# CIG: Cultural Islands and Games

**John Dickerson M. Vanina Martinez Diego Reforgiato V.S. Subrahmanian**

{jdicker1,mvm,diegoref,vs}@cs.umd.edu

Department of Computer Science and
University of Maryland Institute for Advanced Computer Studies (UMIACS)
College Park, MD 20742

**Abstract**

In this paper, we propose the concept of a "cultural island" – a framework with which users can virtually immerse themselves within a culture, friendly or not. Cultural islands provide a virtual world technology for a wide range of applications ranging from sampling tourist destinations to helping national security and defense analysts understand how best to reason about a particular part of the world. In this paper, we describe a formal architecture for cultural islands, and describe a prototype implementation of part of this architecture. We show how our framework has been used to build an initial prototype game appropriate for training members of the US military.

## Introduction

There are numerous applications in the world where there is a need to understand a foreign culture. Businessmen traveling overseas to China may want a quick "virtual experience" of how to greet their counterparts there, and what the do's and don'ts are when having a meal with their hosts. A tourist might want to "sample" certain parts of the world before she decides where to go – her virtual experiences may make her avoid certain parts of the world where she feels uncomfortable, and enthuse her about other parts. A UN peacekepeer expecting to be deployed in the Sudan might want to engage in a virtual training session alerting him to the cultural nuances of serving in that part of the world and have a deeper understanding of the groups that reside there. A US military analyst might want to understand what changes might occur on the ground, and what kinds of different reactions may occur when US forces take certain actions in a foreign land.

A cultural island is a computational environment within which it is possible for users to:

- Develop an understanding of the background of the culture and its socio-political economic practices.
- Learn how to interact with members of these groups, based on a rich understanding of their behaviors, and based on an automatically learned, statistically valid model of groups behaviors.
- Identify the history of activities of a particular group (this could be a socio-economic group, a religious group, a tribal group, a political organization).
- Interact with computational models of these groups.
- Experiment with "what if" scenarios.
- Forecast what the group will do in a given situation and use that in order to determine how best to work with that cultural group in order to best achieve ones objectives.

While the focus of the applications that we have built with cultural islands is national security focused, the notion of cultural islands is appropriate to tourism, training, and many other applications where an appreciation of a foreign culture is critical. In this paper, we describe the architecture of a Cultural Island Toolkit (CIT) which, when complete, will allow:

- Multiple analysts and other knowledgeable experts to congregate on a (virtual) island on an appropriately secure computer network, and build a community of experts who can share experiences.
- Provide an online game theoretic environment for cultural games that allows analysts to play out "if I do this and the group does this and a third party does this and I do this and they do that..." kinds of scenarios well into the future.
- Leverage statistically accurate models of the behaviors of groups in determining how a given group will respond to a given user's policy or tactical behaviors in the virtual environment.
- Provide a multi-player online gaming environment where a mix of human players and artificial intelligence bots play differing roles.
- Provide analytics on the playouts of such games that will provide valuable input to an analyst or decision maker making policy decisions.

The paper is organized as follows. In the following sections we first provide a description of the architecture of CIT. The *Cultural Island Authoring Tool* section decribes a case study of the CIT architecture – a game we developed that will allow US soldiers and analysts to be more effective on the ground in Afghanistan, both in military and in reconstruction operations. Finally, we conclude with a description of the path forward.

*Note that, to date, we have not fully implemented the CIT architecture.* The design of the architecture is complete, and this is what we describe in the paper. Those parts that still need to be implemented will be clearly specified.

## The CIT Architecture

Figure 1 below presents a detailed view of the CIT architecture. Boxes shown in black are still to be built. Boxes shown in gray are done, while boxes shown in white are at some intermediate stage of completion. The CIT architecture supports an end to end, real time operation, starting with the ingestion of data in real time, the construction of behavioral models of the groups in a particular part of the world, and the actual game environment itself that uses these as inputs in order to support the user's play.

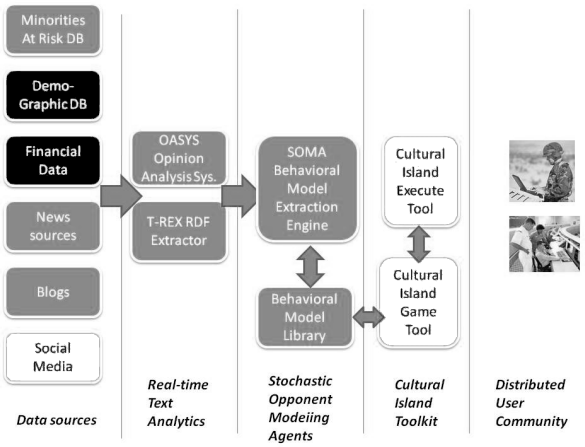


Figure 1: Architecture of the Cultural Island Toolkit

The cultural island architecture takes data sources (both legacy sources and real time sources) and feeds them directly into any game that is built using the toolkit. These data sources are analyzed using text analytic programs such as the OASYS Opinion Analysis System (Cesarano et al. 2006) and The RDF EXtractor (T-REX for short) (Albanese and Subrahmanian 2007) which automatically extracts information of interest from the live data sources. Currently, T-REX and OASYS both track news and blog sources; also, T-REX tracks some social media sources such as a small amount of *YouTube* comments (not the raw video).

In addition, the Minorities at Risk data set was created through extensive manual analysis of data by political scientists. For each of 117 ethnopolitical groups at risk of falling into terrorism, this data set contains information on about 150 variables, coded on an annual basis. Thus information about each of these ethnopolitical groups is represented as a table. Each row in the table represents a year (up to 25 years), and each column represents a variable. The variables fall into two categories: environmental variables describing the environment in which the group functioned in a given year (including information on actions taken by other players that might affect the group being modeled) and action variables that describe various inappropriate actions taken by the group during the year (*e.g.*, kidnappings, armed attacks, transational attacks on governmental security apparatus, etc.)

The SOMA Extraction Engine automatically extracts rules of the form: "if some condition $C$ is true over the environmental variables, then the group in question will execute action $A$ with a probability of $L$ to $U\%$." To date, the SOMA Extraction Engine has extracted rules on about 36 groups ranging from Morocco in the west to Afghanistan in the east. These stochastic rules constitute the behavioral model library used by our Cultural Island Toolkit.

The heart of this paper focuses on the computational environment that allows analysts to use a multi-player game oriented framework in order to interact with each other, with the computational realization of the extraction SOMA models, and with the environment.

### Cultural Island Authoring Tool

The *Cultural Island Authoring Tool* allows users to author a cultural island game. We assume the existence of a set $S$ of game states that is somehow implicitly defined. We have developed an initial game called **CAVE** (Cultural Afghan Village Environment) which is intended as a mechanism to help US soldiers enter an Afghan village. In CAVE, there is one US soldier interested in entering an Afghan village and several villagers that he encounters during his visit. Each villager has an initial tendency to support Americans (or this American in particular). This tendency ranges from $-1$ (completely against) to $+1$ (completely for). Each vector of the form $\langle s_1, s_2, ..., s_n \rangle$ where $s_i$ denotes the tendency of the $i$-th Afghan villager towards the US soldier forms part of the state of the game. This is called the cultural compatibility index vector (CCIN for short).

However, the game state also consists of other parts. One key part is a video clip corresponding to video content to be shown as part of a given node. Another component of a game state consists of a question and a set of possible answers to that question.

Thus, a CIG game is characterized by three components:

- A set $C$ of CCIN vectors specified either implicitly or explicitly.
- A set $Q$ of questions expressed in English and for each question $q$ in $Q$, a set $ANS(q)$ of possible answers (also in English).
- A set $V$ of video clips (in our CIG implementation, these video clips were recorded using the Second Life environment (Linden Research, Inc 2003)).

A cultural island game $G$ based on $(C, Q, V)$ is a tree where:

- Each node in $G$ is labeled with a triple $(c, q, ANS(q))$ where $c$ is in $C$, $q$ is in $Q$, and $Ans(q)$ is the set of all answers to $q$.
- If $N$ is a node labeled with $(c, q, ANS(q))$, then $N$ has one child node associated with each answer in $Ans(q)$.

Thus, a CIG game is represented as a tree. Each node in the tree represents a game state consisting of a cultural compatibility index vector, a question, and the (deterministic) set

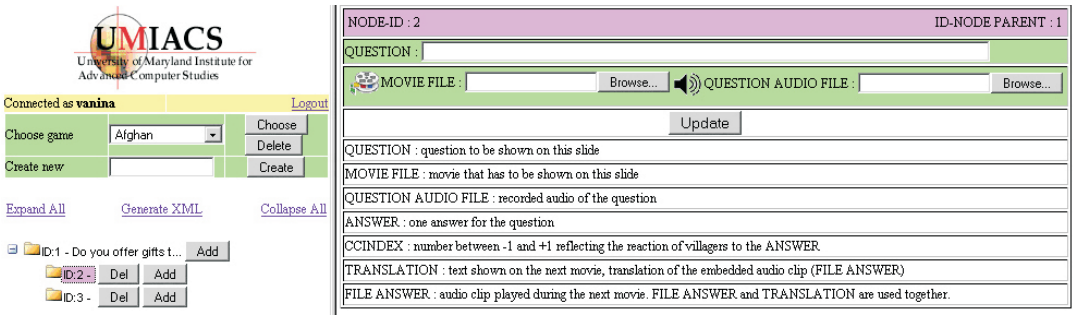


Figure 2: CIG Authoring Tool Screenshot

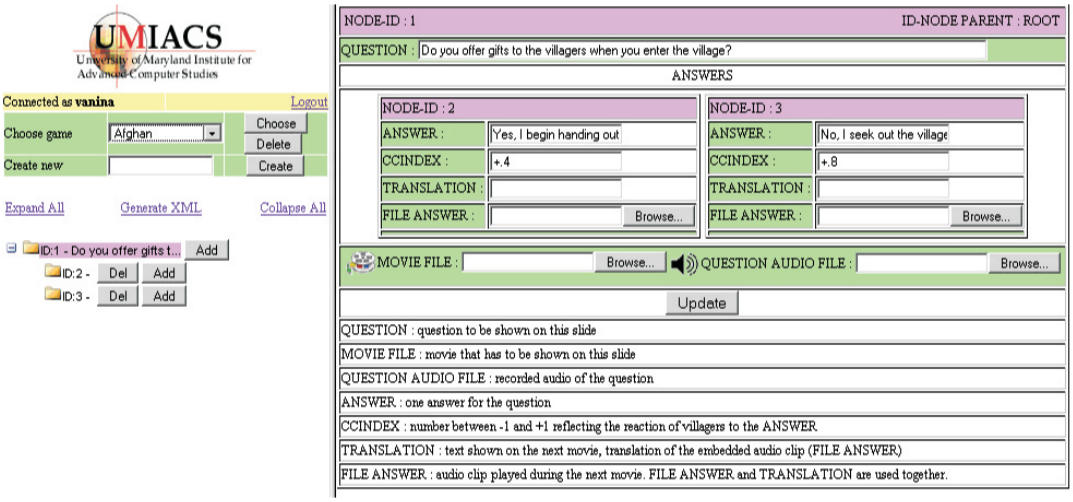


Figure 3: Screenshot when a set of answers to a question is provided

of answers to the question. Depending on which answer the user selects, a new node is chosen.

As SOMA rules are stochastic, we do have plans to allow the answer to a question to cause a non-deterministic (probabilistic) transition to a child state. However, we have not incorporated this into the CIG design thus far.

The Cultural Island Authoring Tool is a password protected online tool for authoring such CIG games. A user can use this tool to:

- Specify the content of a node.
- Specify what video is associated with a node.
- Specify the question and set of answers to that question associated with the node.
- Specify what the children of the node are.

Thus, the author of a CIG game can generate these nodes in a tree with very little knowledge of computer science. This is important as it allows graphic artists, policy experts, and others to come together around a CIG game in order to best orchestrate the game. Figures 2, 3, and 4 show screenshots of the CIG Authoring Tool.

The reader can see from Figure 2 above that the user can add a node and that this node causes various fields within the node to be generated. The user can instantiate a node by specifying the question to be asked (both textually and/or via a recorded audio clip), the movie file to be shown, the answer, the cultural compatibility index, as well as translations of the answer (if they are given in a foreign language).

Figure 3 shows the situation when the user has a question with two possible answers. One node is created for each possible answer and the user can fill in the appropriate state information.

The author of a CIG game can easily instantiate the fields shown in the figure. The structure of the tree describing the game can also be easily visualized by the user as shown in Figure 4 below.

## Cultural Island Execute Tool

The Cultural Island Execute Tool is focused on providing the execution environment for a game authored within the CIG environment. The current implementation of the Cultural Island Execution Tool distributes the game produced in Section  above as a single player game available on a DVD.

However, we have done some work on the design of a multi-player version of CIG that will be accessible online. In the multi-player version of CIG, there is a central database called *GameDB* that tracks the game state at any given point in time.

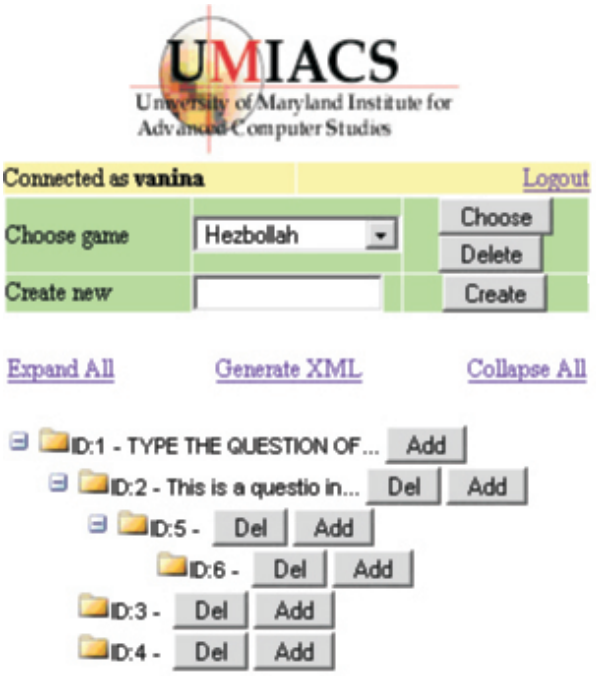


Figure 4: Structure of the CIG tree

A *multiagent game state* associates a cultural compatibility index with each player in the game. Thus, if a villager (played by a bot in the game) is confronted with three players, all representing the US military, he will have a cultural compatibility index for each of them which may vary from one US player to another. We let $CCIN(i, j)$ denote the cultural compatibility index of bot $i$ towards human player j. Thus, each player $P$ has an associated vector $CCIN(P)$ containing a vector of CCINs that the different bots in the game hold toward him.

Let us suppose that we have a single agent game $G(P)$ associated with each player $P$ having the form described earlier in this paper. A *multiagent CIG game* is also a tree where:

- Each node $MN$ is a partial mapping from players (or player's ids) to triples of the form $(c, q, ANS(q))$ and
- If $(c(P), q(P), ANS(q(P)))$ is a child of the node labeled $(c, q, ANS(q))$ in $G(P)$, then there is a child $MN'$ of $MN$ such that $MN'(P)$ is $(c(P), q(P)ANS(q(P)))$.

In other words, a multiagent CIG game has nodes that combine nodes from single player CIG games. Figure 5 shows a simple example where there are two players. Figure 5a) shows the game for Player $P_1$, while Figure 5b) shows the tree for Player $P_2$. Figure 6 shows the combined multiagent CIG game tree when Players $P_1$ and $P_2$ start playing at the same time.

As a consequence, we can see that it is possible to *automatically merge* the trees of individual games when they start to form a multiplayer game.

## CAVE: Cultural Afghan Village Experience Game

Based on the principles articulated in the preceding sections, we have developed a game focused on supporting US soldiers who are entering an Afghan village. The aim of the game is for the soldiers to be culturally sensitive and make a good impression on the Afghan villagers. Players do well in the game if the Afghan villagers feel the players respect and understand them and their culture, are reliable partners who care about them and are likely to assist them in the improvement of their lives.

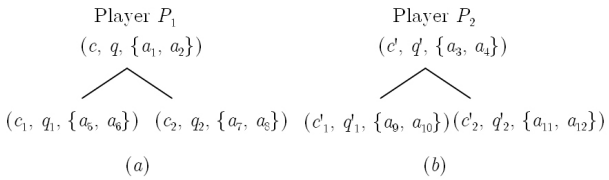


Figure 5: Game trees for two players $P_1$ and $P_2$

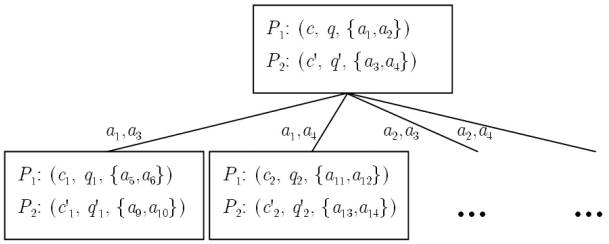


Figure 6: Combined game tree when $P_1$ and $P_2$ play at the same time

The definition of multiagent CIG game makes it possible for players to join a game after another player has already started playing. Figure 7 shows what the multiagent game tree looks like in the above example when Player $P_2$ joins the game one unit of time after player $P_1$ started playing.

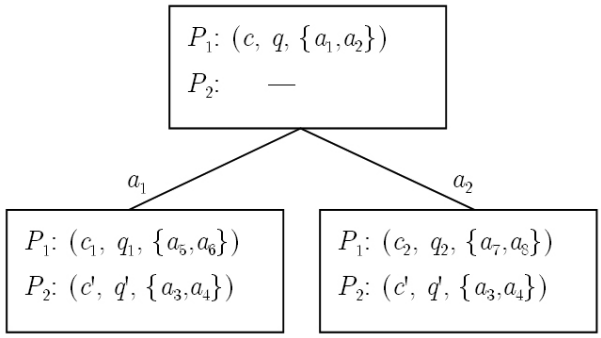


Figure 7: Merged game tree when player $P_2$ joins the game a unit of time after player $P_1$

The CAVE game itself has three levels. The first level is a basic introductory level where a soldier drives up to a vil-

lage, introduces himself to the villagers, and asks to meet the elders. The level goes all the way from the vehicular approach to the village, to the initial greetings from the villagers, to meetings in the village Shura room with village elders, to the soldiers departure in this vehicle.

The second level is reached only if the soldier achieved a passing score on the first level. This level focuses on social welfare of the village including understanding how best the soldier might help achieve better health care, educational, and reconstruction goals. A significant part of this level focuses on the conversation within the Shura room with village elders.

Finally, the third level focuses on how the soldier might enlist the villagers in helping gather information that could save lives of both villagers and US personnel. The idea is to determine if there are terrorists hiding in the village, or whether there are roadside bombs planted near the village, and how terrorists may use the village in furthering their goals. In total, CAVE consists of about 3800 nodes.

Figures 8 shows a screenshot of the US soldier entering the village. Players who play the game get a score at the end of each level. Figure 10 shows one such sample screenshot.


Figure 8: Screenshot of US soldier entering the village

Figure 9, on the other hand, shows a screenshot of the US soldier engaged in a discussion with village elders inside the Shura room.

## Implementation Details

The CAVE Game is a Windows-specific application developed primarily on commercially available software using well-documented programming languages. The majority of the development time was spent in three environments:

- Linden Lab's Second Life – provides gameplay visualizations of soldier interacting with villagers in a virtual Afghan village – contains around 500 lines of Linden Script Language.
- Macromedia's Director MX 2004 – provides user-run options, ties game structure (an XML file) with the Second Life visualizations – contains around 3000 lines of Lingo.


Figure 9: Screenshot of the US soldier meeting village elders inside the village Shura room.

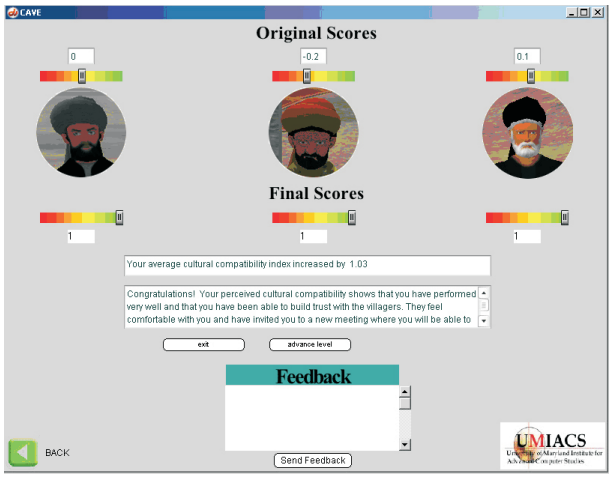


Figure 10: Screenshot of user's score after completing the first level

- Adobe's Flash CS3 – provides animated main menu, load/save games from XML files – contains around 500 lines of ActionScript 2.0.

In conjunction with the software suites mentioned above, other readily available tools were used. These include:

- Autodesk's $3d$ Studio Max – created models and respective animations for importing into the Afghan village environment in Second Life – employed around 300 lines of MaxScript.
- Adobe's Photoshop CS3 – created textures for the Second Life models as well as the logos and cover art.
- Adobe's After Effects CS3 – edited videos and added subtitle capability.
- Adobe's Dreamweaver CS3 – used to help edit the nearly 4,000 nodes describing the CAVE game.

- Ruby programming language – used around 700 lines of scripts to automate various tasks.

The game itself runs on any somewhat modern laptop or desktop that runs the Windows operating system. It does not require any specialized hardware beyond a pair of speakers and a mouse. A full installation requires about 1.5 GB of space, while the game itself easily fits onto a standard DVD.

## Related Work and Conclusions

We are not aware of any work on building cultural games. There is, of course, a rich history of work on gaming in general and multimedia presentations (of which genre this work forms one important part).

CIG is based on our prior work on multimedia presentations (Candan, Lemar, and Subrahmanian 2000; Candan, Prabhakaran, and Subrahmanian 1996) in which one of the authors defined the concept of a multimedia presentation and an interactive multimedia presentation as a tree. Prior work (Buchanan and Zellweger 1993; Little and Ghafoor Apr 1990; Adali, Sapino, and Subrahmanian 1999) had focused on non-interactive multimedia presentations and had focused on important issues such as delivery across a computed network. CIG builds on these efforts by explicitly focusing on the states contained within these games, and on building a platform within which such cultural games can be constructed. In addition, CIG brings together the important idea of a multiagent game not found in these previous works. Importantly, CIG games have video components that we have prototyped within Second Life.

More importantly, CIG is being designed to eventually build on top of live news and blog feeds that will dynamically change the game state and will support the use of real models of behaviors of real cultural groups and/or terror groups as described in the CARA cultural modeling architecture (Subrahmanian et al. 2007; Khuller et al. 2007). Our system already contains automatically extracted rules for 36 terrorist groups from Morocco to Afghanistan.

Several important questions remain open. How easy is it to construct cultural games using CIG? How useful are these cultural games in helping policy makers arrive at decisions? What additional features must be present in CIG to better support such needs? These and other questions are the topic of our ongoing research efforts.

## Acknowledgements

The authors gratefully acknowledge funding support for this work provided by the Air Force Office of Scientific Research through the Laboratory for Computational Cultural Dynamics (LCCD) and AFOSR grants FA95500610405 and FA95500510298, the Army Research Office under grant DAAD190310202, and the National Science Foundation under grant 0540216. Any opinions, findings, or recommendations in this document are those of the authors and do not necessarily reflect the views of sponsors.

## References

Adali, S.; Sapino, M. L.; and Subrahmanian, V. S. 1999. A multimedia presentation algebra. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, 121–132. New York, NY, USA: ACM.

Albanese, M., and Subrahmanian, V. 2007. T-rex: A system for automated cultural information extraction. *First International Conference on Computational Cultural Dynamics*.

Buchanan, M. C., and Zellweger, P. T. 1993. Automatically generating consistent schedules for multimedia documents. 1(2):55–67.

Candan, K. S.; Lemar, E.; and Subrahmanian, V. S. 2000. View management in multimedia databases. *The VLDB Journal* 9(2):131–153.

Candan, K. S.; Prabhakaran, B.; and Subrahmanian, V. S. 1996. Chimp: a framework for supporting distributed multimedia document authoring and presentation. In *MULTIMEDIA '96: Proceedings of the fourth ACM international conference on Multimedia*, 329–340. New York, NY, USA: ACM.

Cesarano, C.; Dorr., B.; Picariello, A.; Reforgiato, D.; Sagoff, A.; and Subrahmanian, V. 2006. Opinion analysis in document databases. *Proc. 2006 AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs, Stanford, CA*.

Khuller, S.; Martinez, M. V.; Nau, D.; Simari, G.; Sliva, A.; and Subrahmanian, V. 2007. Computing most probable worlds of action probabilistic logic programs: scalable estimation for $10^{30,000}$ worlds. *Annals of Mathematics and Artificial Intelligence* 51(2):295–331.

Linden Research, Inc. 2003. Second Life, Official Site of the 3D Online Virtual World: http://www.secondlife.com/.

Little, T., and Ghafoor, A. Apr 1990. Synchronization and storage models for multimedia objects. *Selected Areas in Communications, IEEE Journal on* 8(3):413–427.

Subrahmanian, V.; Albanese, M.; Martinez, V.; Reforgiato, D.; Simari, G. I.; Sliva, A.; Udrea, O.; and Wilkenfeld, J. 2007. CARA: A Cultural Reasoning Architecture. *IEEE Intelligent Systems* 22(2):12–16.


# ICCCD 2008 Proceedings

This paper was published in the *Proceedings of the Second International Conference on Computational Cultural Dynamics,* edited by V. S. Subrahmanian and Arie Kruglanski (Menlo Park, California: AAAI Press).

The 2008 ICCCD conference was held at the University of Maryland, College Park, Maryland, USA, 15–16 September, 2008.