Crystal Cassie: Use of a 3-D Gaming Environment for a Cognitive Agent

John F. Santore and Stuart C. Shapiro
University at Buffalo, The State University of New York
201 Bell Hall Box 602000
Buffalo, NY 14260-2000
{jsantore|shapiro}@cse.buffalo.edu

Abstract

The SNePS research group has built several implementations of an embodied computational cognitive agent called Cassie, based on the Grounded Layered Architecture with Integrated Reasoning (GLAIR). In this document we describe a new implementation, in which Cassie's body and the world are simulated in Crystal Space, an environment for building 3-D games. We describe the implementation of Cassie in a Crystal Space environment including her current suite of actions and her simulated vision system. Crystal Cassie is a tool for cognitive modeling and testing cognitive theories. We briefly discuss our first use of the resulting simulated cognitive agent and our experiences using an off-the-shelf 3-D product to build a cognitive agent.

1 Introduction

The SNePS research group has built several implementations of an embodied computational cognitive agent called Cassie [Shapiro, 1998; Shapiro et al., 2000; Shapiro and Ismail, 2001], based on the Grounded Layered Architecture with Integrated Reasoning (GLAIR) [Hexmoor et al., 1993; Hexmoor and Shapiro, 1997; Shapiro and Ismail, 2003]. There has been one major hardware implementation of Cassie, using a commercial Nomad robot, and several simulated versions using various graphical user interfaces for her environment and to display her behavior to human observers. In this document, we describe a new implementation, in which Cassie's body and the world are simulated in Crystal Space [Tyberghein et al., 2002], an environment for building 3-D games. In Section 2 we briefly describe the architecture we are using. In Section 3, we discuss the implementation of the architecture and how the pieces of the simulation fit together. In Section 4 we briefly describe the current first use of this simulation as a tool for cognitive modeling. And in Section 5 we discuss our experiences using Crystal Space as an off-the-shelf 3-D gaming environment.

2 The Architecture

The simulation is based on the Grounded Layered Architecture with Integrated Reasoning (GLAIR) [Hexmoor et al., 1993; Hexmoor and Shapiro, 1997; Shapiro and Ismail, 2003], a three-layer architecture for cognitive robotics and modeling cognitive agents. It consists of the Knowledge Level, the Perceptuo-Motor Level, and the Sensory-Actuator Level.

The Knowledge Level (KL) is the "conscious" level of the cognitive agent. It is the location of symbols accessible to reasoning and to natural language interaction, including the "abstract-level representations of objects" discussed in [Coradeschi and Saffiotti, 2001a; Coradeschi and Saffiotti, 2001b; Shapiro and Ismail, 2003]. That is, the KL is the location of the concepts that the cognitive agent has, including its concepts of objects in the world.

The Perceptuo-Motor Level (PML) is the level of the cognitive agent's physical representation of objects. At this level, the objects are represented by n-tuples of their physical characteristics such as shape, material, and size, rather than by their KL concepts. The PML is also the location of the cognitive agent's well defined skills, including natural language understanding and generation and the primitive actions of the KL—those skills which do not require conscious reasoning from the agent. In practice, the PML is often separated into three parts [Shapiro, 1998; Shapiro and Ismail, 2003], referred to as the PMLa, PMLb, and PMLc. Details of the implementation are to be found in Section 3, below.

The Sensory-Actuator Level (SAL) is where the low level control of the cognitive agent's motors and sensors (real or simulated) is located.

3 The Implementation of a GLAIR-based Cognitive Agent in a Crystal Space Environment

3.1 The Four Processes of Crystal Cassie

The Crystal Space version of Cassie (Crystal Cassie) is composed of four separate processes. Two of the processes explicitly implement parts of the GLAIR architecture, one is used for natural language interaction with a human user, and the fourth contains the simulation of the agent's physical body and the world itself. The processes are connected using standard IP sockets. Each of these processes is described in more detail below. Figure 1 shows the four processes, their socket connections, and the parts of the GLAIR architecture that

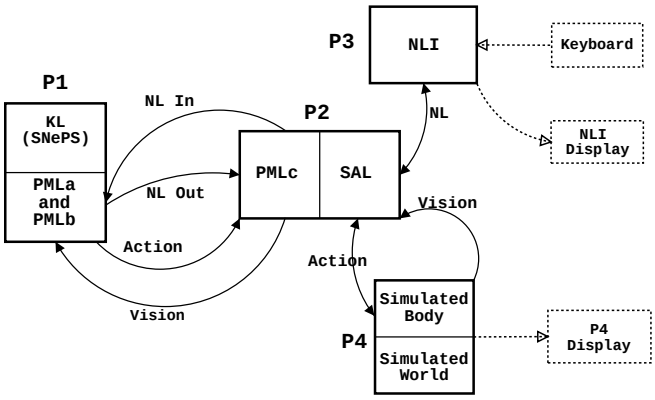

Figure 1: A schematic of the four processes, showing how they communicate with each other and with the user, and how the various parts of the GLAIR architecture are distributed among them.

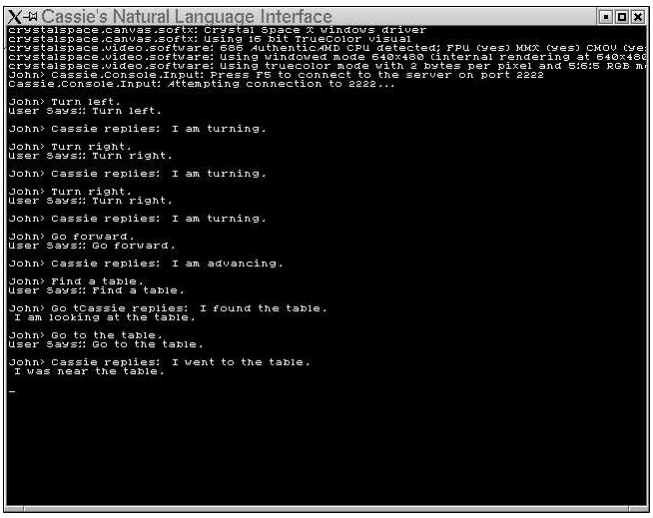

Figure 2: The users view of the NLI process during a sample interaction.

they implement. The figure also shows the interface between Crystal Cassie and a human user/interlocutor/observer.

Process P1 implements the KL, and the top two parts of the PML. It is is written in Allegro Common Lisp. The KL is identical to the FEVAHR knowledge level described in [Shapiro and Ismail, 2003] but with different domain knowledge. It is implemented using the SNePS knowledge representation and reasoning system [Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992; Shapiro and the SNePS Implementation Group, 2002]. Symbols in the KL are implemented as terms in the SNePS logic [Shapiro, 1993; Shapiro, 2000].

Process P1 also contains parts of the PML. The top sublevel of the PML (the PMLa) is where the agent's well-defined skills (KL primitive acts) are implemented. It is also the place where natural language understanding and generation occurs. Natural language interaction is implemented using a GATN grammar [Shapiro, 1982; Shapiro, 1989].

The second sub-level of the PML (the PMLb) implements the connection between the PMLa and the rest of the simulated agent. In previous simulations, [Shapiro, 1998; Shapiro and Ismail, 2003], the connection was made using the Lisp foreign function interface. In Crystal Cassie, the connections are made using socket connections. The PMLb has four one way connections to the PMLc (described below). These connections each represent one of the agent's cognitive modalities [Shapiro and Ismail, 2003].

The first connection is a one-way natural language (NL) input connection (a "hearing" modality), over which sentences are sent as strings. These strings are then sent to the GATN parser at the PMLa level, which translates them into the SNePS KR language.

The second connection is a one-way NL output connection (the agent's "speech" modality) from the PMLb to the PMLc. Sentences and phrases from the GATN generator are sent through this connection.

The Action connection (the movement modality) is used to send action requests from the PMLb to the PMLc/SAL layer.

The Vision connection (the vision modality) is a one-way connection from the PMLc to the PMLb, over which is sent a series of n-tuples with the relevant features of all of the objects that are currently visible to the agent's simulated vision system. See Section 3.4 for more about the simulated vision system.

The remaining three processes all currently use the Crystal Space tools [Tyberghein et al., 2002] and are written in C++.

Process P2 implements the PMLc and the SAL. The PMLc mediates between the PMLb and the SAL. This process controls Cassie's sensors and actuators and connects all the other processes to each other. The PMLc has the four socket connections to the PMLb discussed above. The SAL has three more connections to the remaining processes. Each of these connections and its function is described below in the paragraph devoted to the process it is connected to.

The Natural Language Interface (NLI) process (P3) uses the crystal space console window as a console that the user uses to type natural language text to the agent. This process connects to the SAL via a socket connection using the Crystal Space C++ socket wrappers. The NLI sends one sentence at a time, as a single line, to the SAL, and listens to its network connection for replies. It prints the full text of any text it receives prepended with the string "Cassie replies:" to its display. Figure 2 shows the NLI display.

The last process, P4, implements the simulation of Cassie's body, and the simulation of the environment that Cassie interacts with. P4 displays what Cassie can see, showing the results of her interactions with the environment. See Figures 3 and 6 for views of what P4's display looks like. The simulated body connects to the SAL with two socket connections. The Action Connection is a two-way connection. The simulated robot body receives action requests (requests to turn motors on or off, forward or backward) from the SAL. Cassie's body will then act on those requests. These actions may or may not produce a change in the world. (e.g. asking Cassie to move forward when she is up against a table will not accomplish anything.) In order to better model ac-

Figure 3: The P4 display showing a sample view of what Cassie can see: a computer lab with two people in it.

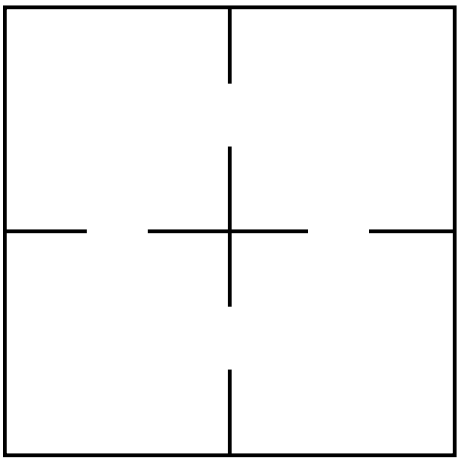

Figure 4: The floor plan of the smaller suite of interconnected rooms.

tual robots, there is some uncertainty built into the simulated robot. When the simulated body receives action requests, the actual distance traveled may or may not be the amount requested by the higher levels. When an action is finished (with either success or failure) the simulated body sends the SAL a short "action complete" message over the Action Connection.

The Vision Connection is a one-way socket connection from the simulated robot body to the SAL. The body sends vision information in the form of a string consisting of a space-deliminated 4-tuple of {material, shape, location, size} values. The SAL translates this simulated sensor data into the "physical-level representations" [Coradeschi and Saffiotti, 2001a; Coradeschi and Saffiotti, 2001b; Shapiro and Ismail, 2003] appropriate to the PML.

Cassie's simulated vision system can only see and send visual information about objects which appear within her first person perspective of the world. See Figures 3 and 6 for samples of what Cassie can see. Cassie's PML only receives feature tuples for those objects within her visual field, that is, those shown on P4's display.

Together, these four processes implement a working cognitive agent based on the GLAIR robotic architecture.

3.2 Crystal Cassie's Environments.

Crystal Space environments are defined in a Crystal Space specific XML text file format. The file defines the geometry of all of the objects in the environment, including the rooms, and materials associated with those objects.

Objects are first defined by a series of vertices given by <x, y, z> world-space coordinates; the x, y and z values of the coordinates are each floating point numbers. These vertices are then used to define polygons. In Crystal Space, as an optimization, only one side of a polygon is visible: the visible side is the one defined by the list of vertices in clockwise order. Polygons are then grouped together to form objects—either rooms or objects in the rooms.

In order to be visible, each polygon must also have a material associated with it. Materials are images of the visible features of objects in the world. For example, in Figure 3 "wood" is a material on the table object. A material is generated from an image file in a Crystal Space data library . Most standard image file formats are supported. Crystal Space data libraries are implemented as zip archives referenced in a Crystal Space specific configuration file. The height and width of material images must be some power of two, though the height and width need not be equal. Crystal Space will then tile the material across the associated polygons. The syntax allows materials to be scaled before being tiled on the polygon, so that users can create better looking worlds.

Crystal Space calls objects in rooms meshes. Meshes are prototyped once as a group of vertices, polygons and associated materials, and can then be used in any or all rooms defined in the file. Meshes and rooms can both have "keys" associated with them. These keys are ignored by the Crystal Space engine, but can be queried by user programs. This way objects in the world can have customized user data associated with them. Each object in our environments has two keys associated with it, one for shape and one for material. We use these keys in our simulated vision routine discussed in Section 3.4.

Most commercial game engines have a graphical "world editor" for users to design their environments. Crystal Space does not yet have a native world editor, although the Crystal Space project has supplied several programs to convert environment files generated by some commercial world editors to Crystal Space format. At the time we were designing our environment, however, these tools did not make use of many of the Crystal Space features, so we built our environment files using a text editor.

We did use the automated tools in one instance. The very complex shapes of the robots and people in the world (See Figures 3, 6, 7, and 8) were generated from public domain and freely available Quake II model files. These meshes are each composed of several hundred triangles. We used Crystal Space conversion tools to convert the Quake II format to crystal space format and then merged the resulting file into our environment definition file.

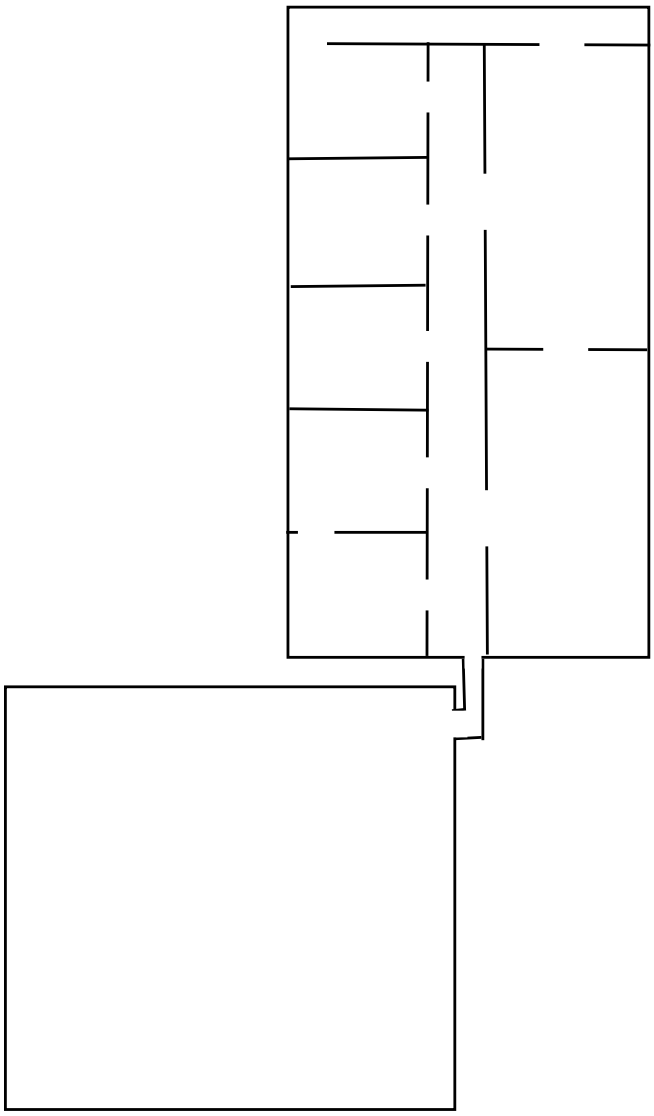

Figure 5: The floor plan of the larger suite of rooms, with a garage attached at the bottom.

We are currently using two different environments with the Crystal Cassie simulation. The first is a small suite of four interconnected rooms whose floor plan can be seen in Figure 4. The second environment is a much larger suite of rooms whose floor plan can be seen in Figure 5.

The rooms in the smaller suite contain tables, chairs, drinking glasses, bottles and sometimes robots. The larger suite is intended to resemble a wing of an academic building. It includes: two classrooms with chairs, tables and white boards; a computer lab with chairs, tables, computers, monitors and keyboards; a lounge with a stove, table and Pepsi machine; and a second lab with bulletin boards, a table with a large machine on it, and a filing cabinet. The larger suite is connected to a parking garage with a single car in it. Either suite can also have people or robots wandering through it. Figure 3 shows a view of the computer lab in the larger suite, containing some chairs, tables, computers and monitors, and two people. Figure 6 shows a view of a room in the smaller suite, containing a table with a bottle and several glasses on it, and two robots roaming about.

3.3 Crystal Cassie's current suite of actions

Currently, Crystal Cassie has a small repertoire of primitive acts which she can use to build more complex behavior by reasoning and planning at the KL. At the most basic, Cassie can be told to turn left or right, or to go forward or backward. Essentially, in this way she can be tele-operated through natural language. Cassie also has a primitive act for looking, which is used to invoke the simulated vision (see Section 3.4).

Finding an object that is in the current room is also a primitive act. Cassie will look and then turn and then look again, repeating this sequence until either she finds the object, or completes a 360 degree rotation. If she cannot find the object in this room, she must reason at the KL about how to look for it in another room.

Currently, Cassie's final primitive act is to go to an object. In order to go to an object she has to be looking at it, which means that she has found it. When going to an object, Cassie uses the location and size values in the vision 4-tuple (see Section 3.4) of the PML description of the object. The location is the object's center of gravity. The size is the object's largest radius in the <x,z> plane. When Cassie goes to an object, she plots a straight line course from her current position to a position close to the object. This new position is calculated using the location value for the object and the sum of the size value and a small constant. Cassie will then move along this straight line until she reaches the new point. (Obstacle-avoidance is planned for the near future.)

Unlike previous versions of Cassie, following a moving object is not a primitive act in Crystal Cassie. In order to follow a moving object, Cassie must reason through a series of finding and going-to acts.

3.4 Simulated Vision.

Processes P1, P2, and P4 of Crystal Cassie all play parts in the simulated vision system. Vision information is only sent by the simulated robot body when it receives a look action request. Cassie begins the simulation not looking at anything, and will only consciously look at the world if her reasoning at the KL triggers an action that requires looking. When she does the PML act of looking, the request is sent to the SAL, which asks the robotic body to look at the world and inform it of the objects that are currently visible.

At the level of the simulated robot body, the objects visible to Cassie are those also visible to the user viewing P4's display. (see Figures 3 & 6) One exception to this is that because of a peculiarity in how Crystal Space marks things as visible, Cassie cannot see through doors into adjacent rooms. This limitation is discussed further below. Cassie has a viewing angle of about 60 to 70 degrees in a single direction. The view is similar to what is visible in most commercial immersive 3-D computer games.

The look act, at the level of the simulated robot body, is implemented as a P4 method which is called from the Crystal Space libraries via the C++ callback mechanism. When

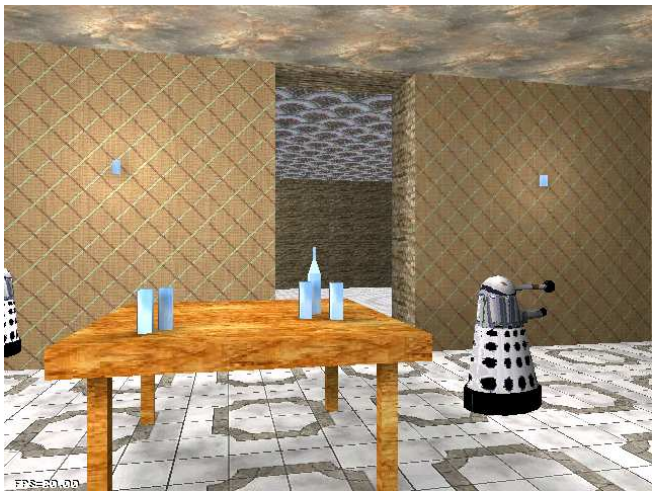

Figure 6: A view of the smaller environment from the P4 display, showing a room containing a table, on which is a bottle and several glasses, and two robots.

requested, Crystal Space's rendering algorithm will call this method for every object that Crystal Space marks as visible. This includes both the meshes that are visible, and the visible rooms: the current room and nearby visible rooms. For each object, the shape and material keys (see Section 3.2) are extracted. We are using the key value for material rather than the name of the actual material for two reasons. The first reason is a limitation of the Crystal Space API in the version we are using. The name of the material is not available from the polygon object associated with it. The second reason for using a material key (should the first issue be resolved) is a simplification of the simulated vision. Some objects, such as the glasses, table, and bottles in Figure 6 have a single material on all of their polygons. However, other objects, such as the computer monitors in Figure 3 have multiple different materials; some polygons have one material and other polygons have another. In order to simplify the simulated vision algorithm, we chose the material that we felt was most salient as the material key value. This eliminates the problem of recognizing an object from different viewing angles.

In addition to the shape and material keys, the vision callback also retrieves the object's current center point in world coordinates and calculates the radius of the object. These data points form the 4-tuple {material, shape, location, size} for the current object. When the callback has finished with all of the currently visible objects, the 4-tuples of those objects are sent to the SAL in P2.

The simulated sensor controls in the SAL currently send the 4-tuples from the vision system unchanged to the PML.

At the PMLa in P1, a vision tuple is "parsed" to see what kind of thing it is. In the PML there is a data structure containing a list of "alignments" between {material, shape} pairs and Cassie's concepts of the categories of things she knows about (for example, the category of wooden tables). Once this kind of basic object recognition is done at the PML, the KL must be invoked to reason about exactly which specific object Cassie is looking at. The KL uses the location information from the 4-tuple to reason about which object Cassie is looking at. It is impossible to distinguish between many of the objects in the environment using only their shape and material. The glasses in Figure 6 cannot be distinguished with only shape and material nor can the robots in the same figure. However, Cassie can reason about the identity of the glass she is looking at by reasoning about its location.

The Crystal Space team recently released a new version of the Crystal Space graphics tools. The original callback method that we used for doing simulated vision was removed in this new version. The original version used to return a list of objects visible in the current room only. The replacement mechanism that we are using now, will return all those objects marked as "probably visible" by the system. Objects marked probably visible are all those with at least part of their bounding box within the view frustum of the P4 and either in the current room or in a nearby room. Unfortunately this can include objects that are completely occluded by the walls separating adjacent rooms. As a simple way of correcting this, we cull out all meshes that are not in the current room. However doors and the adjacent rooms themselves are still visible. In order to see objects in an adjacent room, Cassie will have to find a doorway and go through it.

4 The First Use of Crystal Cassie.

The first use of Crystal Cassie is as a test-bed for developing a computational theory of identifying perceptually indistinguishable objects [Santore and Shapiro, 2002]. Two objects are perceptually indistinguishable to a cognitive agent if the agent cannot find any difference in their appearance by using its sensors. By identifying perceptually indistinguishable objects we mean the following: when an agent finds an object that is perceptually indistinguishable from one it has encountered before, the agent identifies the object if it successfully decides whether the object is the same one it encountered previously, or if it is a new object. If the object has been encountered before, and the agent has encountered more than one such object before, the agent should also know which one it is currently encountering.

We are developing a computational theory of identifying perceptually indistinguishable objects based on human performance at the task. We have used the same environment that Cassie operates in to perform a protocol-analysis experiment with 69 subjects on a series of tasks that require them to identify perceptually indistinguishable objects. The data from these experiments are currently being analyzed and will be reported in [Santore et al., Forthcoming].

The tasks performed by the human subjects, which Crystal Cassie will also have to perform, are counting perceptually indistinguishable stationary and moving objects, and following moving objects. The counting tasks are done in the smaller suite of rooms. The following tasks are done in the larger suite of rooms.

There are two conditions in each of the counting experiments. When counting stationary objects, in the first condition, the rooms themselves all have a different material, and so are perceptually distinguishable. In the second condition, the rooms diagonally opposite each other have the same ma-

Figure 7: A robot used in the tasks that human subjects performed, and which Cassie must perform.

Figure 8: A second robot used in the human subjects' and in Cassie's tasks.

terial and are therefore perceptually indistinguishable themselves. There are also two conditions in the experiment in which the subject must count moving objects. In the first condition, all of the objects look like the robot shown in Figure 7. In the second condition, some of the objects look like the robot in Figure 7 while the rest look like the robot in Figure 8.

There are three conditions in the following experiments. In the first, the object to be followed looks like the robot in Figure 7 and there are several perceptually indistinguishable robots wandering randomly in the suite. In the second following condition, the object to be followed again looks like Figure 7 and several perceptually indistinguishable robots are following their own paths in the suite. The third condition requires the subject to follow a person while several perceptually distinguishable people follow their own paths through the suite. A view from this last condition is shown in Figure 3.

The large suite of rooms is complicated enough for future experiments with other tasks for Crystal Cassie.

5 Experiences using Crystal Space

When we decided to use an existing 3-D graphics/gaming engine for the next simulated robot to be built on the GLAIR architecture, we needed a tool with a published API. We also wanted, if possible, an engine with publicly available source code, so we could build the engine ourselves for different operating systems. We had two possibilities, we could use a commercial game engine that had released the source code for its game under a public license, or we could use a publicly available engine built by a group of hobbyists for their own enjoyment.

At the time, id Software[1] had released the software for their original Quake game engine; they have since released the software for some of their newer Quake engines as well. The benefit of using this type of engine was that the API would be well defined and fixed. The drawback was that the company was no longer supporting the product, so we would have to fix any bugs or add any new features we needed.

The 3-D engines produced by hobbyists had a complementary set of benefits and drawbacks. Bugs are always being fixed, and new features are being added. However, this can make the API quite unstable. On the other hand, the original designers are available to answer questions and provide support.

Given these considerations, and the tools available when this project started, we chose the Crystal Space engine, which is a hobbyist project. Because Crystal Space development is ongoing, we have to adjust our code with every release to reflect API changes. The Crystal Space developers include a list of most of the necessary changes with each release. Recent releases have reduced the number of necessary changes dramatically. The community of hobbyists that design and use the Crystal Space engine supports projects built using Crystal Space tools by quickly answering most questions. The chances are fairly good that someone will know which of the hundreds of C++ classes that compose Crystal Space will be the best to use for a particular need.

We have used Crystal Space without modification as a set of shared libraries, on both Linux and Solaris platforms as an off the shelf solution to our 3-D engine needs. It provides enough flexibility in building environments to allow a reasonably complex environment for our agent to explore.

Crystal Space has performed adequately as a library for developing a simulated cognitive agent. Crystal Space still has a few holes in its API, and the Crystal Space Designers are working to fix them, and are committed to doing so without creating too many incompatibilities with previous versions. Overall we have been able to use Crystal Space successfully.

6 Conclusions

Crystal Cassie is a simulated cognitive agent, built on GLAIR, an existing architecture for representing the control of cognitive agents, SNePS, a knowledge representation and

[1] http://www.idsoftware.com/

reasoning system, and Crystal Space, an off the shelf 3-D engine produced by a community of hobbyists. Crystal Cassie can understand and generate sentences in a fragment of English, and has a small suite of primitive acts which she can use to perform a larger repertoire of actions through the intervention of her reasoning system. GLAIR is a three-layer architecture, the middle layer of which is separated into three sublayers. These five layers are implemented in two processes, written in different computer languages, that communicate over four socket connections representing the modalities of speech, hearing, vision, and action. The lowest GLAIR layer communicates with two additional processes, implementing Cassie's sensor and effector organs, over two two-way, and one one-way socket connections, for natural language interaction (two-way), action (two-way), and vision (one-way). The process that contains Cassie's vision organ and action effectors also contains a program that operates the simulated world that she occupies. A human user/interlocutor/observer can talk with Cassie using a keyboard and display connected to her natural-language-interaction process, and can observe the simulated world "through Cassie's eyes" via a display attached to the world simulator.

Crystal Cassie was designed and implemented in order to develop a computational theory of identifying perceptually indistinguishable objects. We have designed a number of tasks that she will have to perform to demonstrate this ability, and have recorded protocols from a number of human subjects performing the same tasks using the keyboard and world-simulation display to "drive" through the environment. The data from these experiments are currently being analyzed, and the theory is currently being formulated.

We have found Crystal Space to be a satisfactory set of tools for implementing a reasonably complex environment for a simulated cognitive agent. We are willing to share this environment with other interested researchers, who should contact the first author of this paper.

Acknowledgments

The authors are grateful to the other members of The University at Buffalo SNePS Research Group for their contributions to the design and implementation of SNePS and GLAIR, and for their comments throughout the course of this project.

References

[Coradeschi and Saffiotti, 2001a] Coradeschi, S. and Saffiotti, A. (2001a). Forward. In Coradeschi, S. and Saffiotti, A., editors, *Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems: Papers from the 2001 AAAI Fall Symposium, Technical Report FS-01-01*, page viii, Menlo Park CA. AAAI Press.

[Coradeschi and Saffiotti, 2001b] Coradeschi, S. and Saffiotti, A. (2001b). Perceptual anchoring of symbols for action. In *Proceedings of the 17th international joint conference on artificial intelligence (IJCAI-01)*, pages 407–412, San Francisco CA. Morgan Kaufman.

[Hexmoor et al., 1993] Hexmoor, H., Lammens, J., and Shapiro, S. C. (1993). Embodiment in GLAIR: a grounded layered architecture with integrated reasoning for autonomous agents. In Dankel, II, D. D. and Stewman, J., editors, *Proceedings of The Sixth Florida AI Research Symposium (FLAIRS 93)*, pages 325–329. The Florida AI Research Society.

[Hexmoor and Shapiro, 1997] Hexmoor, H. and Shapiro, S. C. (1997). Integrating skill and knowledge in expert agents. In P. J. Feltovic, K. M. Ford, R. R. H., editor, *Expertise in Context*, pages 383–404. AAAI/MIT Press, Cambridge, MA.

[Santore et al., Forthcoming] Santore, J. F., Segal, E., and Shapiro, S. C. (Forthcoming). Human identification of perceptually indistinguishable objects. Forthcoming.

[Santore and Shapiro, 2002] Santore, J. F. and Shapiro, S. C. (2002). Identifying perceptually indistinguishable objects: Is that the same one you saw before? In Baral, C. and McIlraith, S., editors, *Cognitive Robotics (CogRob2002), Papers from the AAAI Workshop, Technical Report WS-02-05*, pages 96–102, Menlo Park, CA. AAAI Press.

[Shapiro, 1982] Shapiro, S. C. (1982). Generalized augmented transition network grammars for generation from semantic networks. *The American Journal of Computational Linguistics*, 8(1):12–25.

[Shapiro, 1989] Shapiro, S. C. (1989). The CASSIE projects: An approach to natural language competence. In Martins, J. P. and Morgado, E. M., editors, *EPIA 89: 4th Portugese Conference on Artificial Intelligence 390*, pages 362–380. Springer-Verlag.

[Shapiro, 1993] Shapiro, S. C. (1993). Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2 & 3):225–235.

[Shapiro, 1998] Shapiro, S. C. (1998). Embodied Cassie. In *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium, Technical Report FS-98-02*, pages 136–143. AAAI Press, Menlo Park, CA.

[Shapiro, 2000] Shapiro, S. C. (2000). SNePS: A logic for natural language understanding and commonsense reasoning. In Łucja Iwańska and Stuart C. Shapiro, editor, *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*, pages 175–195. AAAI Press/MIT Press, Menlo Park CA.

[Shapiro and Ismail, 2001] Shapiro, S. C. and Ismail, H. O. (2001). Symbol-anchoring in Cassie. In Coradeschi, S. and Saffioti, A., editors, *Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems: Papers from the 2001 AAAI Fall Symposium, Technical Report FS-01-01*, pages 2–8, Menlo Park, CA. AAAI Press.

[Shapiro and Ismail, 2003] Shapiro, S. C. and Ismail, H. O. (2003). Symbol anchoring in a grounded layered architecture with integrated reasoning. *Robotics and Autonomous Systems*. in press.

[Shapiro et al., 2000] Shapiro, S. C., Ismail, H. O., and Santore, J. F. (2000). Our dinner with Cassie. In *Working Notes for the AAAI 2000 Spring Symposium on Natural Dialogues with Practical Robotic Devices*, pages 57–61, Menlo Park, CA. AAAI.

[Shapiro and Rapaport, 1987] Shapiro, S. C. and Rapaport, W. J. (1987). SNePS considered as a fully intensional propositional semantic network. In Cercone, N. and McCalla, G., editors, *The knowlege frontier*, pages 263–315. Springer-Verlag, New York.

[Shapiro and Rapaport, 1992] Shapiro, S. C. and Rapaport, W. J. (1992). The SNePS family. *Computers and Mathematics with Applications*, 23(2-5):243–275. Reprinted in [**?**, pp. 243–275].

[Shapiro and the SNePS Implementation Group, 2002] Shapiro, S. C. and the SNePS Implementation Group (2002). *SNePS 2.6 User's Manual*. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo NY.

[Tyberghein et al., 2002] Tyberghein, J., Zabolotny, A., Sunshine, E., and et. al. (2002). *Crystal Space: Open Source 3D Engine Documentation*. http://crystal.sourceforge.net, .94 release edition. ftp://crystal.sourceforge.net/pub/crystal/docs/download/.