CAUSAL COGNITIVE ARCHITECTURE 3 (CCA3): A SOLUTION TO THE BINDING PROBLEM

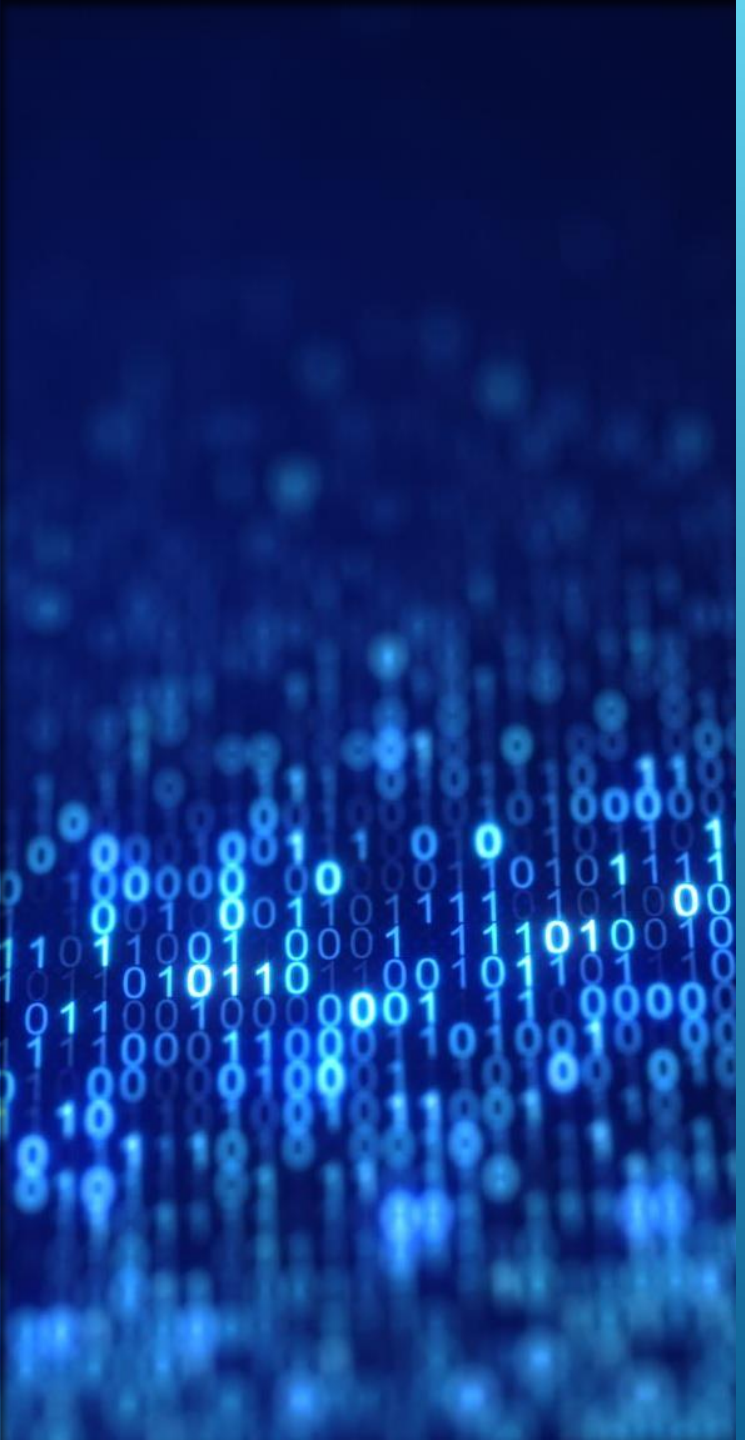
Howard Schneider

Sheppard Clinic North, Ontario, Canada

Cognitive Systems Research, in press
Supplementary Video File

GITHUB Username: "CausalCog"
https://github.com/CausalCog

VIDEO #2



- CCA3 Overview ✔
- Binding Problem Overview ✔
- Software Overview ⟵
- Operations Overview
- Operations Causal
- Software in More Detail
- More videos, code on GitHub "CausalCog"

(If interest, continued updating on GitHub)

SOFTWARE OPERATION LINKED TO "CCA3 – A SOLUTION TO BINDING PROBLEM" PAPER

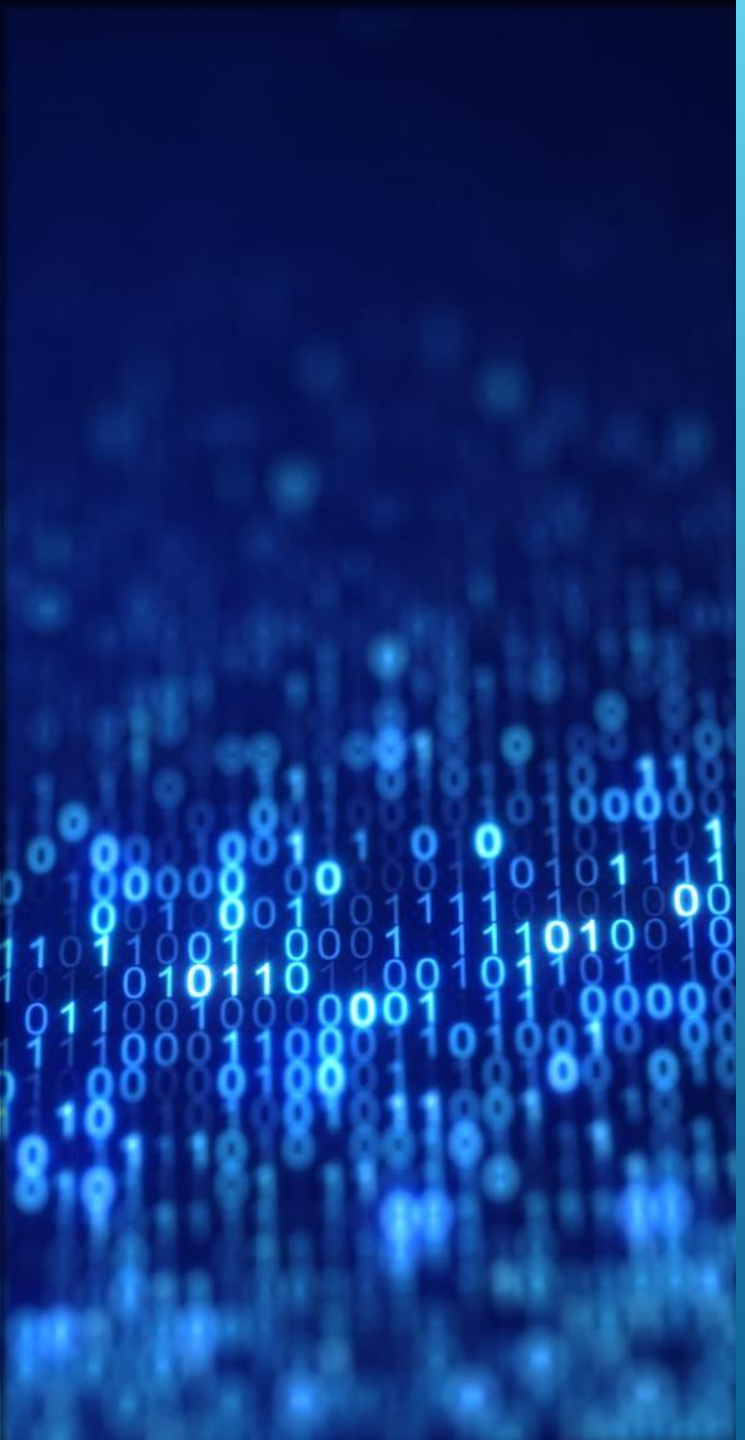
```
self.visual_inputs_zoom_out_assocn_module   ['patient,
self.visual_inputs_zoom_out_motion_modules   []
self.radar_inputs_assocn_module   []

These set of LNM's represent lnm(t) Equation 23
Also, at this time we have effectively extracted motio
as per equation 44  s'_series(t), although not the ful
```

SOFTWARE WILL RUN ON NORMAL LAPTOP

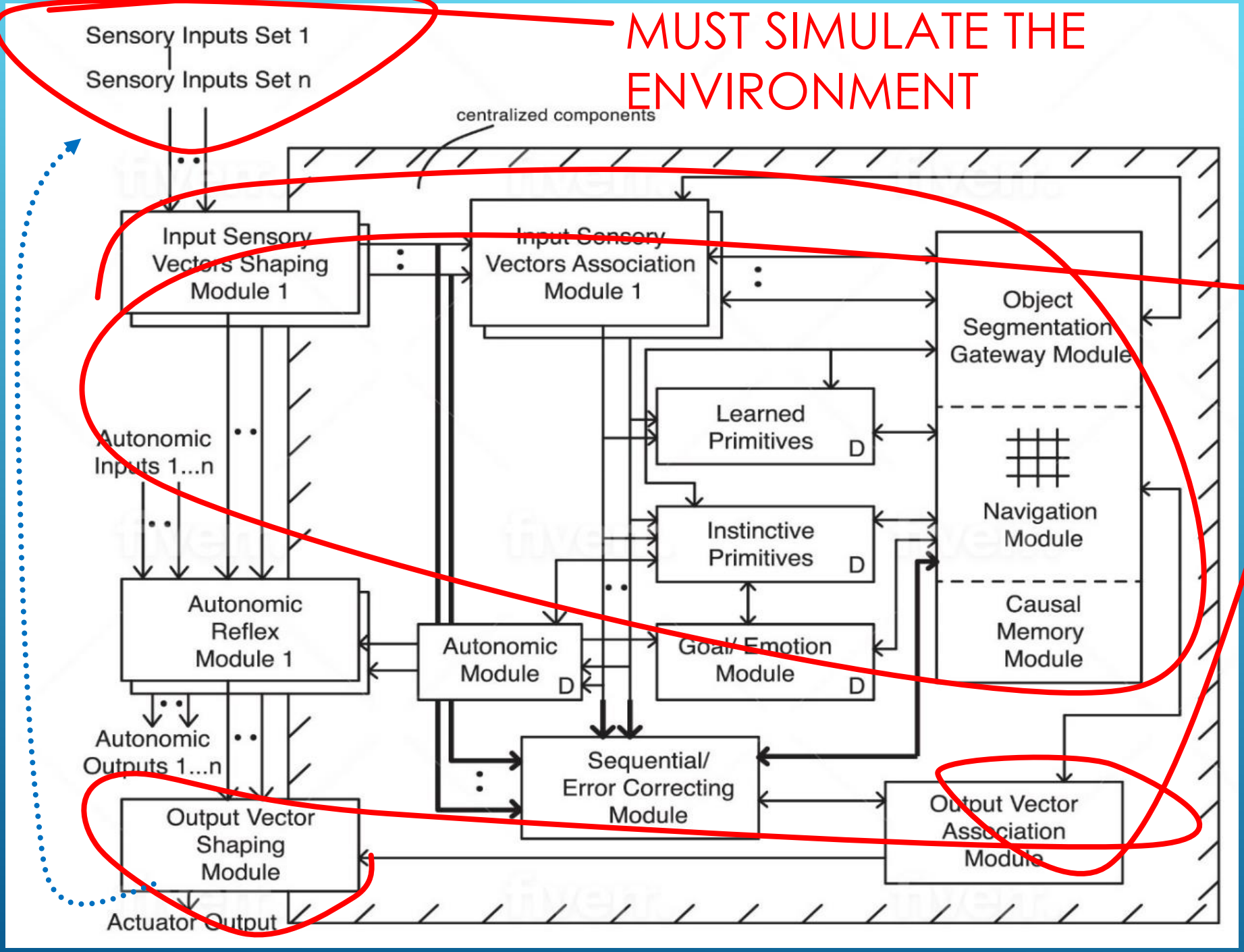
```
Causal Cognitive Architecture 4 (CCA4)
Sept 2021 rewrite for CSR Manuscript

-- Demonstrate architecture
-- Link to equations in the CSR Manuscript
-- Allow users to run on normal Win or Linux system without GPU
-- Purpose is to show reader what steps the Causal Cognitive Archite
   is taking, how it is accomplishing them, etc
```



BRIEF OVERVIEW:

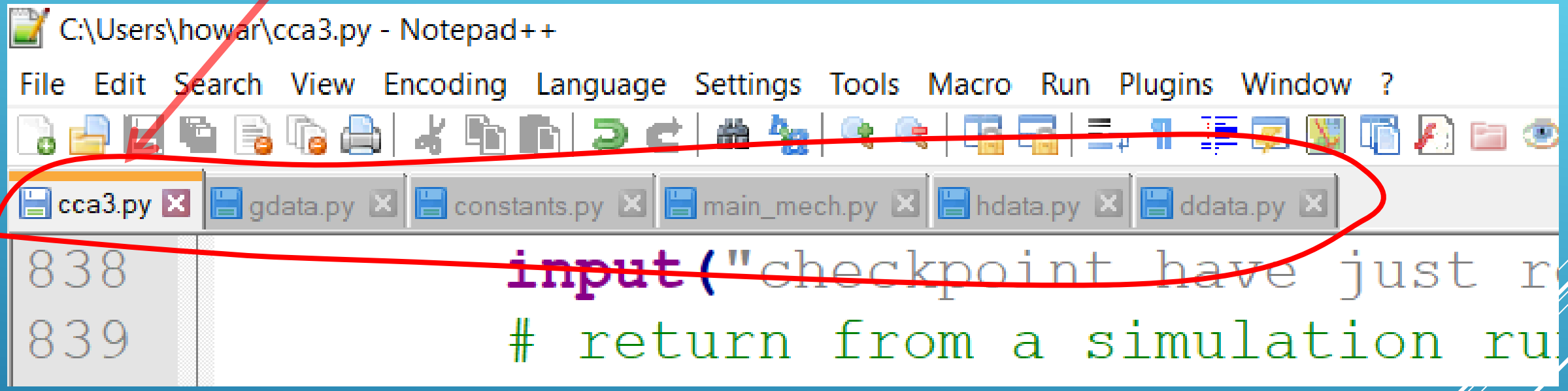
CCA3 SIMULATION SOFTWARE – STRUCTURE



MUST SIMULATE THE CCA3

MUST SIMULATE THE OUTPUTS AND THE EFFECT ON THE ENVIRONMENT

CCA3 (and environment) Simulation Code



Note: the code to simulate **both** the environment **and** to simulate the CCA3 architecture is in these same modules.

main_eval()

- Top level simulation cycles
- Enter hyperparameters

```
830     for g.mission_counter in range(1, LIFE
831         h, m = choose_simulation(g, h, m)
832         d = choose_starting_scene(d, g, h)
833         d, g, h, m = main_mech.cycles(d, g
834         print_event_log_memory(g)
835         if not run_again():
836             break
837     exit_program(g)
```

Each “mission” or “run” we load in a set of sensory scenes corresponding to some environment (e.g., taking care of patient in hospital room, e.g., playing game of sudoku

```
830     for g.mission_counter in range(1, LIFE
831         h, m = choose_simulation(g, h, m)
832         d = choose_starting_scene(d, g, h)
833         d, g, h, m = main_mech.cycles(d, g
834         print_event_log_memory(g)
835         if not run_again():
836             break
837     exit_program(g)
```

```
Please choose type of "hippocampus"/"brain" which, of course
approximates the biological equivalent (you are effectively
0. SAME AS LAST ENVIRONMENT, DO NOT ERASE/REFURBISH THE MEMO
1. Lamprey-like brain analogue
2. Fish-like brain
3. Reptile-like brain
4. Mammalian-like brain - note: meaningfulness, precausal
5. Human-like brain - note: meaningfulness plus full causal
6. Augmented Human level 1 - simultaneous application of mul
7. Augmented Human level 2 - enhanced generative abilities
```

```
for g.mission_counter in range(1, LIFE
    h, m = choose_simulation(g, h, m)
    d = choose_starting_scene(d, g, h)
```

“main()” of CCA3

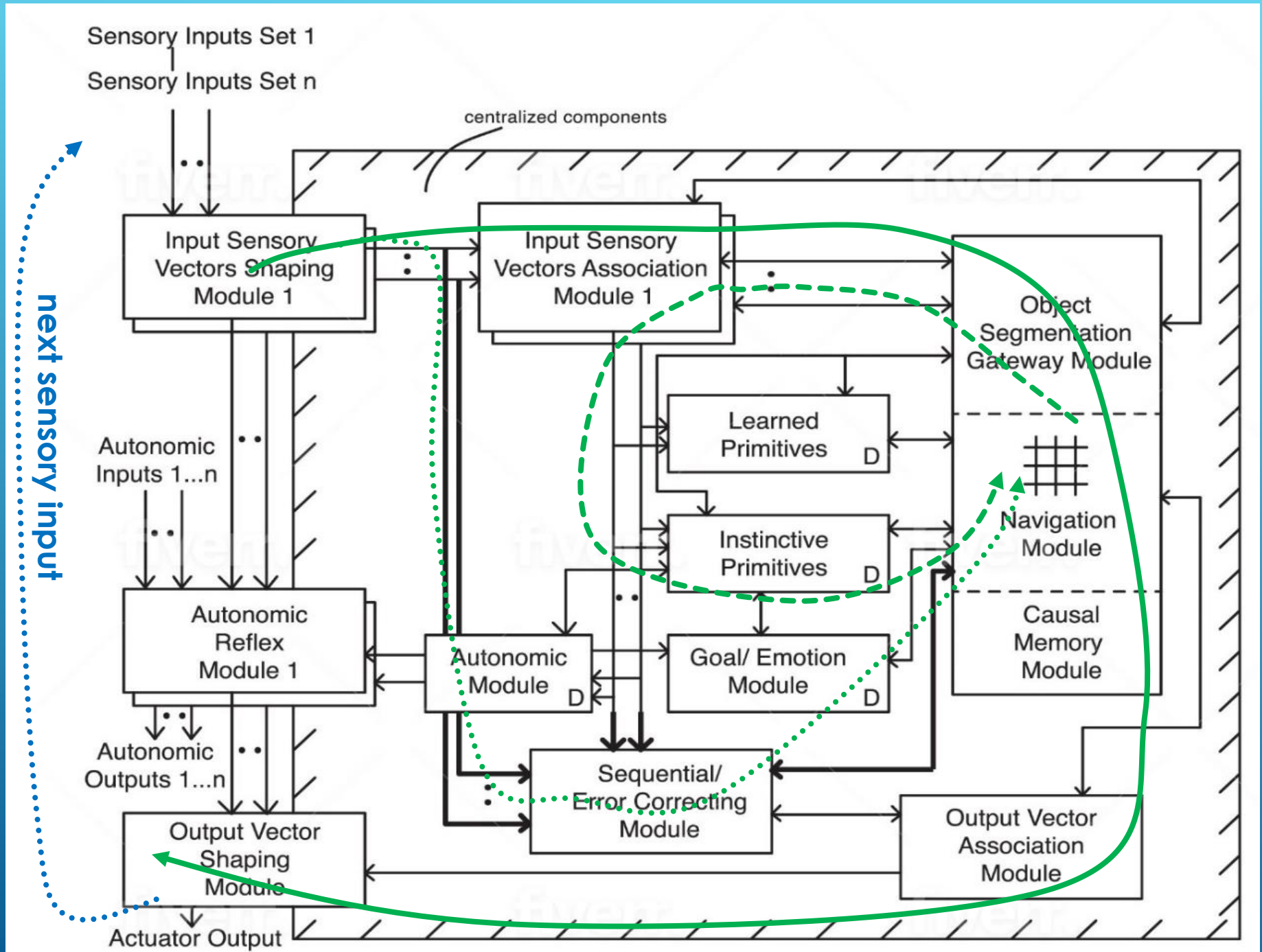
```
830     for g.mission_counter in range(1, LIFE
831         h, m = choose_simulation(g, h, m)
832         d = choose_starting_scene(d, g, h)
833         d, g, h, m = main_mech.cycles(d, g
834         print_event_log_memory(g)
835         if not run_again():
836             break
837     exit_program(g)
```

main_mech.cycles()

```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)
        h.navigation_module(d, g)
        h.output_vector_assocation_module
        if (next_scene_from_envrt < 0 or
            d, g, h = update_expected_val
            return d, g, h, m
```

“cycle” == “processing cycle”

```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)
        h.navigation_module(d, g)
        h.output_vector_assocation_module
        if (next_scene_from_envrt < 0 or
            d, g, h = update_expected_val
            return d, g, h, m
```



"cycle" == "processing cycle"

Equations 3 – 92 each "cycle"

```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)
        h.navigation_module(d, g)
        h.output_vector_assocation_module
        if (next_scene_from_envrt < 0 or
            d, g, h = update_expected_val
            return d, g, h, m
```

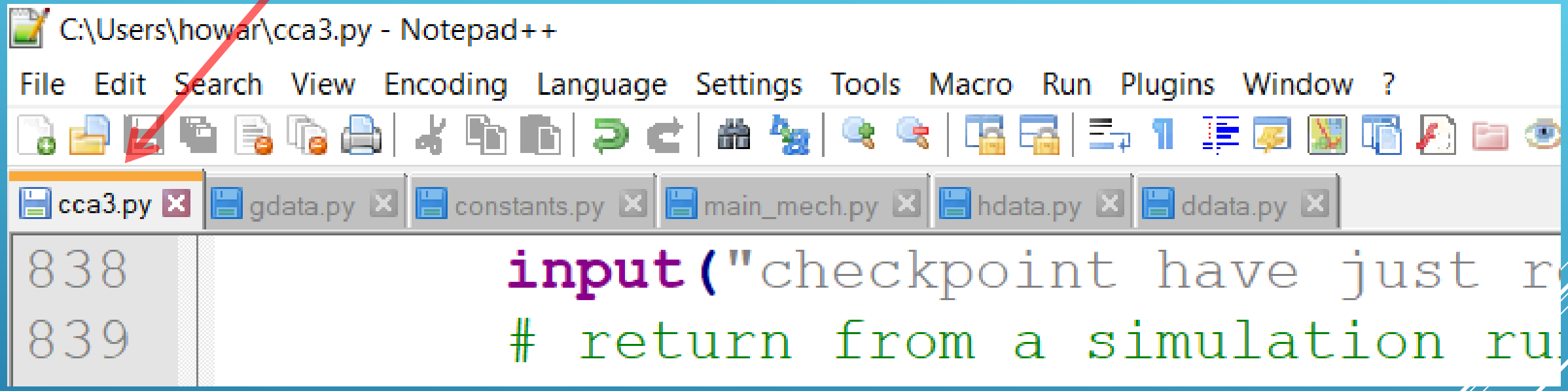
```
Sensory inputs from a simulated environment have been fed i
CCA3 Input Sensory Vectors Shaping Modules which in turn fe
and the Input Sensory Vectors Associations Modules.
EQUATIONS 3 - 13....
Sensory systems defined: visual_far, visual_close, auditory
EQ #13 s'(t) output here, albeit as labelled groups in prog
Effectively we now have transformed the input sensory data
```

```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt
        h.input_sensory_vectors_associat
```

Where actually is code?

```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)
        h.navigation_module(d, g)
        h.output_vector_assocation_module
        if (next_scene_from_envrt < 0 or
            d, g, h = update_expected_val
            return d, g, h, m
```

CCA3 (and environment) Simulation Code



https://github.com/CausalCog

- no GPU needed, will run on ordinary Windows (Linux) laptop
- → code *will* run in Python envr't
- linked to equations in paper "CCA3 – A Solution to the Binding Problem"

requirements.txt + Python 3.9 environment

Note: “cca4.py” replaces old “cca3.py”

cca4.py	main_eval() + others	enter hyperparameters
main_mech.py	cycles() + others	run code for “processing cycles”
hdata.py	h == NavMod()	cca3 data structures and methods
ddata.py	d == MapData()	helper methods and deprecated code
gdata.py	g == MultSessionsData()	helper methods and deprecated code
constants.py	constants	initiation and constant values
other files	e.g., .jpg files, .txt files	associated data files that are needed
import xx	pypi.org imports	much of code is from python libraries

Data Structures

$$\mathbf{NM_{mapno}} \in R^{mxnxo}, \mathbf{IPM_{mapno}} \in R^{mxnxo}, \mathbf{LPM_{mapno}} \in R^{mxnxo} \quad (24)$$

$$\mathbf{\Theta\_NM} := \text{total NM's} \in N, \ \mathbf{\Theta\_IPM} := \text{total IPM's} \in N, \ \mathbf{\Theta\_LPM} := \text{total LPM's} \in N \quad (25)$$

$$\boldsymbol{all\_LNMs_t} := [\boldsymbol{all\_maps_{1,t}}, \boldsymbol{all\_maps_{2,t}}, \boldsymbol{all\_maps_{3,t}}, \ldots, \boldsymbol{all\_maps_{n\_\sigma,t}}] \quad (26)$$

$$\boldsymbol{all\_NMs_t} := [\mathbf{NM_{1,t}}, \mathbf{NM_{2,t}}, \mathbf{NM_{3,t}}, \ldots, \mathbf{NM_{\Theta\_NM,t}}] \quad (27)$$

$$\boldsymbol{all\_IPMs_t} := [\mathbf{IPM_{1,t}}, \mathbf{IPM_{2,t}}, \mathbf{IPM_{3,t}}, \ldots, \mathbf{IPM_{\Theta\_IPM,t}}] \quad (28)$$

$$\boldsymbol{all\_LPMs_t} := [\mathbf{LPM_{1,t}}, \mathbf{LPM_{2,t}}, \mathbf{LPM_{3,t}}, \ldots, \mathbf{LPM_{\Theta\_LPM,t}}] \quad (29)$$

$$\boldsymbol{all\_navmaps_t} := [\boldsymbol{all\_LNMs_t}, \boldsymbol{all\_NMs_t}, \boldsymbol{all\_IPMs_t}, \boldsymbol{all\_LPMs_t}] \quad (30)$$

self.gb
== h.gb

self.gb[] stores navigation maps

```
329         self.gb = np.empty((self.total_maps, 6,
330         # self.gb = np.empty((1000,6,6,6,11), dt
331         # gb[n,x,y,z,s]
```

features within a cube

map number

x,y,z cube within a given navigation map

self.gb[] – primitives are also in navmap's

```
eg, self.gb[n,0,0,0,10] = 'instinctive' means this is an instinctive primit
eg, self.gb[n,0,0,0,10] = 'navmap' means this is an multisensory navigatior
```

Data Structures

$$\mathbf{S_1} \in R^{m\_1xn\_1xo\_1} \quad (3)$$

$$\mathbf{S_{1,t}} := \text{visual inputs(t)} \quad (4)$$

$$\mathbf{S_2} \in R^{m\_2xn\_2xo\_2} \quad (5)$$

$$\mathbf{S_{2,t}} := \text{auditory inputs(t)} \quad (6)$$

$$\mathbf{S_3} \in R^{m\_3xn\_3xo\_3} \quad (7)$$

$$\mathbf{S_{3,t}} := \text{olfactory inputs(t)} \quad (8)$$

$$\sigma := \text{sensory system identification code} \in N \quad (9)$$

$$n\_\sigma := \text{total number of sensory systems} \in N \quad (10)$$

$$s(t) = [\, \mathbf{S_{1,t}}, \mathbf{S_{2,t}}, \mathbf{S_{3,t}}, \ldots, \mathbf{S_{n\_\sigma,t}}] \quad (11)$$

self.ext
== h.ext

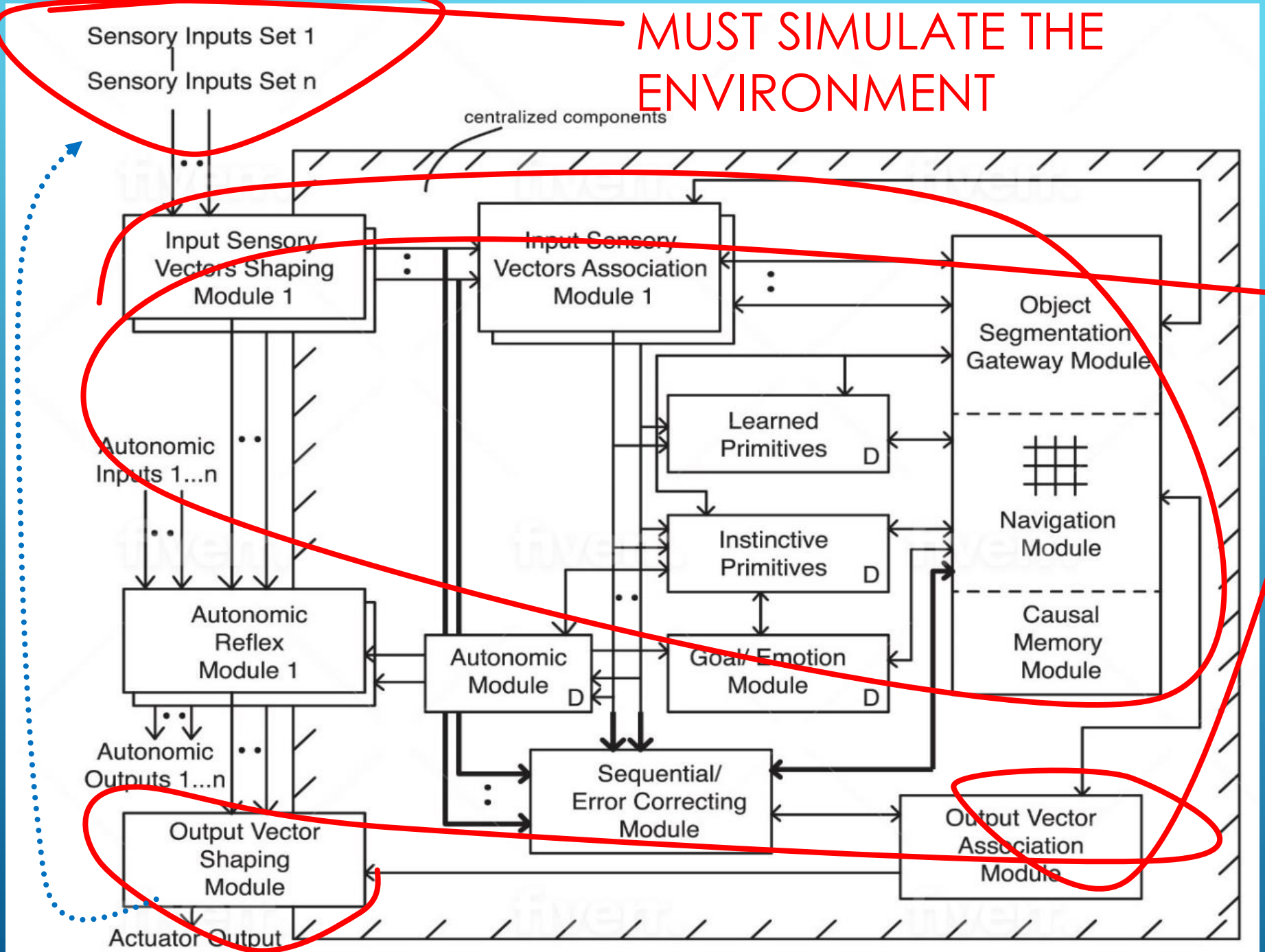
self.ext[] – since no actual inputs, we need to simulate the external world's sensory inputs

```
363             self.ext = np.empty((self.total_envir
364             # [environment_number, scene_number x
```

```
elf.total_environments, self.total_scenes, self.tota
 scene_number x20, sensory_stream_number x20]
```

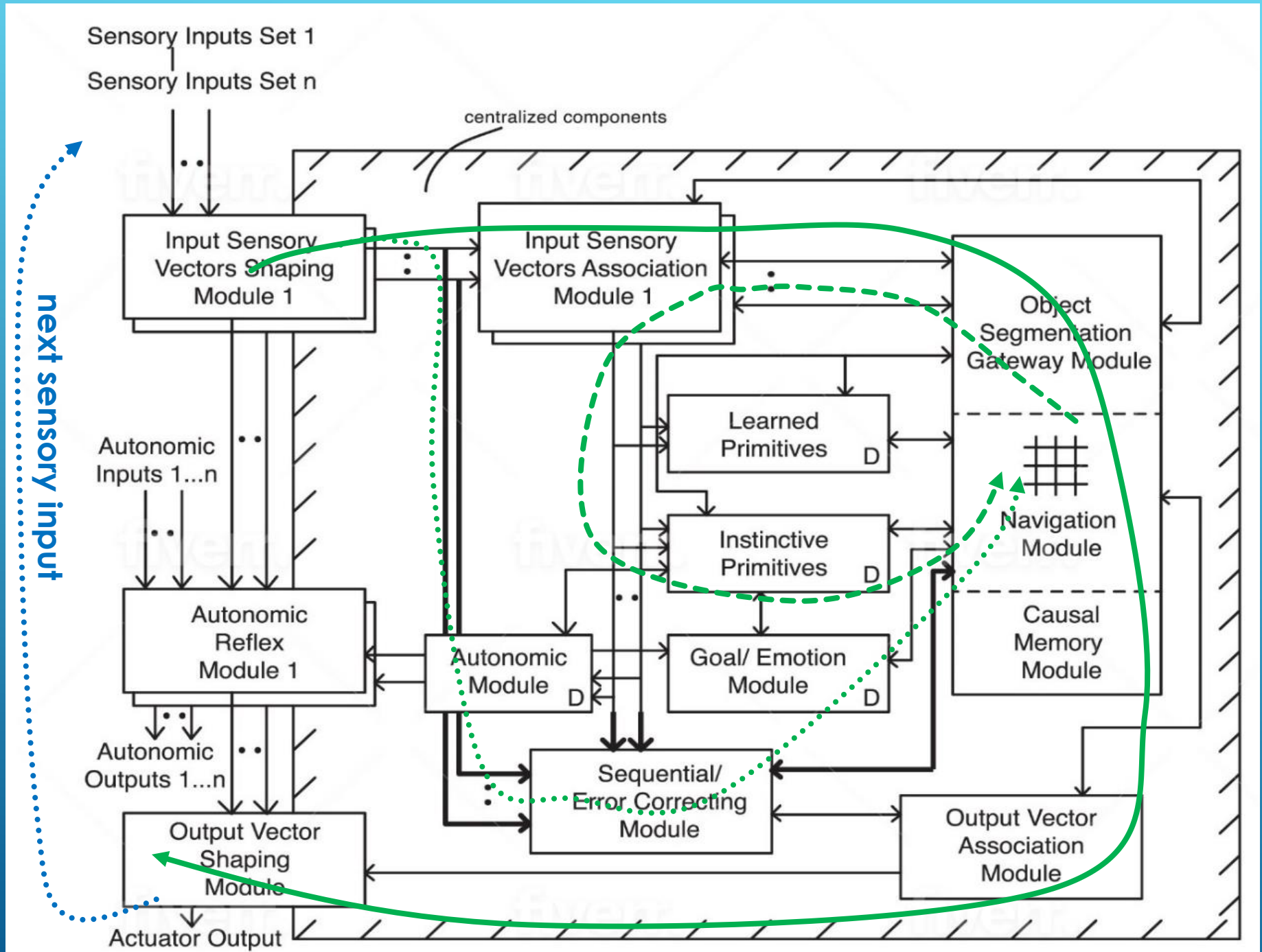
```
self.ext[2, 0, 0] = "visual_far:patient,walker, patien
self.ext[2, 0, 1] = "visual_close:left_hand>walker"
self.ext[2, 0, 2] = "visual_close:right_hand>walker"
```

Recap



MUST SIMULATE THE CCA3

MUST SIMULATE THE OUTPUTS AND THE EFFECT ON THE ENVIRONMENT



"cycle" == "processing cycle"

main_eval()

- Top level simulation cycles
- Enter hyperparameters

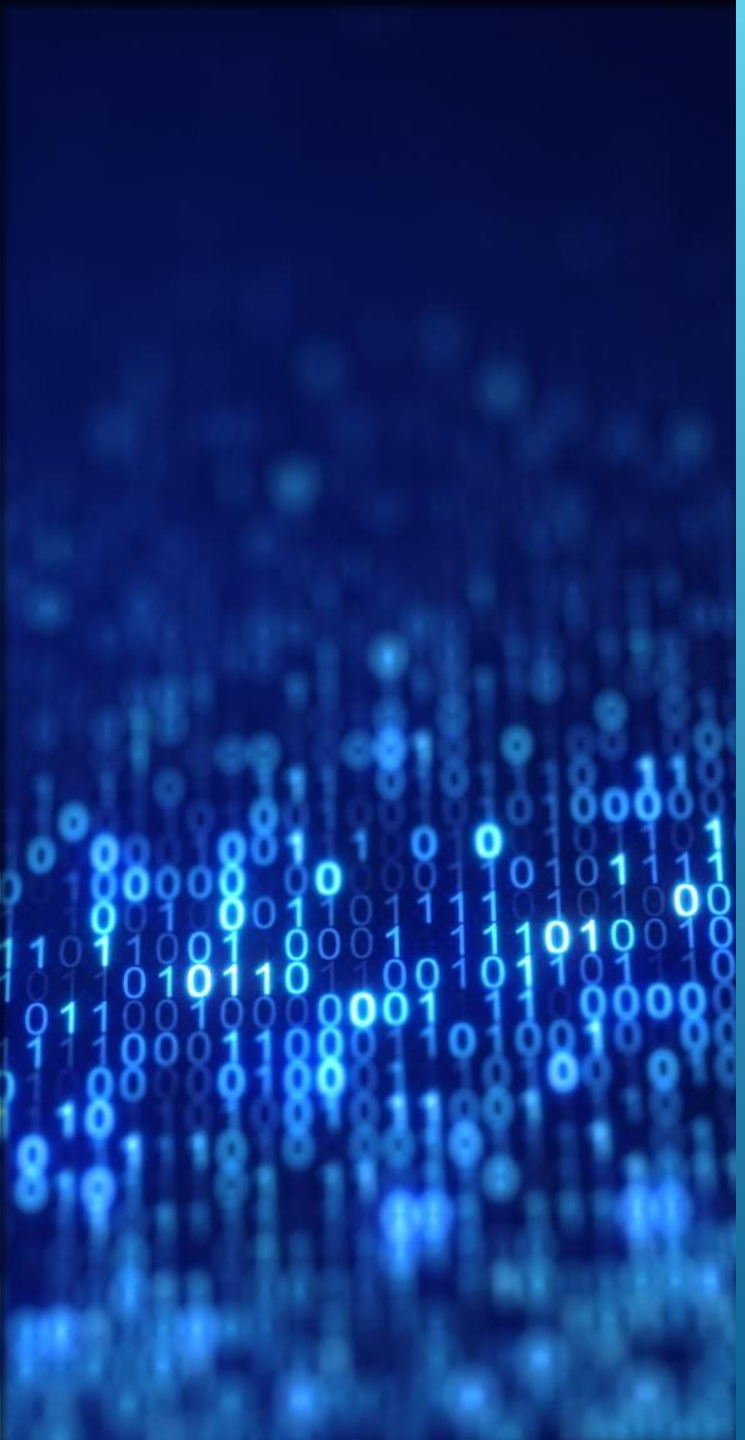
```
830     for g.mission_counter in range(1, LIFE
831         h, m = choose_simulation(g, h, m)
832         d = choose_starting_scene(d, g, h)
833         d, g, h, m = main_mech.cycles(d, g
834         print_event_log_memory(g)
835         if not run_again():
836             break
837     exit_program(g)
```

main_mech.cycles()

```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)
        h.navigation_module(d, g)
        h.output_vector_assocation_module
        if (next_scene_from_envrt < 0 or
            d, g, h = update_expected_val
            return d, g, h, m
```

```
les(d, g, h, m):
->SENSORY INPUTS -> CCA3 -> MOTOR
 d.evaluation_cycles in range(sys.
 autonomic_check(g)
 next_scene_from_envrt = (h.envrt_
 h.input_sensory_vectors_associati
 h.sequential_error_correcting_mod
 h.object_segmentation_module(g)
 h.navigation_module(d, g)
 h.output_vector_assocation_module
 if (next_scene_from_envrt < 0 or
     d, g, h = update_expected_val
     return d, g, h, m
```

Now, let's look at each of these methods that occur each cycle, and how they relate to the equations of the paper....



BRIEF OVERVIEW:

CCA3 SIMULATION SOFTWARE – OPERATION

….continued in VIDEO 3

balloon from powerpoint stock image