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

Howard Schneider

Sheppard Clinic North, Ontario, Canada

*Cognitive Systems Research, in press*

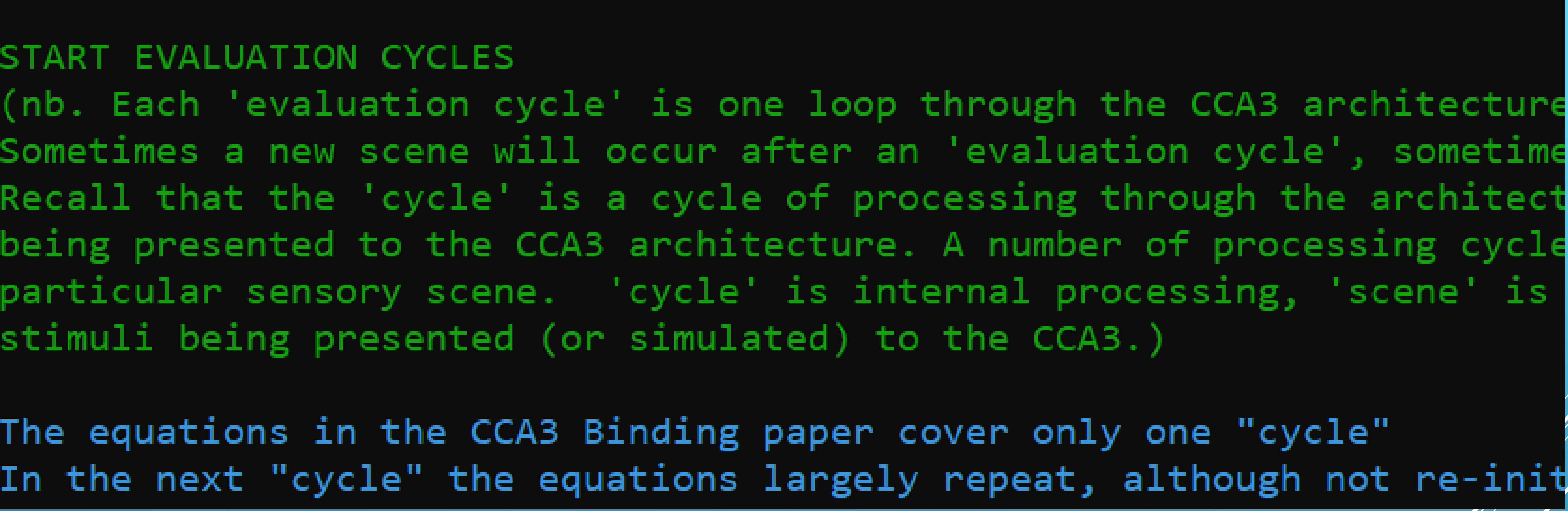
Supplementary Video File

GITHUB Username: “CausalCog”
https://github.com/CausalCog

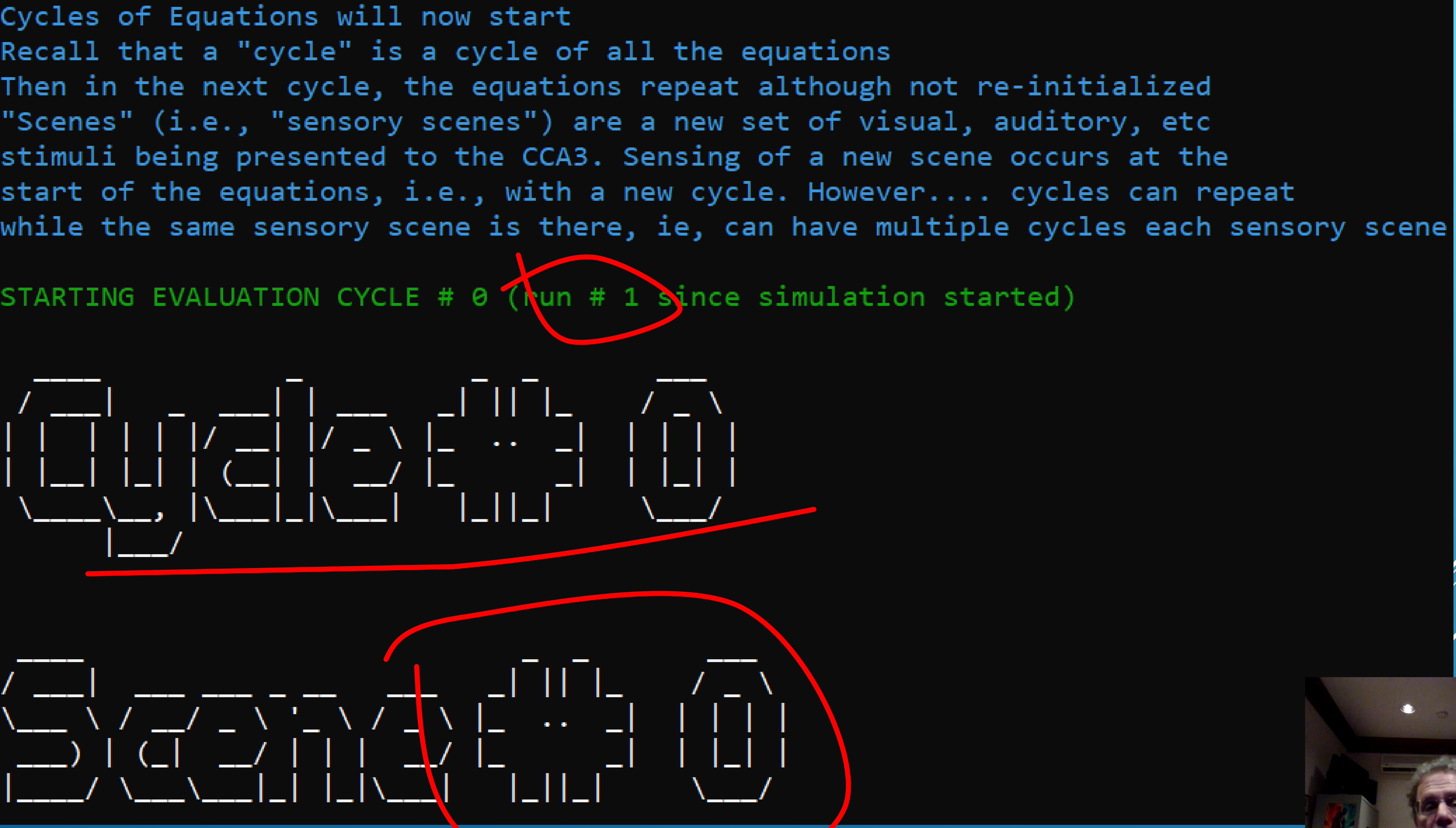
VIDEO #4

- 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)

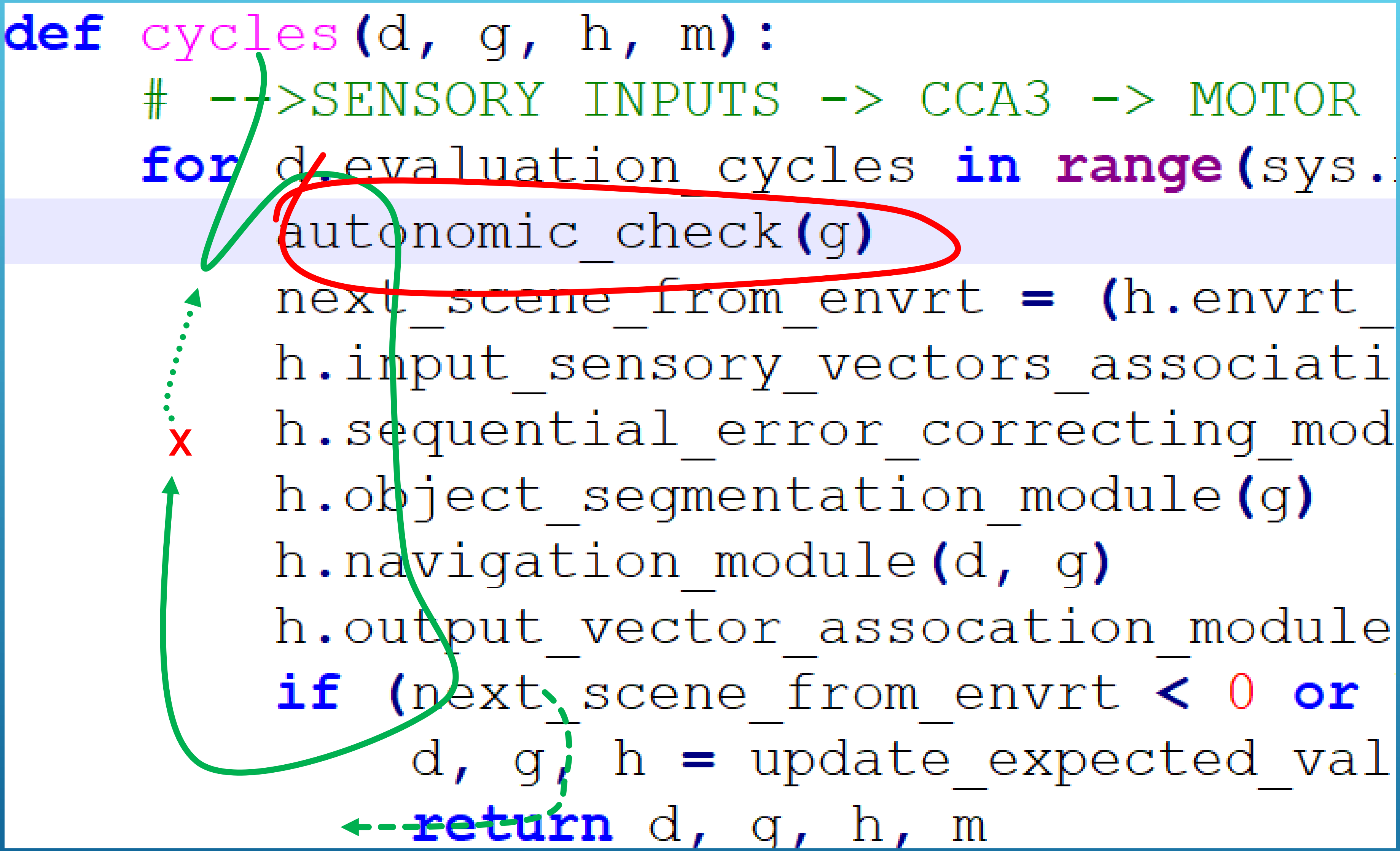


```
START EVALUATION CYCLES
(nb. Each 'evaluation cycle' is one loop through the CCA3 architecture
Sometimes a new scene will occur after an 'evaluation cycle', sometime
Recall that the 'cycle' is a cycle of processing through the architect
being presented to the CCA3 architecture. A number of processing cycle
particular sensory scene.  'cycle' is internal processing, 'scene' is
stimuli being presented (or simulated) to the CCA3.)

The equations in the CCA3 Binding paper cover only one "cycle"
In the next "cycle" the equations largely repeat, although not re-init
```

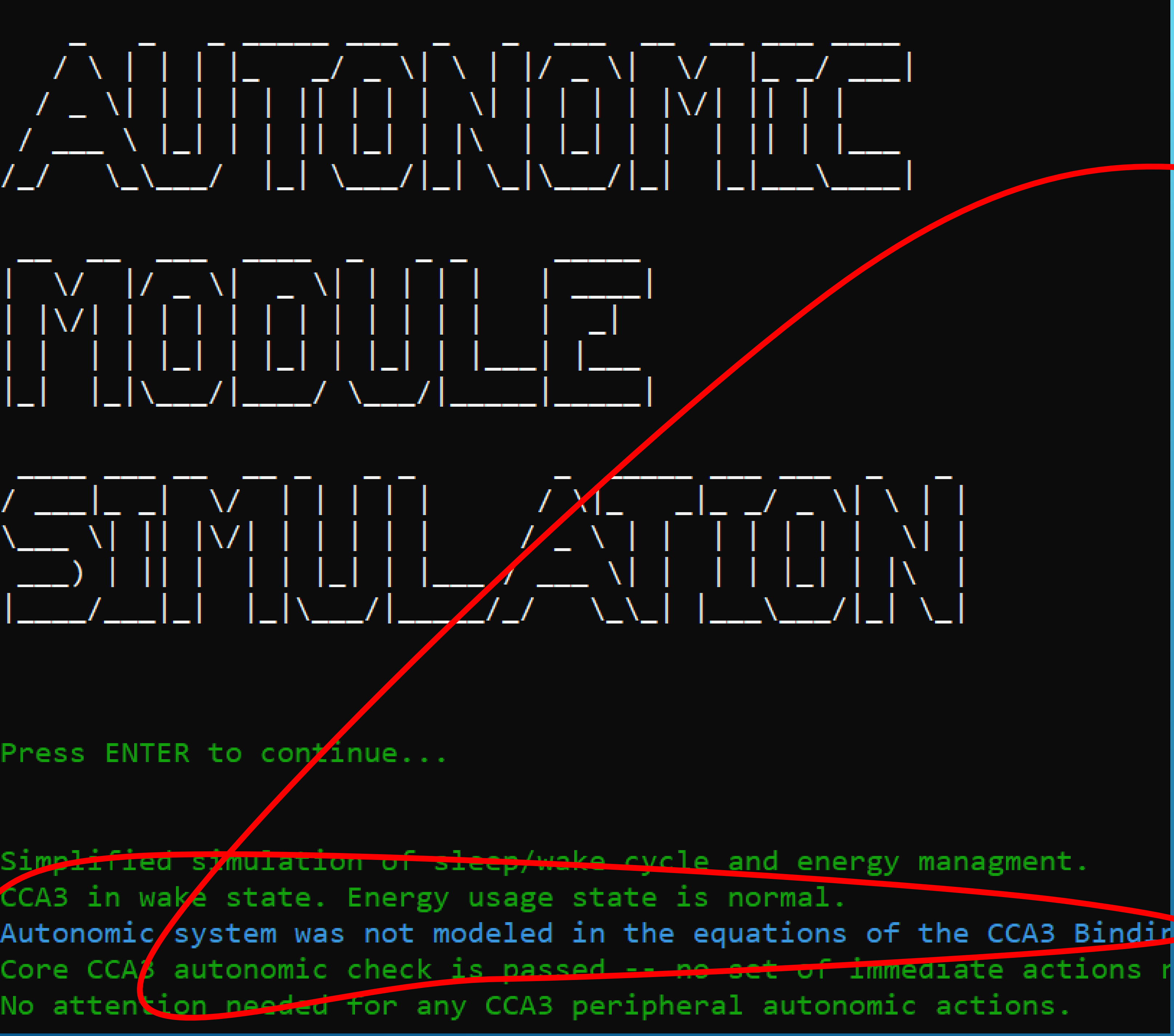


```
Cycles of Equations will now start
Recall that a "cycle" is a cycle of all the equations
Then in the next cycle, the equations repeat although not re-initialized
"Scenes" (i.e., "sensory scenes") are a new set of visual, auditory, etc
stimuli being presented to the CCA3. Sensing of a new scene occurs at the
start of the equations, i.e., with a new cycle. However.... cycles can repeat
while the same sensory scene is there, ie, can have multiple cycles each sensory scene

STARTING EVALUATION CYCLE # 0 (run # 1 since simulation started)

  ____           _        _  _      ___
 / ___|   _  ___| | ___ _| || |_   / _ \
| |  | | | |/ __| |/ _ \_  ..  _| | | | |
| |__| |_| | (__| |  __/_      _| | |_| |
 \____\__, |\___|_|\___| |_||_|    \___/
      |___/


 ____                       _  _      ___
/ ___|  ___ ___ _ __   ___ | || |_   / _ \
\___ \ / __/ _ \ '_ \ / _ \_  ..  _| | | | |
 ___) | (_|  __/ | | |  __/_      _| | |_| |
|____/ \___\___|_| |_|\___| |_||_|    \___/
```

# main_mech.cycles()



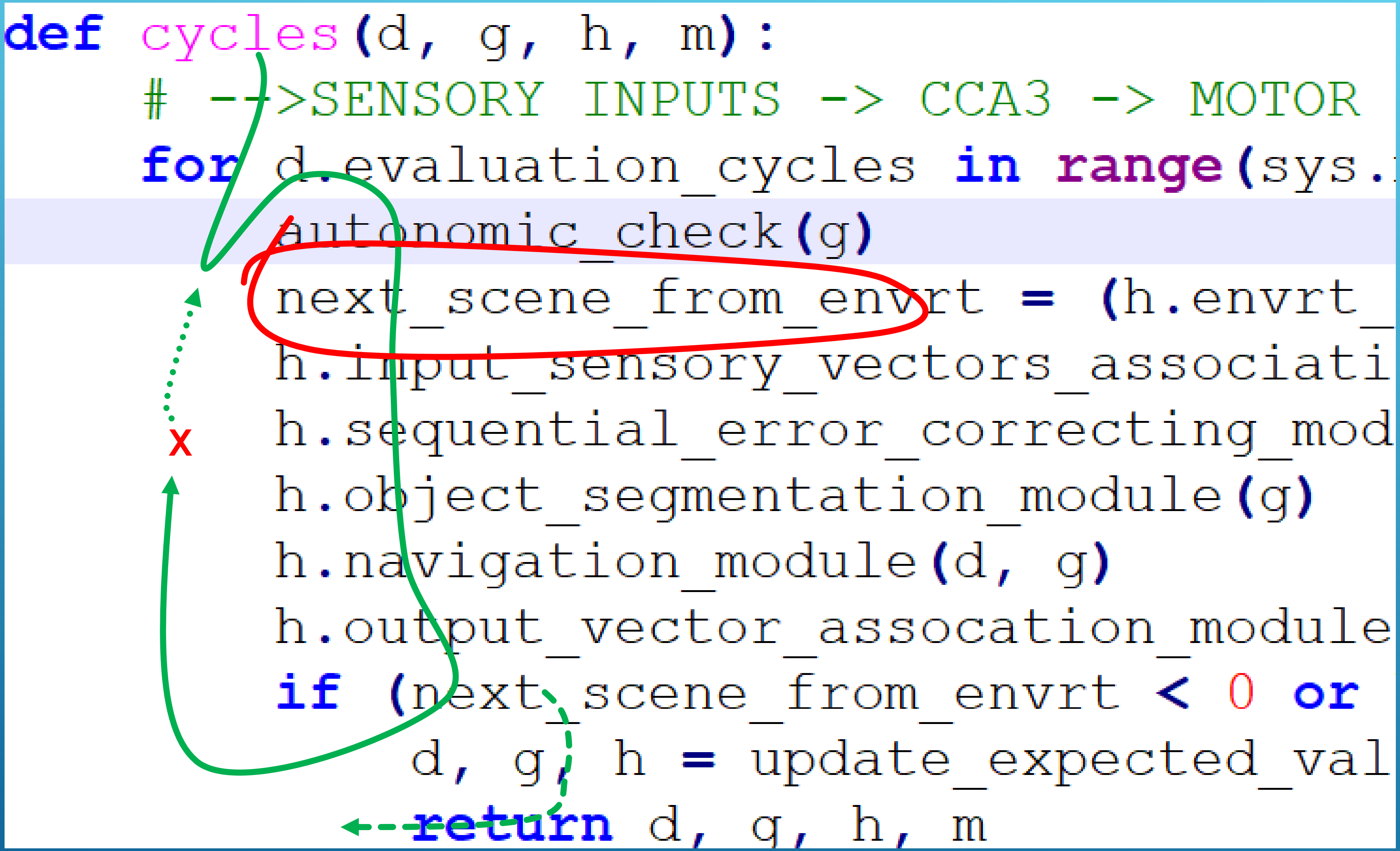
```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)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



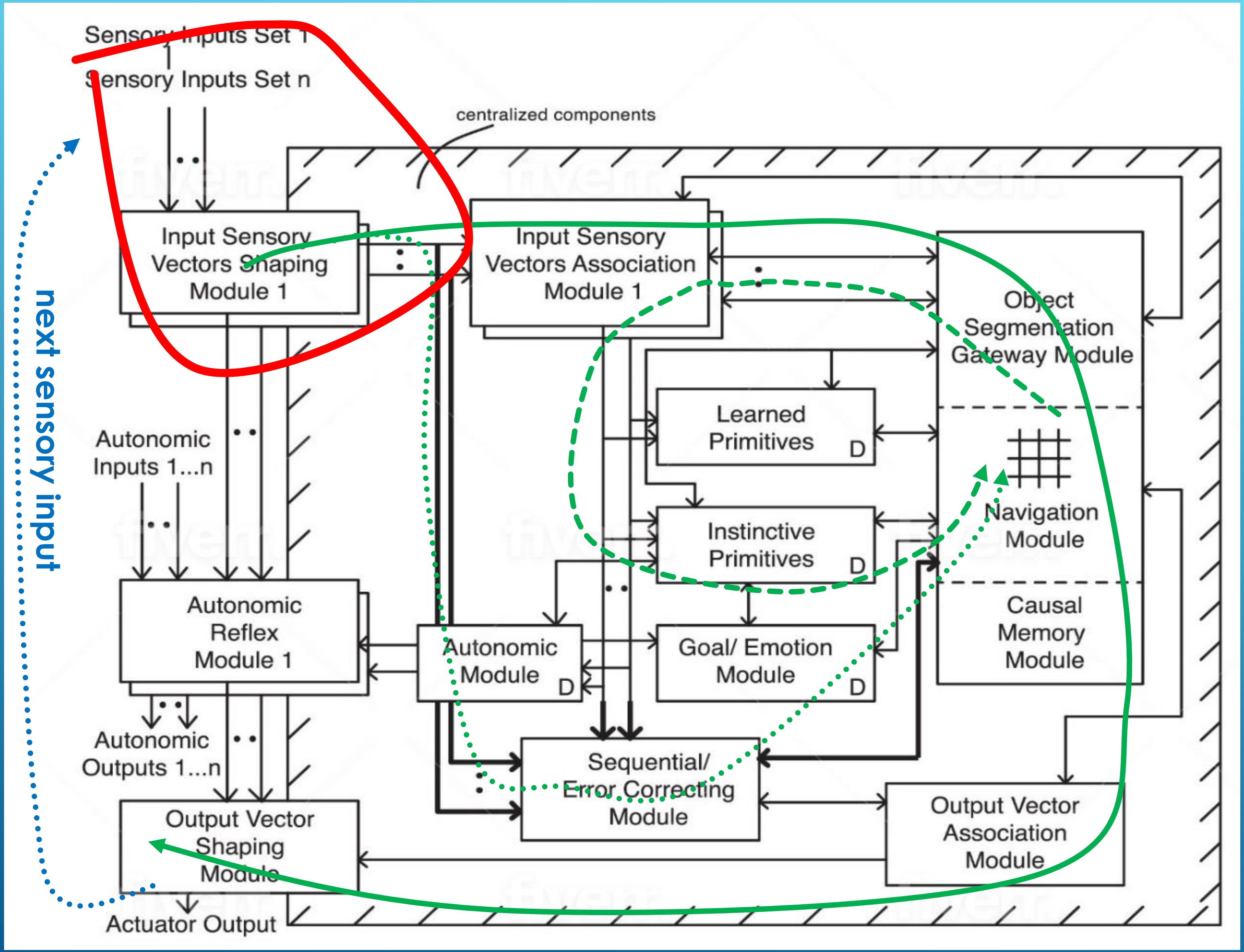
```
Press ENTER to continue...


Simplified simulation of sleep/wake cycle and energy managment.
CCA3 in wake state. Energy usage state is normal.
Autonomic system was not modeled in the equations of the CCA3 Bindi
Core CCA3 autonomic check is passed -- no set of immediate actions r
No attention needed for any CCA3 peripheral autonomic actions.
```

Autonomic not modelled in equations

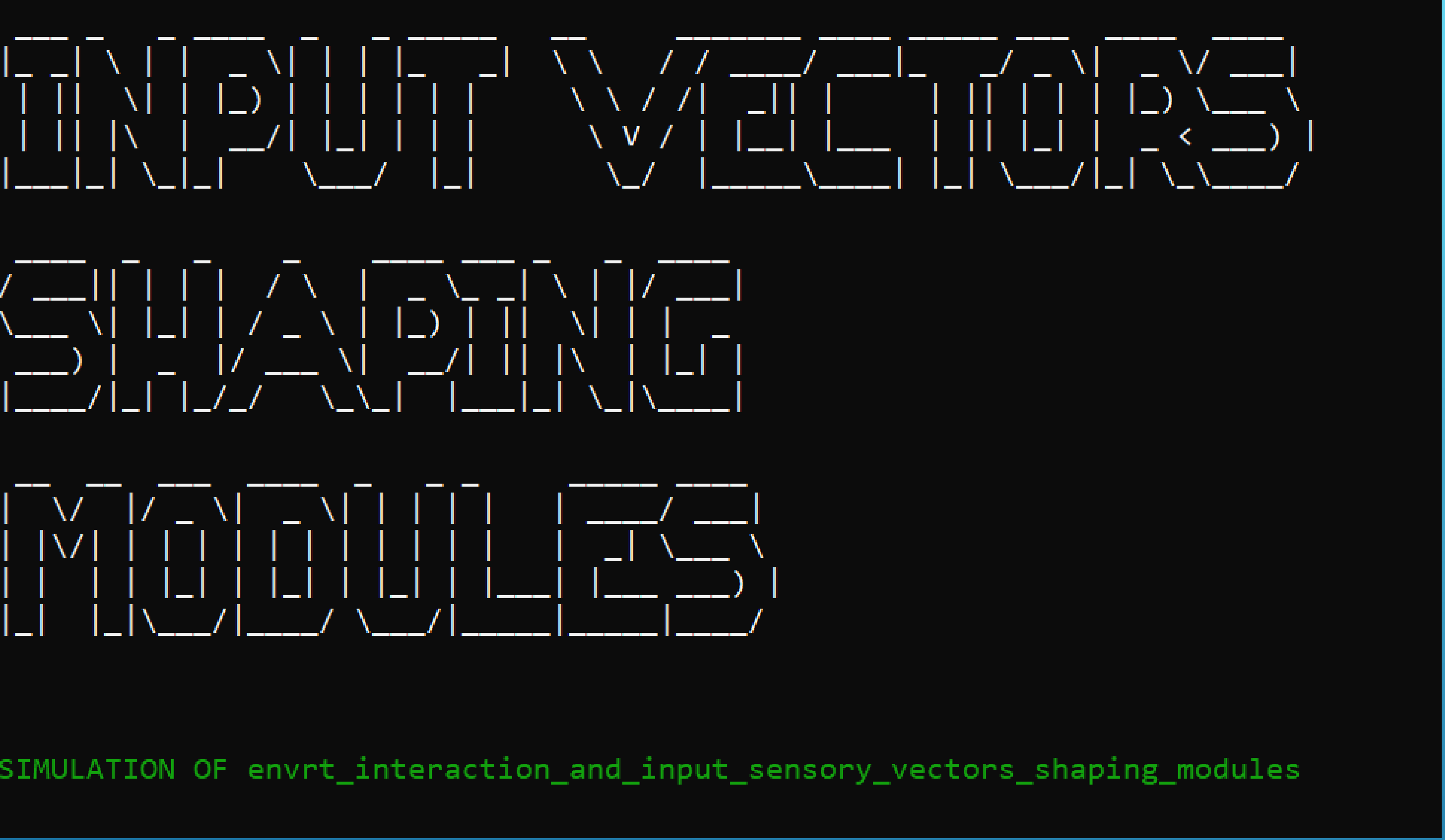
# main_mech.cycles()



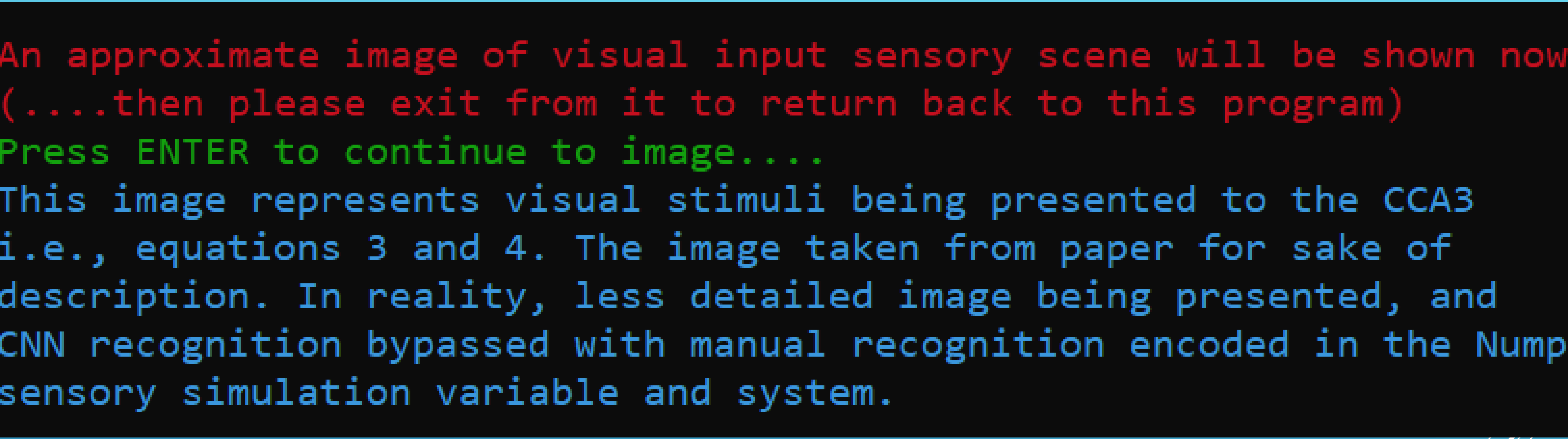
```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)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



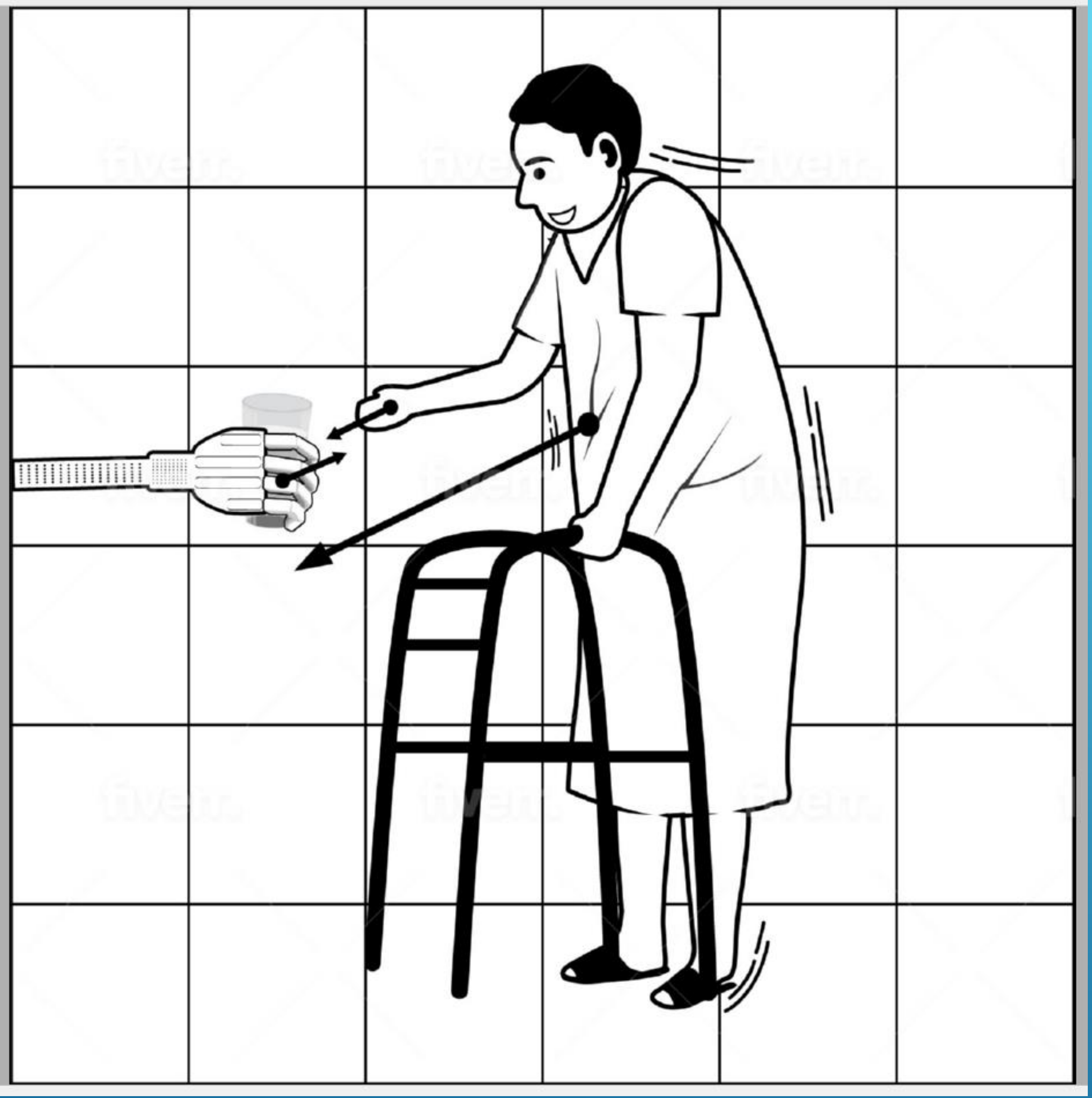
Sensory Inputs Set 1
Sensory Inputs Set n
centralized components
Input Sensory Vectors Shaping Module 1
Input Sensory Vectors Association Module 1
Object Segmentation Gateway Module
Learned Primitives D
Instinctive Primitives D
Navigation Module
Causal Memory Module
Autonomic Inputs 1...n
Autonomic Reflex Module 1
Autonomic Module D
Goal/ Emotion Module D
Autonomic Outputs 1...n
Sequential/ Error Correcting Module
Output Vector Association Module
Output Vector Shaping Module
Actuator Output
next sensory input

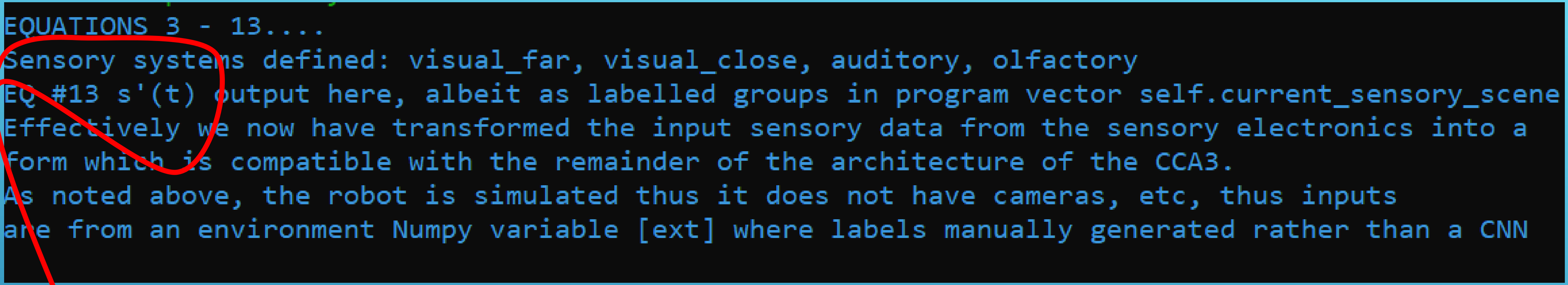


```
INPUT VECTORS
SHAPING
MODULES

SIMULATION OF envrt_interaction_and_input_sensory_vectors_shaping_modules
```



```
An approximate image of visual input sensory scene will be shown now
(....then please exit from it to return back to this program)
Press ENTER to continue to image....
This image represents visual stimuli being presented to the CCA3
i.e., equations 3 and 4. The image taken from paper for sake of
description. In reality, less detailed image being presented, and
CNN recognition bypassed with manual recognition encoded in the Nump
sensory simulation variable and system.
```

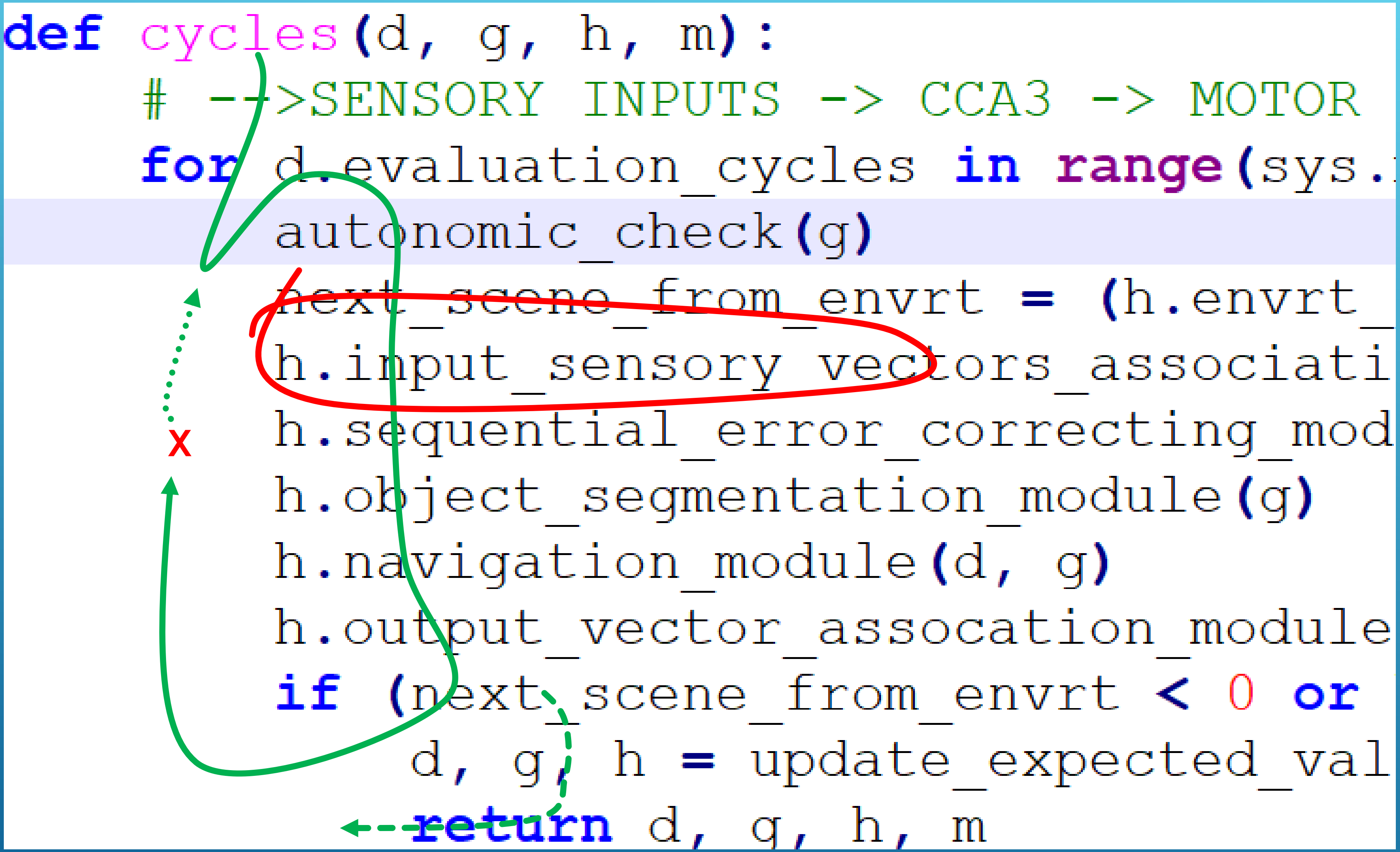




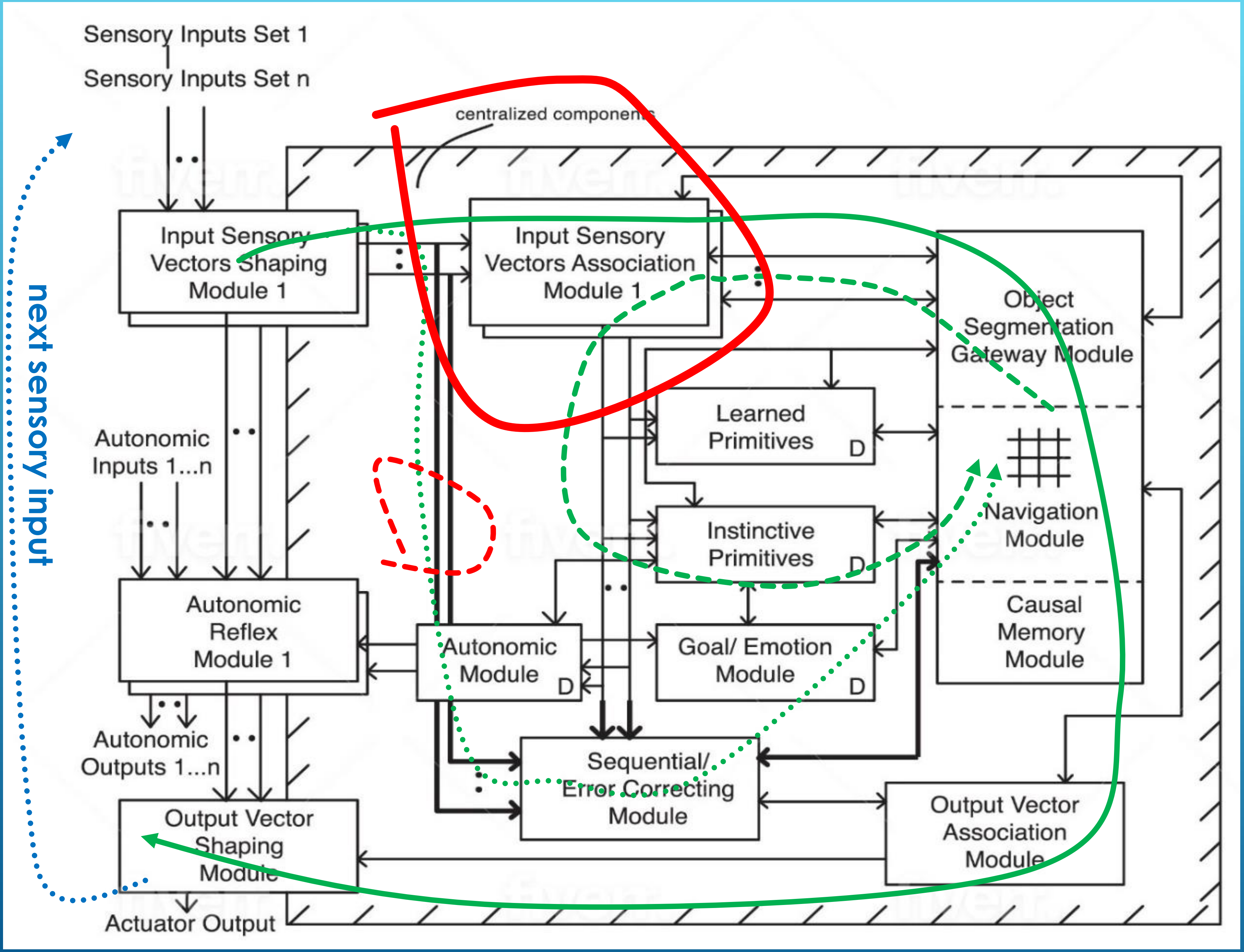
```
EQUATIONS 3 - 13....
Sensory systems defined: visual_far, visual_close, auditory, olfactory
EQ #13 s'(t) output here, albeit as labelled groups in program vector self.current_sensory_scene
Effectively we now have transformed the input sensory data from the sensory electronics into a
form which is compatible with the remainder of the architecture of the CCA3.
As noted above, the robot is simulated thus it does not have cameras, etc, thus inputs
are from an environment Numpy variable [ext] where labels manually generated rather than a CNN
```

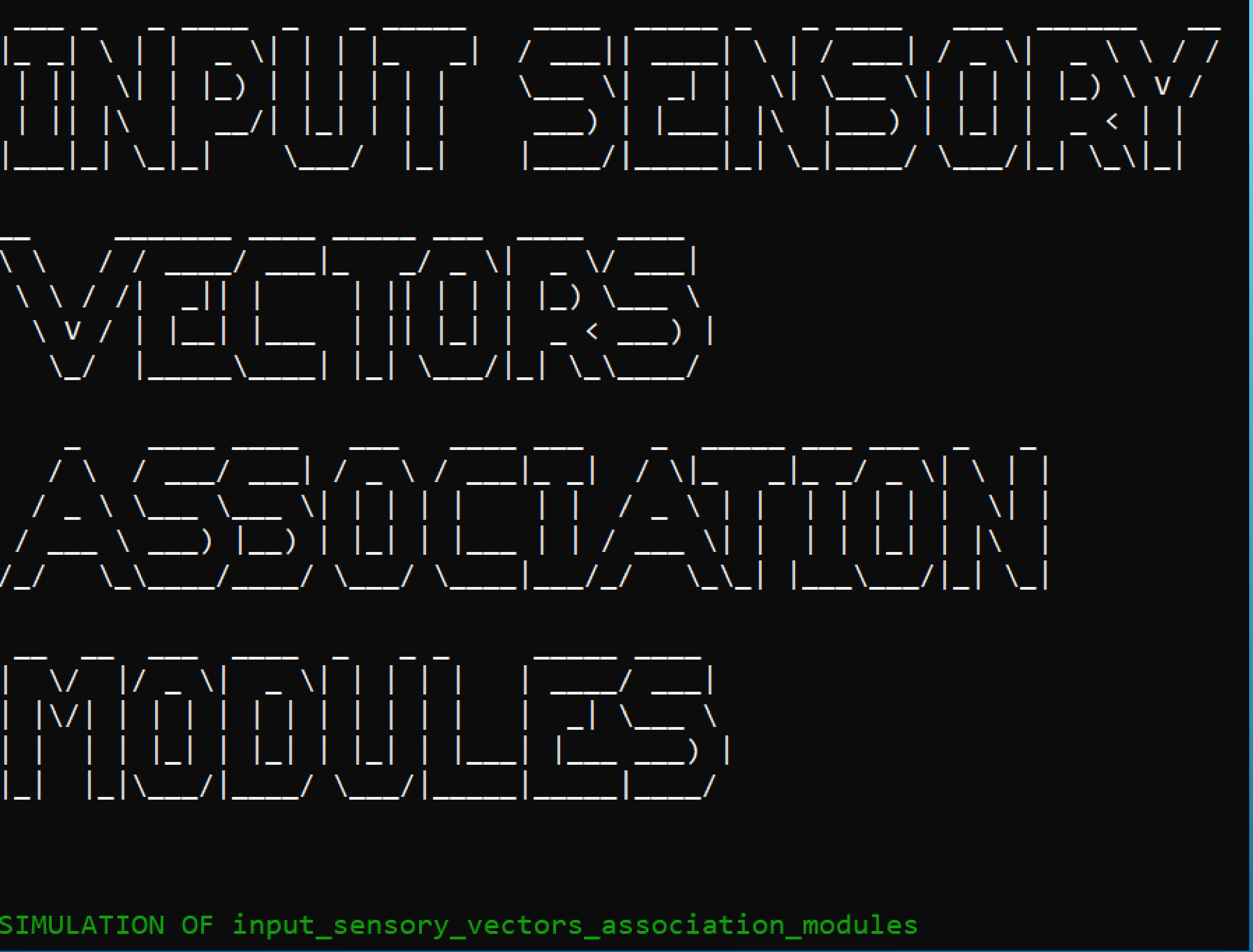
(3) – (13) occur here
s'(t) created (12/13)

# main_mech.cycles()

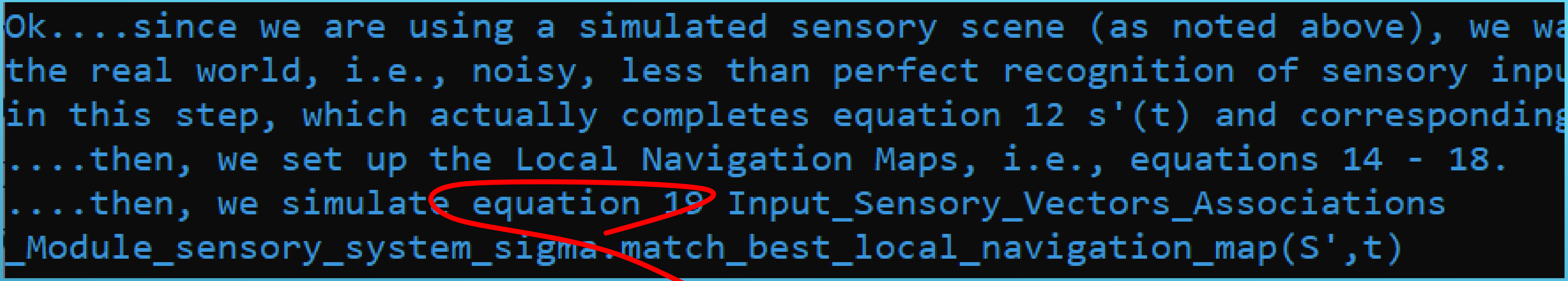


```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)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


Sensory Inputs Set 1
Sensory Inputs Set n
centralized components
Input Sensory Vectors Shaping Module 1
Input Sensory Vectors Association Module 1
Object Segmentation Gateway Module
Learned Primitives D
Instinctive Primitives D
Navigation Module
Causal Memory Module
Autonomic Inputs 1...n
Autonomic Reflex Module 1
Autonomic Module D
Goal/ Emotion Module D
Autonomic Outputs 1...n
Sequential/ Error Correcting Module
Output Vector Association Module
Output Vector Shaping Module
Actuator Output
next sensory input



```
 ___ _   _ ____  _   _ _____   ____  _____ _   _ ____   ___  ______   __
|_ _| \ | |  _ \| | | |_   _| / ___|| ____| \ | / ___| / _ \|  _ \ \ / /
 | ||  \| | |_) | | | | | |   \___ \|  _| |  \| \___ \| | | | |_) \ V /
 | || |\  |  __/| |_| | | |    ___) | |___| |\  |___) | |_| |  _ < | |
|___|_| \_|_|    \___/  |_|   |____/|_____|_| \_|____/ \___/|_| \_\|_|

__     _______ ____ _____ ___  ____  ____
\ \   / / ____/ ___|_   _/ _ \|  _ \/ ___|
 \ \ / /|  _|| |     | || | | | |_) \___ \
  \ V / | |__| |___  | || |_| |  _ < ___) |
   \_/  |_____\____| |_| \___/|_| \_\____/

    _    ____ ____   ___   ____ ___    _  _____ ___ ___  _   _
   / \  / ___/ ___| / _ \ / ___|_ _|  / \|_   _|_ _/ _ \| \ | |
  / _ \ \___ \___ \| | | | |    | |  / _ \ | |  | | | | |  \| |
 / ___ \ ___) |__) | |_| | |___ | | / ___ \| |  | | |_| | |\  |
/_/   \_\____/____/ \___/ \____|___/_/   \_\_| |___\___/|_| \_|

 __  __  ___  ____  _   _ _     _____ ____
|  \/  |/ _ \|  _ \| | | | |   | ____/ ___|
| |\/| | | | | | | | | | | |   |  _| \___ \
| |  | | |_| | |_| | |_| | |___| |___ ___) |
|_|  |_|\___/|____/ \___/|_____|_____|____/


SIMULATION OF input_sensory_vectors_association_modules
```



```
Ok....since we are using a simulated sensory scene (as noted above), we wa
the real world, i.e., noisy, less than perfect recognition of sensory inpu
in this step, which actually completes equation 12 s'(t) and correspondin
....then, we set up the Local Navigation Maps, i.e., equations 14 - 18.
....then, we simulate equation 19 Input_Sensory_Vectors_Associations
_Module_sensory_system_sigma.match_best_local_navigation_map(S',t)
```

$$\mathbf{LNM}_{(\boldsymbol{\sigma},\mathbf{mapno})} \in \mathrm{R}^{mxnxo} \quad (16)$$

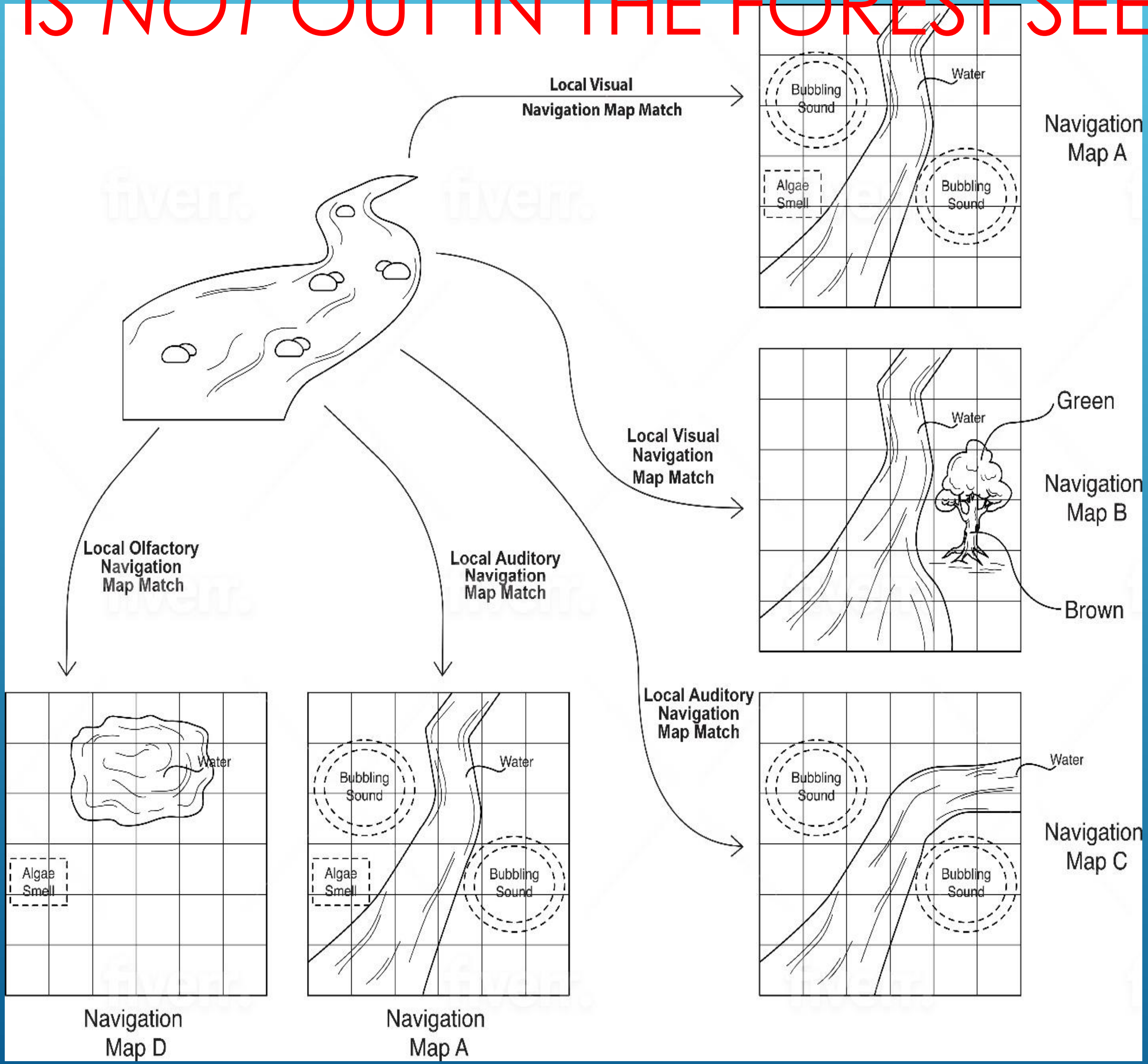
$$\boldsymbol{all\_maps}_{\sigma,t} = [\mathbf{LNM}_{(\boldsymbol{\sigma},1,t)}, \mathbf{LNM}_{(\boldsymbol{\sigma},2,t)}, \mathbf{LNM}_{(\boldsymbol{\sigma},3,t)}, \ldots, \mathbf{LNM}_{(\boldsymbol{\sigma},\,\boldsymbol{\theta},\,t)}] \;(17)$$

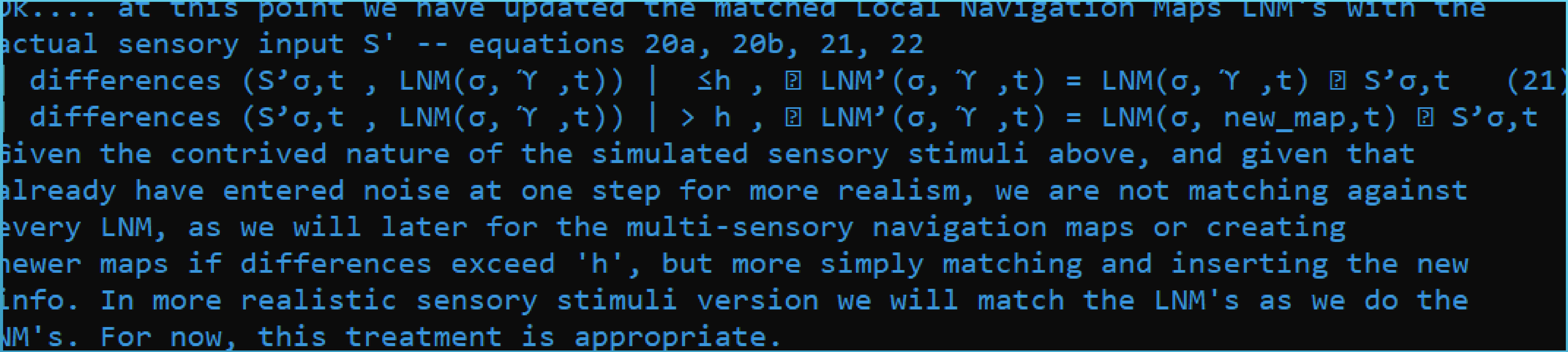
$$\Upsilon := \mathbf{mapno} \text{ of best matching map in set of navigation maps} \in \mathbf{mapno} \quad (18)$$

$$\mathbf{LNM}_{(\boldsymbol{\sigma},\,\Upsilon,\,t)} = \text{Input\_Sensory\_Vectors\_Associations\_Module}_{\boldsymbol{\sigma}}\text{.match\_best\_local\_navigation\_map}(\, \mathbf{S'}_{\boldsymbol{\sigma},t}\,) \quad (19)$$

*“then a miracle occurs”* – summarize some algorithmic portions with dot notation

**THIS IS EXAMPLE ONLY OF MATCHING NAVMAPS**
CCA3 IN CURRENT EXAMPLE IS WORKING AS A PATIENT-AIDE
IT IS *NOT* OUT IN THE FOREST SEEING RIVERS





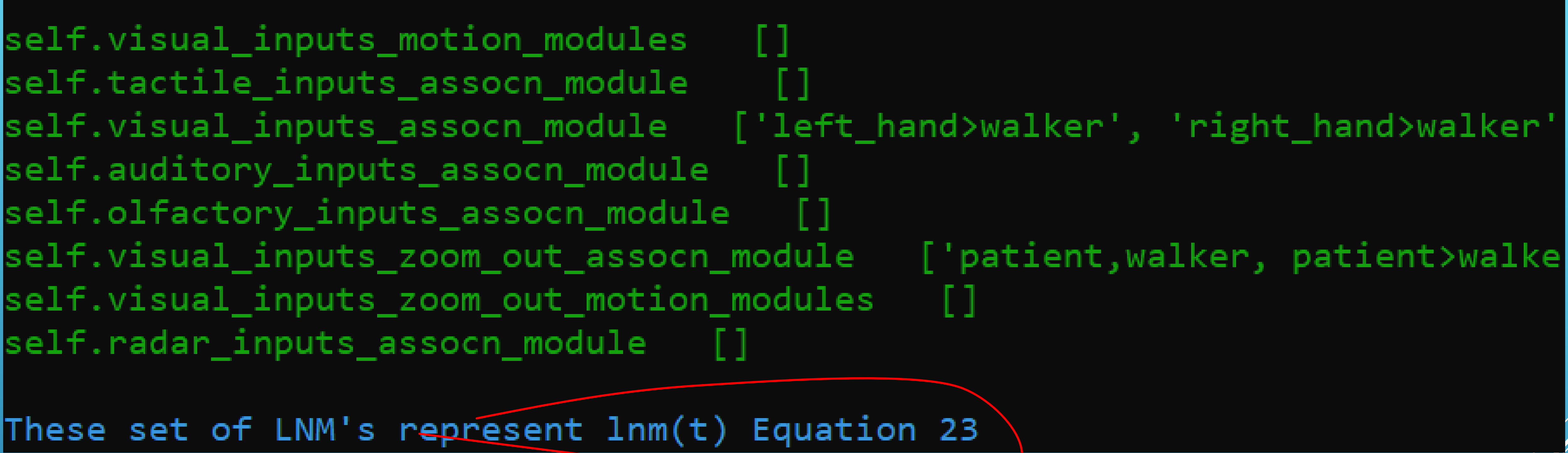
```
K.... at this point we have updated the matched Local Navigation Maps LNM's with the
ctual sensory input S' -- equations 20a, 20b, 21, 22
 differences (S'σ,t , LNM(σ, Ύ ,t)) |  ≤h , ▯ LNM'(σ, Ύ ,t) = LNM(σ, Ύ ,t) ▯ S'σ,t    (21
 differences (S'σ,t , LNM(σ, Ύ ,t)) | > h , ▯ LNM'(σ, Ύ ,t) = LNM(σ, new_map,t) ▯ S'σ,t
iven the contrived nature of the simulated sensory stimuli above, and given that
lready have entered noise at one step for more realism, we are not matching against
very LNM, as we will later for the multi-sensory navigation maps or creating
ewer maps if differences exceed 'h', but more simply matching and inserting the new
nfo. In more realistic sensory stimuli version we will match the LNM's as we do the
NM's. For now, this treatment is appropriate.
```

**h =** number of differences allowed copied onto existing map $\in$ R (20a)

**new_map** : = **mapno** of new local navigation map added to **σ** $\in$ **mapno** (20b)

$$| \text{differences } (\mathbf{S'}_{\sigma,t} , \mathbf{LNM}_{(\sigma, \Upsilon ,t)}) | \leq \mathbf{h} , \Rightarrow \mathbf{LNM'}_{(\sigma, \Upsilon ,t)} = \mathbf{LNM}_{(\sigma, \Upsilon ,t)} \cup \mathbf{S'}_{\sigma,t} \quad (21)$$

$$| \text{differences } (\mathbf{S'}_{\sigma,t} , \mathbf{LNM}_{(\sigma, \Upsilon ,t)}) | > \mathbf{h} , \Rightarrow \mathbf{LNM'}_{(\sigma, \Upsilon ,t)} = \mathbf{LNM}_{(\sigma, \text{new\_map},t)} \cup \mathbf{S'}_{\sigma,t} \quad (22)$$

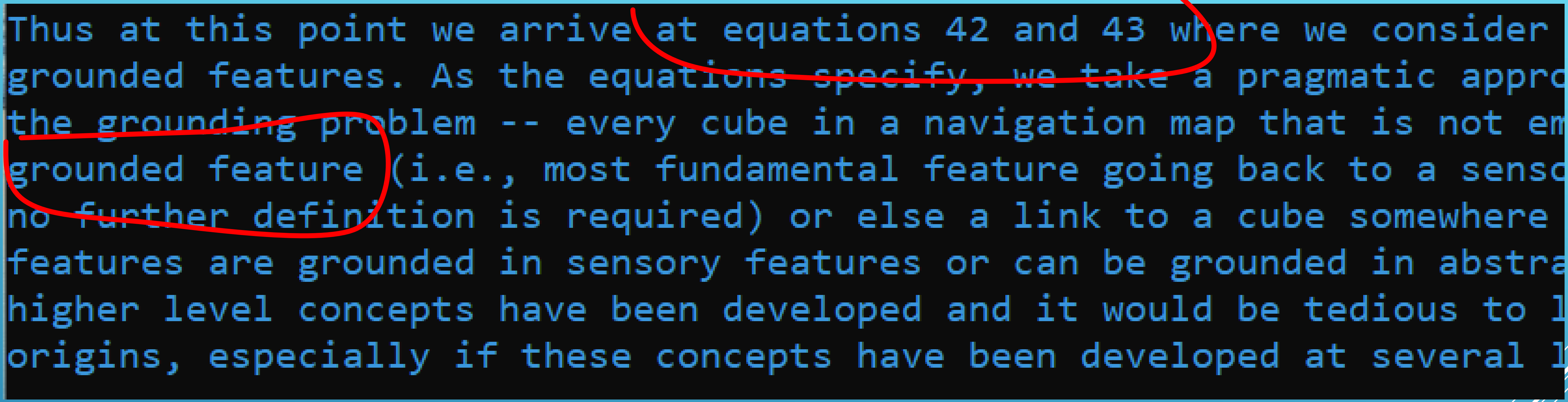


```
self.visual_inputs_motion_modules    []
self.tactile_inputs_assocn_module     []
self.visual_inputs_assocn_module    ['left_hand>walker', 'right_hand>walker'
self.auditory_inputs_assocn_module    []
self.olfactory_inputs_assocn_module    []
self.visual_inputs_zoom_out_assocn_module    ['patient,walker, patient>walke
self.visual_inputs_zoom_out_motion_modules    []
self.radar_inputs_assocn_module    []

These set of LNM's represent lnm(t) Equation 23
```

$$\mathit{lnm}_t = [\mathbf{LNM'}_{(1,\Upsilon,t)}, \mathbf{LNM'}_{(2,\Upsilon,t)}, \mathbf{LNM'}_{(3,\Upsilon,t)}, \ldots, \mathbf{LNM'}_{(\boldsymbol{n\_\sigma},\Upsilon,t)}] \quad (23)$$

$$\mathbf{NM}_{\mathbf{mapno}} \in \mathrm{R}^{mxnxo}, \mathbf{IPM}_{\mathbf{mapno}} \in \mathrm{R}^{mxnxo}, \mathbf{LPM}_{\mathbf{mapno}} \in \mathrm{R}^{mxnxo} \quad (24)$$

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



Thus at this point we arrive at equations 42 and 43 where we consider
grounded features. As the equations specify, we take a pragmatic appro
the grounding problem -- every cube in a navigation map that is not em
grounded feature (i.e., most fundamental feature going back to a senso
no further definition is required) or else a link to a cube somewhere
features are grounded in sensory features or can be grounded in abstra
higher level concepts have been developed and it would be tedious to l
origins, especially if these concepts have been developed at several l

## Symbol Grounding Problem

- Harnad 1990
- symbolic model of mind – symbol strings, rules can manipulate
- how can capture thoughts or beliefs?
- e.g., learn Chinese from Chinese-Chinese dictionary

- Barsalou 2020
- how can abstract symbols of a "cognition module" understand the world?

## CCA3 Pragmatic Grounding Solution:

- every cube in a navmap must contain a grounded feature or a link somewhere
- links can be to actual low-level sensory features or to higher concepts

$$\textbf{\textit{grounded\_feature}} := \forall_{\textit{feature}} : \textbf{\textit{feature}} \in \textbf{\textit{all\_LNMs}}_{\chi} \quad (42)$$

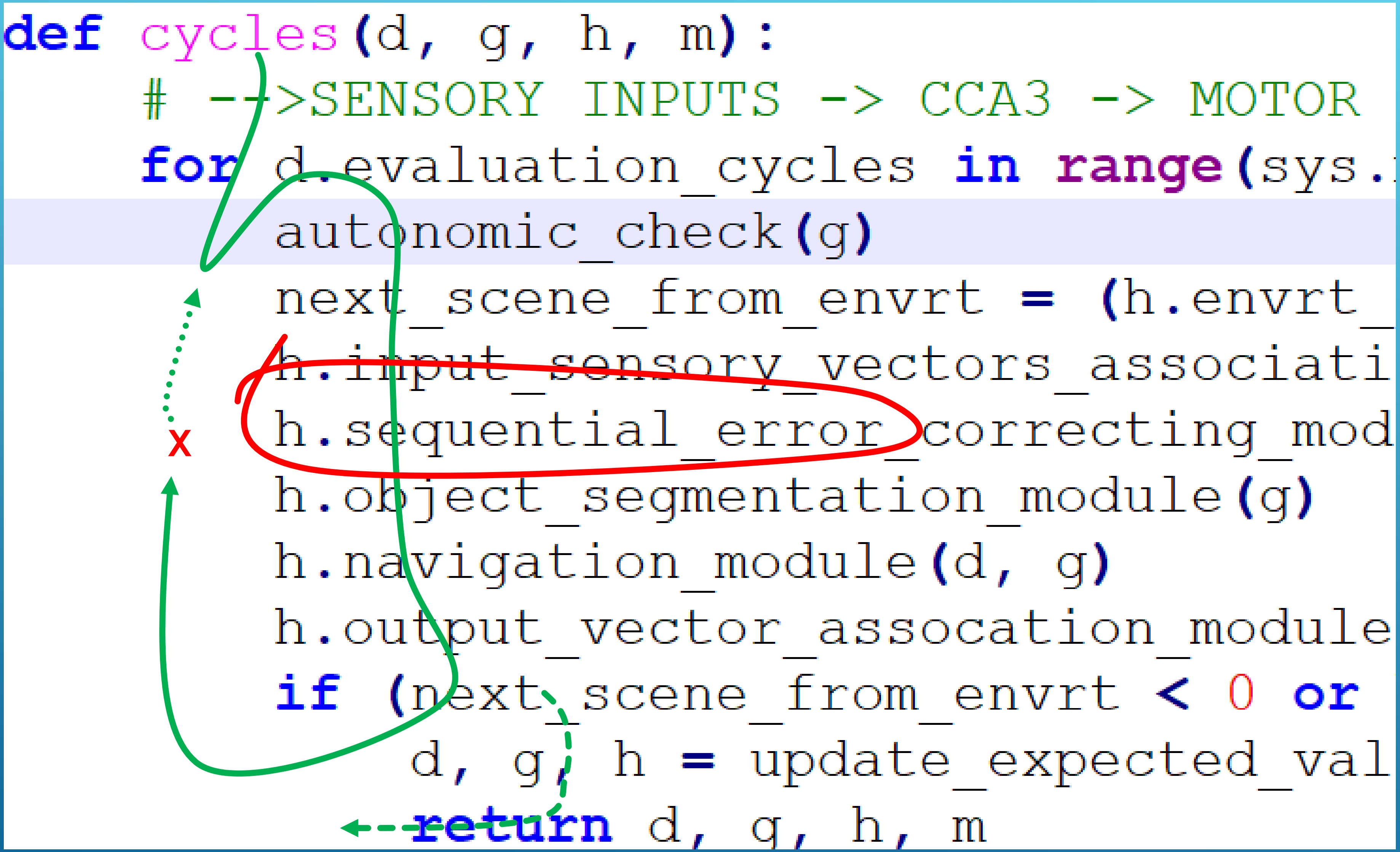
$$\forall_{\chi,t} : \textbf{\textit{all\_navmaps}}_{\chi,t} = \textbf{\textit{grounded\_feature}}$$

$$\text{OR } \textbf{\textit{link(all\_navmaps}}_{\chi,t}) \neq [\ ]$$

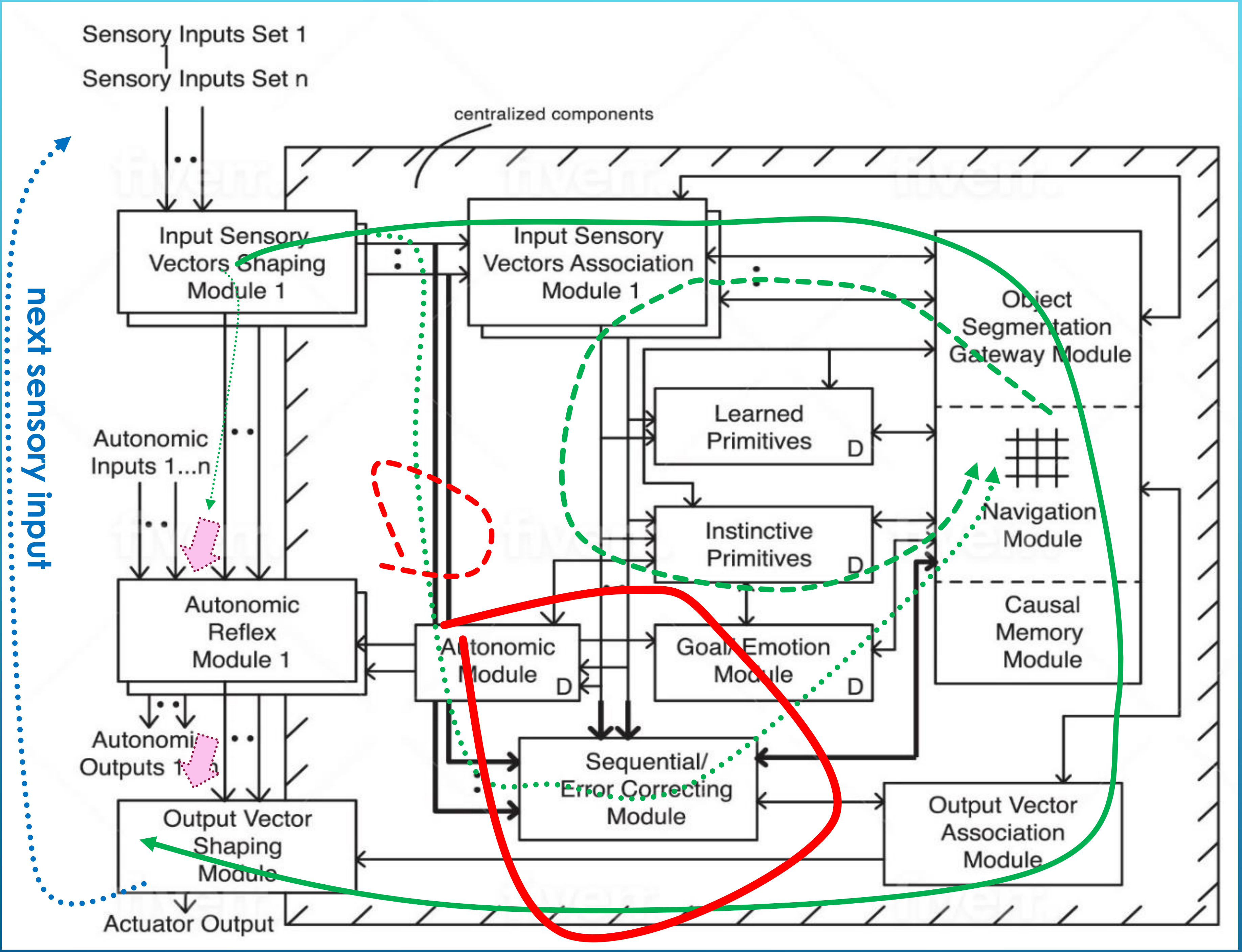
$$\text{OR } \textbf{\textit{all\_navmaps}}_{\chi,t} = [\ ] \quad (43)$$

for all values of address $\chi$, cube contains a grounded feature or a link somewhere

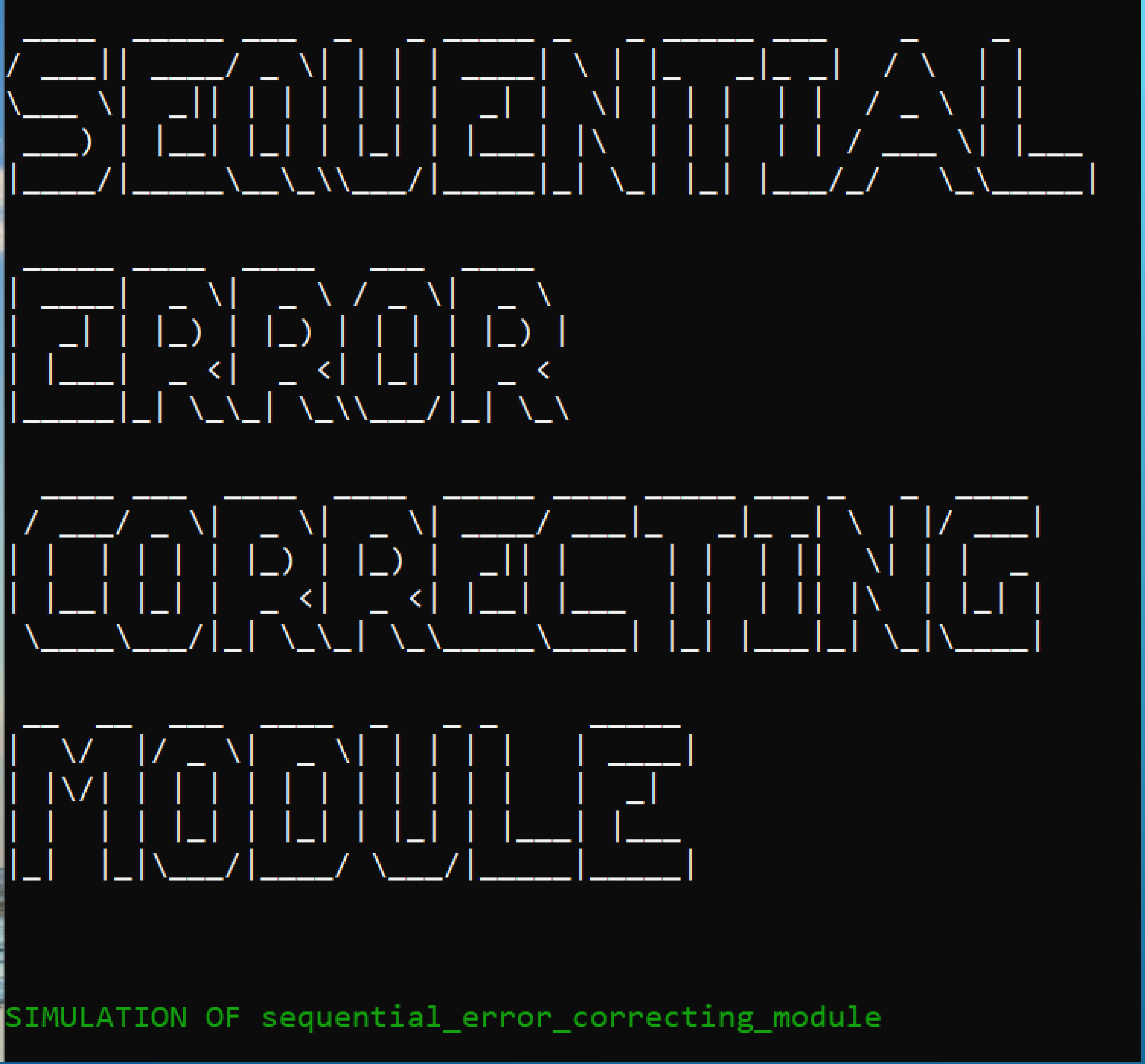
# main_mech.cycles()



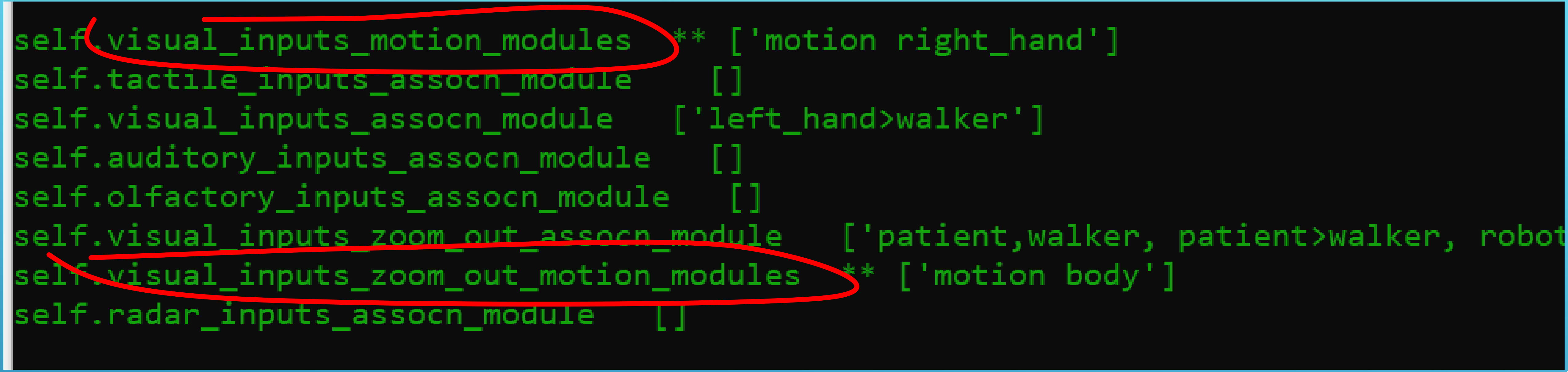
```
def cycles(d, g, h, m):
    # -->SENSORY INPUTS -> CCA3 -> MOTOR
    for d.evaluation_cycles in range(sys.
        autonomic_check(g)
        next_scene_from_envrt = (h.envrt_
        h.input_sensory_vectors_associati
        h.sequential_error_correcting_mod
        h.object_segmentation_module(g)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



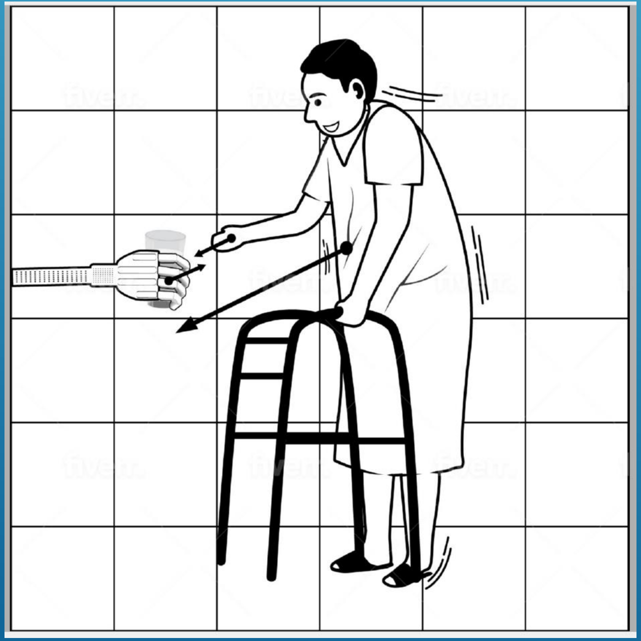
Sensory Inputs Set 1
Sensory Inputs Set n
centralized components
next sensory input
Input Sensory Vectors Shaping Module 1
Input Sensory Vectors Association Module 1
Object Segmentation Gateway Module
Learned Primitives D
Instinctive Primitives D
Navigation Module
Causal Memory Module
Autonomic Inputs 1...n
Autonomic Reflex Module 1
Autonomic Module D
Goal/Emotion Module D
Autonomic Outputs 1...n
Sequential/ Error Correcting Module
Output Vector Association Module
Output Vector Shaping Module
Actuator Output

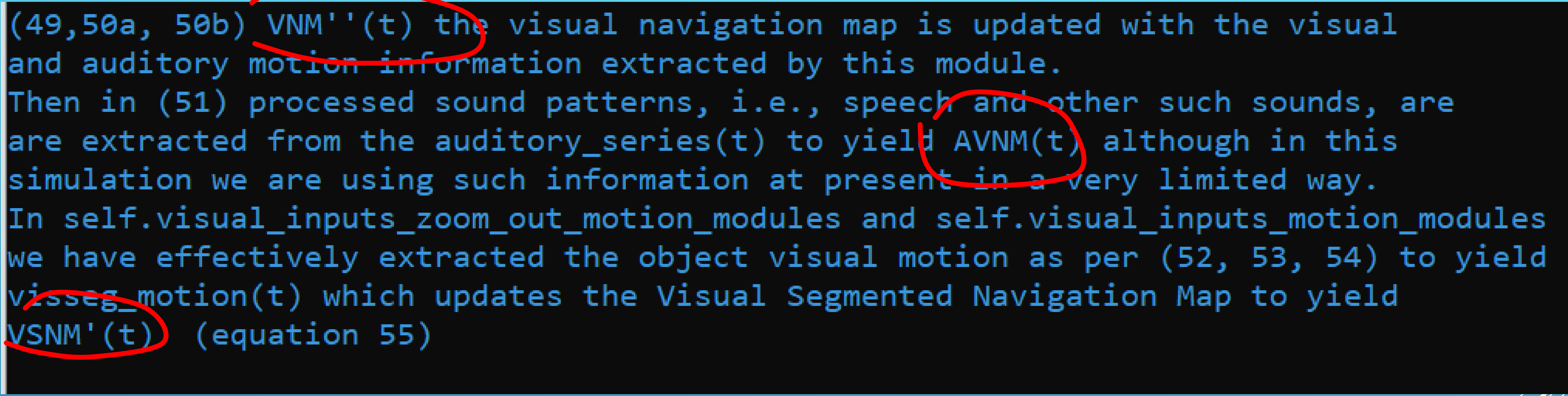


```
 ____  _____ ___  _   _ _____ _   _ _____ ___    _    _     
/ ___|| ____/ _ \| | | | ____| \ | |_   _|_ _|  / \  | |    
\___ \|  _|| | | | | | |  _| |  \| | | |  | |  / _ \ | |    
 ___) | |__| |_| | |_| | |___| |\  | | |  | | / ___ \| |___ 
|____/|_____\__\_\\___/|_____|_| \_| |_| |___/_/   \_\_____|

 _____ ____  ____   ___  ____  
| ____|  _ \|  _ \ / _ \|  _ \ 
|  _| | |_) | |_) | | | | |_) |
| |___|  _ <|  _ <| |_| |  _ < 
|_____|_| \_\_| \_\\___/|_| \_\

  ____ ___  ____  ____  _____ ____ _____ ___ _   _  ____ 
 / ___/ _ \|  _ \|  _ \| ____/ ___|_   _|_ _| \ | |/ ___|
| |  | | | | |_) | |_) |  _|| |     | |  | ||  \| | |  _ 
| |__| |_| |  _ <|  _ <| |__| |___  | |  | || |\  | |_| |
 \____\___/|_| \_\_| \_\_____\____| |_| |___|_| \_|\____|

 __  __  ___  ____  _   _ _     _____ 
|  \/  |/ _ \|  _ \| | | | |   | ____|
| |\/| | | | | | | | | | | |   |  _|  
| |  | | |_| | |_| | |_| | |___| |___ 
|_|  |_|\___/|____/ \___/|_____|_____|


SIMULATION OF sequential_error_correcting_module
```



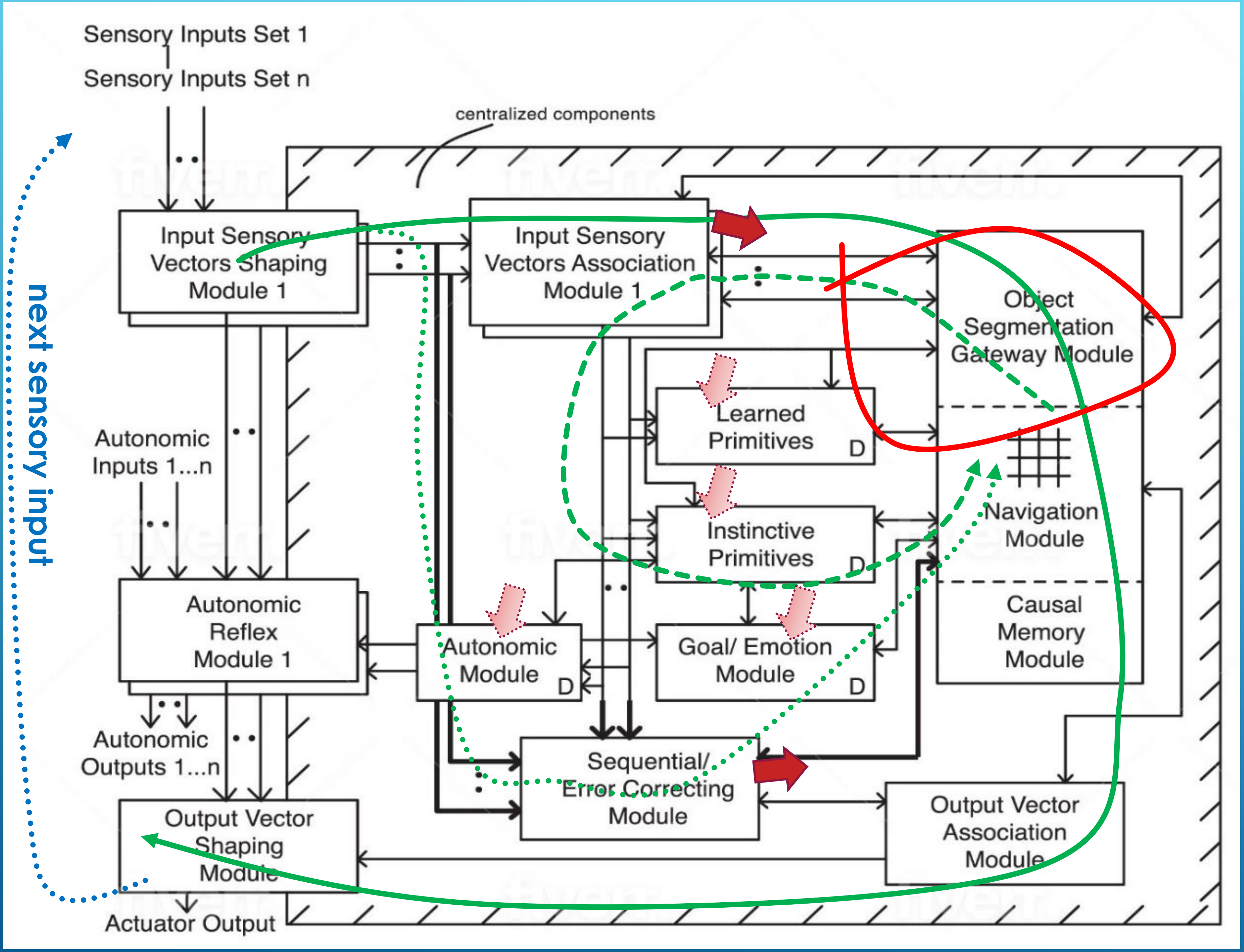
```
self.visual_inputs_motion_modules   ** ['motion right_hand']
self.tactile_inputs_assocn_module     []
self.visual_inputs_assocn_module   ['left_hand>walker']
self.auditory_inputs_assocn_module    []
self.olfactory_inputs_assocn_module    []
self.visual_inputs_zoom_out_assocn_module    ['patient,walker, patient>walker, robot
self.visual_inputs_zoom_out_motion_modules  ** ['motion body']
self.radar_inputs_assocn_module    []
```



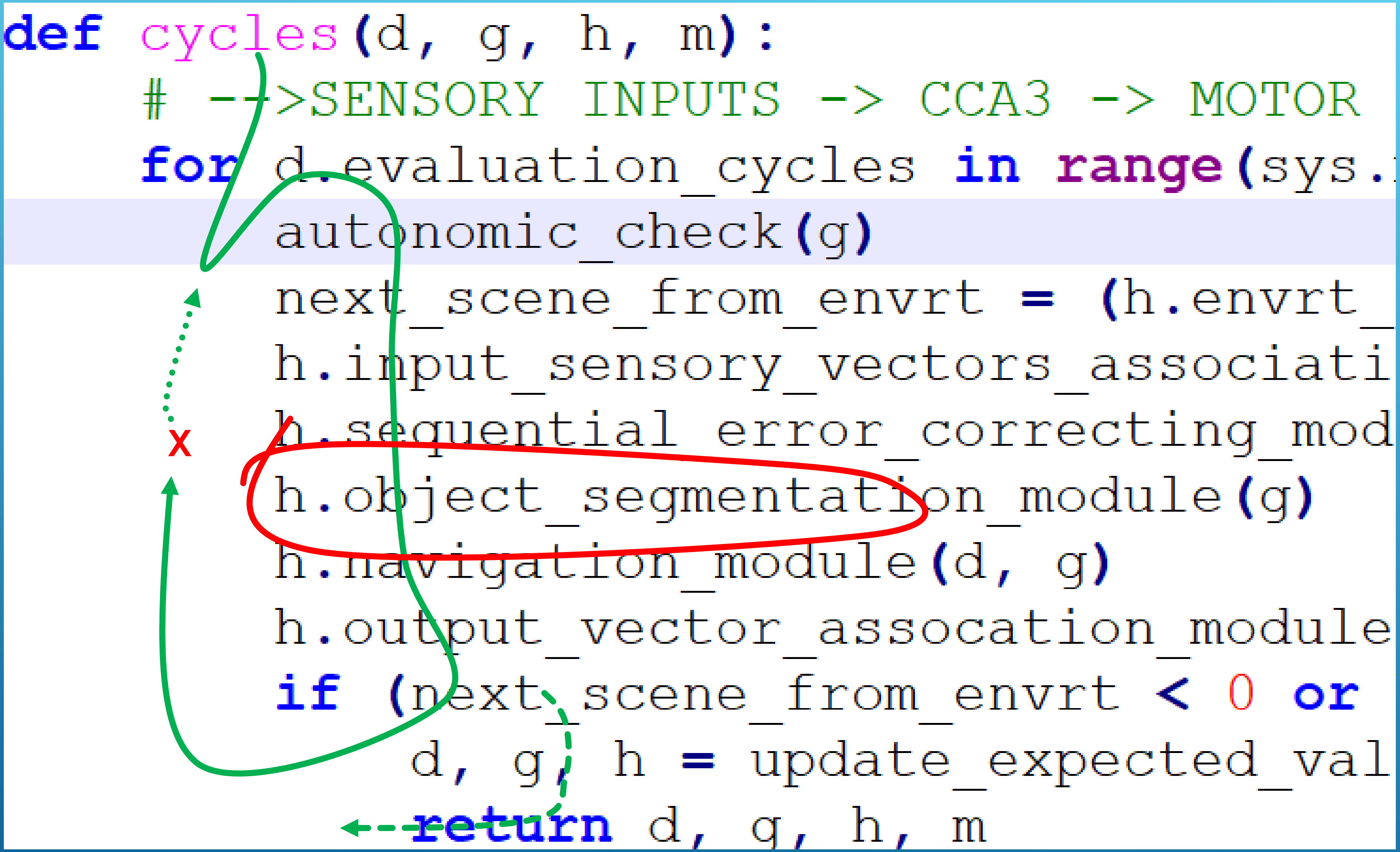


```
(49,50a, 50b) VNM''(t) the visual navigation map is updated with the visual
and auditory motion information extracted by this module.
Then in (51) processed sound patterns, i.e., speech and other such sounds, are
are extracted from the auditory_series(t) to yield AVNM(t) although in this
simulation we are using such information at present in a very limited way.
In self.visual_inputs_zoom_out_motion_modules and self.visual_inputs_motion_modules
we have effectively extracted the object visual motion as per (52, 53, 54) to yield
visseg_motion(t) which updates the Visual Segmented Navigation Map to yield
VSNM'(t)  (equation 55)
```

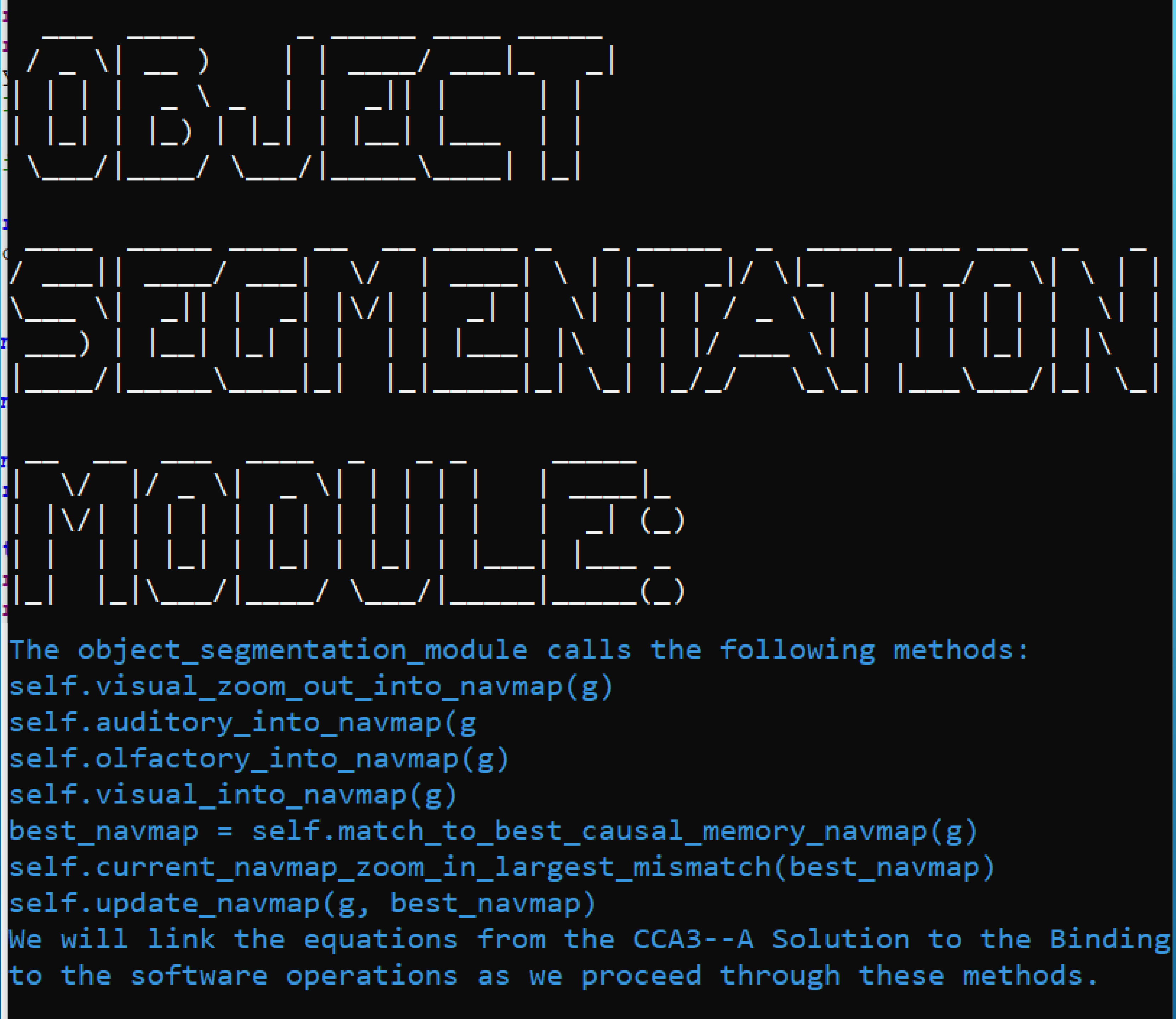
**VNM''(t)** Vector Navigation Map updated with *visual_motion(t) and auditory_motion(t)*
**AVNM(t)** Audio Vector Navigation Map from auditory_series(t)
**VSNM'(t)** Visual Segmented Navigation Map is sent from Object Segmentation Mod at t-3, t-2, t-1, t visual input sensory data → visseg_motion(t) vector which is bound to VSNM(t) creating **VSNM'(t)**


Sensory Inputs Set 1
Sensory Inputs Set n
centralized components
Input Sensory Vectors Shaping Module 1
Input Sensory Vectors Association Module 1
Object Segmentation Gateway Module
Learned Primitives D
Instinctive Primitives D
Navigation Module
Causal Memory Module
Autonomic Inputs 1...n
Autonomic Reflex Module 1
Autonomic Module D
Goal/ Emotion Module D
Autonomic Outputs 1...n
Sequential/ Error Correcting Module
Output Vector Association Module
Output Vector Shaping Module
Actuator Output
next sensory input

# 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
```



# OBJECT SEGMENTATION MODULE:

```
The object_segmentation_module calls the following methods:
self.visual_zoom_out_into_navmap(g)
self.auditory_into_navmap(g
self.olfactory_into_navmap(g)
self.visual_into_navmap(g)
best_navmap = self.match_to_best_causal_memory_navmap(g)
self.current_navmap_zoom_in_largest_mismatch(best_navmap)
self.update_navmap(g, best_navmap)
We will link the equations from the CCA3--A Solution to the Binding
to the software operations as we proceed through these methods.
```

....continued in VIDEO 5

balloon from powerpoint stock image