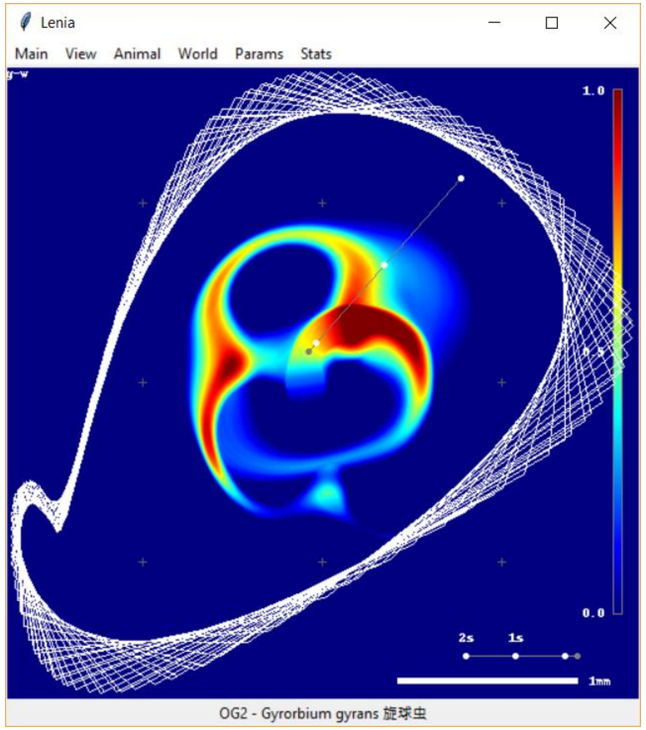
# How to Evolve Life in a Computer using Python

Bert Chan

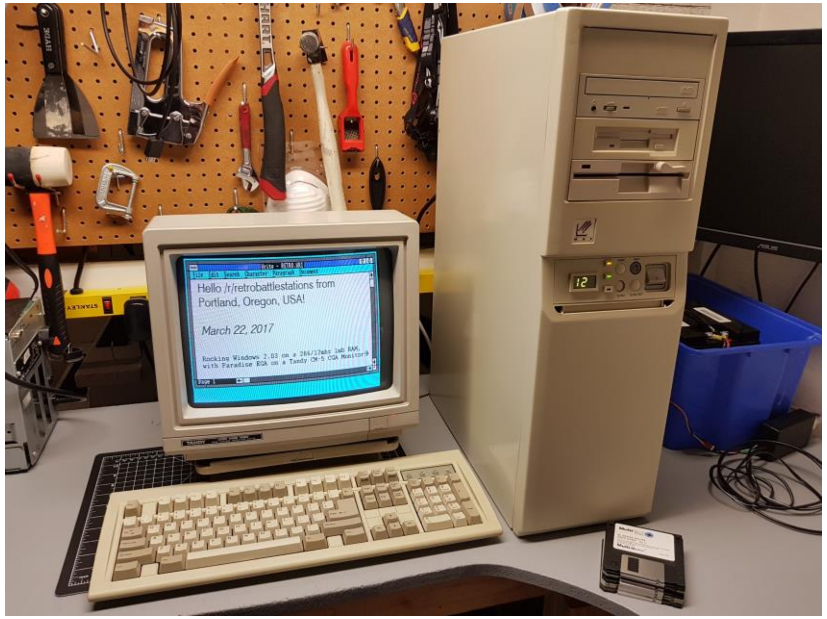
*Big Data Consultant @ ASL*

PyCon HK / Code Conf 2018



# Programming in the 1990's

- PC: **80286** (8MHz, 8MB RAM)
- OS: MS-DOS (Win3.1 too slow!)
- **Pascal**
  - Simulate life
  - Simulate gravity, fractals
  - Hack & decode games
- **Assembly**
  - Main loop – very fast!
  - Direct write to video cache



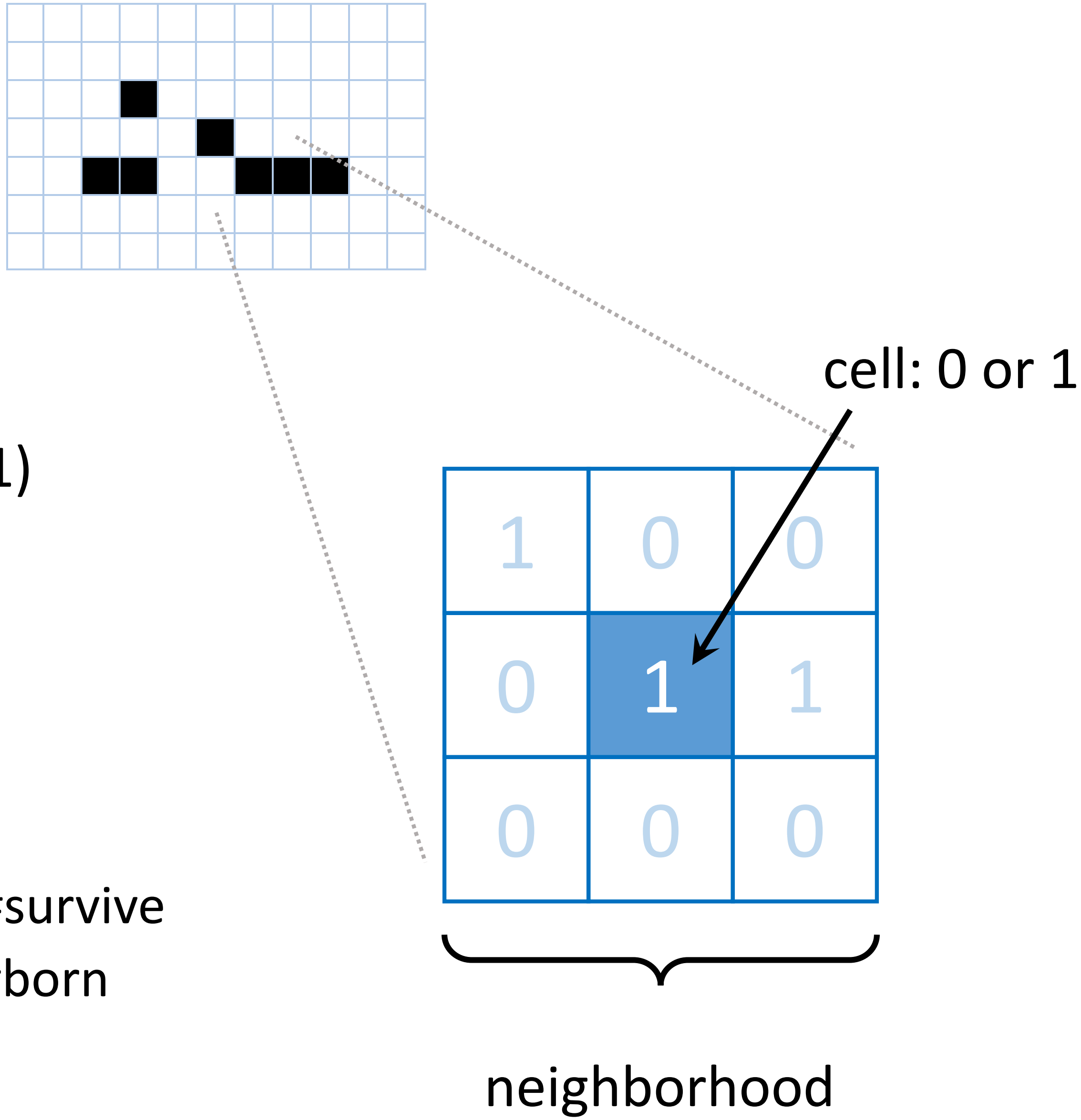
*If you did coding and hacked stuffs in the 90’s, you’re a…*

# Simulate Life

- **Conway’s Game of Life**
    - John Conway 1970
    - Cellular Automata – array of **cells**  (0 or 1)
    - **Neighborhood**  (8 cells, sum)

- Simple **if-then-else** rule

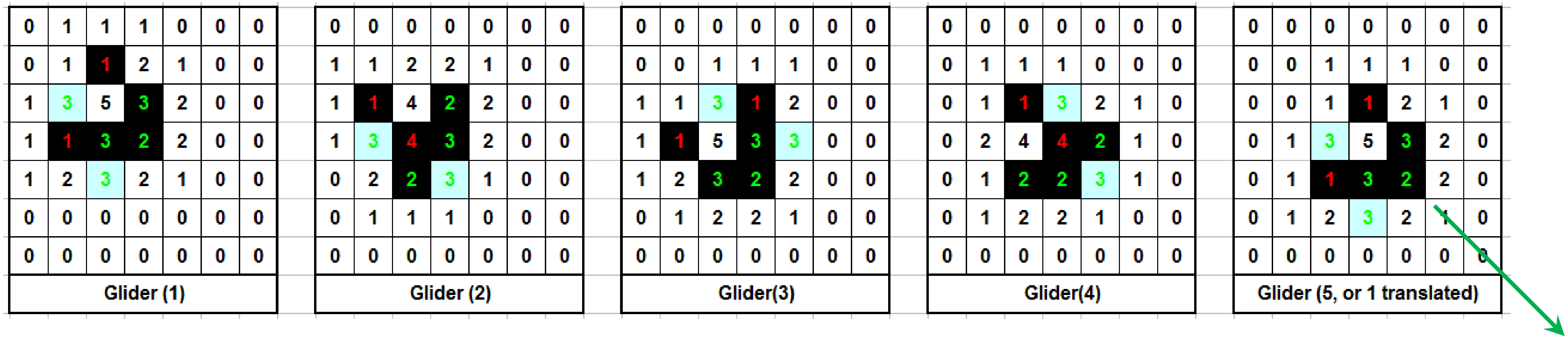
```
for cell in cells:
    if cell==1 and sum in [2, 3]: cell = 1    #survive
    elif cell==0 and sum in [3]:  cell = 1    #born
    else: cell = 0    #die
```



| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

# Conway’s Game of Life

- The moving **glider**
- Green cells survive, blue cells are born, red cells die



| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 2 | 1 | 0 | 0 |
| 1 | 3 | 5 | 3 | 2 | 0 | 0 |
| 1 | 1 | 3 | 2 | 2 | 0 | 0 |
| 1 | 2 | 3 | 2 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Glider (1)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 2 | 2 | 1 | 0 | 0 |
| 1 | 1 | 4 | 2 | 2 | 0 | 0 |
| 1 | 3 | 4 | 3 | 2 | 0 | 0 |
| 0 | 2 | 2 | 3 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Glider (2)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 3 | 1 | 2 | 0 | 0 |
| 1 | 1 | 5 | 3 | 3 | 0 | 0 |
| 1 | 2 | 3 | 2 | 2 | 0 | 0 |
| 0 | 1 | 2 | 2 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Glider(3)

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 3 | 2 | 1 | 0 |
| 0 | 2 | 4 | 4 | 2 | 1 | 0 |
| 0 | 1 | 2 | 2 | 3 | 1 | 0 |
| 0 | 1 | 2 | 2 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

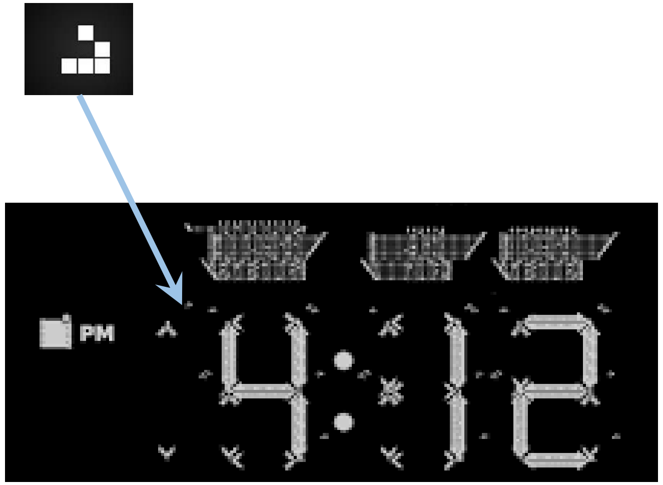
Glider(4)

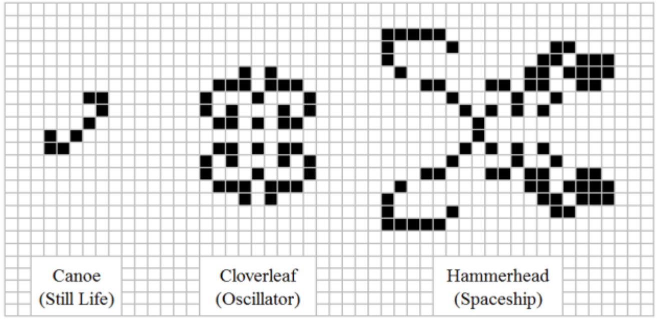
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 2 | 1 | 0 |
| 0 | 1 | 3 | 5 | 3 | 2 | 0 |
| 0 | 1 | 1 | 3 | 2 | 2 | 0 |
| 0 | 1 | 2 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

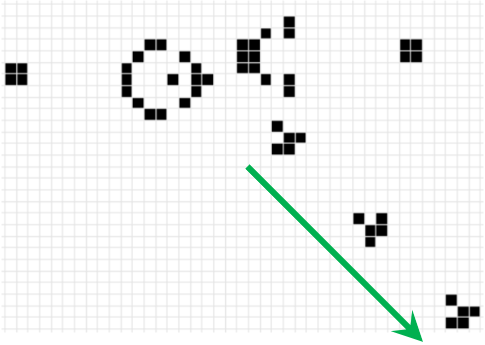
Glider (5, or 1 translated)

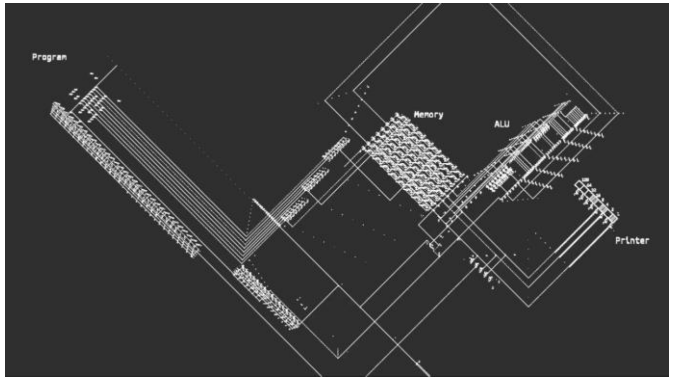
# Conway's Game of Life

- Spaceships, glider gun…
- Logic gate, clock, computer…
- Hackers love it!
- Good way to learn programming!



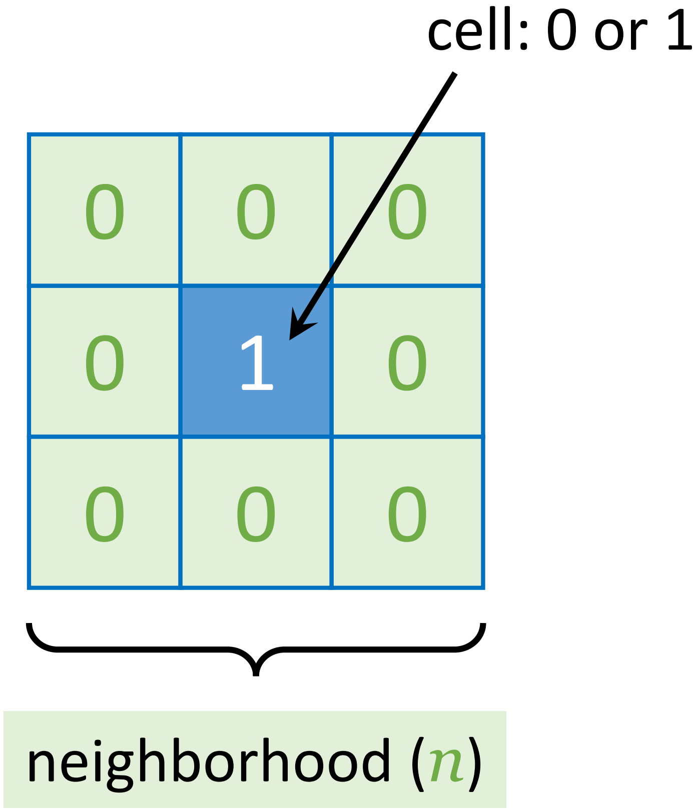






# Play with the rules

- What if we…



cell: 0 or 1

|   |   |   |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

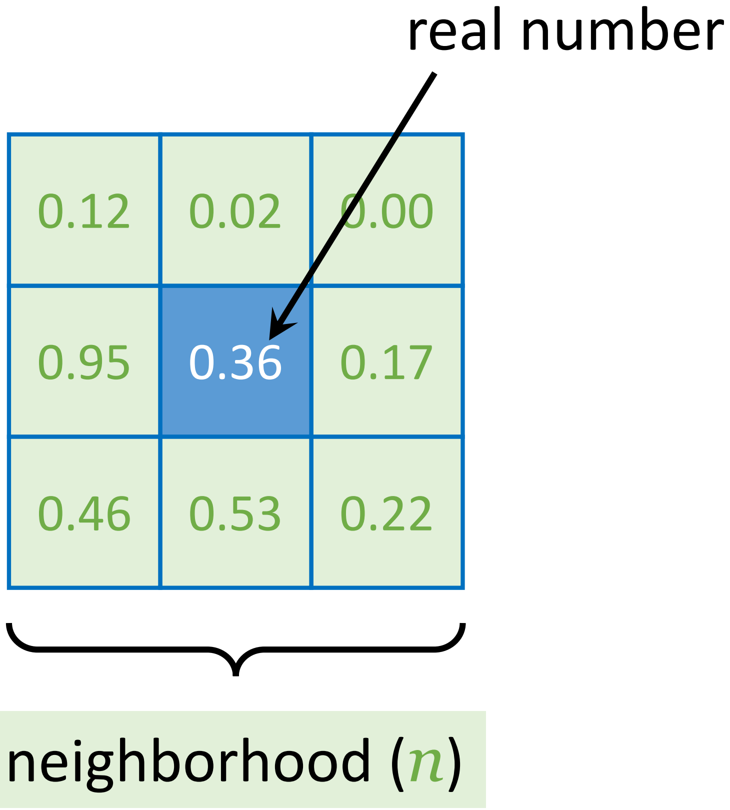
neighborhood ($n$)

sum = $\sum n$

cell = (if sum … then … else …)

# Play with the rules

- What if we…
  - Use floating point?



real number

|      |      |      |
|------|------|------|
| 0.12 | 0.02 | 0.00 |
| 0.95 | 0.36 | 0.17 |
| 0.46 | 0.53 | 0.22 |

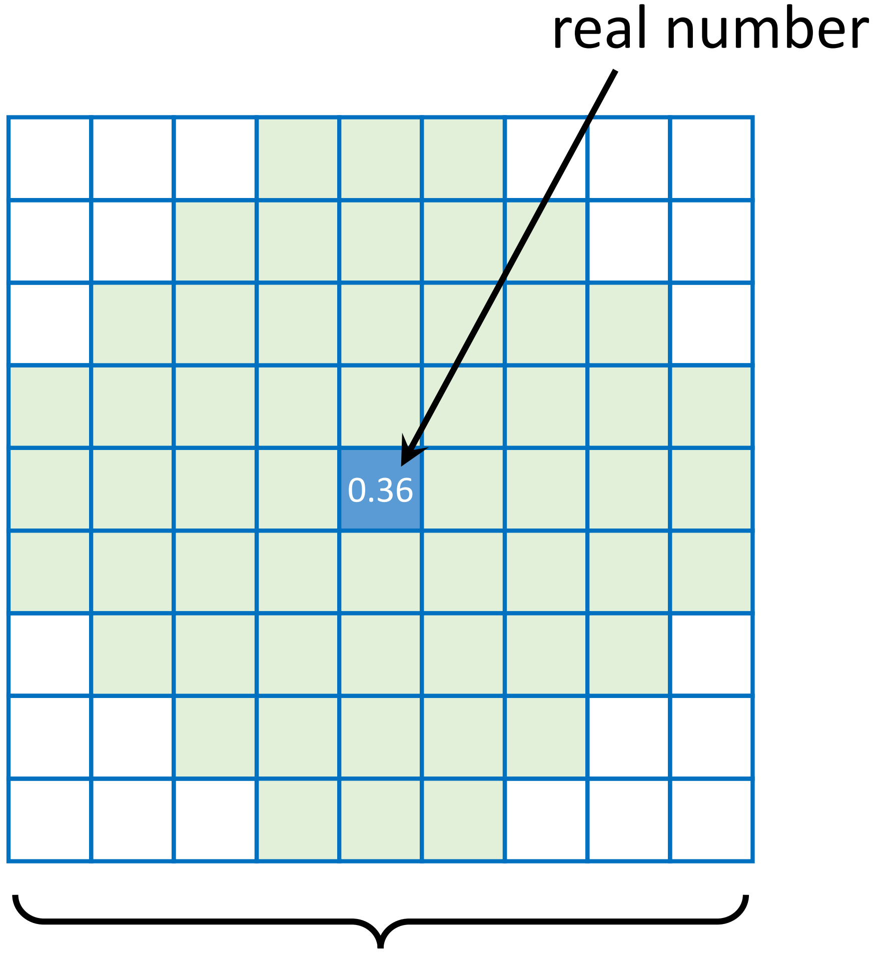
neighborhood ($n$)

sum = $\sum n$

cell = (if sum … then … else …)

# Play with the rules

- What if we…
  - Use floating point?
  - Bigger neighborhood? Circular?



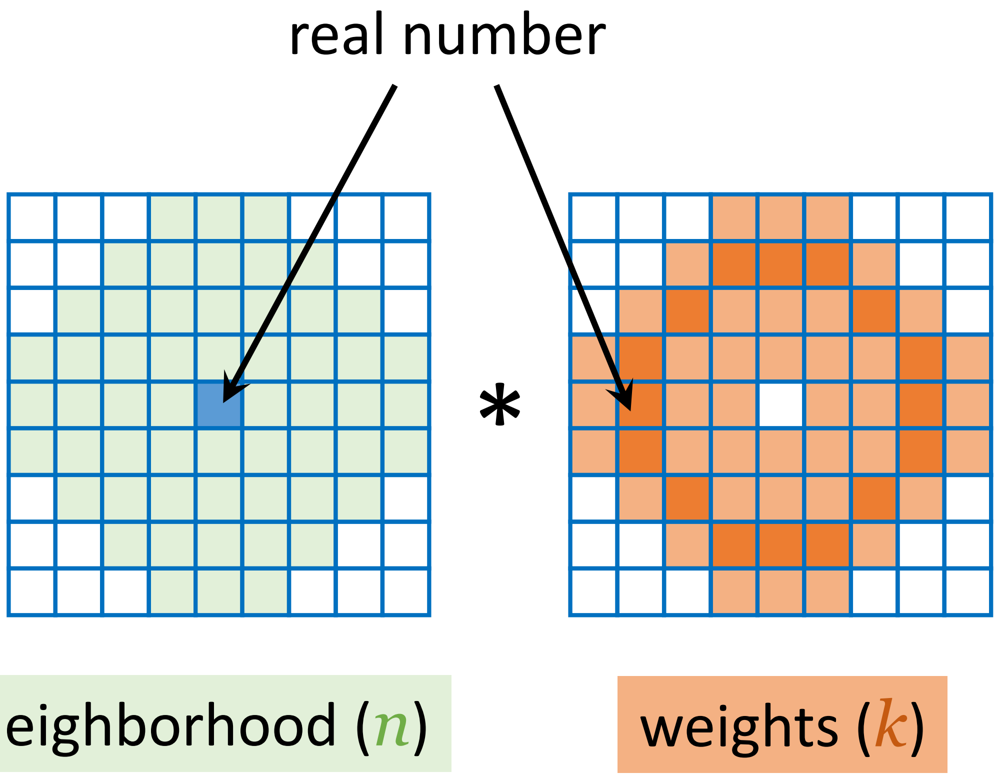
neighborhood ($n$)

sum = $\sum n$

cell = (if sum … then … else …)

# Play with the rules

- What if we…
  - Use floating point?
  - Bigger neighborhood?  Circular?
  - Weighted sum?



real number

*

neighborhood ($n$)

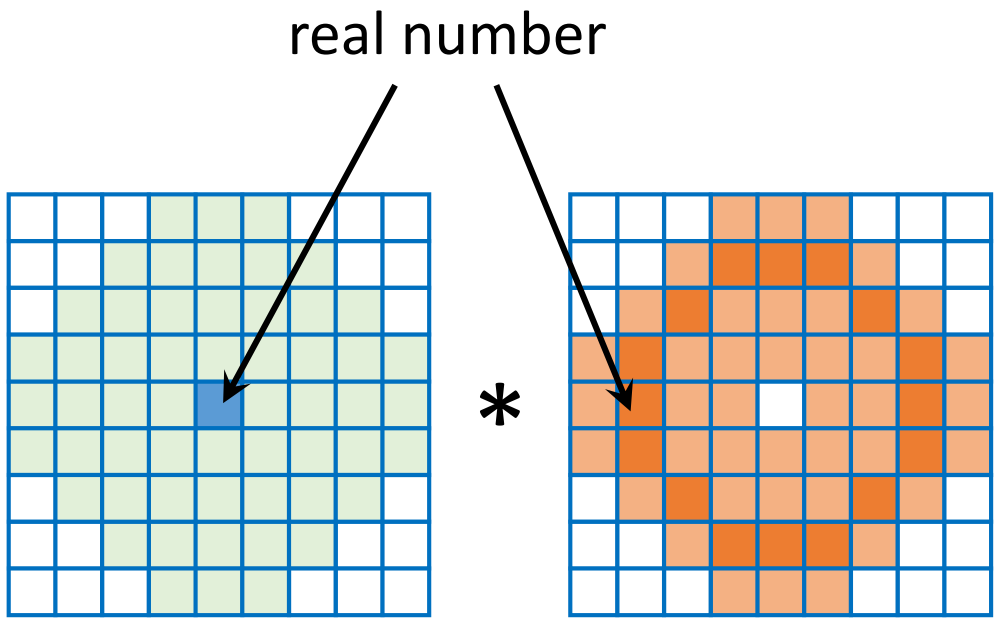
weights ($k$)

sum = $\sum nk$

cell = (if sum … then … else …)

# Play with the rules

- What if we…
    - Use floating point?
    - Bigger neighborhood?  Circular?
    - Weighted sum?
    - Smooth update?

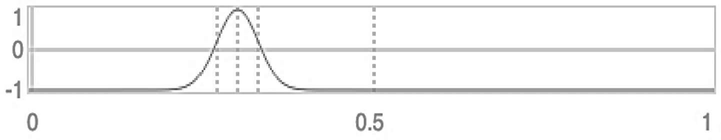


neighborhood ($n$)

weights ($k$)

sum = $\sum nk$

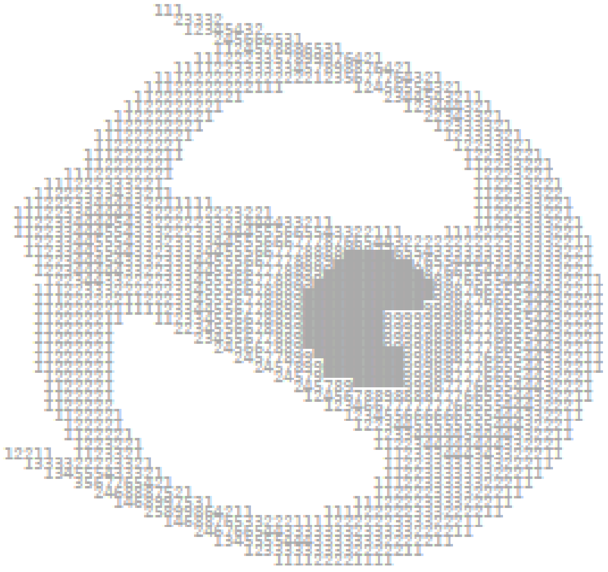
cell = cell + 0.1 * f(sum)



# Play with the rules

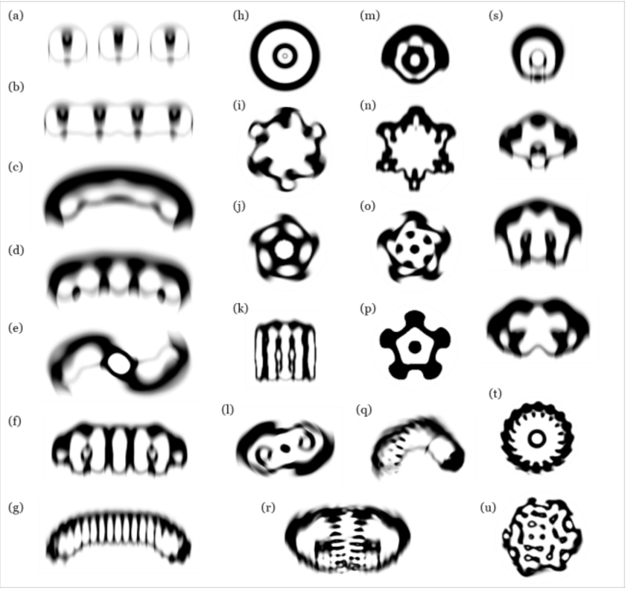
- What if we…
  - Use floating point?
  - Bigger neighborhood?  Circular?
  - Weighted sum?
  - Smooth update?

- Spooky things happened…



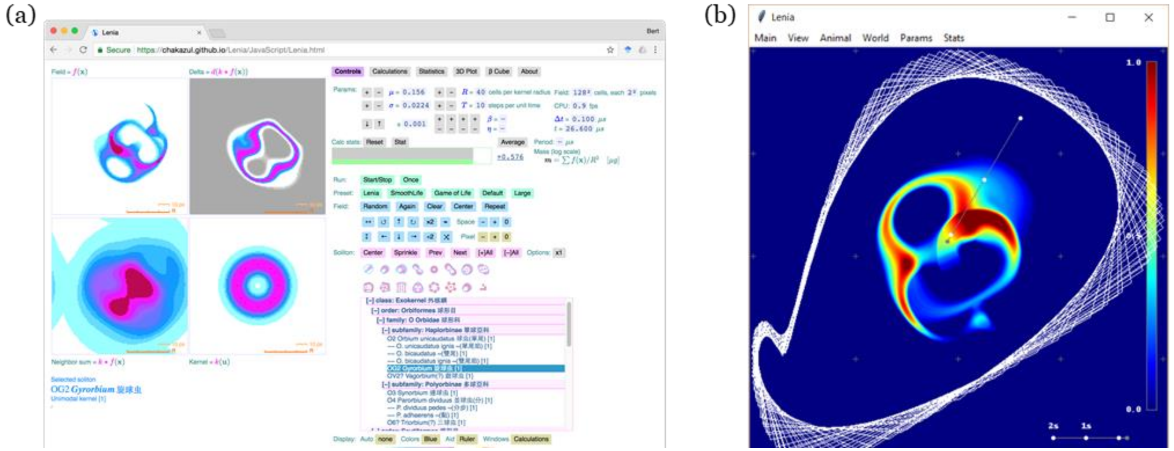
OMG WHAT IS THIS??

# Lenia

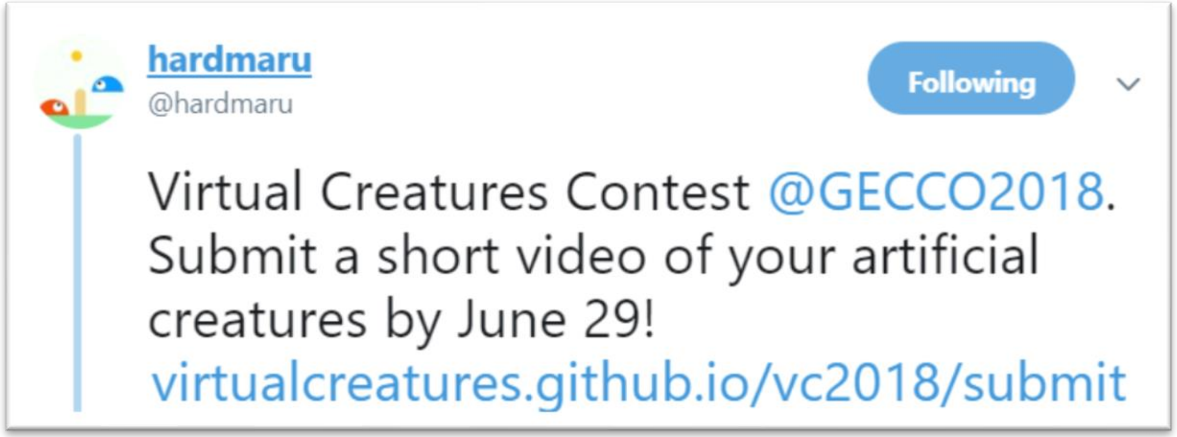


- New kind of **Artificial Life**
  - Microorganism-like creatures
  - Discovered 400+ species
  - Study their anatomy, behavior, physiology…

- Good programming exercise
  - JavaScript, C#, MATLAB, Python


(a)
Lenia
https://chakazul.github.io/Lenia/JavaScript/Lenia.html
Field = f(x)
Delta = d(k * f(x))
Neighbor sum = k * f(x)
Kernel = k(u)
Controls
Calculations
Statistics
3D Plot
β Cube
About
Params:
μ = 0.156
σ = 0.0224
R = 40 cells per kernel radius
T = 10 steps per unit time
Field: 128² cells, each 2² pixels
CPU: 0.9 fps
Δt = 0.100 μs
t = 26.600 μs
Calc stats: Reset Stat Average
Run: Start/Stop Once
Preset: Lenia SmoothLife Game of Life Default Large
Field: Random Again Clear Center Repeat
Soliton: Center Sprinkle Prev Next [+]All [−]All
[−] class: Exokernel
[−] order: Orbiformes
[−] family: O Orbidae
[−] subfamily: Haplorbinae
[−] subfamily: Polyorbinae
Selected soliton
OG2 Gyrorbium 旋球虫
Display: Auto none Colors Blue Aid Ruler Windows Calculations
(b)
Lenia
Main View Animal World Params Stats
1.0
0.0
2s
1s

# Video



hardmaru
@hardmaru

Virtual Creatures Contest @GECCO2018. Submit a short video of your artificial creatures by June 29! virtualcreatures.github.io/vc2018/submit

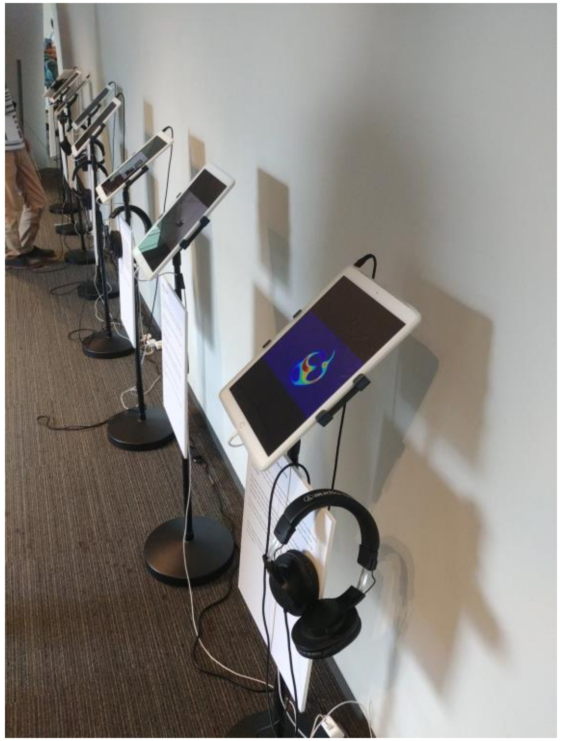
- Python → showcase video
- https://vimeo.com/277328815

# Kyoto

- Won GECCO Virtual Creatures Contest, Kyoto
- Honorable Mention in ALIFE Art Award, Tokyo
- Meet my AI hero – @hardmaru
  - David Ha (Google Brain Tokyo)

GECCO

ALIFE 2018



# Using Python

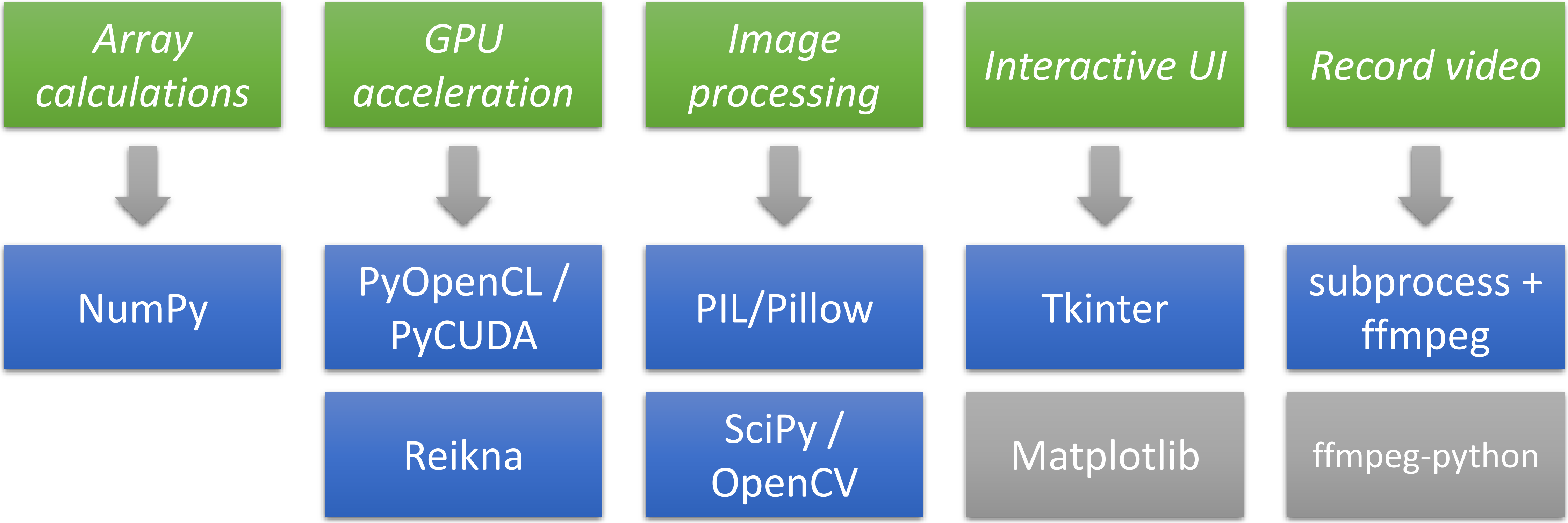
for PyCon HK

# Why Python?

- Good performance
- Fast coding
- Nice syntax (indent, list comprehension, etc)
- Lots of useful libraries
- Vibrant community (PyCon, GitHub…)

# Python Libraries

- “Rule 34” of Python
  - *“If there is a need, there is a Python library for it.”*



*Array calculations* → NumPy

*GPU acceleration* → PyOpenCL / PyCUDA, Reikna

*Image processing* → PIL/Pillow, SciPy / OpenCV

*Interactive UI* → Tkinter, Matplotlib

*Record video* → subprocess + ffmpeg, ffmpeg-python

# NumPy

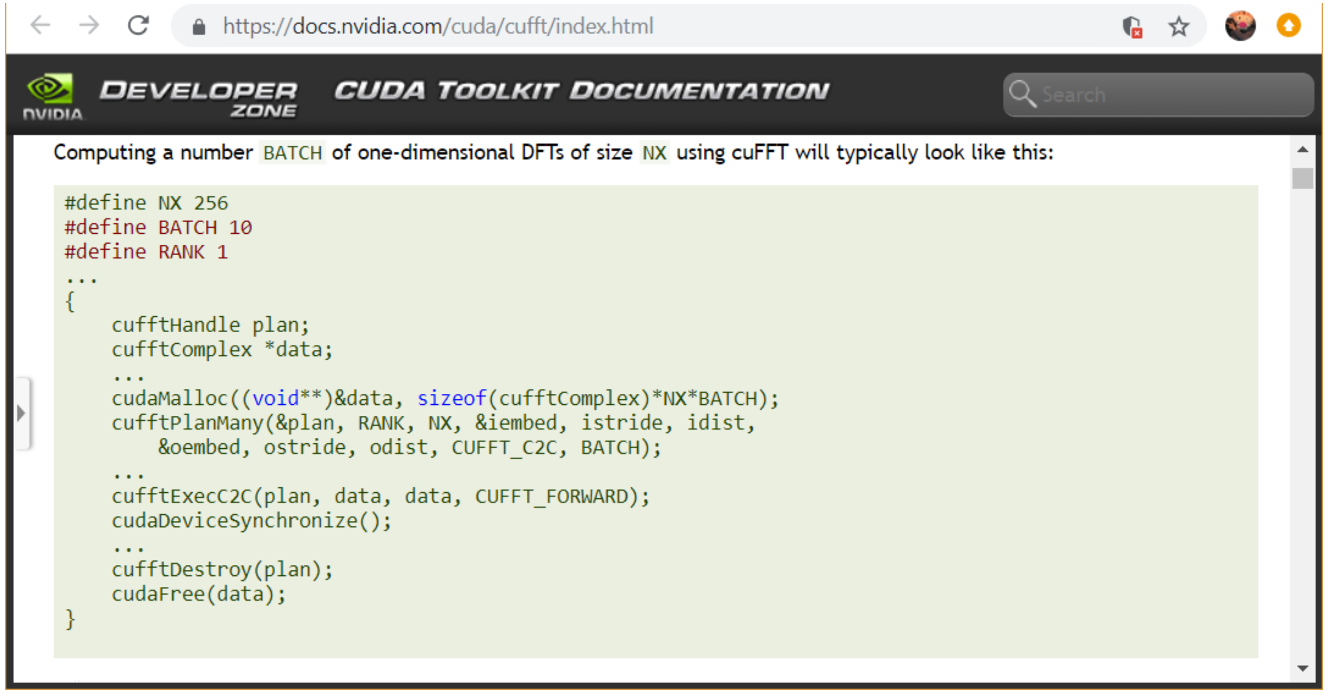
- Fast array calculations
  - ✓Machine learning, deep learning
  - ✓Basis of image processing, time-series
  - ✓Cellular automata (weighted sum using FFT)

- Main loop of Lenia in 3 lines

```
potential_fft = np.fft.fft2(cells) * kernel_fft
potential = np.fft.fftshift(np.real(np.fft.ifft2(potential_fft))
cells_new = np.clip(cells + dt * g(potential, m, s), 0, 1)
```

# PyOpenCL/PyCUDA + Reikna

- GPU acceleration
  - (NVIDIA) CUDA → PyCUDA
  - (Apple) OpenCL → PyOpenCL


https://docs.nvidia.com/cuda/cufft/index.html
NVIDIA
DEVELOPER ZONE
CUDA TOOLKIT DOCUMENTATION
Search

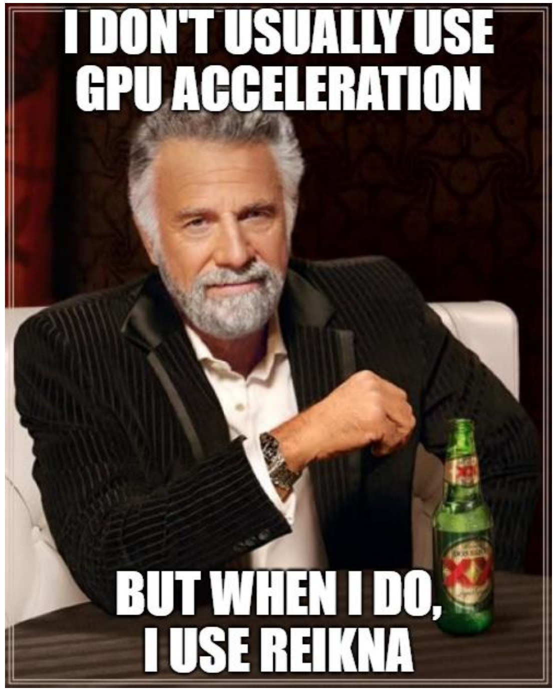
Computing a number BATCH of one-dimensional DFTs of size NX using cuFFT will typically look like this:

```
#define NX 256
#define BATCH 10
#define RANK 1
...
{
    cufftHandle plan;
    cufftComplex *data;
    ...
    cudaMalloc((void**)&data, sizeof(cufftComplex)*NX*BATCH);
    cufftPlanMany(&plan, RANK, NX, &iembed, istride, idist,
        &oembed, ostride, odist, CUFFT_C2C, BATCH);
    ...
    cufftExecC2C(plan, data, data, CUFFT_FORWARD);
    cudaDeviceSynchronize();
    ...
    cufftDestroy(plan);
    cudaFree(data);
}
```

# PyOpenCL/PyCUDA + Reikna

- Reikna
  - PyOpenCL/PyCUDA wrapper
  - Compiles the GPU code for you
  - GPU accelerated FFT in a few lines

```
gpu_thr = reikna.cluda.any_api().Thread.create()
gpu_fft = reikna.fft.FFT(cells.astype(np.complex64)).compile(gpu_thr)
op_dev = gpu_thr.to_device(cells.astype(np.complex64))
gpu_fft(op_dev, op_dev, **kwargs)
cells = op_dev.get()
```


I DON'T USUALLY USE GPU ACCELERATION
BUT WHEN I DO, I USE REIKNA

# PIL/Pillow, SciPy, OpenCV

- Image handling
  - PIL (Python Image Lib) → pillow
  - Create image, draw lines/texts, save GIF…

```
img = PIL.Image.frombuffer('P', buffer.shape, buffer, …)
draw = PIL.ImageDraw.Draw(img)
img[0].save(path, format='GIF', append_images=self.gif[1:], loop=0 …)
```

- Image processing
  - SciPy

```
scipy.ndimage.rotate(A, reshape=False, order=0, mode='wrap’)
```

  - OpenCV-Python

# Tkinter vs Matplotlib

- Interactive UI
  - Real-time 2D image display
  - Menu, keyboard binding, clipboard
- Matplotlib
  - For data visualization
  - Powerful but slow…
- Tkinter (Toolkit interface)
  - Basic and fast
  - Others: wxPython, PyQt, PyGTK…

matplotlib

Tcl/Tk

# Tkinter vs Matplotlib

- Interactive UI

  win = **tk**.Tk()

  **tk**.Canvas()

  **tk**.Menu()

  win.bind('<Key>', key_press_event)

  win.clipboard_get()

- Python 3

  import **tkinter** as tk

# subprocess + ffmpeg

- Pythonic FFmpeg wrappers
  - ffmpeg-python, ffmpy, etc.

- Pipe video to ffmpeg

```
cmd = ['/usr/local/bin/ffmpeg’, '-f','rawvideo', …]
video = subprocess.Popen(cmd, stdin=subprocess.PIPE)
for img in imgs:
        video.stdin.write(img.convert('RGB').tobytes())
video.stdin.close()
```

# About ALife and AI

for HK Code Conf

# Lenia

- Not just funny creatures
- Using **AI** to create **ALife**

# About ALife

- **Artificial Life**
  - Simulate biological life or create new lifeforms
  - → Create a body

**Wet ALife** = biochemistry, synthetic life
Artificial cell, expanded genetic code…

**Hard ALife** = hardware, robots & machines
Humanoids, *Strandbeest*…

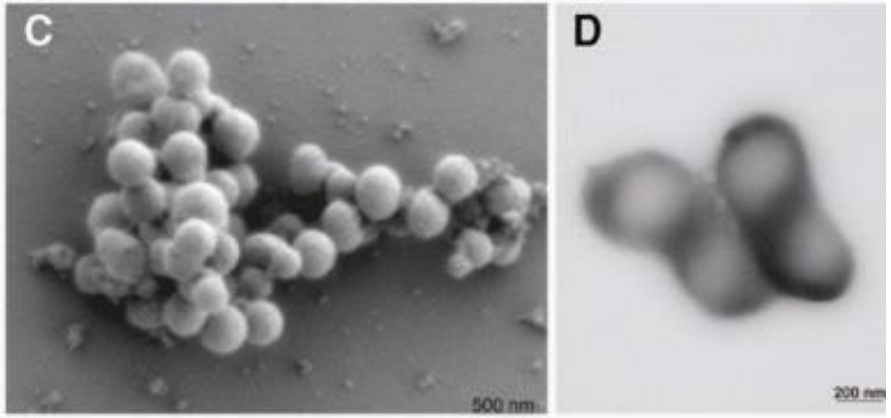
**Soft ALife** = software, simulations
Cellular automata, virtual creatures…

# About ALife

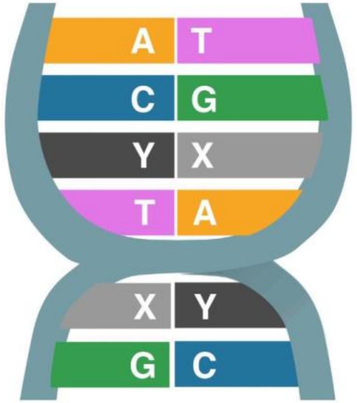
**Wet ALife** = biochemistry, synthetic life
Artificial cell, expanded genetic code…

**Hard ALife** = hardware, robots & machines
Humanoids, *Strandbeest*…

**Soft ALife** = software, simulations
Cellular automata, virtual creatures…



Synthetic cell (*JCVI*, 2010)



Expanded DNA (*TSRI*, 2014)

# About ALife

**Wet ALife** = biochemistry, synthetic life
Artificial cell, expanded genetic code…

**Hard ALife** = hardware, robots & machines
Humanoids, *Strandbeest*…

**Soft ALife** = software, simulations
Cellular automata, virtual creatures…

Strandbeest (*Theo Jansen*, 1990)

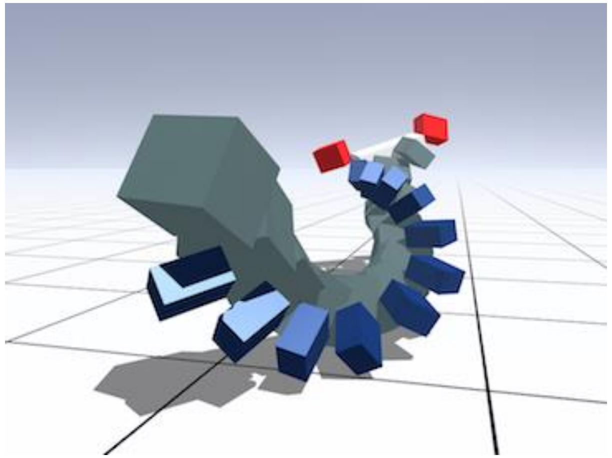
Atlas (*Boston Dynamics*, 2017)

# About ALife

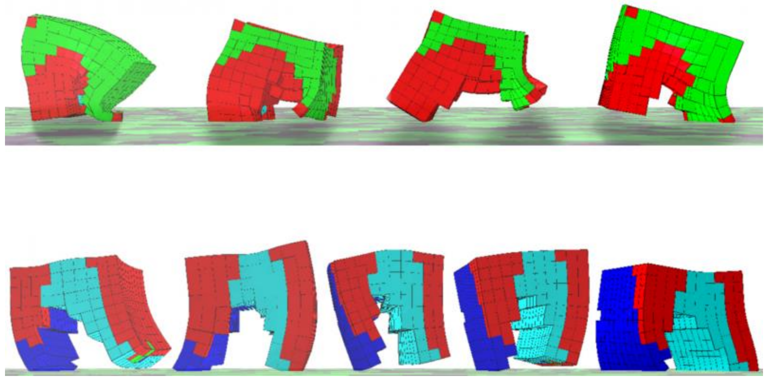
**Wet ALife** = biochemistry, synthetic life
Artificial cell, expanded genetic code…

**Hard ALife** = hardware, robots & machines
Humanoids, *Strandbeest*…

**Soft ALife** = software, simulations
Cellular automata, virtual creatures…



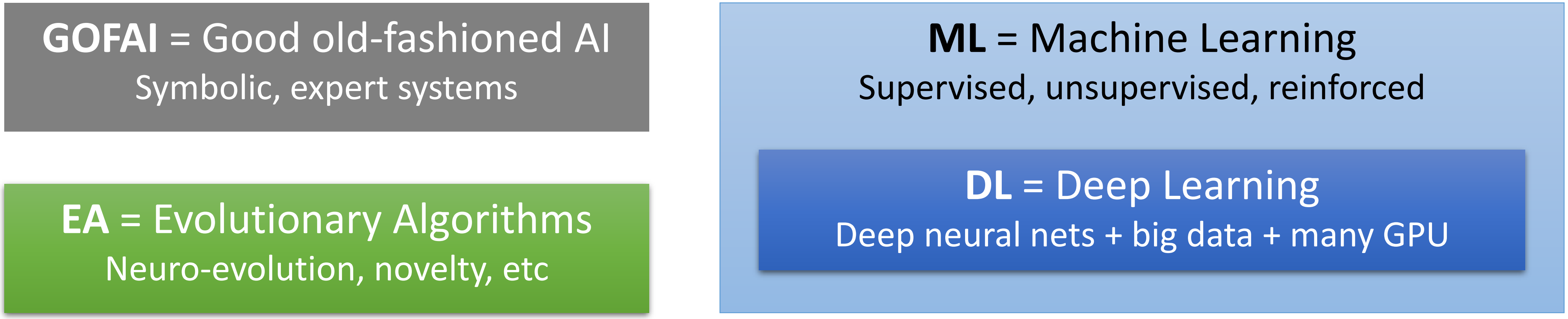
Virtual creatures (*Karl Sims*, 1994)



Soft robots (*Nick Cheney*, 2014)

# About AI

- **Artificial Intelligence**
  - Machines do: learning, planning, vision, language, emotion, art
  - → Create a mind



**GOFAI** = Good old-fashioned AI
Symbolic, expert systems

**EA** = Evolutionary Algorithms
Neuro-evolution, novelty, etc

**ML** = Machine Learning
Supervised, unsupervised, reinforced

**DL** = Deep Learning
Deep neural nets + big data + many GPU

# About AI

**GOFAI** = Good old-fashioned AI

Symbolic, expert systems

**ML** = Machine Learning

Supervised, unsupervised, reinforced

**DL** = Deep Learning

Deep neural nets + big data + many GPU

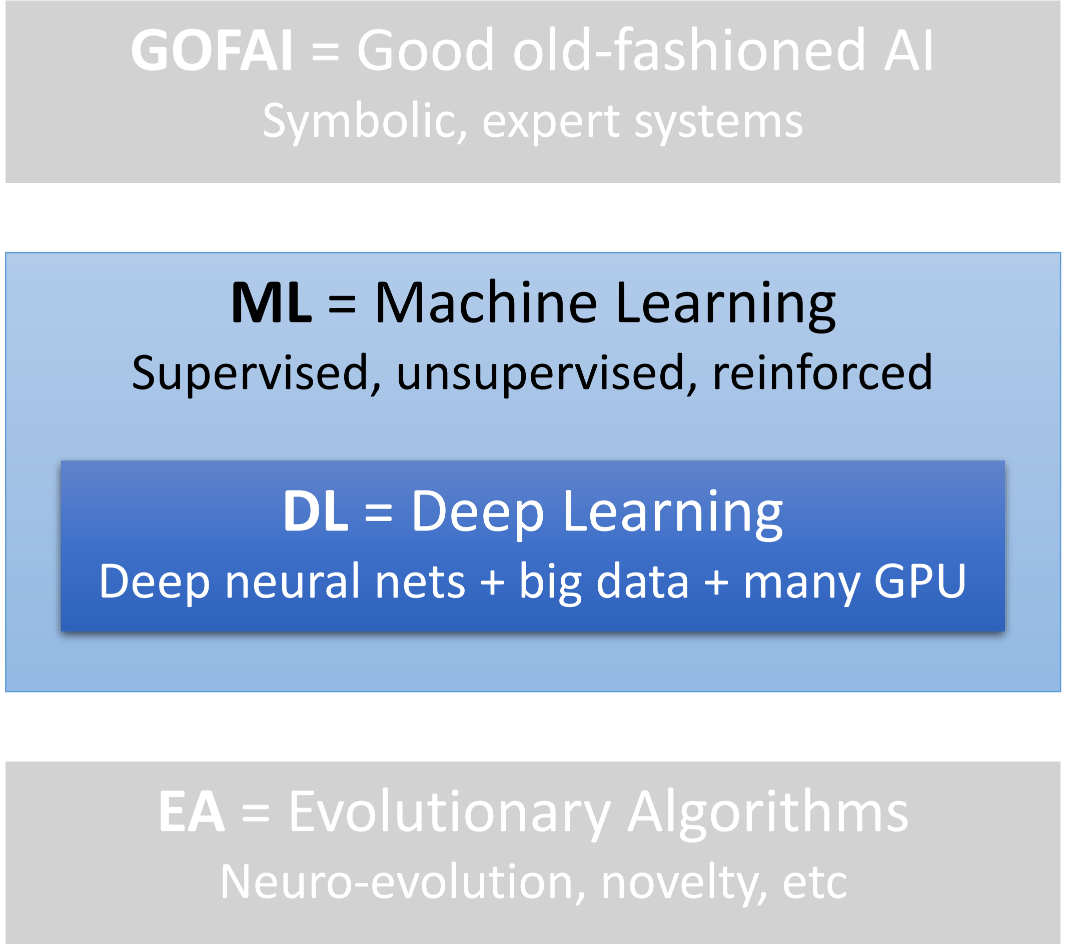
**EA** = Evolutionary Algorithms

Neuro-evolution, novelty, etc

Deep Blue vs. Kasparov (*IBM*, 1997)

Watson in Jeopardy! (*IBM*, 2011)

# About AI



**GOFAI** = Good old-fashioned AI

Symbolic, expert systems

**ML** = Machine Learning

Supervised, unsupervised, reinforced

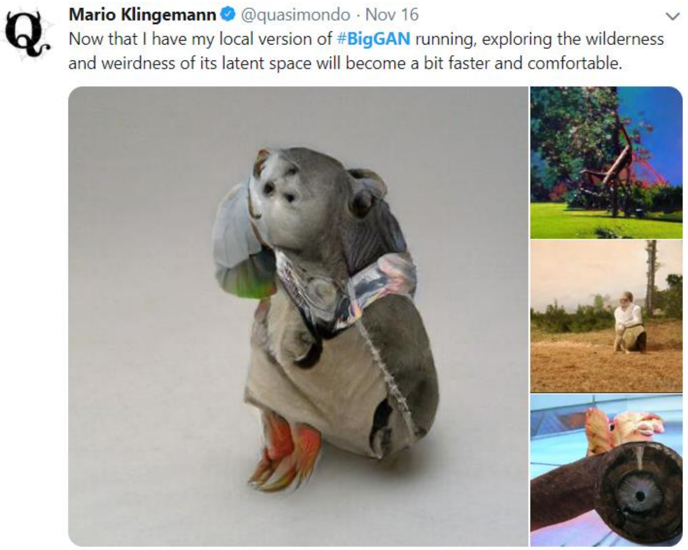
**DL** = Deep Learning

Deep neural nets + big data + many GPU

**EA** = Evolutionary Algorithms

Neuro-evolution, novelty, etc

AlphaGo vs. Lee Sedol (*DeepMind*, 2016)



**Mario Klingemann** @quasimondo · Nov 16

Now that I have my local version of #BigGAN running, exploring the wilderness and weirdness of its latent space will become a bit faster and comfortable.

BigGAN (*Andrew Brock*, 2018)

Autopilot (*Tesla*, 2014)

# About AI

**GOFAI** = Good old-fashioned AI

Symbolic, expert systems

**ML** = Machine Learning

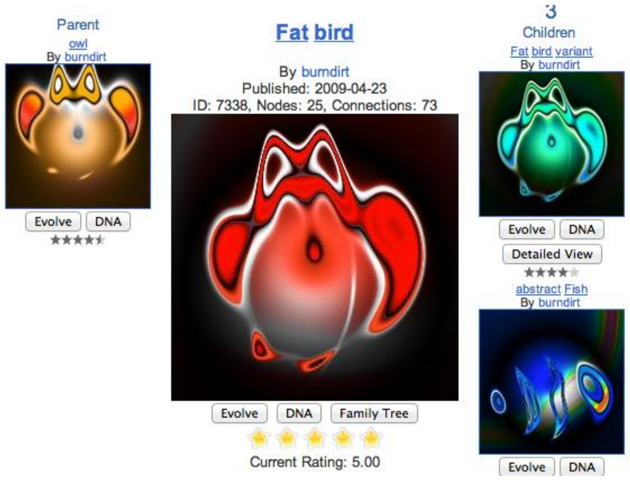
Supervised, unsupervised, reinforced

**DL** = Deep Learning

Deep neural nets + big data + many GPU

**EA** = Evolutionary Algorithms

Neuro-evolution, novelty, etc

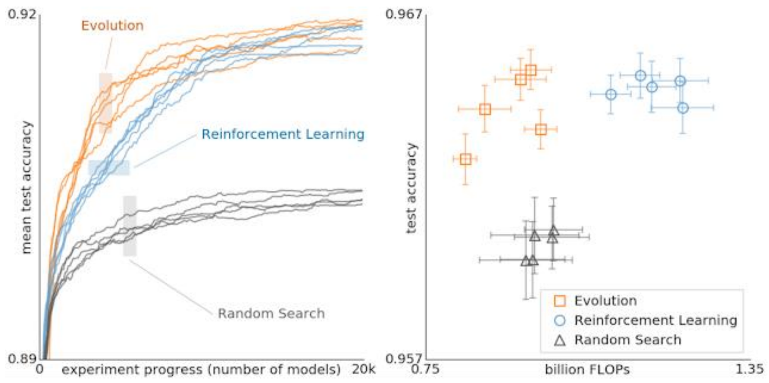


Published: 2009-04-23

ID: 7338, Nodes: 25, Connections: 73

Current Rating: 5.00

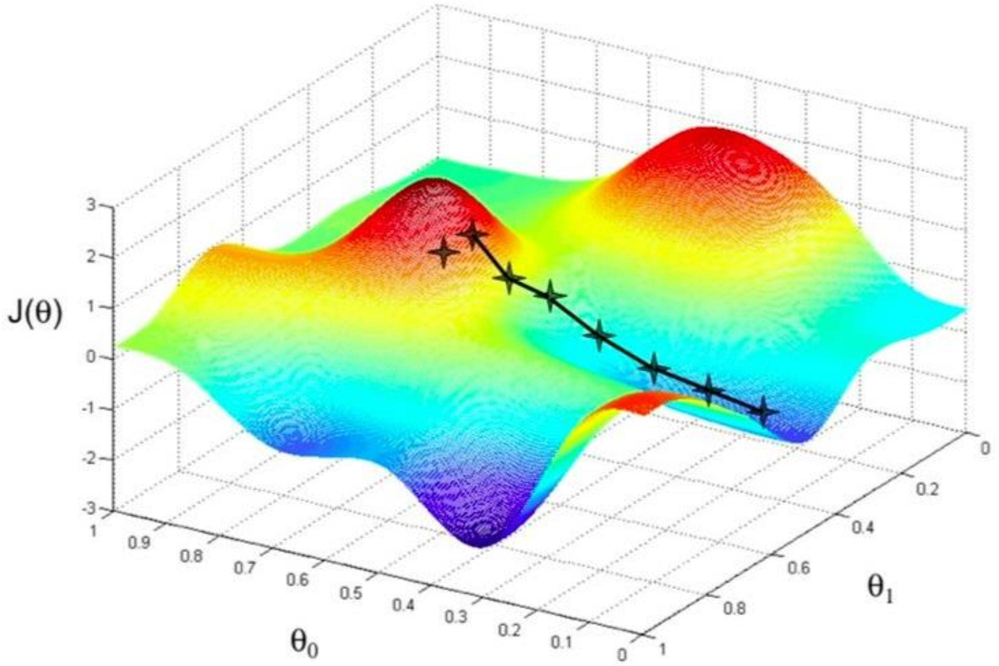
PicBreeder (*EPlex*, 2007)



Evolutionary AutoML (*Google Brain*, 2017)

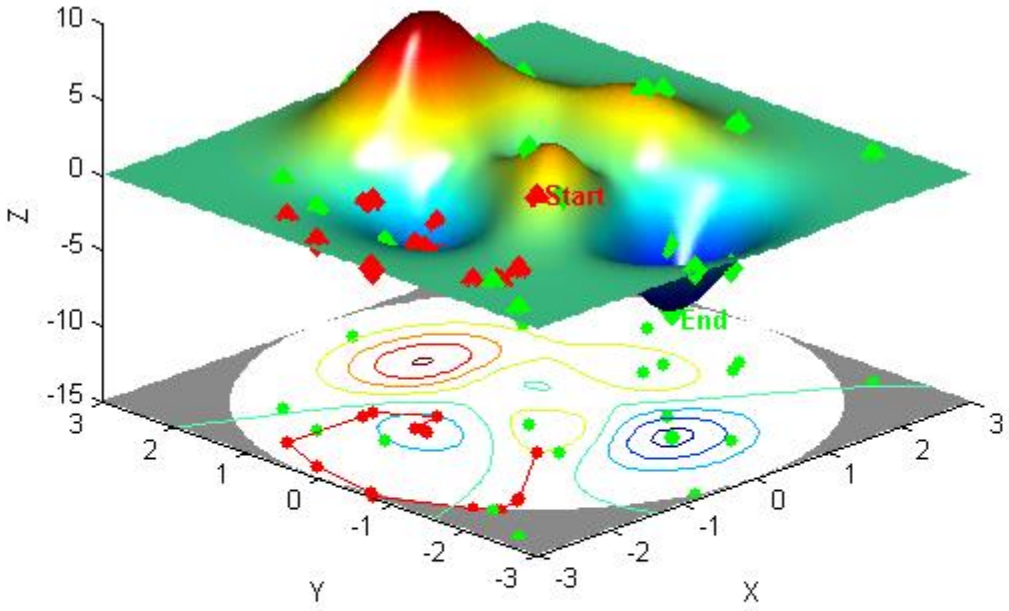
**DL = Deep Learning**

(Gradient Descent)



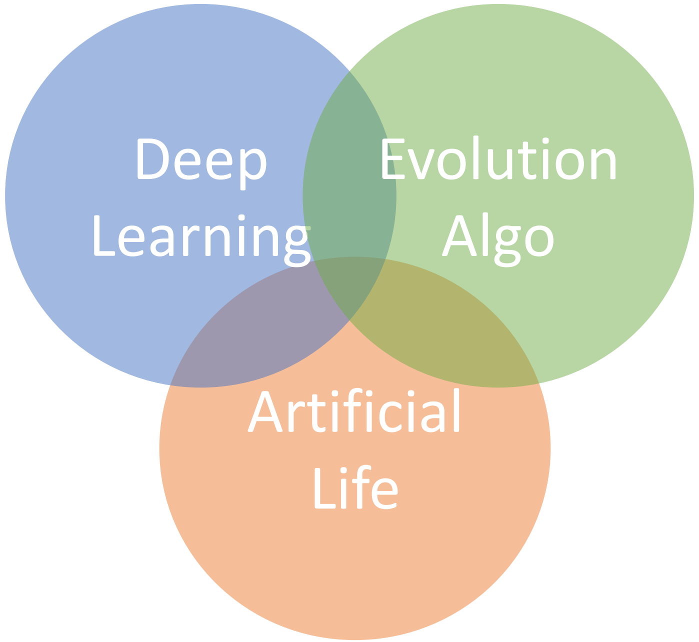
**EA = Evolutionary Algorithm**

(Natural Selection)



# Towards AGI

- Artificial **General** Intelligence
  - Sapience, sentience, consciousness
  - When?  How?  Should we?

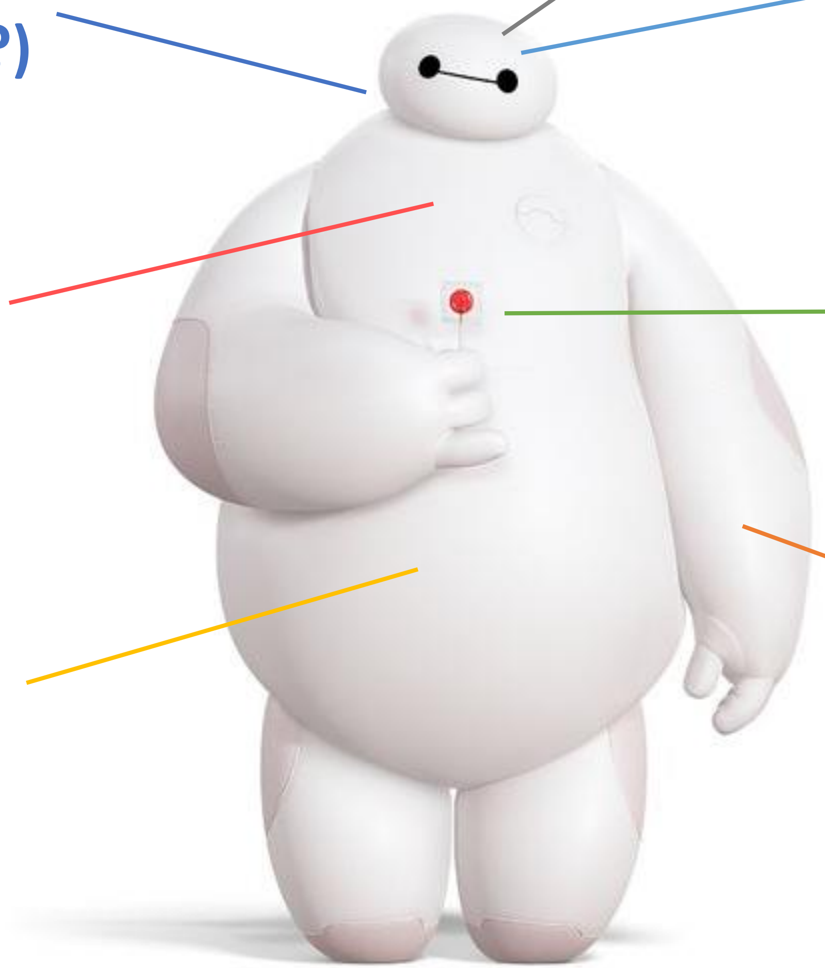


Deep Learning

Evolution Algo

Artificial Life

# Towards AGI



Consciousness?
Mind? Soul?

Knowledge, Reasoning
(Symbolic AI?)

Senses, Communication
(Deep Learning?)

Learning, Planning
(Reinforcement Learning?)

Emotions, Empathy
(Artificial Empathy)

Curiosity, Creativity
(Evolutionary Algorithms?)

Lenia

Safety, Ethics
(AI Safety)

Body, Actions
(Artificial Life)

# Use Lenia to…

- Understand evolution by writing programs?
- Teach AI to be curious and creative?
- Teach AI to understand life?

# Thank You!

Bert Chan
chakazul.github.io
@BertChakovsky