci sequence

Binary tree L-system

0: 0
1: 1[0]0
2: 11[1[0]0]1[0]0
3: 1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0

Variables: 0 1
Constants: [ ]
Start: 0
Rules: (0 → 1[0]0), (1 → 11)

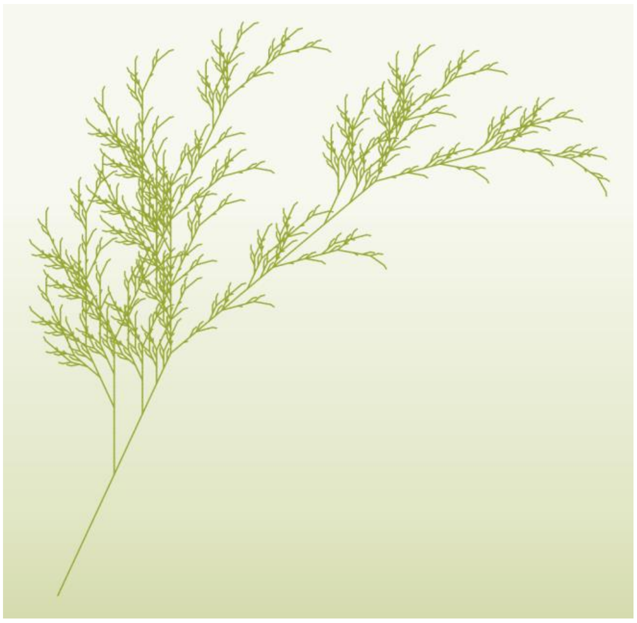
Interpretation (turtle):
0: draw final line
1: draw line
[: push position and angle, turn left 45°
]: pop position and angle, turn right 45°



Fractal plant

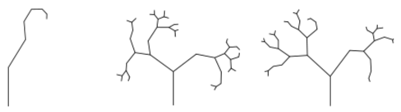
Variables: X F
Constants: + - [ ]
Start: X
Rules: X → F+[[X]-X]-F[-FX]+X]
F → FF

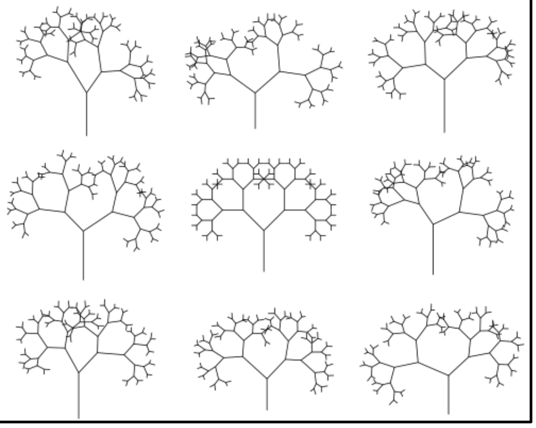
Interpretation (turtle):
F: draw line
-/+: turn left/right 25
[/]: push/pop position & angle
X: *do nothing*



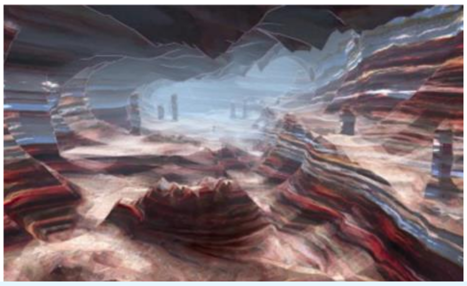
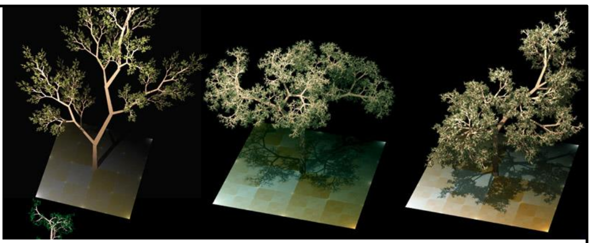
Stochastic L-Systems

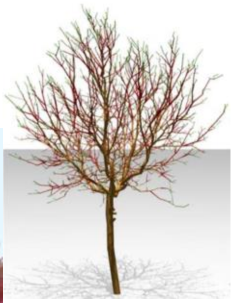
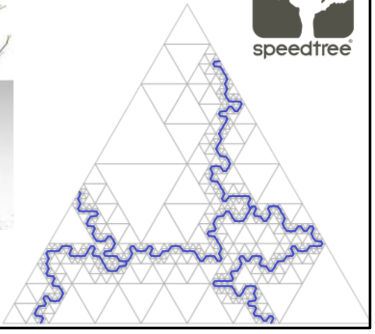
- Add some randomization to rules to break up symmetry
- For example:
  - Randomly peturb angle
  - Randomly peturb line-length
  - Randomly select one rule
  - Don’t use a rule





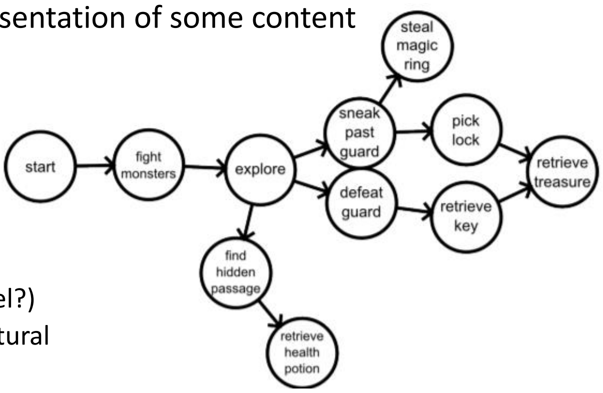
L-system examples

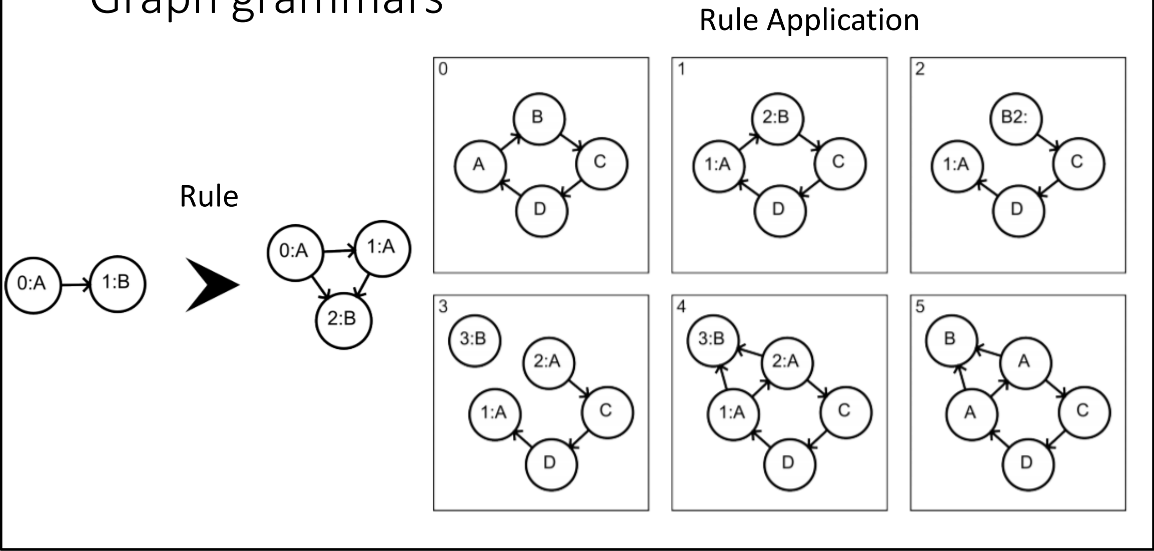
Graph grammars

- Originated in 1960s – pattern recognition and compilers
- Graphs are nice abstract representation of some content
- Problems:
  - How to turn it into content (level?)
  - Describing a grammar is less natural



As with a bunch of other stuff (Noise, Evolution, L-Systems) it originated somewhere else, but has a great use in PCG.
Picture from pcgbook.com

Graph grammars



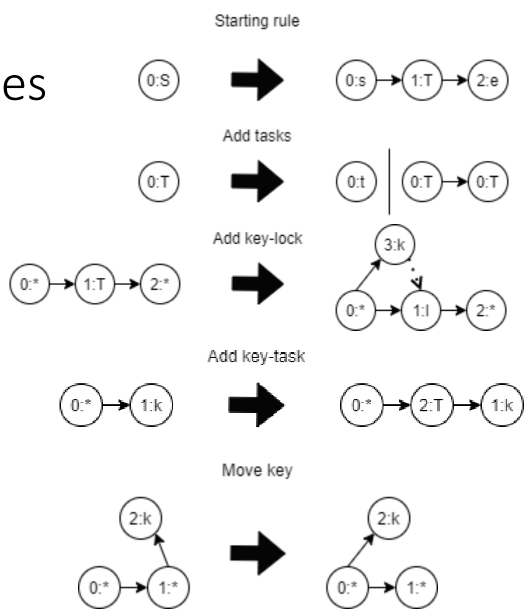
0. Graph
1. Find structure matching rule left-side
2. Remove all edges between match
3. Modify/create nodes by rule right-hand side
4. Alter edges by rule right-hand side
5. Remove temporary marks

Graph grammars -> content

- Graph usually isn't the end result
- Four generic options:
  - Graph first, then independently space to fit it in
  - Graph first, nodes and edges also prescribe how to generate space (as prescription)
  - Graph and space at the same time
  - Space first, generate graph in that space
- Each has different pros/cons
- None of them are simple

Side-note:
Lock and key puzzles

Can be easily generated

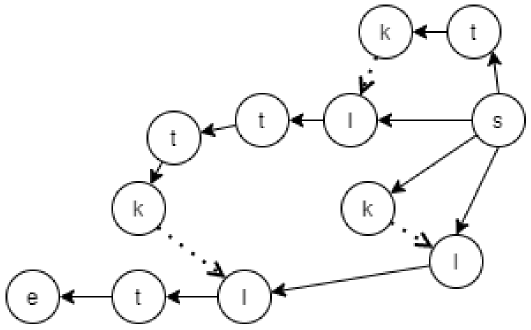


Node annotations (uppercase letters = non-terminals, lowercase = terminals):
S – Start node (of the grammar)
e – level entrance
g – goal (exit)
T – Task
t – some task (perhaps monster battle)
b – boss
k, l – key, lock (connected by crossed "unlock arrow")

Side-note: Lock and key puzzles

- De-linearizes levels nicely
- Very common approach
- Are they fun?

- Interesting compromise:
    - Unexplored (Joris Dormans)
    - Many variations
        - Potion of resist-fire + lava-barrier
        - Scroll revealing secret door
        - …



Possible result

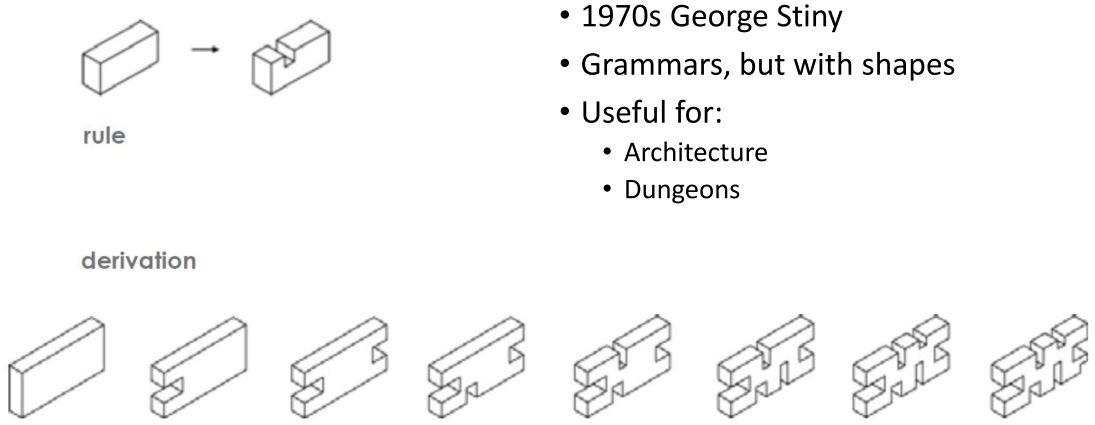
Other interesting things about Unexplored:
- It solves the issue of difficult transformation of graph into grid, by working with a grid directly (and having the rules operate on the grid)
- Works directly with cycles (the level starts as a cycle)
- The chapter (Cyclic Generation in Procedural Content Generation in Game Design) dissects lock-and-key puzzles
    - Permanent, Reversible, Temporary, Collapsing, One-Directional, Unsafe, Multi-purpose, …

Another thing that has a lot of lock-key varieties is the Zelda series.

Side-note 2: Genetic Programming

- Genetic Programming (GP) – evolutionary approach for trees
- Can be extended to allow non-tree patterns, but is somewhat clumsy

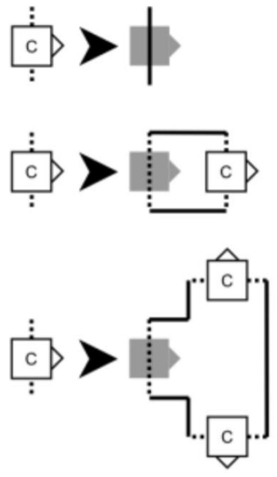
Shape grammars



- 1970s George Stiny
- Grammars, but with shapes
- Useful for:
  - Architecture
  - Dungeons

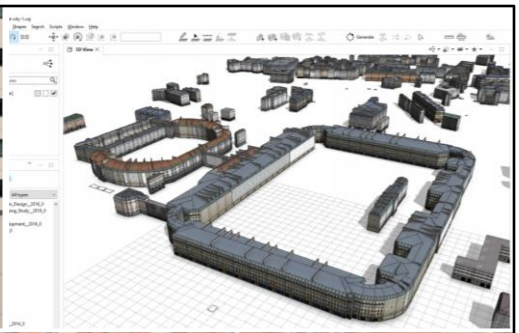
Shape grammars

- Can be used to layout a mission-graph
- Can generate things on a grid



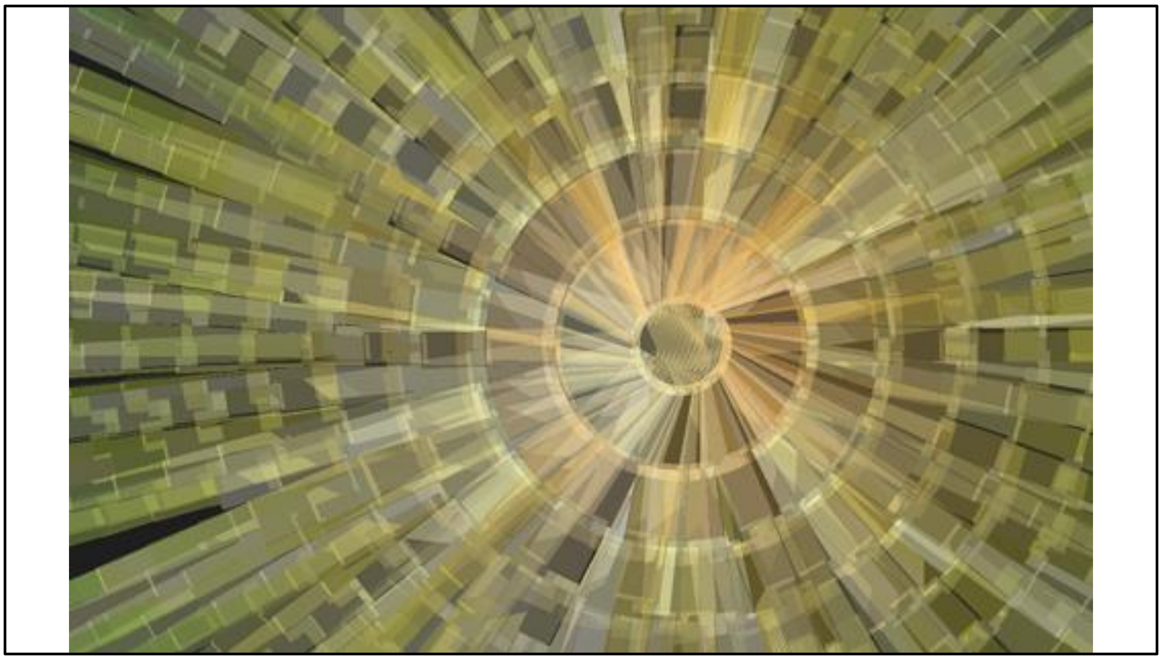
Shape grammar examples

- City Engine

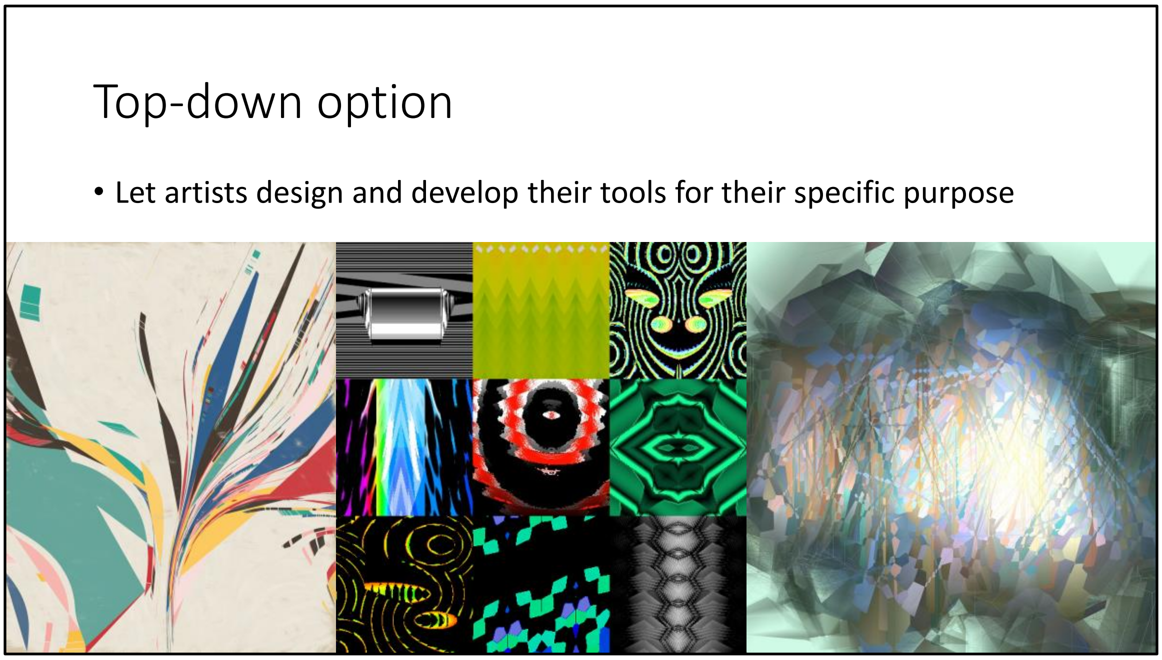


City Engine is a tool for generating cities (similiarly to how Terragen is for terrain and Speedtree for vegetation)
However, City Engine is quite complex and requires a lot of work. Uses CGA (presumably Computer Generated Architecture) grammar

Visual art



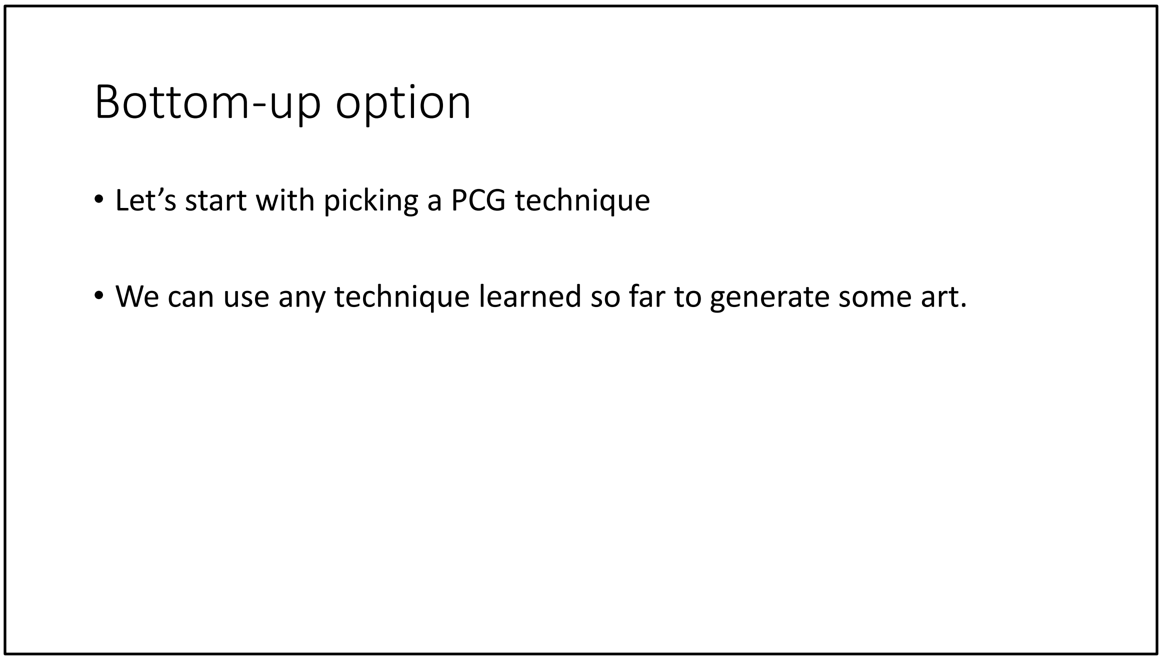
"Octopod" by Mikael Hvidtfeldt Christensen



Top-down option

- Let artists design and develop their tools for their specific purpose

https://www.flickr.com/photos/144063771@N05/
https://tylerxhobbs.com/essays
http://quil.info/examples

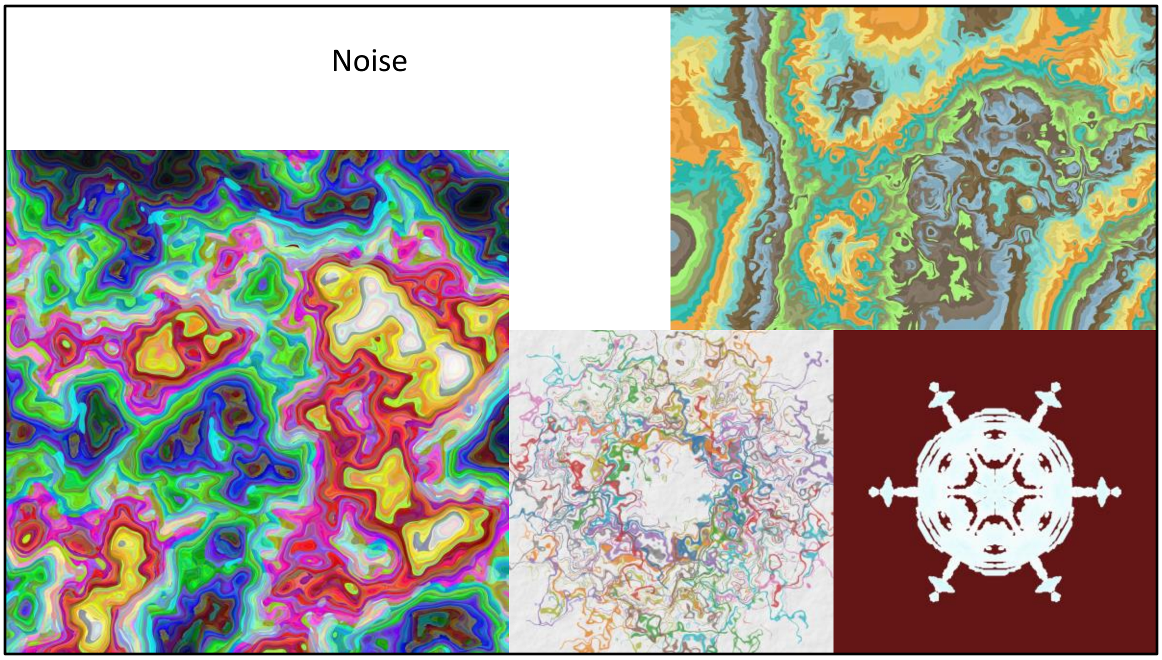


Bottom-up option

- Let’s start with picking a PCG technique
- We can use any technique learned so far to generate some art.

Also, since games are a form of art, it is useful to know these methods at least for inspiration.

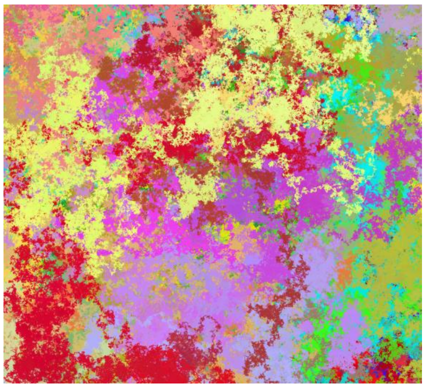
Noise

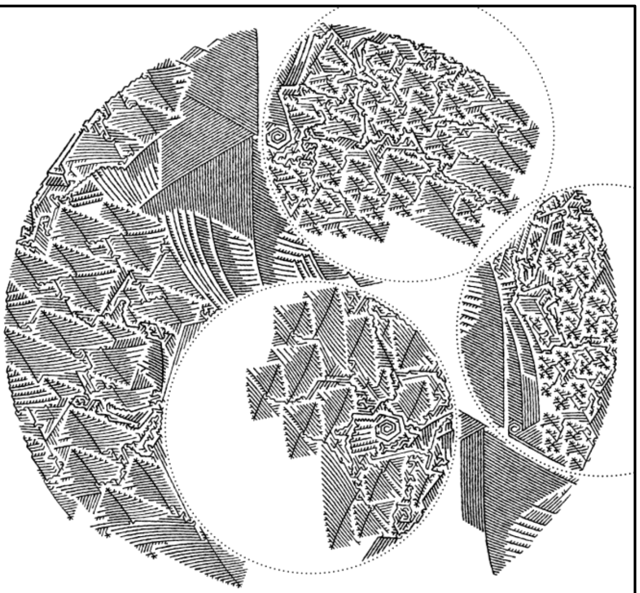


https://thingonitsown.blogspot.com/
https://allrgb.com/ (art using all 24-bit RGB colors)

The on the left is several noises (which produces floats) in HSV, converts them to RGB, and orders the pixels -

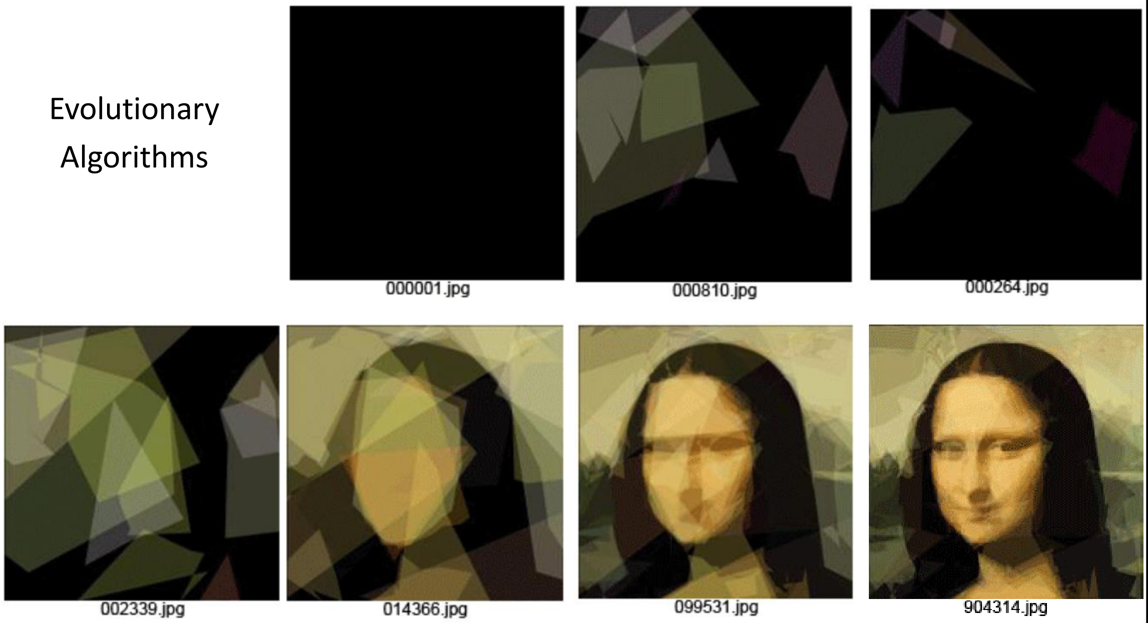
Random Walks





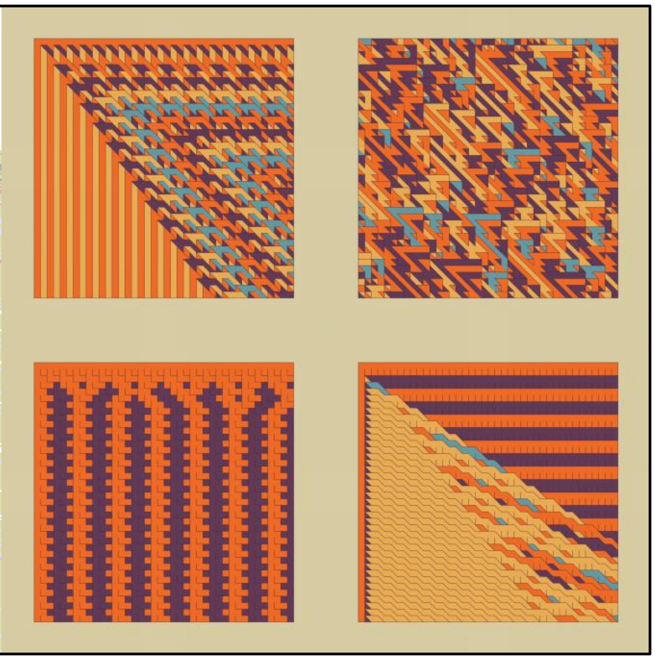
https://twitter.com/Paavo_Toivanen/status/1191795117759385601/photo/2
https://www.etsy.com/au/listing/678439855/random-walker-algorithm-art-print-poster

Evolutionary Algorithms



Mona Lisa from 50 translucent triangles, using Genetic Programming.

Cellular



https://twitter.com/kGolid/status/1069764925722103808/photo/1
https://www.dartmouth.edu/~matc/math5.pattern/FinalProject/Hoke.html
More AI, but awesome cellular automata - https://distill.pub/2020/growing-ca/

Fractals
(L-systems)

If you like fractals and games (who doesn’t?), make sure to check out Marble Marcher - https://www.youtube.com/watch?v=9U0XVdvQwAI

Summary – most common methods

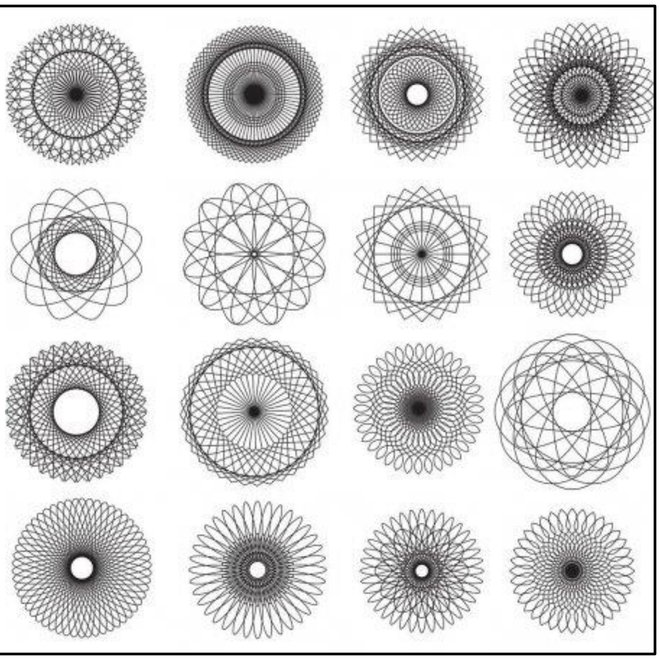
- Cellular automata
- Fractals (often by L-Systems or other grammars)
- Generative Adversarial Networks
- Evolutionary algorithms

Art Toys

- Just playing with PCG

Toy = Game – Goal

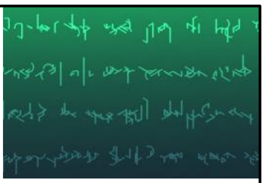
- Useful in games (e.g. flowers)
- Less limitations (doesn't need to be playable)



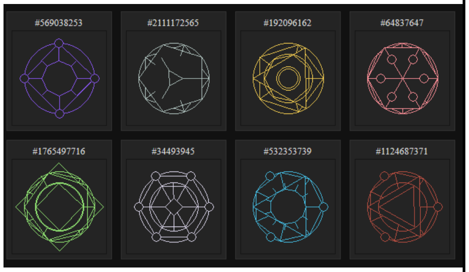
Picture of results from a spirograph.
Also, original Minecraft (alpha) as basically an art toy (no objectives, just create stuff).
It got millions of people hooked.

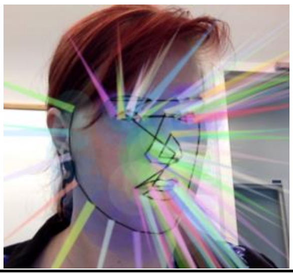
It is good to distinguish Art Toys vs. Art Tools (e.g. Sketchpad)
- Toys have less control and more power in what you create
- So you won't create stuff exactly as you want it, but it will be fast and empowering

Art Toys

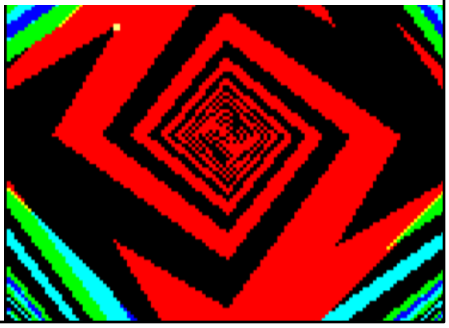


- Take any inputs (keys, video, current price of bitcoin, …)
- Pick a target representation
  - Pixels, strokes, lines, polygons, …
- Apply transformations
  - Maybe use some symmetries
- Postprocess things
  - Add details / effects





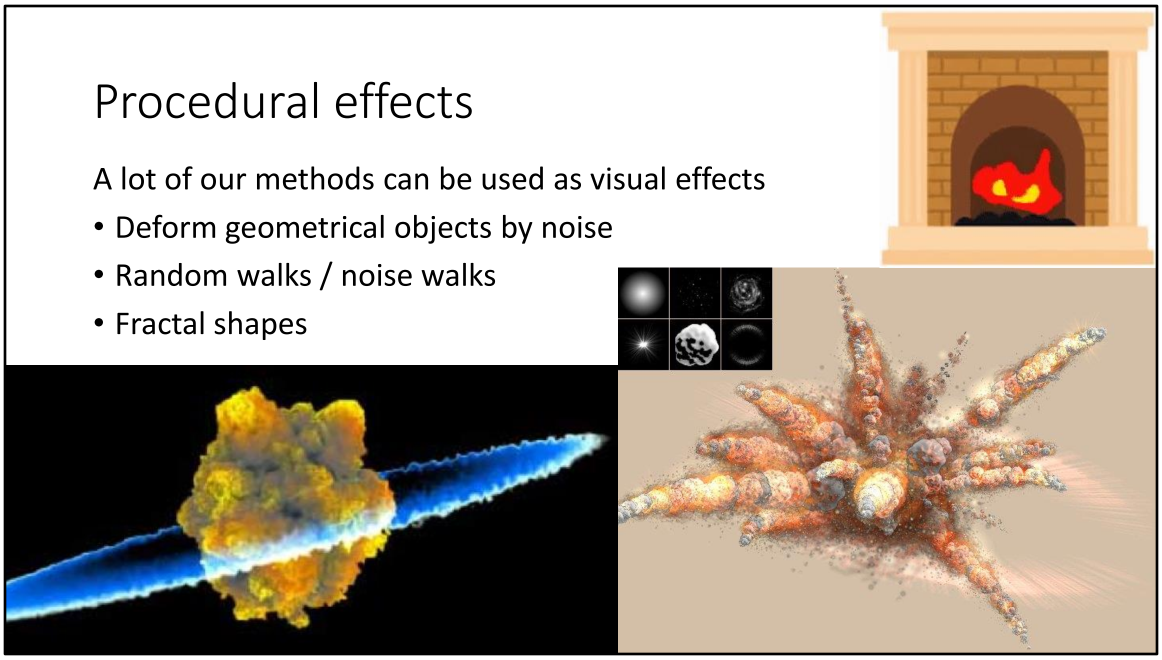
https://github.com/CiaccoDavide/Alchemy-Circles-Generator basic elements = strokes
https://github.com/AdrianMargel/glyphs Various symmetries and polygons utilized
Picture from chapter 15 of Procedural Art in Game Design – usage of half-toning algorithm with different weights to create an artistic effect.



You may want to generate art because you are not good at making art by hand ☺

Become a great artist in just 10 seconds http://ludumdare.com/compo/ludum-dare-27/?action=preview&uid=4987
Desolus – atmospheric game created by not-an-artist.
http://picbreeder.org/ (Users are the interactive fitness in an evaluator)

Procedural effects

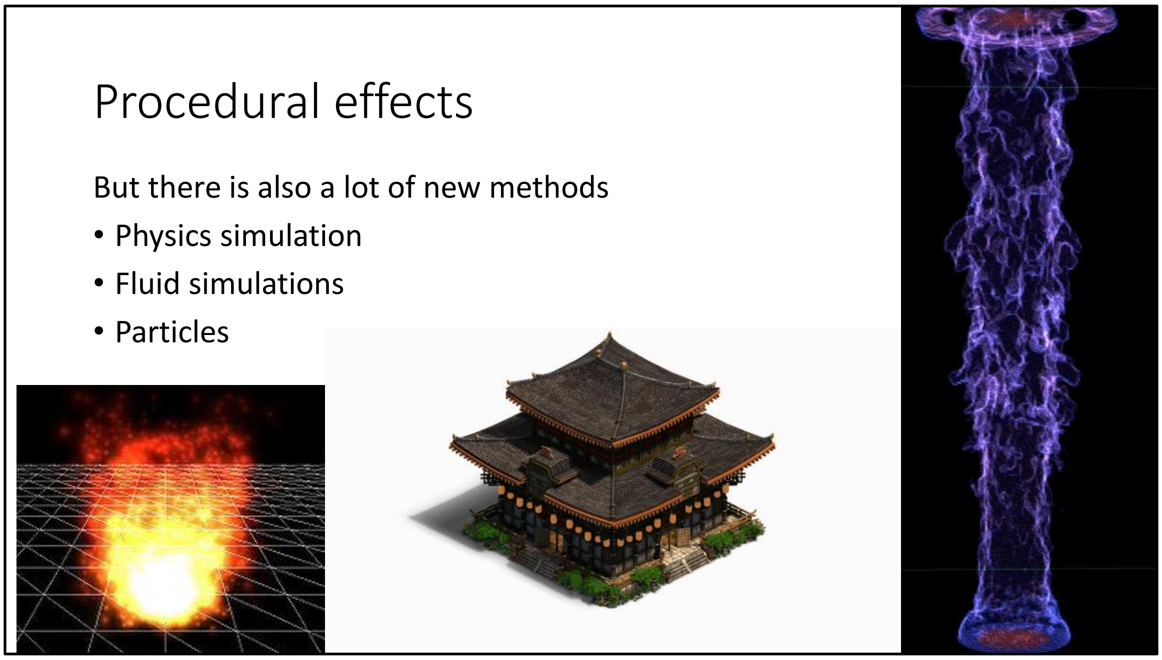


Procedural effects

A lot of our methods can be used as visual effects

- Deform geometrical objects by noise
- Random walks / noise walks
- Fractal shapes

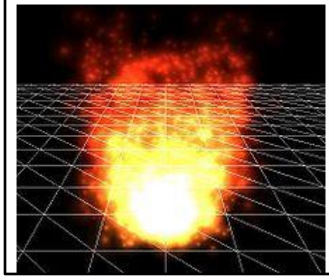
https://www.nvidia.fr/docs/IO/8343/RealTime-Procedural-Effects.pdf (somewhat old, but nice as intro)

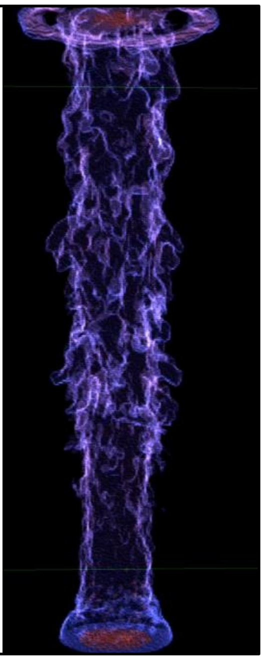


Procedural effects

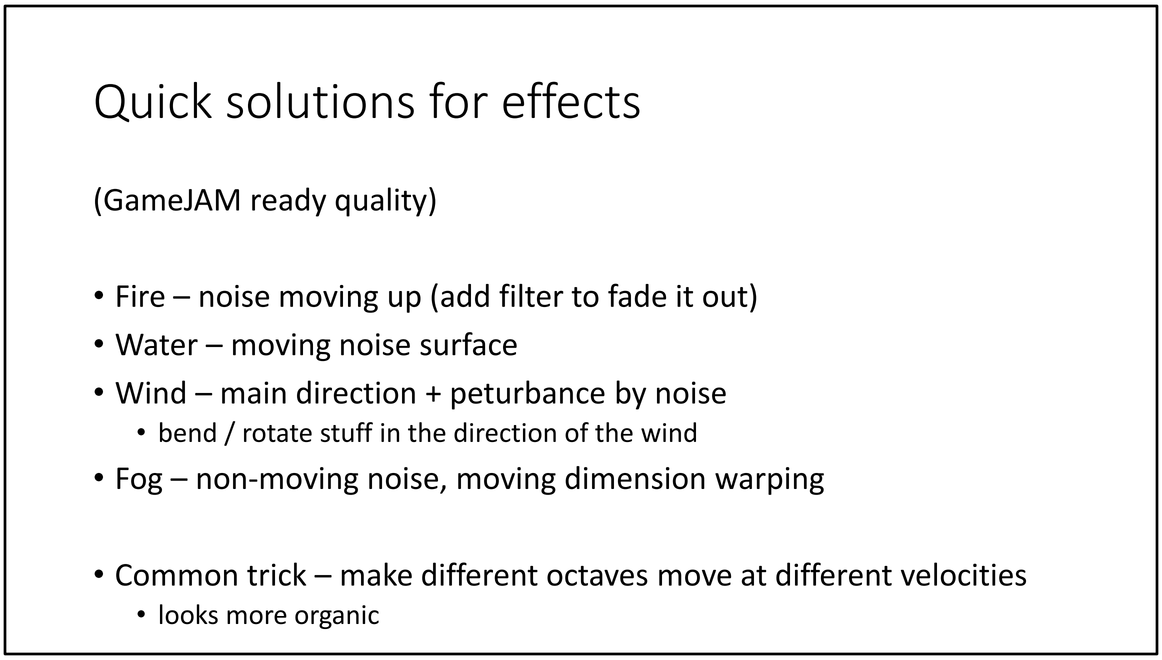
But there is also a lot of new methods

- Physics simulation
- Fluid simulations
- Particles





https://softologyblog.wordpress.com/2019/02/28/jos-stams-fluid-simulations-in-3d/
The famous powder game https://dan-ball.jp/en/javagame/dust/



Quick solutions for effects

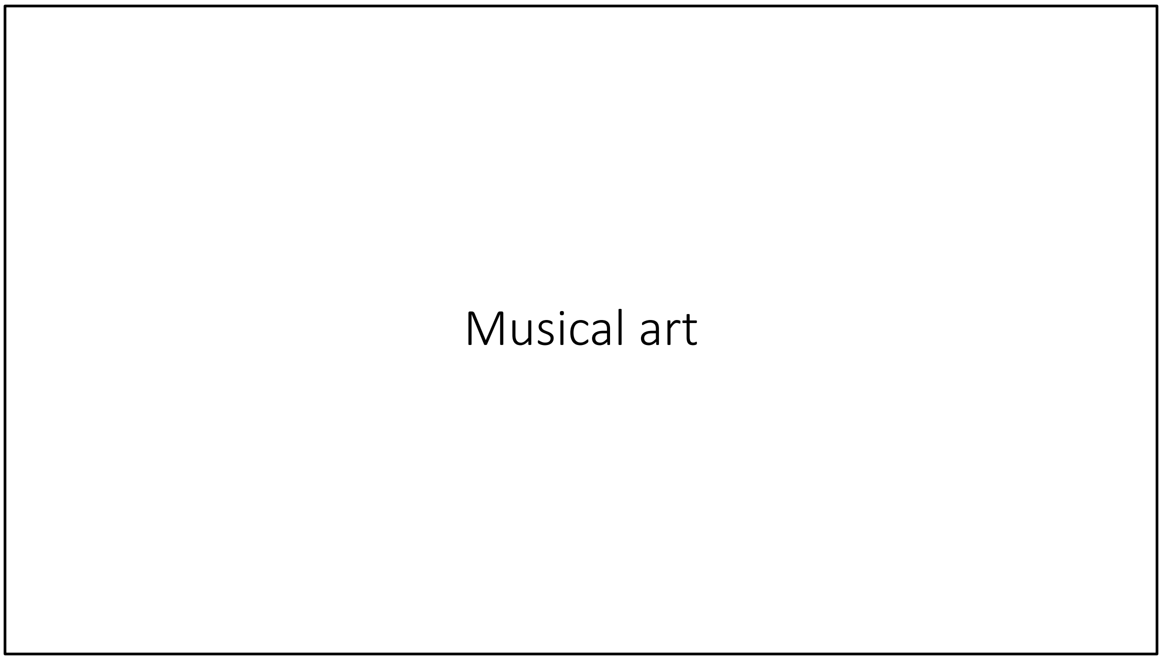
(GameJAM ready quality)

- Fire – noise moving up (add filter to fade it out)
- Water – moving noise surface
- Wind – main direction + peturbance by noise
  - bend / rotate stuff in the direction of the wind
- Fog – non-moving noise, moving dimension warping

- Common trick – make different octaves move at different velocities
  - looks more organic

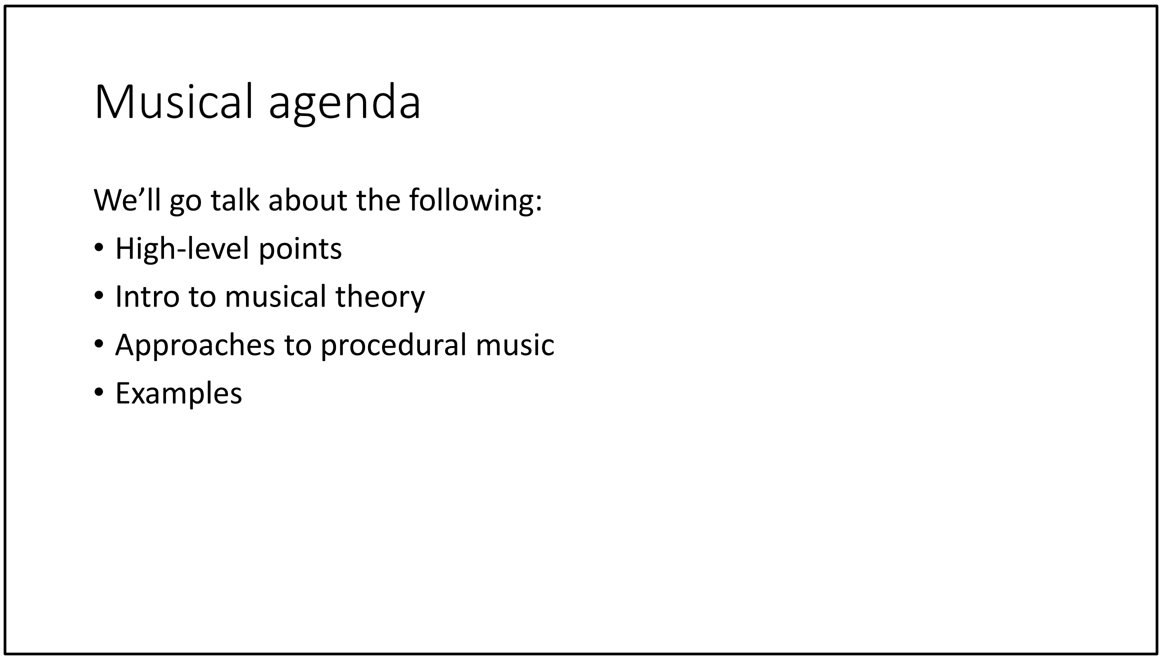
https://www.youtube.com/watch?v=CI3JZ-3cabg (fire by playing with noises, rising up)
https://www.youtube.com/watch?v=QEaTsz_0o44 (fog)
https://www.youtube.com/watch?v=DpOg8L-vYsU (water)

If you're doing a game with a larger scope, you might have to do some research. Graphical artists are creative and coming up with new things all the time.



Musical art

Best quick-start tutorial - http://www.procjam.com/tutorials/en/music/
A wonderful, slightly more in-depth tutorial - https://learningmusic.ableton.com/
A very nice example is also in Procedural Generation in Game Design, Chapter 17 – Audio and Composition.



Musical agenda

We'll go talk about the following:
- High-level points
- Intro to musical theory
- Approaches to procedural music
- Examples

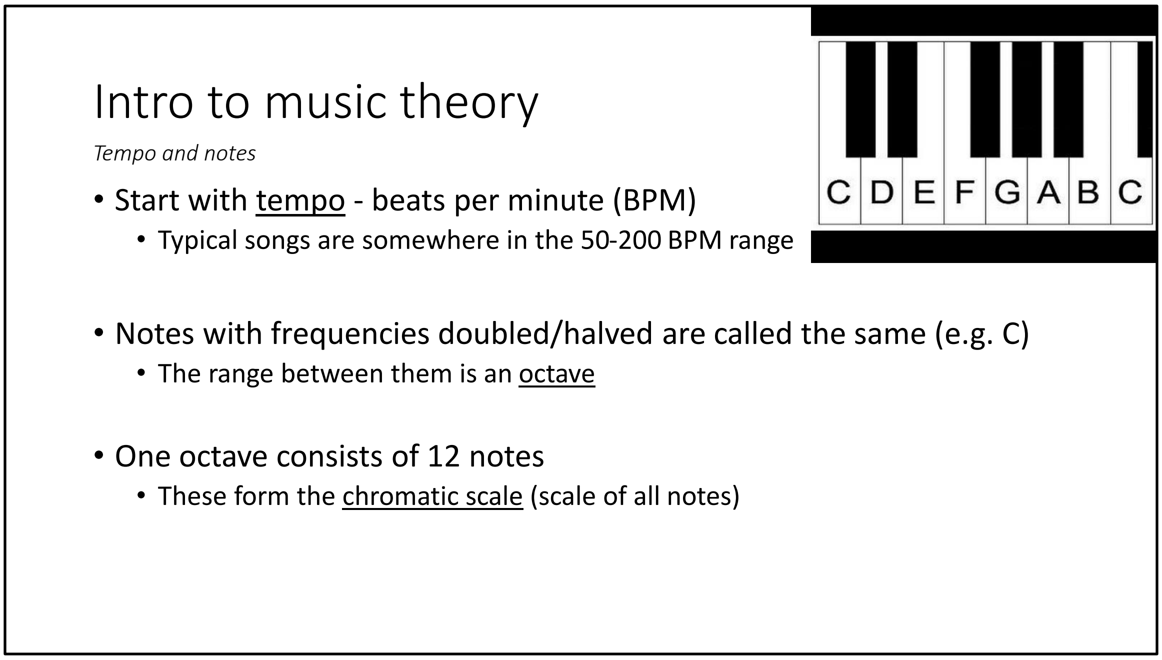
This really is a super-fast intro. We will not talk about stuff like the circle of fifths, etc. We will also completely skip all the underlying theory about how sound is formed, and the physical attributes of it. We start with beats, notes and instruments.

Also, music is complicated. We will at some points oversimplify things, so beware.

High-level points

- There's a lot that can be described by maths/physics
- But a lot of things just go beyond that
  - There are a lot of rules, which are meant to be (sometimes) broken
- You can think of it as dungeon
  - Opening
  - Some easy/hard fight sections
  - Prepare for boss fight
  - Boss fight
  - Outro

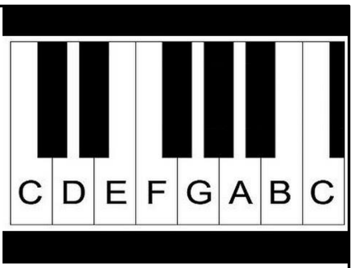
Quote by Rick Sanchez. http://bit.ly/2xblCUd



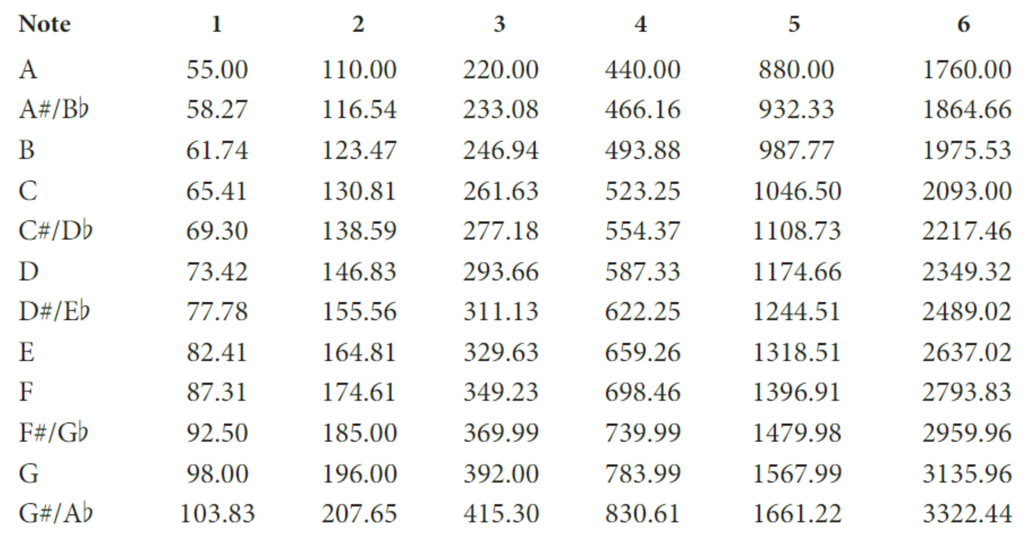
Intro to music theory

Tempo and notes

- Start with tempo - beats per minute (BPM)
  - Typical songs are somewhere in the 50-200 BPM range
- Notes with frequencies doubled/halved are called the same (e.g. C)
  - The range between them is an octave
- One octave consists of 12 notes
  - These form the chromatic scale (scale of all notes)



Note is a sound of constant frequency.
Notice the similiarity of octaves in musical theory and in noise.



Note	1	2	3	4	5	6
A	55.00	110.00	220.00	440.00	880.00	1760.00
A#/B♭	58.27	116.54	233.08	466.16	932.33	1864.66
B	61.74	123.47	246.94	493.88	987.77	1975.53
C	65.41	130.81	261.63	523.25	1046.50	2093.00
C#/D♭	69.30	138.59	277.18	554.37	1108.73	2217.46
D	73.42	146.83	293.66	587.33	1174.66	2349.32
D#/E♭	77.78	155.56	311.13	622.25	1244.51	2489.02
E	82.41	164.81	329.63	659.26	1318.51	2637.02
F	87.31	174.61	349.23	698.46	1396.91	2793.83
F#/G♭	92.50	185.00	369.99	739.99	1479.98	2959.96
G	98.00	196.00	392.00	783.99	1567.99	3135.96
G#/A♭	103.83	207.65	415.30	830.61	1661.22	3322.44

Frequencies of notes (Hz)

Notice the doubling of frequency each octave. Also, neighboring notes are different by a factor of “12th root of 2” (apx. 1.059463).

Remark: Starting with A makes somewhat more sense than with C, at least for us.

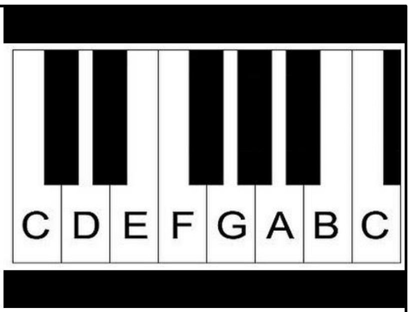
Intro to music theory

Scales

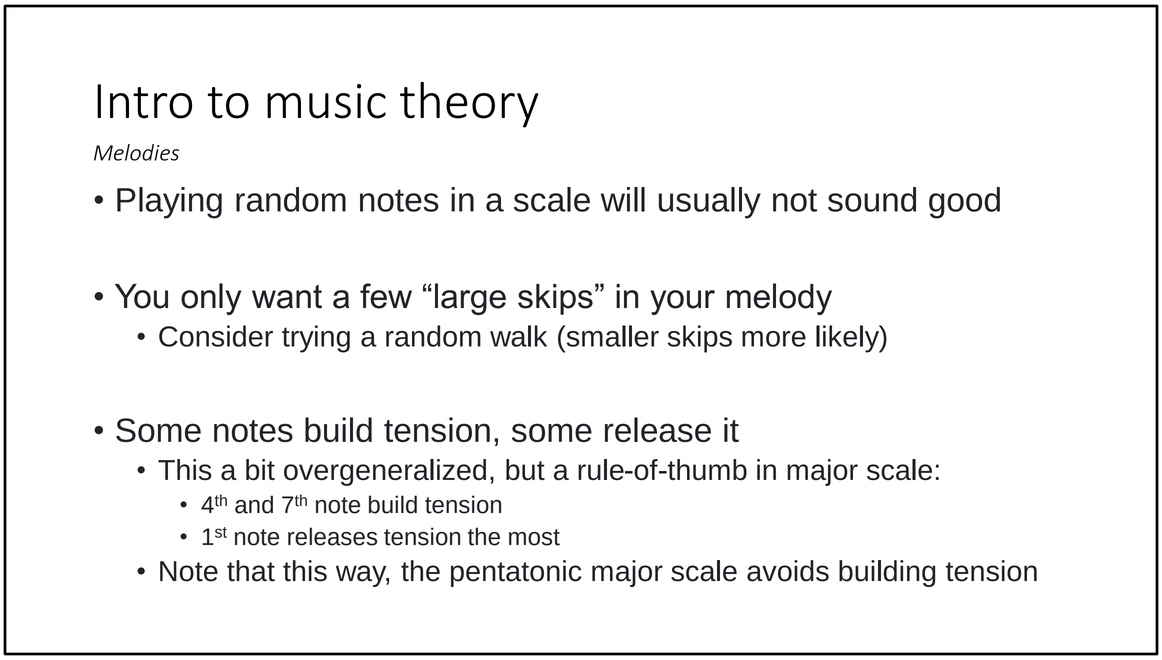
However, you don’t want to use all the notes

Useful other scales (intervals between notes):

- Major (2, 2, 1, 2, 2, 2, 1)
  - White keys on a piano if you start from C
  - Most classic, “happy”-sounding
- Natural Minor (2, 1, 2, 2, 1, 2, 2)
  - “Sad”-sounding
  - Notice it’s just a major scale starting on 6th note
- Major Pentatonic (2, 2, 3, 2, 3)
  - Just 5 notes, contains no semitones



Scales are really just things that were generally accepted to sound good.
https://en.wikipedia.org/wiki/Minor_scale



Intro to music theory

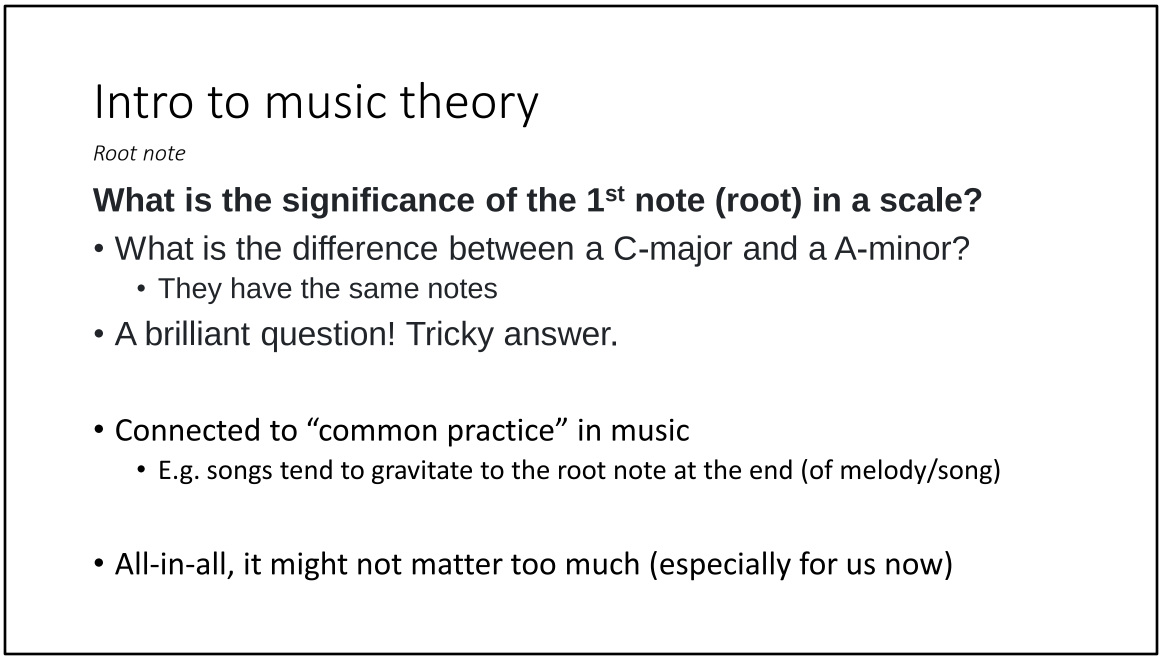
Melodies

- Playing random notes in a scale will usually not sound good
- You only want a few “large skips” in your melody
  - Consider trying a random walk (smaller skips more likely)
- Some notes build tension, some release it
  - This a bit overgeneralized, but a rule-of-thumb in major scale:
    - 4th and 7th note build tension
    - 1st note releases tension the most
  - Note that this way, the pentatonic major scale avoids building tension

Also, noise can be applied, either directly, or to further smoothen random-walk.

Scales are really just things that were generally accepted to sound good.
https://en.wikipedia.org/wiki/Minor_scale

The concept with tension and release is a useful tactic for generating nice melodies.
Also, everybody **absolutely should** watch this TED talk:
https://www.ted.com/talks/benjamin_zander_the_transformative_power_of_classical_music



Intro to music theory

Root note

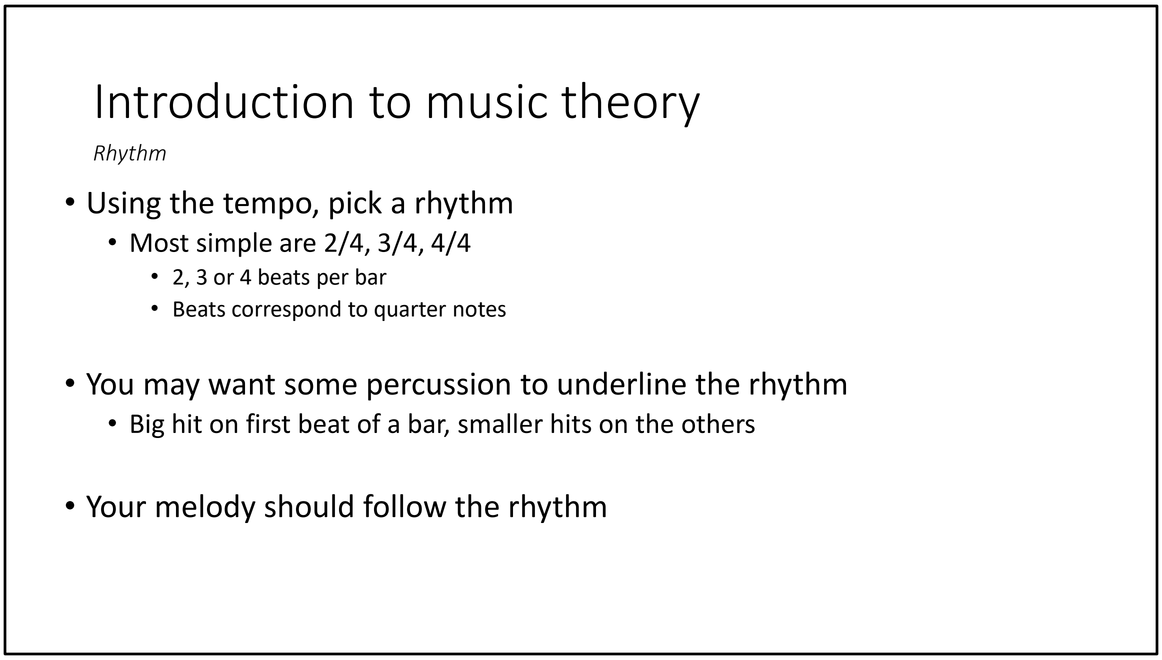
What is the significance of the 1st note (root) in a scale?

- What is the difference between a C-major and a A-minor?
  - They have the same notes
- A brilliant question! Tricky answer.

- Connected to "common practice" in music
  - E.g. songs tend to gravitate to the root note at the end (of melody/song)

- All-in-all, it might not matter too much (especially for us now)

A song can start on any note (usually in it's scale) and can end on any note.
Good answers here https://music.stackexchange.com/questions/27030/when-is-a-piece-in-a-minor-versus-c-major

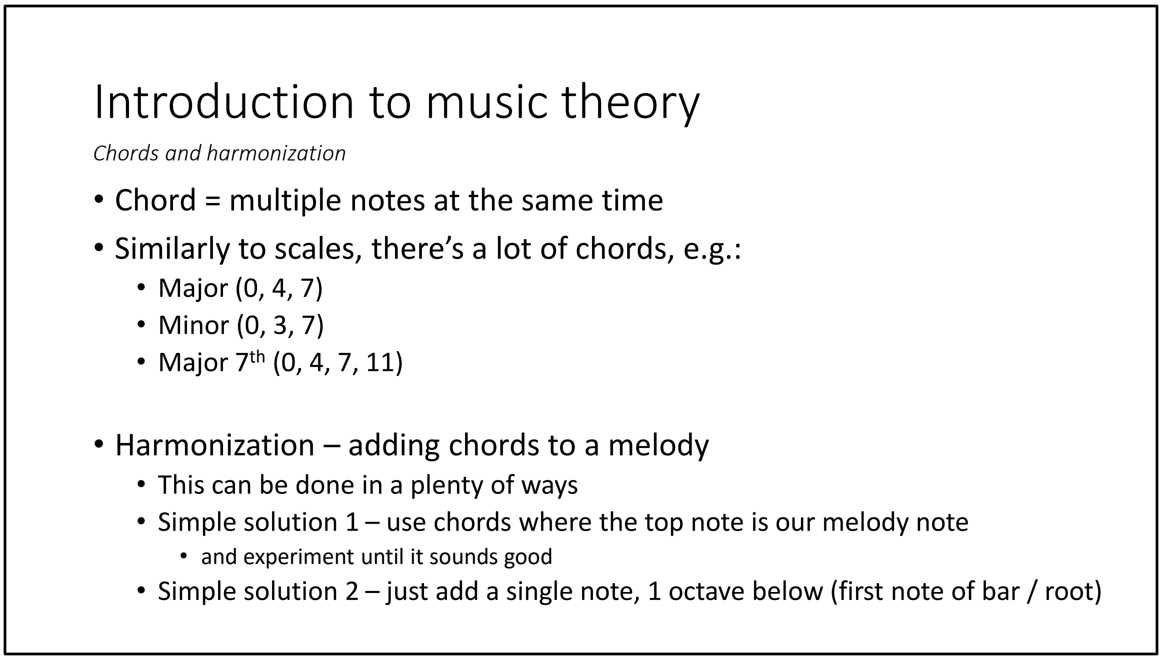


Introduction to music theory

Rhythm

- Using the tempo, pick a rhythm
  - Most simple are 2/4, 3/4, 4/4
    - 2, 3 or 4 beats per bar
    - Beats correspond to quarter notes
- You may want some percussion to underline the rhythm
  - Big hit on first beat of a bar, smaller hits on the others
- Your melody should follow the rhythm

It should be a rule that BPM is measured in the number behind the slash (60BPM = 60 quarter notes / minute)
But sometimes you find weird shenaniganz in music.



Introduction to music theory

Chords and harmonization

- Chord = multiple notes at the same time
- Similarly to scales, there's a lot of chords, e.g.:
  - Major (0, 4, 7)
  - Minor (0, 3, 7)
  - Major 7th (0, 4, 7, 11)
- Harmonization – adding chords to a melody
  - This can be done in a plenty of ways
  - Simple solution 1 – use chords where the top note is our melody note
    - and experiment until it sounds good
  - Simple solution 2 – just add a single note, 1 octave below (first note of bar / root)

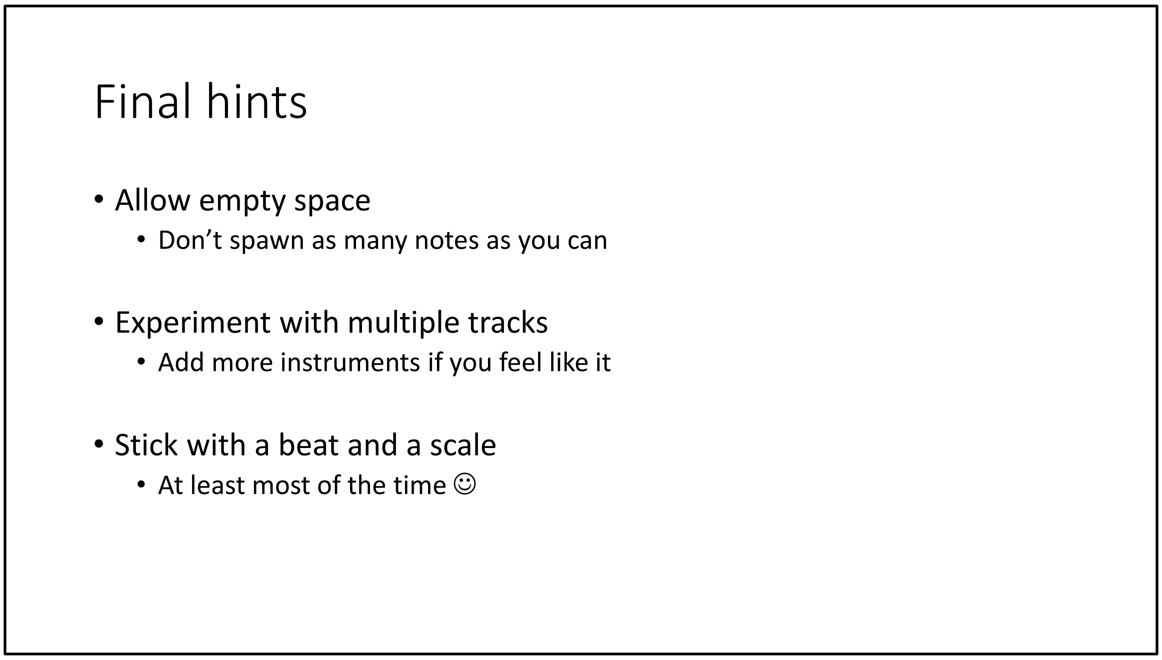
Similar notation on chords as to scales, they can start with any note and then the intervals go up.
e.g C-major 7th is C-E-G-B

Introduction to music theory

Song structure

- Unlike in terrain, you want **patterns** in your music
- Repeat generated melodies
  - Maybe put some small changes in some repetitions

- Intro – Verse – Chorus – Verse – Chorus – Bridge – Chorus
  - Common pattern in modern songs
  - Verse is somewhat “calmer”
  - Chorus carries the main melody
  - Bridge is a build-up for the last chorus



Final hints

- Allow empty space
  - Don’t spawn as many notes as you can
- Experiment with multiple tracks
  - Add more instruments if you feel like it
- Stick with a beat and a scale
  - At least most of the time ☺

If you would want to dig into more theory, here are some pointers:
0xdeadbeef
https://www.musictheory.net/lessons/57

Musical approaches and examples

Best quick-start tutorial - http://www.procjam.com/tutorials/en/music/
A very nice example is also in Procedural Generation in Game Design, Chapter 17 – Audio and Composition.

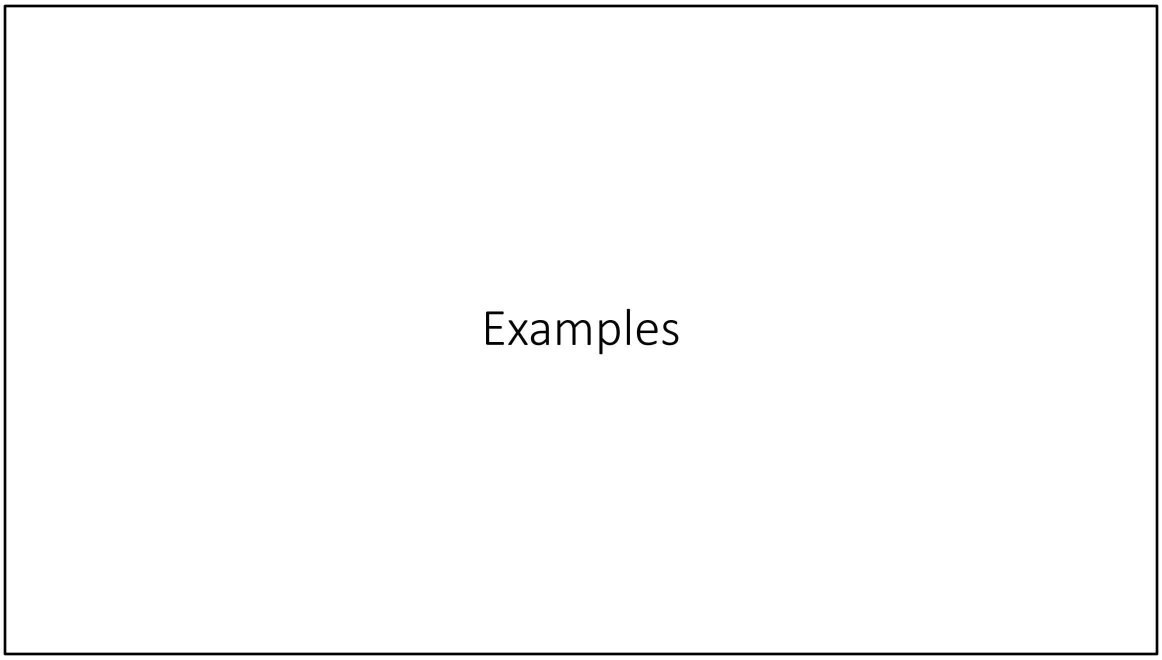
Composition

Using basic theory

- We can use some random-walks
- Maybe smoothed by noise
- Repeat some parts into a final structure

Other computer-science approaches:

- Model your melody as a Markov-chain
  - each note depends on *n* predecessors
- Use cellular automata (they naturally form patterns)
- Train neural networks



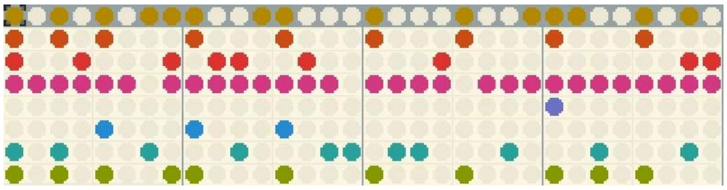
Examples

All examples are playable.
Output of ProcJAM tutorial http://www.procjam.com/tutorials/en/music/
Cellular automata in Wolfram http://tones.wolfram.com/generate
Very cool approach using a tiny bit more musical theory – youtube with approach https://youtu.be/olp93S6yVBk https://roycramer.itch.io/harpgen
Another great example with ambient music for a game
https://thesaplinggame.com/devlogs/music.html

Output of ProcJAM tutorial

• link in the notes

(recommended read)



• notice the empty space between the notes

http://www.procjam.com/tutorials/en/music/

WolframTones Created 2005

Wolfram's Approach to generated music

- based on cellular automata

Roy Cramer's generator

hand-made procedural generator

- uses a tiny bit more musical theory than we talked about

THE SAPLING

simple creation of ambient music for a game

- notice the repeating patterns with slight changes

Take-away points

- Cellular automata can make coherent surfaces / patterns
- L-Systems are good for “fractal” shapes (e.g. vegetation)
- Shape grammars can do level layouts, architecture, …

- Art can be made by most of the techniques taught so far

- With just a couple of rules, you can be a music composer

Q & A

cerny@gamedev.cuni.cz
discord.gg/Zts98PGw6z

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Material has been produced within and supported by the project
„Zvýšení kvality vzdělávání na UK a jeho relevance pro potřeby trhu práce“
kept under number CZ.02.2.69/0.0/0.0/16_015/0002362.