The awakening of conscious bots:

Conscious-Robots team, winner of the *2K BotPrize 2010* contest, explains how their bot is designed.

by Raúl Arrabales and Jorge Muñoz

Most of current efforts in the development of believable bots – bots that behave like human players – are based on classical AI techniques. These techniques are based on relatively old principles, which nevertheless are being progressively improved or wisely adapted increasing their performance in order to satisfy new game requirements. Taking a different perspective, the approach that we adopted for the design of our bot (CC-Bot2) was rather opposed to this trend. Specifically, we implemented a computational model of the Global Workspace Theory (Baars, 1988), a kind of shared memory space where different agents – that we call *specialized processors* - can collaborate and compete with each other dynamically (see Figure 1). We believe that applying new techniques from the field of Machine Consciousness might also provide good results, even in the short term.

In this article we briefly describe the design of CC-Bot2, the winning Unreal Tournament bot developed by the Conscious-Robots team for the third edition of the 2K BotPrize. The BotPrize competition is a version of the Turing test adapted to the domain of FPS video games (Hingston, 2009). The ultimate goal of the contest is to develop a computer game bot able to behave the same way humans do. Furthermore, a bot would be considered to pass the Turing test (in this particular domain) if it is undistinguishable from human players.

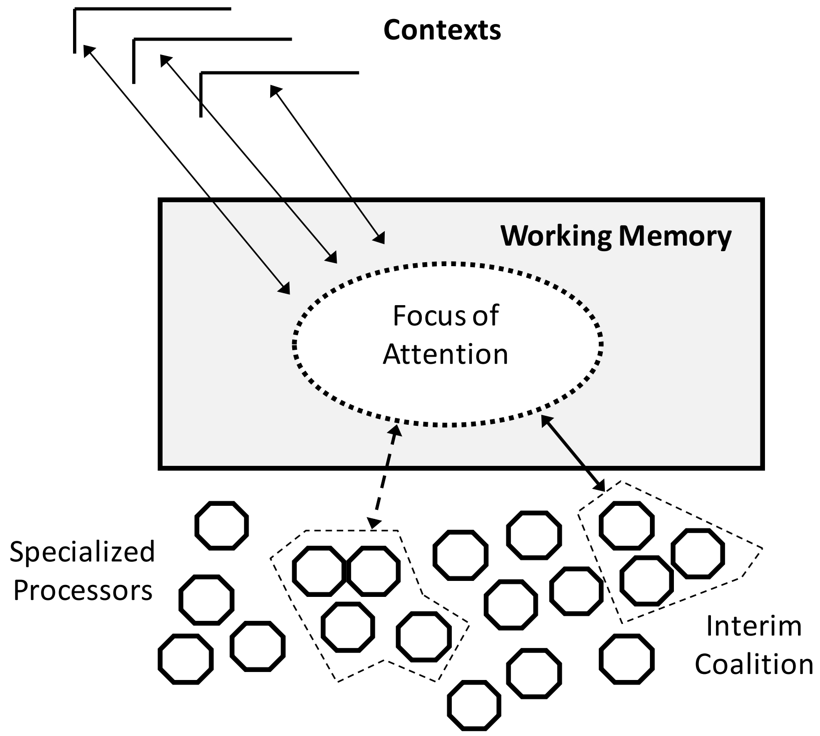

Figure 1. Global Workspace Model.

1. CERA-CRANIUM Cognitive Architecture and CC-Bot2

As a result of our research line on Machine Consciousness we have developed a new cognitive architecture called CERA-CRANIUM (Arrabales et al. 2009), which has been the basis for the development of CC-Bot2 (CERA-CRANIUM Bot 2). CERA-CRANIUM is a cognitive architecture, designed to control autonomous agents, like physical mobile robots or Unreal Tournament bots, and based on a computational model of consciousness. The main inspiration of CERA-CRANIUM is the Global Workspace Theory (Baars, 1988). CC-Bot2 is a Java implementation of the CERA-CRANIUM architecture specifically developed for the 2K BotPrize competition.

CERA-CRANIUM consists of two main components (see Figure 2):

- CERA, a control architecture structured in layers, and
- CRANIUM, a tool for the creation and management of high amounts of parallel processes in shared *workspaces*.

As we explain below, CERA uses the services provided by CRANIUM with the aim of generating a highly dynamic and adaptable perception processes orchestrated by a computational model of consciousness.

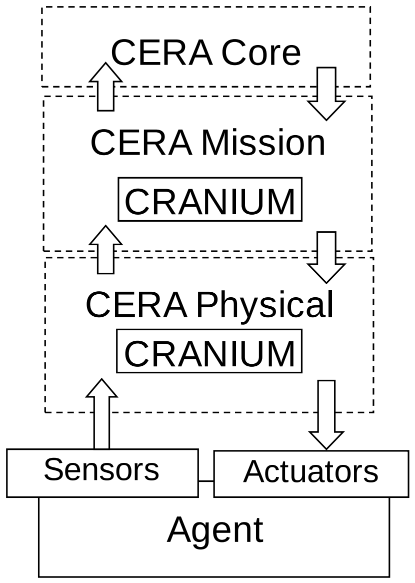

Figure 2. Overview of CERA-CRANIUM architecture.

Basically, in terms of controlling a bot, CERA-CRANIUM provides a mechanism to synchronize and orchestrate a number of different specialized processors that run concurrently.

These processors can be of many kinds, usually they are detectors for given sensory conditions, like the "*player approaching detector*" processor, or they are behavior generators, like the "*run away from that bully*" processor.

1.1 CERA

CERA is a layered cognitive architecture designed to implement a flexible control system for autonomous agents. Current definition of CERA is structured in four layers (see Figure 3): sensory-motor services layer, physical layer, mission-specific layer, and core layer. As in classical robot subsumption architectures, higher layers are assigned more abstract meaning; however, the definition of layers in CERA is not directly associated to specific behaviors. Instead, they manage any specialized processors that operate on the sorts of representations that are handled at that particular level, i.e. physical layer deals with data representations closely related to raw sensory data, while mission layer deals with more high-level task-oriented representations.

CERA sensory-motor services layer comprises a set of interfacing and communication services which implement the required access to both sensor readings and actuator commands. These services provide the physical layer with a uniform access interface to agent's physical (or simulated) machinery. In the case of CC-Bot2, the CERA sensory-motor layer is basically an adaptation layer to Pogamut 3.

CERA physical layer encloses agent's sensors and actuators low-level representations. Additionally, according to the nature of acquired sensory data, the physical layer performs data preparation and preprocessing. Analogous mechanisms are implemented at this level with actuator commands, making sure for instance that command parameters are within safety limits. The representation we have used for sensory data and commands in CC-Bot2 physical layer is, in most of the cases, actually that of Pogamut 3, like "*player appeared in my field of view*" or "*I am being damaged*".

CERA mission-specific layer produces and manages elaborated sensory-motor content related to both agent's vital behaviors and particular missions (in the case of a deathmatch game the mission is relatively clear and simple). At this stage single contents acquired and preprocessed by the physical layer are combined into more complex pieces of content, which have some specific meaning related to agent's goals (like "*this player is my enemy*" or "*enemy x is attacking me*"). The mission-specific layer can be modified independently of the other CERA layers according to assigned tasks and agent's needs for functional integrity.

CERA core layer, the highest control level in CERA, encloses a set of modules that perform higher cognitive functions. The definition and interaction between these modules can be adjusted in order to implement a particular cognitive model. In the case of CC-Bot2, the core layer contains the code for the attention mechanism (many other modules could be added in the future). The main objective of these core modules is to regulate the way CERA lower layers work (the way specialized processors run and interact with each other).

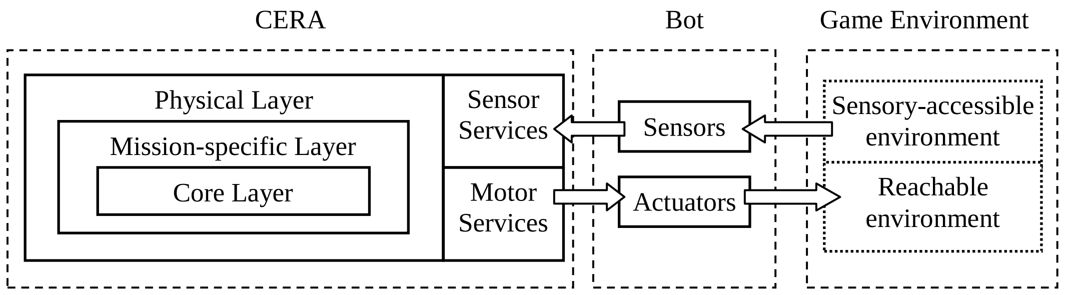

Figure 3. CERA cognitive architecture layered design.

Physical and mission-specific layers are characterized by the inspiration on cognitive theories of consciousness, where large sets of parallel processes compete and collaborate in a shared workspace in the search of a global solution. Actually, a CERA controlled agent is endowed with two hierarchically arranged workspaces which operate in coordination with the aim to find two global and interconnected solutions: one is related to perception and the other is related to action. In short, CERA has to provide an answer for the following questions continuously:

- What must be the next content of agent's conscious perception?
- What must be the next action to execute?

Typical agent control architectures are focused on the second question while neglecting the first one. Here we argue that **a proper mechanism to answer the first question is required in order to successfully answer the second question in a human-like fashion**. Anyhow, both questions have to be answered taking into account safety operation criteria and the mission assigned to the agent. Consequently, CERA is expected to find optimal answers that will eventually lead to human-like behavior. As explained below, CRANIUM is used for the implementation of the workspaces that fulfill the needs established by the CERA architecture.

1.2 CRANIUM

CRANIUM provides a subsystem in which CERA can execute many asynchronous but coordinated concurrent processes. In the CC-Bot2 implementation (Java), CRANIUM is based on a task dispatcher that dynamically creates a new execution thread for each active processor. A CRANIUM workspace can be seen as a particular implementation of a *pandemonium*, where daemons compete with each other for activation. Each of these daemons or specialized processors is designed to perform a specific function on certain types of data. At any given time the level of activation of a particular processor is calculated based on a heuristic estimation of how much it can contribute to the global solution currently sought in the workspace. The concrete parameters used for this estimation are established by the CERA core layer. As a general rule, CRANIUM workspace operation is constantly modulated by commands sent from the CERA core layer.

In CC-Bot2 we use two separated but connected CRANIUM workspaces integrated within the CERA architecture. The lower level workspace is located in the CERA physical layer, where specialized processors are fed with data coming from CERA sensor services (Pogamut). The second workspace, located in the CERA mission-specific layer, is populated with higher-level specialized processors that take as input either the information coming from the physical layer or information produced in the workspace itself (see Figure 4). The perceptual information flow is organized in packages called single percepts, complex percepts, and mission percepts.

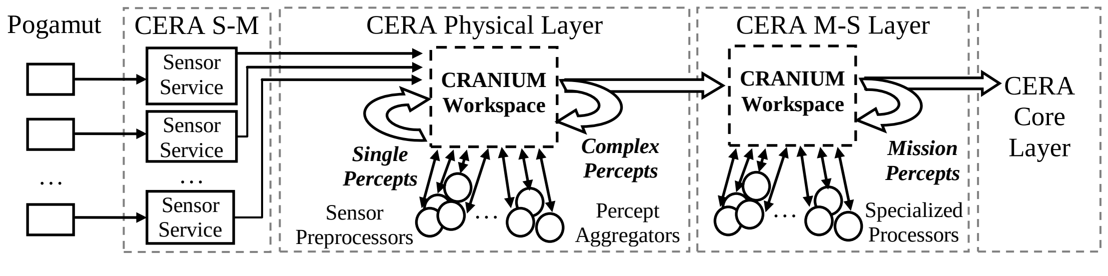

Figure 4. CERA-CRANIUM bottom-up flow: perception.

In addition to the bottom-up flow involving perception processes, a top-down flow takes place simultaneously in the same workspaces in order to generate bot's actions. Physical layer and mission-specific layer workspaces include single actions (directly translated into Pogamut commands), simple behaviors, and mission behaviors (see Figure 5).

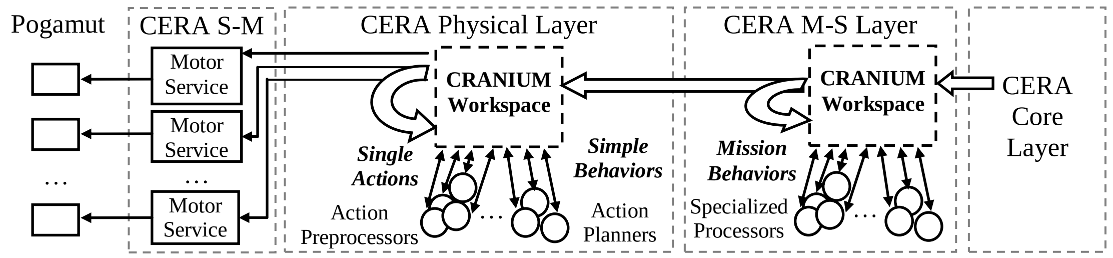

Figure 5. CERA-CRANIUM top-down flow: behavior generation.

One of the key differences between CERA-CRANIUM bottom-up and top-down flows is that while percepts are being iteratively composed in order to obtain more complex and meaningful representations, high level behaviors are iteratively decomposed until a sequence of atomic actions is obtained. Top-down flow could be considered, to some extent, to be equivalent to behavior trees, in the sense that behaviors are associated to given contexts or scopes. However, the way CERA-CRANIUM selects the next action is quite different, as current active context is periodically updated by the CERA Core layer. At the same time, the active context is calculated based on input from the sensory bottom-up flow. Having an active context mechanism implies that out of the set of possible actions that could be potentially executed; only the one which is located closer to the active context will be selected for execution. In the next subsection, we describe how the behavior of the agent is generated using this approach.

1.3 Behavior generation

Having a shared workspace, where sensory and motor flows converge, facilitates the implementation of the multiple feedback loops required for adapted and effective behavior. The winning simple behavior is continuously confronted to new options generated in the physical layer, thus providing a mechanism for interrupting behaviors in progress as soon as they are no longer considered the best option. In general terms, the activation or inhibition of perception and behavior generation processes is modulated by CERA according to the implemented cognitive model of consciousness. In other words, behaviors are assigned an activation level according to their distance to the active context in terms of the available sensorimotor space. Only the most active action is the one executed at the end of each “cognitive cycle”.

Distance to a given context is calculated based on sensory criteria like relative location and time. For instance, if we have two actions: Action A: “shoot to the left” and Action B: “shoot to the right”, and an active context pointing to the left side of the bot (because there is an enemy there), action A will be most likely selected for execution, and action B will be either discarded or kept in the execution queue (while it is not too old).

Figure 6 shows a schematic representation of typical feedback loops produced in the CERA architecture. These loops are closed when the consequences of actions are perceived by the bot, triggering adaptive responses at different levels.

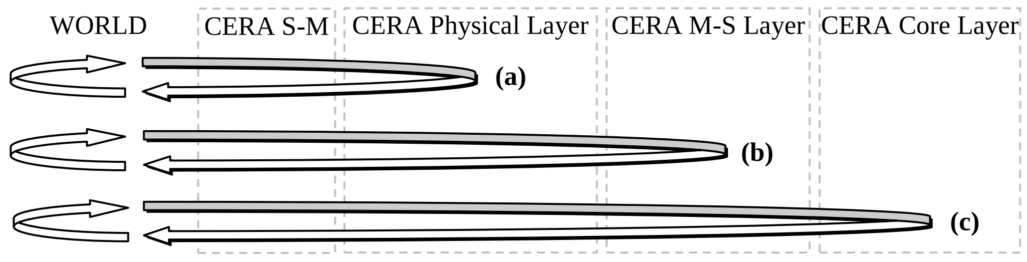

Figure 6. Different feedback loops produced in the CERA-CRANIUM.

Curve (a) in Figure 6 represents the feedback loop produced when an instinctive reflex is triggered. Figure 6 curve (b) corresponds to a situation in which a mission-specific behavior is being performed unconsciously. Finally, curve (c) symbolizes the higher level control loop, in which a task is being performed consciously. These three types of control loops are not mutually exclusive; in fact, same percepts will typically contribute to simultaneous loops taking place at different levels.

CRANIUM workspaces are not passive short-term memory mechanisms. Instead, their operation is affected by a number of workspace parameters that influence the way the pandemonium works. These parameters are set by commands sent to physical and mission-specific layers from the CERA core layer. In other words, while CRANIUM provides the mechanism for specialized functions to be combined and thus generate meaningful representations, CERA establishes a hierarchical structure and modulates the competition and collaboration processes according to the model of consciousness specified in the core layer. This mechanism closes the feedback loop between the core layer and the rest of the architecture: core layer input (perception) is shaped by its own output (workspace modulation), which in turn determines what is perceived.

2. The CC-Bot2 Implementation

In the following table some of the main specialized processors implemented in CC-Bot2 are briefly described (note that a number of processors performing the very same task but using different techniques might coexist in the same workspace).

Specialized Processor	**CERA Layer**	**Task**
AttackDetector	Physical	To detect conditions compatible with enemy attacks (health level decreasing, enemy fire, etc.).
AvoidObstacle	Physical	To generate a simple avoiding obstacle behavior.

BackupReflex	Physical	To generate a simple backup movement in response to an unexpected collision.
ChasePlayer	Mission	To generate a complex chasing player behavior.
EnemyDetector	Physical	To detect the presence of an enemy based on given conditions, like previous detection of an attack and presence of other players using their weapons.
GazeGenerator	Physical	To generate a simple gaze movement directed towards the focus of attention.
JumpObstacle	Physical	To generate a simple jump movement in order to avoid an obstacle.
KeepEnemiesFar	Mission	To generate a complex run away movement in order to maximize the distance to detected enemies.
LocationReached	Physical	To detect if bot has reached the spatial position marked as goal location.
MoveLooking	Physical	To generate a complex movement combining gaze and locomotion.
MoveToPoint	Physical	To generate a simple movement towards a given location.
ObstacleDetector	Physical	To detect the presence of an obstacle (which might prevent the bot to follow her path).
RandomNavigation	Physical	To generate a complex random wandering movement.
RunAwayFromPlayers	Mission	To generate a complex movement to run away from certain players.
SelectBestWeapon	Mission	To select the best weapon currently available.
SelectEnemyToShoot	Mission	To decide who is the best enemy to attack to.

In our current implementation, specialized processors are created programmatically (see sample code below), and they are also assigned dynamically to their corresponding CERA layer. It is our intention to create a more elegant mechanism for the programmer to define the processors layout (configuration text file or even a GUI).

```
[…]

// ** ATTACK DETECTOR *
// Generates a BeingDamaged percept every time the health level decreases
_CeraPhysical.RegisterProcessor( new AttackDetector() );

// ** OBSTACLE DETECTOR **
// Generates a Obstacle single percept if there is any obstacle in the
// direction of the movement
_CeraPhysical.RegisterProcessor( new ObstacleDetector() );

// ** EMEMY DETECTOR **
// Generates a Enemy Attacking complex percept every time the bot is
// being damaged and possible culprit/s are detected.
_CeraMission.RegisterProcessor( new EnemyDetector() );

[…]
```

3. Conscious-Robots Bot in action

The following is an excerpt of a typical flow of percepts that ultimately generates the bot's behavior (see Figure 7):

1. The processor *EnemyDetector* detects a new enemy, and creates a new "*enemy detected*" percept.

2. The “*enemy detected*” percept is in turn received by the *SelecEnemyToShoot* processor, which is in charge of selecting the enemy to shoot. When an enemy is selected, the corresponding fire action is generated.

3. Two processors receive the fire action, one in charge of aiming at the enemy and shoot, and other that creates new movement actions to avoid enemy fire.

4. As the new movement actions have more priority than actions triggered by other processors, like the *RandomMove* processor, these actions are more likely to be executed.

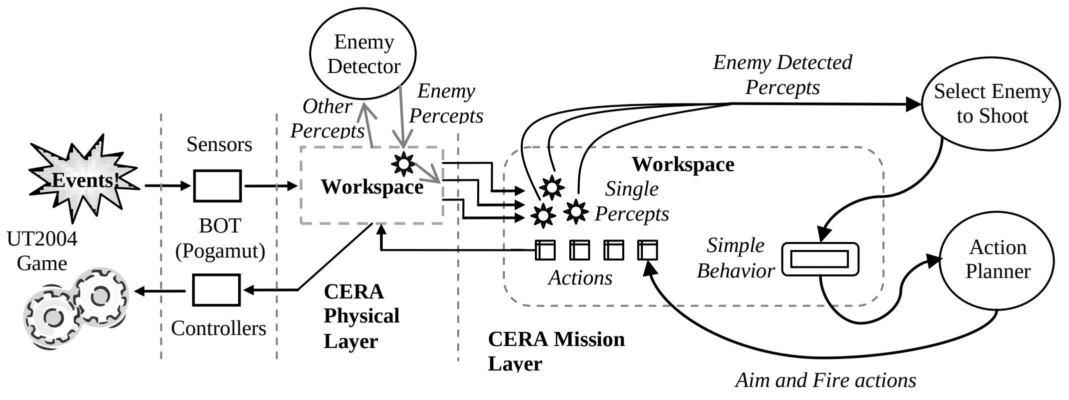

Figure 7. Simplified scheme of percept and action flow in CERA-CRANIUM.

This is a very simple example that how the bot works. However, it is usual to have much more complex scenarios in which several enemies are attacking the bot simultaneously, and the selected target might be any of them. In these cases, the attention mechanism plays a key role. CERA-CRANIUM implements an attention mechanism based on active contexts. Percepts that are closer to currently active context are more likely to be selected and further processed. This helps maintaining more coherent sequences of actions.

The following video shows the bot in action.

http://www.youtube.com/watch?v=9pmYPROqoxM

4. Future Work

CC-Bot2 is actually a partial implementation of the CERA-CRANIUM model. Our Machine Consciousness model includes much more cognitive functionality that is unimplemented so far. It is our aim to enhance the current implementation with new features like a model of emotions, episodic memory, different types of learning mechanisms, and even a model of the self. After a hard work, we expect CC-Bot3 to be a much more human-like bot. We also plan to use the same design for other games like TORCS or Mario.

Although CC-Bot2 could not completely pass the Turing test, it achieved the highest humanness rating (31.8%). As of today, the Turing test level intelligence has never been achieved by a machine. There is still a long way to go in order to build artificial agents that are clever enough to parallel human behavior. Nevertheless, we think we are working in a very promising research line to achieve this ambitious goal.

Acknowledgements

We wish to thank Alex J. Champandard for his helpful suggestions and comments on the drafts of this article.

References

Philip Hingston. **A Turing Test for Computer Game Bots**, **IEEE** Transactions on Computational Intelligence and AI In Games, Vol. 1, No. 3, pp 169-186, September 2009.

Arrabales, R. Ledezma, A. and Sanchis, A. "**CERA-CRANIUM: A Test Bed for Machine Consciousness Research**". International Workshop on Machine Consciousness 2009. Hong Kong. June 2009.

Baars, B.J. 1988. A Cognitive Theory of Consciousness: Cambridge University Press (**About GWT**).

Arrabales, R. Ledezma, A. and Sanchis, A. "**Towards Conscious-like Behavior in Computer Game Characters**", in Proceedings of the IEEE Symposium on Computational Intelligence and Games 2009 (CIG-2009) pp. 217-224. ISBN 978-1-4244-4815-9.

Muñoz, J., Arrabales, R. et al., “2K BotPrize 2010 winner bot: steps toward passing the Turing test”. Forthcoming.