Cognitive Agent and its Implementation in the Blender Game Engine Environment

Janusz A. Starzyk
Electrical Engineering and Computer Science
Ohio University, Athens, OH 45701, USA and
Department of Applied Information Systems, University of Information Technology and Management, Rzeszow, Poland
starzykj@gmail.com

Paweł Raif
Dept. of Biosensors and Biomedical Signals Processing
Silesian University of Technology, Gliwice, Poland
pawel.raif@gmail.com

***Abstract*— The paper presents a structural model of a cognitive agent and its Blender implementation. Built in a virtual world, the agent is able to act autonomously, observe its environment and learn from its actions using principles of motivated learning. We discuss both its organizational structure and tools we used to develop virtual implementation of the agent. In this paper, we explain why it is important to consider cognitive agent's model together with restrictions imposed by its sensory and motor functions and the properties of objects the agent interacts with in the virtual world.**

Keywords—cognitive model, building blocks of cognitive agents, memory organization, computational simulation, Blender.

I. INTRODUCTION

In research on cognitive architectures modeling and simulation play an important role. There is a disagreement whether the simulation result is a good research material or not. Some researchers claim that only the robotic (embodied and situated) agents are useful in cognitive agent research. However, many say that simulation is a very useful substitute of expensive experiments with physical robots, especially in the initial phases of research on the development of intelligent systems. We support this view that placing a learning agent in unknown virtual environment and letting it develop is important for advancement of cognitive architectures. The role of computer simulation in cognitive systems research is discussed in [1], [1] and [3].

The problems of using reinforcement learning approach [4] in design and simulation of autonomous humanoid agents and robots has been addressed in papers [5],[6]. Several examples of using simulation in autonomous systems with intrinsic motivation, are presented in [7],[8]. The motivated learning systems allow the agents to develop their own motivations and hierarchy of goals. This approach is based on primitive and abstract needs that motivate an agent to learn abstract concepts through the interaction with environment. This mechanism has been discussed in [9],[10]. The goal-driven mechanism in motivated learning can also be combined with reinforcement learning algorithms. Such combination was proposed and examined in [11].

In simulation of virtual cognitive systems, game programming libraries as well as virtual simulation environment [12],[13] are widely used [14],[15]. In this paper we present a functional organization of a cognitive system and an example simulation framework which can be used for cognitive agents' research.

Functional memory organization is the model we develop implementing its major operations on symbolic level and illustrating effects of its cognitive work by agent's actions. This framework is implemented in the Blender Game Engine (BGE) environment. BGE is a real-time simulation engine implementing realistic physics in created virtual 3D world. Among other functions, it allows modeling, preparing animations, and developing video games that illustrate behavior of virtual agents. It facilitates simulation of such behaviors as learning how to move and control agents' virtual "body". An agent can interact with the environment using: sensors, actuators and controllers.

Sensors are associated with stimuli from the environment and actuators reflect actions the agent performs on the objects in the environment. The internal logic that drives agent's behavior is associated with motor controllers. All the essential features of the sensors, actuators and controllers, useful in agent's simulations, are presented in the subsequent paragraphs. The following section presents a model of the memory organization of a cognitive agent. Selected building blocks of this agent have been implemented in Blender game engine functionality. In the subsequent sections, we address the problem of modeling of selected parts of the cognitive agent in Blender Game Engine.

II. COGNITIVE AGENT MEMORY ORGANIZATION

Cognitive agent memory should be organized both functionally and structurally to help it function in an open environment and perform all the cognitive and noncognitive processing. While noncognitive processing takes place concurrently in various parts of the memory, in our model cognitive processing is basically sequential and relies on attention switching and mental saccades [16] to serve possibly several concurrent tasks that the agent may be involved with. While cognitive processing in the brain emerges from parallel interactions of interconnected neuronal networks, we aim to implement its symbolic form responsible for thought processes (i.e., reasoning, perception, judgment, and memory).
Memory organization must be structured such that

cognitive tasks are feasible in autonomous agent. To illustrate needs in this regard let us first consider a hypothetical scenario and use it to demonstrate the proposed system operation and to debug its structural organization.

A. Scenario

Imagine an agent's walk in the neighborhood and let us analyze tasks (mental and motor) that the agent must be able to perform. She goes at a rapid exercise pace making sure that she does not trip but also looking around to enjoy nice views and sunshine. Thus her attention is switched to various objects along her path as well as to her own thoughts.

Her main task is marching on along the selected route. Since she knows this route, her cognitive attention seldom is used to verify where she is along the route in order to choose proper turn at the next cross section. The progress is marked in her sequential procedural memory. Cognitive interrupts are obtained from action evaluation that updates the currently active nodes in the procedural memory.

At the same time this cognitive observation of the scene provides scene description and event significance for episodic memory. Thus episodic memory is written to as separate events, structured first in the working memory in the form of scenes with significance of scene elements. Typically, a scene will contain familiar objects filling-in typical objects from its semantic memory, thus the cognitive effort to build a scene is focusing on new elements of the scene. For instance, a view of aggressive neighbor's dog may trigger her attention to evaluate a potential threat and register this as an event.

A scene is not a snapshot at a single moment in time, rather it contains elements of the scene over a time window during which the scene is void of significant changes due to changing scenario that results from motion, appearance of unexpected objects or events. Many scenes will be ignored as insignificant, allowing the agent to fill in the gaps in episodic memory from its semantic knowledge of the neighborhood. Significant scenes will be written to sequential episodic memory with strength that corresponds to overall significance of the entire episode.

She walks fast and it is a hot day, so she is sweating. She pulls her handkerchief while walking and sweep away her sweat from her face. So she must be able to perform two activities (walking and drying her face). Thus a number of parallel activities must be supervised by her working memory. They are initiated by a cognitive decision to respond to a specific need, so the system is driven by various motivations and a strong primitive motivation (pain or discomfort) will activate an attention switch.

An action can be also initiated by a cognitive thought in opportunistic behavior. The agent evaluates situation in the environment and decides if she needs to take an action (for instance to satisfy her higher order need) or to avoid making an unwanted action (action that may have a negative impact on his motivations or values). But whatever she is currently doing, her cognitive mind is always busy. She may think of the incoming party, who will be there, how she would dress, what food she will make. Her inner life is full of unexpected associations and new thoughts. They are not triggered by the external events, at least not the current one, so where they are coming from? In general, they are driven by her internal needs and motivations that prompted her to take this morning walk. In addition, they are supported by associations within her semantic and episodic memories. To be sure, external events affect her thoughts and actions, but if none on these external events is currently switching her attention, her internal state of mind dominates.

There are three sources of attention switching in our model:

1) subconscious switching triggered by the sensory signals. Subconscious sensory signals must have priority over internal cognitive signals, at least until they are cognitively evaluated, since they may represent an incoming threat or pain that must be paid attention to and trigger the immediate response.

2) cognitive signals that represent internal motivations and action monitoring. Internal cognitive signals that represent internal motivations and action monitoring must have higher priority than internal thoughts in attention switching mechanism. A threshold may be applied to these signals if they are frequently repeated.

3) cognitive signals that represent inner thoughts. Internal thoughts compete with each other using mental saccades mechanism, so they do not require thresholds. Instead, an inhibitory signal that is a part of mental saccades mechanism will block the winning concept, so that other concepts can win and be processed. These cognitive signals represent observations, scene building, plans, personal memories, and associations in the semantic memory.

Next, we show a structural organization of a cognitive agent memory providing functional links between various functional blocks and explaining their mutual relationships. Various blocks process their information concurrently sending interrupt signals to redirect focus of attention, change plans, monitor actions and respond to an external threats and opportunities. They also provide the cognitive agent with reach personal memories, accumulated knowledge, skills and desires, making agent's operations fully autonomic, satisfying its need, building its motivations and affecting its emotions.

Fig. 1 shows building blocks of a cognitive agent. The links between various memory areas indicate flow of information between them. Dashed lines indicate one directional links with arrows that indicate the direction of information flow.

B. Building blocks of a cognitive agent

Fig. 1 shows major building blocks of a cognitive agent.
It contains seven processing areas subdivided as follows:

1) *Sensory-motor area*
   a) *Sensory processing*
   b) *Motor processing*
2) *Motivations and goal creation*
   a) *Subconscious pains, rewards, and emotions*
   b) *Cognitive motivations*
3) *Semantic and episodic memory*
   a) *Mental saccades*
   b) *Attention focus*
   c) *Episodic memory*

4) *Motor control*
   a) *Procedural memory*
   b) *Visual saccades*
5) *Working memory*
   a) *Action planning*
   b) *Action evaluation*
   c) *Action monitoring*
   d) *Scene building*
   e) *Episodic management*
6) *Subconscious attention switching*
7) *Winner takes all (WTA) attention switching*

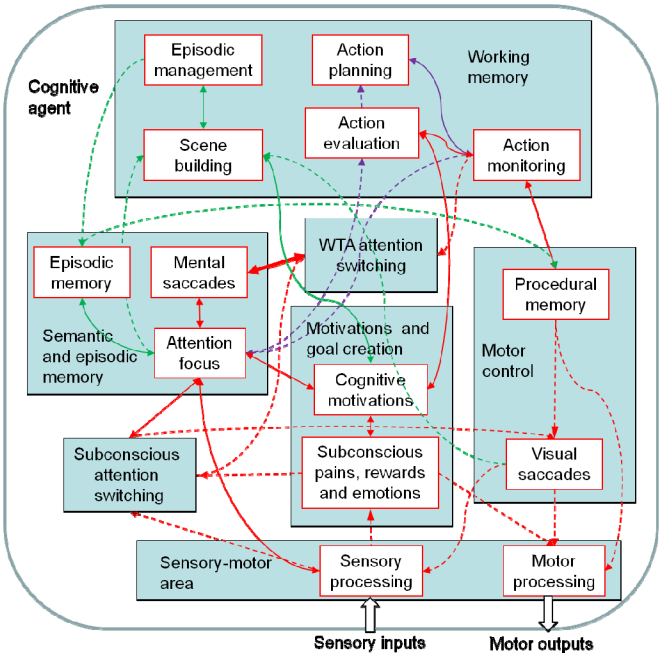

Figure 1. Building blocks of a cognitive agent

Next we briefly discuss functionality of various blocks.

1.a. Sensory processing. This block receives the sensory information from the environment through its sensors, processes it and sends the result to semantic memory where it can invoke a cognitive response. While this is its main function, it also triggers primitive pain signals that regulate agent's motivations, emotions or other noncognitive signals that are used in the sensory (subconscious) attention switching block to trigger attention focus on the specific sensory input. Sensory processing block also receives feedback signals from the semantic memory to help identify object of attention. Other feedback signals it receives come from the Visual saccades block that changes the focal point in the visual input.

1.b. Motor processing. It sends actuator control signals to its actuators, causing its embodiment actions. The major input it receives comes from the Procedural memory that controls execution of a sequence of operations and also controls specific actuators in a single motor action. An input from the Visual saccades is used to move the visual focus on the selected target, and input it receives from the Subconscious pains, rewards and emotions block triggers involuntary reaction to remove the source of pain and to show emotions (for instance by changing facial expressions).

2.a. Subconscious pains, rewards, and emotions. It receives inputs from the Sensory processing and sends its outputs to the Motor processing as discussed in 1.a and 1.b. In addition, it sends signals to the Subconscious attention switching that switches attention to the pain source, and to the Cognitive motivations to influence abstract goal creation. It receives emotional feedback from the Cognitive motivations responding both to painful and pleasant events. It affects significance estimation that the Cognitive motivations assigns to Scene building. Emotional responses build up also as a result of subconscious pains and rewards without cognitive explanation of the emotional level. Emotions have strong influence on the cognitive process and motor activities changing agent's behavior. They affect agent's operations through global signals by changing thresholds for sensory inputs and activation of motor responses.

2.b. Cognitive motivations. It receives primitive pain signals from the Subconscious pains, rewards and emotions as discussed in 2.a. Based on the change of these signals, it creates higher level goals and sends pain information to Action evaluation block, when such information is requested through a bidirectional connection. Thus, useful actions can be learned and new motivations established, while harmful actions can be avoided. Another bidirectional connection to Cognitive motivations is from Attention focus block. If the system attention is focused on a cognitive motivation it may trigger action evaluation and planning. The link to Attention focus provides a priming signal in the semantic memory area.

3.a. Mental saccades. It provides a major search mechanism for internal thoughts and plans based on the primed signal values in the semantic memory. It receives inputs from all primed signals in the semantic memory and sends them to WTA attention switching block. Once declared a winner by WTA competition, a feedback signal from WTA attention switching is send back to attention focus highlighting selected area of the semantic memory. Notice, that many signals from the semantic memory (indicated here by a thick line) compete for attention together with interrupt signals from subconscious attention switching block, where the interrupt signals have priority over other competing signals.

3.b. Attention focus. Attention focus block highlights a selected concept in the semantic memory and is responsible for all cognitive aspects of the system operation. It provides cognitive interpretation to all observations, actions, plans, thoughts, and memories. Besides previously discussed bidirectional links to Sensory processing, Cognitive motivations, and Mental saccades, this block receives inputs from Action monitoring as well as bidirectional links to Subconscious attention switching and Episodic memory. It also sends triggering signals to Action evaluation block. These signals trigger response from Cognitive motivations requested by Action evaluation. The main role of the Attention focus in action evaluation is to provide information about the value of the planned action through cognitive motivations. A direct link from Attention focus to Action evaluation helps to assess the

feasibility of the planned action, while indirect link through Cognitive motivations provides value based assessment of the planned action. The link from Action monitoring provides the attention focus on the activities performed currently or those planned ahead, in order to complete or abort the action. Finally, the bidirectional link from Subconscious attention switching, helps to focus system's attention on the source of interrupt, and helps to cognitively recognize this source.

3.c. Episodic memory. It receives scene information from Episodic management block and writes it to sequential memories with significance information used to determine duration and importance of the episode stored in the memory. A link from Attention focus is used to recall episodes based on association with the focus of attention. A link to Attention focus is used for episodic recall and semantic memory learning. Finally, a link to Procedural memory is used to learn procedures based on the episodes. "Writing to memory" should not be confused here with computer memory operation, but rather should be understood as forming associations across multiple modalities, activating spatio-temporal storage in groups of neurons. Such storage could be obtained using LTM cells described in [17]

4.a. Procedural memory. This is a major block of memory responsible for performing learned actions. In this block, sequential memory cells are responsible for learning and recalling the learned sequences of operations. Initially, learned sequences involve more cognitive attention, but as the system performs learned operations more frequently, cognitive supervision is less intensive. Procedural memory has bidirectional links to action monitoring. Action monitoring is responsible for checking progress in action completion and preparing for the next stage in a sequence of operations. Through these links, Action monitoring is advancing the sequence of actions after a single step completion, and is receiving information about the next planned element of the action sequence. Finally, two one directional links out of Procedural memory control motor processing to execute the action and visual saccades to observe it.

4.b. Visual saccades. This block receives control signals from procedural memory to focus visual attention on a selected object or action. This includes searches of the environment in a special procedure that is responsible for the scene building. Another input to visual saccades is from Subconscious attention switching to help determine the source of the interrupt signal. Outputs from Visual saccades are to Motor processing, in order to implement saccadic movement, or to Sensory processing area to simulate such movement.

5.a. Action planning. Action planning is used in the planning stage to check cognitively all the aspects of the planned action and its value to the system. It initiates planned action monitoring, checking for consistency of the environment conditions with the planned action. Thus action planning relies on the information stored in the procedural memory to determine the sequence of operations needed to complete the planned action. Once the action plan is completed, action can be initiated by action evaluation block and is changed from planned action to action currently executed. If the plan cannot be completed, the planned action is abandoned and is removed from Action planning block.

5.b. Action evaluation. Action evaluation block responds to an opportunity of an action, when the Attention focus selects a new object or idea considered useful by the Cognitive motivation block. Action evaluation block activates action planning to initiate plan for a new action by setting an active link to the planned action. Action planning proceeds to plan the action according to a script in Procedural memory. Once action planning was successfully completed, Action evaluation initiates new action by activating link to a real action monitoring. Once the action is completed or abandoned such link is deactivated (removed from the working memory). It also removes link to a planned action, once it was determined that such action would be harmful or cannot take place due to environment conditions.

5.c. Action monitoring. It has, previously discussed, bidirectional link to Procedural memory to monitor the progress of sequential actions, and one directional links to Attention focus block. Another one directional link to WTA attention switching, forces focus of attention to switch to the planned action. Signals from Action monitoring to WTA attention switching have higher strength than signals from mental saccades, but lower than those from Subconscious attention switching. This helps to focus on the performed activities, while being alert to significant, observable changes in the external environment. Action monitoring block also has bidirectional links to Action planning and Action evaluation blocks. Incoming links initiate the process of action monitoring (real or planned), while outgoing links inform Action evaluation and Action planning about the progress made.

5.d. Scene building. Scene building is important for learning, episodic memory formation, active searches and orientation map building. It includes cognitive recognition of the scene elements and their location, activities performed, significance of the activities and the entire scene, time markers or other relative scene information. It receives one directional links from attention focus and visual saccades to obtain elements of the observed scene and their location. It uses visual saccade control signals to determine object locations, and it uses object attention focus for its cognitive meaning. It has a bidirectional link to motivations and goal creation, to determine significance of the scene and its elements.

5.e. Episodic management. Episodic management is a part of the working memory responsible for initiation of scene building and formation of episodes for episodic memory. Obtained scenes are arranged in episodes and are written to Episodic memory with significance information used to determine duration and importance of the episode stored in the memory. Once a scene is completed and is written to Episodic memory, a new scene is initiated by Episodic management.

6. Subconscious attention switching. It receives inputs from the sensory processing area and sends the interrupt signals to Attention focus. It also receives signals from subconscious pains, rewards and emotions to alert the system about pains and to help focus its attention on the pain source. Emotions

change response thresholds of Subconscious attention switching, resulting in a faster or slower response. Finally, it cooperates with Attention focus for object recognition and cognitive identification. It sends a blocking interrupt signal to WTA attention switching to prevent it from selecting a cognitive focus of attention.

7. WTA attention switching. WTA attention switching cooperates with the mental saccades mechanism to process thoughts, plans, and associations based on the semantic and episodic memory activities. It receives the interrupt signals from subconscious attention switching and action monitoring. Since subconscious attention switching takes precedence over cognitive mental saccades or action monitoring, its input blocks WTA attention switching from selecting a cognitive winner – instead focus of attention is shifted towards a dominating subconscious event.

C. *Analysis of the Scenario*

Let us analyze the described cognitive agent architecture to illustrate its activities in presented Scenario. As she walks, her procedural memory controls motor processing to sustain rhythmic walking pace. Since this activity is well developed and automated, she does not pay much attention to individual steps, reacting cognitively only to obstacles or unexpected events, like someone running in front of her on a skate board. This triggered her subconscious attention switching bringing her attention focus to the skateboarder, and interrupting her walk. The interruption was rapid and occurred before she realized that this was her neighbors' boy. It was her reaction to a sudden subconscious fear signal that came first (from subconscious pains and rewards) and interrupted her walk, with cognitive information about the neighbor boy followed in a brief succession. This second information was initiated by subconscious attention switching that focused her attention on the boy, followed by action evaluation that resumed her walking after she realized she was out of danger. Since she was quite frightened, this particular scene will be written by the scene building mechanism with higher significance, so she will have something to talk about at home. On the other hand, she did not pay much attention to how the boy was dressed. He must have been in his sports outfit she figured, but was not sure. She noticed a little scratch on his left elbow and that he was not wearing his helmet.

Her eyes glance around a familiar scene (visual saccades); she enjoys herself in the bright, warm sunlight (subconscious pains and rewards). She recognizes rusty leaves on trees (subconscious attention switching followed by attention focus) that indicate the beginning of the Fall season (cognitive association in semantic memory that activated mental saccade to Fall). She recalled last Fall full of red and yellow and orange leaves (episodic memory), some of which she picked up and saved to decorate her table during Thanksgiving (associated episodic memory).

She reached the cross section and had to stop on the red light (visual saccade to observe the cross section, attention focus on the cross section, action evaluation, and cognitive motivation to check the light, attention on the red light, action evaluation, and action monitoring to stop walking). After crossing the street, she turns left towards her home (action monitoring moves pointer in procedural memory forwards along her path home).

Her scene building mechanism registered the fact she had to stop on the red light, she knows this cross section well so most of the scene is filled-in automatically from her semantic memory, and the scene is written to her episodic memory with a low significance. Her memory of this scene will be short.

She noticed a sweat on her face and reached automatically to her pocket for a handkerchief (subconscious attention switching to focus attention on her sweat, action evaluation, check with cognitive motivation to remove unpleased feeling, action initiation and monitoring, motor processing).

Notice that since this action is concurrent with walking, she initiated another action in her working memory with a separate pointer to procedural memory to reach for handkerchief. This is followed by her drying her face and putting away the handkerchief. Then, as she was wiping her face, she recalled how much she was sweating after she finished her one mile run last week, and she smiled to herself – she was the best in her class (association to episodic memory, recall of her being sweat even more, followed by the scene from her episodic memory of her winning the race, pleasant feeling in cognitive motivations, with a smile showing her emotions).

III. IMPLEMENTATION.

Described in Section II building blocks of a cognitive agent are being implemented using software tools presented in this section.

A. *System Modeling and Simulation in Blender*

Blender [18] is a versatile open source graphical program that allows modeling 3D objects, rendering them, preparing animations, movies and video games. Python API (application programming interface) is one of Blender's most powerful and useful features. It allows to interface Blender with Python programming language for task automation. All operations available for the user in Blender graphical interface can also be accessed from scripts and programs and automated. All new versions of Blender (>2.5) use Python 3 and have new Python API and new, user friendly graphical interface.

B. *Python scripting*

Python is a general-purpose, interpreted, high-level programming language. It supports multiple programming paradigms: object-oriented, imperative and functional programming. Python is strong typed and it features a dynamic type system and automatic memory management [19]. It has a very comprehensive standard library as well as numerous third-party modules and packages available with an open-source license. Among them: SciPy package (Open Source Library of Scientific Tools), the open-source software for mathematics, science, and engineering. The SciPy [20] library depends on NumPy [21], which provides fast, convenient and matlab-like N-dimensional array manipulation. This feature makes using Python very useful in prototyping. In connection with Blender physical simulation features, this allows fast

prototyping of models of robots or agents. There are several robotics projects based on Blender [22].

Python supports Blender environment in many ways. For example, it is possible to write special purpose 3D modeling tools with their own GUI using Python scripts. But what is more important for building cognitive system models, is that it is possible to write user's own specialized controllers for simulations in the blender game engine. From the point of view of any simulation project this is its main advantage.

Using Blender as the simulation environment has a few downsides like the lack of detailed examples how to program game engine in new versions of Blender (>2.5). However, the Blender users' community is very active developing these examples.

C. *Blender Game Engine*

Simulations of the realistic physics behavior in blender are possible due to Blender Game Engine (BGE) module. This module uses a system of graphical logic blocks (called logic bricks) to control behavior and visualization of objects located in the 3D scene.

Interaction between different objects in BGE (as well as objects and environment) is based on available sensors, controllers and actuators (SCA components). Both sensors and actuators are predefined and are accessible for the user in Logic Editor view, but the user can only set and change its parameters. Controllers are the "carriers" of the agent's internal logic, so effectively, they define its behavior. We will present selected types of SCA components, available in BGE, in the subsequent sections.

All object's attributes and methods (functions) of the Blender Game Engine are available for users through Blender Python API, and all of them can be used in custom controllers implemented in Python programming language.

1) *Predefined Sensors*

The following sensors are predefined: Actuator, Always, Keyboard, Mouse, Touch, Collision, Near, Radar, Property, Random, Ray and Message sensor. Especially useful in modeling cognitive agents are: Actuator, Always, Collision, Near, Property, Radar, Ray, Touch [23].

- Actuator Sensor – is set off when other specified actuator is activated.
- Always Sensor – sensor used for things that need to be done every n-th logic tick. Logic ticks is a kind of universal trigger that can be used for different purposes. Pulses coming from sensors can trigger both controllers and actuators linked to it. Pulse can have two values TRUE or FALSE. Logic ticks have default frequency of 60 ticks per second. User can specify the kind of triggering using True level triggering or False level triggering as well as delay between repeated pulses (f parameter).
- Collision Sensor – works like a Touch sensor but is able to filter collision events by property. That means that only objects with selected property will generate a positive pulse upon the collision.
- Near Sensor - this sensor detects objects within a specific distance between them, it is able to filter objects by property.
- Property Sensor – detects changes in the properties of specified objects. This sensor can be set to generate positive pulse when: the property value matches the value in the sensor (Equal option), when the property value differs from the value in the sensor (Not Equal option), when the property value is between Min and Max values of the sensor (Interval option) and as soon as the property value changes (Changed option).
- Radar Sensor - It works similarly as Near sensor but detects objects only within a specific distance and only within an angle from an axis. User can specify such parameters as: local axis of the object, distance and angle. It "sees" through other objects. Radar is able to filter objects by property (Fig 2.a).
- Ray Sensor – It generates positive pulse once emitted ray in specified direction hits something. In X-Ray mode it "sees" through other objects (Fig. 2.b).
- Touch Sensor – it generates positive pulse when the object is in contact with another object, the negative pulse is generated once the objects are no longer in contact.

a)

b)

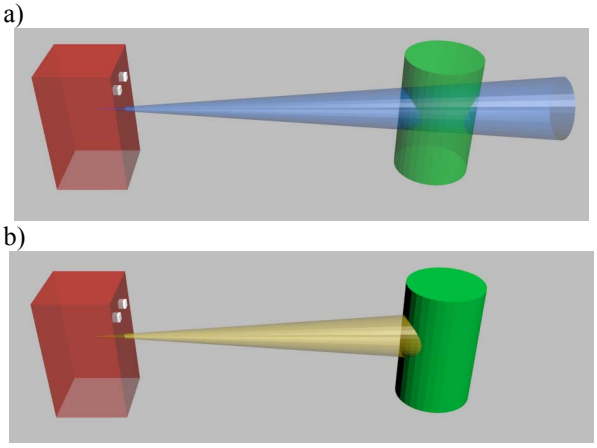

Figure 2. Visualization of the selected sensors: a) Radar Sensor – invisible cone with the top in the object's center, b) Ray Sensor

2) *Predefined Actuators*

Actuators available in Blender SCA include: Visibility, State, Sound, Steering, Scene, Random, Property, Parent, Motion, Message, Game, Filter 2D, Edit object, Constraint, Camera and Action actuator. Some of them have limited application to cognitive agent modeling. Most of them are intended to be used mainly in video games and animations.

In modeling behavior of cognitive agents especially useful are: Motion, Action and Camera actuators.

- Motion Actuator – sets object into motion and rotation. What's important: the motion and rotation details depends on the type of Physics setting of the object. This actuator can be used to moving whole agent or selected parts of it (selected objects). For example agent can move to other position on the scene while rotating its head (and look around).

- Action Actuator – it allows applying animation to objects. For example, it can be useful when the agent is moving forward through the scene with a rhythmical motion of its hands and legs. This requires a typical, automatic cyclic movement that needn't be controlled. Instead, it can be applied as an animation sequence through Action Actuator. This facility allows to simplify project especially in early stages because some basic functions (like arm movements) don't need to be implemented.
- Camera Actuator – it makes the camera follow selected object, which is very useful when scenes are vast and cannot be easily observed from one point of view.
- Property Actuator – it assigns a value or expression to the property. It can be used for detecting and counting specified events (for example: how many times agent touched specified object), counting agent's operation time.

Physics settings can influence behavior of defined in the game objects. In Blender Game Engine, objects can be one of five types: No Collision (for example: on screen display), Static (walls), Dynamic (simplified moving), Rigid Body (realistic moving, but object mesh cannot be deformed), Soft Body (object mesh can be deformed - objects like flags or clothes).

3) Controllers

Controllers are the essence of Blender game engine. Blender has two types of controllers: predefined controllers (logic operators: and, or, nand, nor, xor, xnor) and Python controllers. Python controllers are scripts/programs written in Python programming language using Blender Game Engine Python API (usually using BGE module). Python controllers allow to implement much more complicated functions using different sensors and actuators in a single controller.

4) Other facilities

Besides its modeling and simulation features, Blender has other properties that are very useful not only in implementing but also in simulating cognitive agents. For example, it allows such functions as moving cameras or multiple viewports. Moving camera lets one to observe different objects or actors during the simulation process, while multiple viewport lets visualize selected internal states of the actors.

Fig. 3 shows an example of two viewports.

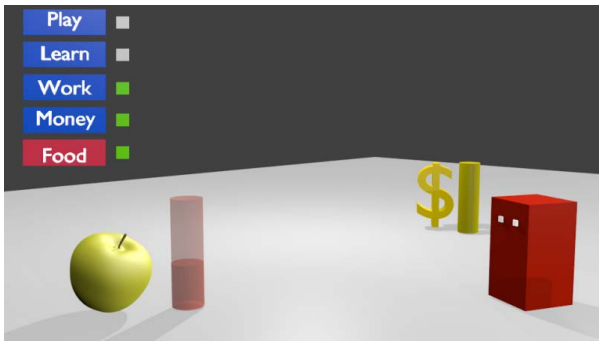

Figure 3. Multiple viewports.

The main viewport shows the agent looking for food. The second (in the top-left corner) shows agent's internal states. From illustrated in Fig. 3 five different resources that the agent can potentially learn, the agent currently knows only three (marked with green squares: Food, Money, Work). Red label (Food) indicates agent's current target.

Example Python script controller file 'viewport.py' defines multiple viewport combining views from two cameras.

```
import bge

def main():

    height = bge.render.getWindowHeight()
    width = bge.render.getWindowWidth()

    scene = bge.logic.getCurrentScene()
    objects_list = scene.objects

    agentcam = objects_list['CameraAgent']
    innercam = objects_list['CameraInner']

    agentcam.setViewport(0,0,width,height)
    innercam.setViewport(0,height-300,400,height)

    agentcam.useViewport = True
    innercam.useViewport = True
```

Figure 4. Example Python controller script.

D. *Agent implementation in BGE*

We have created simulation framework in Blender game engine. Basic functions, like movements, actions animations, of the cognitive agent have been completed; other will be implemented in future work. This approach let us to experiment with different modules of the cognitive agent separately.

Some of the building blocks of the cognitive agent or other objects used in the simulation can be implemented quite easily (using Blender features), but other need more design effort. In particular, the building blocks responsible for higher cognitive functions need to be implemented by careful programming of custom Python controllers. This problem is not trivial due to unreliable behavior of BGE mechanism of the collision detection (especially on custom meshes). Sometimes collisions are not detected which strongly affects the simulation results.

In our implementation the sensory-motor area (number 1 in the Fig. 1) are easily implemented using predefined sensors and actuators. We used Always, Touch, and Property sensors in defining agent's behaviors (mainly associated with interacting with other objects in the environment). We also used Motion, Action, and Property actuators in those tasks.

The Motivation and goal creation building block (Fig. 1, number 2) is more difficult to implement and it needs using Python script controllers. The algorithms, as well as the simulations of this building block, were elaborated in our previous work and are presented in papers [11],[24]. We implemented motivations and goal creation block as Python programs. These scripts will be incorporated into subsequent, more advanced versions of our cognitive agent. Using Blender lets us simplify our work. The motor control building block (Fig. 1, number 4) is beyond the scope of current

implementation, so we can simply automate its function using standard Blender functionality (III.C.2). At the current state of the research, our agent doesn't need advanced motor control, since we only want the agent to be able to move to various locations in the scene and manipulate objects. The motor control level that we need in this simulation is implemented through a simple animation sequence (and applied to agent using Action actuator).

E. Environment implementation in BGE

Useful for the agent's operation, things like environmental resources are modeled as objects with their own sensors, actuators and controllers. Information about the object state in Blender game engine is stored using property mechanism. In our simulation, the information about availability of the environmental resources as well as the state of the other objects is stored in properties associated with specified object. The whole behavior of several entities in the environment is implemented using property sensors and actuators. For example, when the agent wants to use a resource, it only has to find it and touch it. The object detects this event using Touch or Near sensor. Next, the object is changed in accordance with objects associated properties. There is no need to write custom Python scripts to implement this behavior.

Our experiments with implementing the cognitive agent showed that selected parts of the agent can be modeled and simulated in Blender Game Engine. Full model is currently under development and testing stage. We are developing both the agent and its environment, testing its abilities to autonomously learn and work in a virtual world, while interacting with objects equipped with various physical properties.

IV. CONCLUSION

In this paper, we presented and discussed the memory organization of a cognitive agent. We showed an example scenario in order to indicate activities of agent's cognitive memory internal blocks during typical agent's activity in the real world. We also showed how to implement selected internal blocks in Blender Game Engine structures. In addition, we presented the most useful functions of Blender environment in modeling and simulation of cognitive agents.

Blender Game Engine together with Python scripting language is useful in many aspects of simulation and visualization of cognitive autonomous agents. It can be used as a framework which allows to develop and experiment with building blocks of the cognitive agent memory. Many advantages of using BGE include fast prototyping, realistic physics, and flexibility to model virtual environments.

V. ACKNOWLEDGMENT

This work was supported by the grant from the National Science Centre DEC-2011/03/B/ST7/02518.

REFERENCES

[1] R.A. Brooks, New approaches to robotics. Science, Science (253), September 1991, pp. 1227–1232.

[2] R. Pfeifer, J.C. Bongard, How the Body Shapes the Way We Think: A New View of Intelligence, The MIT Press (Bradford Books), 2007.

[3] T. Ziemke, On the Role of Robot Simulations in Embodied Cognitive Science, pp. 389—399, AISB Journal 1(4), 2003.

[4] R. Sutton, A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

[5] J. Peters, S. Vijayakumar, S. Schaal, Reinforcement learning for humanoid robotics. In: Humanoids 2003, Third IEEE-RAS Int. Conf. on Humanoid Robots, Karlsruhe, Germany, Sept.29-30, 2003.

[6] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, F. Nori, An open-source simulator for cognitive robotics research: The prototype of the iCub humanoid robot simulator, Workshop on Performance Metrics for Intelligent Systems, Washington DC, USA, August 19-21, 2008.

[7] Natale, F. Nori, G. Metta, M. Fumagalli, S. Ivaldi, U. Pattacini, M. Randazzo, A. Schmitz, G. Sandini, The iCub platform: a tool for studying intrinsically motivated learning, *Intrinsically motivated learning in natural and artificial systems,* Springer-Verlag Ed.Baldassarre, Gianluca and Mirolli, Marco, 2012.

[8] V. Tikhanoff, P. Fitzpatrick, F. Nori, L. Natale, G. Metta, A. Cangelosi, The iCub humanoid robot simulator. IROS 2008 workshop on robot simulators: available software, scientific applications and future trends. Nice, France, September 22, 2009.

[9] J. A. Starzyk, "Motivation in Embodied Intelligence," in *Frontiers in Robotics, Automation and Control*, I-Tech Education and Publishing, Oct. 2008, pp. 83-110.

[10] J. A. Starzyk, "Motivated Learning for Computational Intelligence," in *Computational Modeling and Simulation of Intellect: Current State and Future Perspectives*, edited by B. Igelnik, IGI Publishing, ch.11, pp. 265-292, 2011.

[11] P. Raif, J.A. Starzyk, "Motivated Learning In Autonomous Systems," Int. Joint Conf. on Neural Networks (IJCNN), San Jose, California, July 31 - August 5, 2011.

[12] W. S. Bainbridge, "The scientific research potential of virtual worlds," Science, vol. 317, no. 5873, pp. 421–534, July 2007.

[13] Wright T., Madey G., 2008 "A Survey of Collaborative Virtual Environment Technologies." Tech. Rep. University of Notre Dame, USA, 2008. http://www.cse.nd.edu/Reports/2008/TR-2008-11.pdf

[14] J. Faust, Ch. Simon, W.D. Smart. A Video Game-Based Mobile Robot Simulation Environment, In "Proceedings of the IEEE/RSJ International Conference on Robots and Systems (IROS 2006)", pages 3749-3754 , Beijing, China, October 2006.

[15] J. Wang , S. Chick, P. J. Sánchez , D. Ferrin , D. J. Morrice, A Game Engine Based Simulation of the NIST Urban Search & Rescue Arenas, Proceedings of the 2003 Winter Simulation Conference, 2003.

[16] J.A. Starzyk, "Mental Saccades in Control of Cognitive Process", Int. Joint Conf. on Neural Networks, San Jose, CA, July 31 - Aug. 5, 2011.

[17] V. A. Nguyen, J. A. Starzyk, W-B. Goh, D. Jachyra, "Neural Network Structure for Spatio-Temporal Long-Term Memory" IEEE Trans. Neural Networks and Learning Syst., vol. 23 no. 6, June, 2012, pp. 971-983.

[18] Blender, http://www.blender.org/

[19] Python, http://www.python.org/

[20] SciPy, www.scipy.org/

[21] NumPy, http://numpy.scipy.org/

[22] MORSE, http://http://www.openrobots.org/wiki/morse/

[23] Blender 2.6 Manual, http://wiki.blender.org/index.php/Doc:2.6/Manual

[24] J. A. Starzyk, J. T. Graham, P. Raif, and A-H. Tan, "Motivated Learning for Autonomous Robots Development", Cognitive Systems Research v.14, no.1, 2012, p.10(16) pp. 10-25.