Second Life as a Platform for Creating Intelligent Virtual Agents

Larry F. Hodges[1], Amy Ulinski[2], Toni Bloodworth[1], Austen Hayes[1], John Mark Smotherman[1], and Brandon Kerr[3]

[1] Clemson University, School of Computing, Clemson, SC 29634-0974
[2] University of Wyoming, Department of Computer Science, Laramie, WY, 82071
[3] University of North Carolina at Charlotte, Department of Computer Science, Charlotte, NC 28223
{lfh,tbloodw,ahayes,jsomthe}@clemson.edu, aulinski@uwyo.edu, bkerr@uncc.edu

Abstract. Intelligent virtual agents (IVAs) are animated characters that interact with humans and with each other using natural modalities such as speech and gestures. Creation of successful IVA applications is challenging since development requires integration of a broad range of specialized tools and skills. We present a discussion of the advantages and technical challenges of using the Second Life programming environment as an accessible platform for undergraduates to develop an Intelligent Virtual Agent and the environment that the IVA inhibits.

Keywords: Intelligent Virtual Agents, Virtual Worlds, Animation, Avatars, Virtual Characters.

1 Introduction and Background

Intelligent virtual agents (IVAs) are animated characters that interact with humans and with each other using natural modalities such as speech and gestures. Recent applications of IVAs include teaching communications skills [1], training nursing students in interview techniques [2], training healthcare workers [3], cultural protocol training [4], children's museum docents [5] and conducting police photo lineups [6]. Creation of successful IVA applications is challenging since development requires integration of a broad range of specialized tools and skills. Typical tasks include modeling (both characters and environment), animation, voice recognition, conversational modeling, text-to-speech with lip syncing, perception of events in both the real and virtual world, and awareness of social conventions. Creating the tool infrastructure to teach just one of these skills is non-trivial and costly. As a result, involving university students in IVA projects, courses and research is significantly hampered by the lack of accessible and usable tools. Commercial products that allow creation and animation of animated characters such as Autodesk Maya [7] or DI-Guy Human Simulation Software [8] are costly, and have steep learning curves—particularly for students with no previous experience. Free tools such as Unity3D [9]

V.G. Duffy (Ed.): Digital Human Modeling, HCII 2011, LNCS 6777, pp. 292–301, 2011.

or Blender [10] provide accessible tools for environmental building but not for custom character creation and animation.

An alternative approach to creating an IVA is to modify components in an existing virtual world. The term *virtual world* can be used to describe any electronic environment that visually depicts or mimics complex physical spaces, and in this environment people can interact with each other and virtual objects while being represented by an animated character (an avatar) [11]. Currently, there are two main types of online virtual worlds. There are the creative-oriented environments such as Linden Labs Second Life (SL) [12] and Active Worlds [13], and there are the large multiplayer online role-playing games such as World of Warcraft (WoW) [14]. Active Worlds and SL are environments where users are visually represented by avatars and have the ability to interact with each other via chat or voice input [15]. Users in SL have the ability to build objects and environments, edit the appearance of his/her avatar, and interact socially with other users. SL is being used as a programming environment by professors and students to create distance learning classrooms, where students have the ability to meet with professors and other students in a virtual classroom. In Second Life there are virtual replicas of entire cities, universities, and companies [16]. WoW is an environment where players create characters with distinctive looks and characteristics ranging from clothes, jewelry, strength, agility and healing powers. In this environment, players work together to complete tasks to advance through levels of play in an environment that is full of different landscapes, objects, animals and fantasy elements [17]. WoW and SL both use avatars to represent users, have large 3-D environments, complex internal economies, voice communication, and simple artificial intelligence. One of the main differences between these two environments is the way the environments are used. WoW focuses on the gaming and fantasy aspect of a virtual world, while the users in SL focus on the social interactions and building aspect of a virtual world. SL affords itself well to the rapid development and prototyping of virtual worlds.

In this paper we present an experiment in which a group of undergraduate students with no previous experience in creating IVAs were challenged to design, model, and implement a virtual world complete with an interactive character that could communicate with a user via voice recognition, gestures, and text-to-speech—all in just eight weeks. Besides the ambitious schedule, the unique aspect to this project was to create the application using tools from the SL programming environment. The underlying assumption for how users will use the SL environment is that the virtual character is an avatar that represents and is under the control of a particular user in the virtual world. Our challenge was to invert this concept and convert a SL avatar from a representation of a user in a virtual world to an intelligent virtual agent that could directly interact with a user. We describe the resulting product and summarize the advantages and challenges of this approach.

2 Project Description and Goals

The Project Team consisted of eight undergraduate students from four different universities who were on the Clemson campus to participate in a summer research program in Human-Centered Computing. All were majoring in computer science or a

related discipline. None of the students had previous experience with second life, voice recognition, or IVAs. Two faculty members and one Ph.D. student served as mentors to advise the team.

The assigned goal for the group was to design a virtual human that could interact naturally with a user via voice recognition, speech, and animated gestures. The context for the interaction was an interactive presentation of the Clemson School of Computing. Projected users of the system would be visiting high school seniors who were potential computer science majors and their parents. The project had to be completed in eight weeks.

During the first week of the project, the students used online tutorials and books to learn basic SL skills: how to control and customize their avatars, how to model objects, and how to program objects using the Linden Scripting Language (LSL). Students were also introduced to the basics of voice recognition and text-to-speech software. In the second week we added daily meetings to discuss content and design of our IVA application. Since we were all (including the mentors) new users of SL we decided to first build an initial prototype to explore and learn more about the available tools before doing the design and implementation of the final product.

Fig. 1. CARA sitting at her desk

2.1 Initial Prototype

The initial prototype was named CARA (**C**lemson **A**utonomous **R**ecruiting **A**gent). CARA was presented on a large screen display and used voice recognition and recorded audio to communicate with a user. The goal was to implement an initial agent that allowed us to test all the capabilities needed for the final project: environmental modeling, avatar creation, animation, voice recognition, agent speech with lip syncing, capabilities of the SL scripting language, embedding multimedia content, and methods for overall program control. The result, which took two weeks to build, was a teenage girl sitting at a desk in an office. CARA greeted the user and asked the user's name. She then stood up and asked the user if they would like to see a short video about one of the current research projects in the School of Computing.

The video played on a virtual large TV screen in the office. At the end of the video, CARA tells the user goodbye and the demo concludes. A simple state machine controlled the flow of the system. Figure 1 shows a screenshot of CARA at her desk.

2.2 Final Design

Once we had the prototype working we began the design of the final product. In addition to identifying several technical problems that required workarounds while creating CARA, we also realized that we had rich tools for fast modeling and a large content space in SL that we had not taken advantage of in the prototype. Instead of an office space we designed an entire island world that included a grizzled older guide (CLEM) who takes the user on a tour that provides an overview of one of the three divisions in the Clemson School of Computing: Computer Science, Human-Centered Computing, or Visual Computing. CLEM carries on a short conversation with the user to determine the preferred tour, and then they set off in his vehicle through a jungle environment complete with rivers, foliage, and creatures. Along the way CLEM relates tales of Clemson history and traditions. Eventually you come to a clearing where CLEM uses posters and videos to explain current research and education projects for the chosen division. CLEM and his world were presented to the user on a large screen display and communication with CLEM was done using voice recognition and text-to-speech.

Fig. 2. CLEM in his vehicle at the Human-Centered Computing exhibit

3 Technical Discussion

3.1 Avatar Creation and Animation

Second Life accounts provide users with a free, customizable avatar that can be easily modified to many different shapes and sizes. Clothing, skin textures, hair styles and

other avatar accessories can be found either free or at low cost. The pre-made, customizable avatar bodies and wide range of available textures meant that our virtual agents could be made quickly without a large time or monetary investment.

Avatar animation can be controlled by the user via a keyboard and mouse or by prerecorded animations that can be triggered via keyboard commands. Animations such as walking and basic gestures come already built-in with the avatar. Controlling a walking movement directly with the keyboard movement keys, however, is nondeterministic. The exact same sequence of keyboard commands may produce a different result in the final location and orientation of the avatar. The consequence is that we could not reliably animate an avatar's movements via capturing and resending a sequence of keyboard commands. Animation files created in a tool such as Blender or Qavimator can be uploaded to SL to allow the user to customize the walking animation or perform other predetermined animations the avatar skeleton is capable of. There is a severe limitation, however, in the duration of prerecorded animations which can be no longer than 60 seconds and the associated file can be no larger than 60 kb. Once the animation is finished, the avatar *snaps* back to its original position and configuration it was in before the animation was triggered. The end result is that both CARA and CLEM had to be restricted to a fixed position or very short animated movements that always returned to an initial position.

Another issue was that both head and eye animations cannot be controlled directly through a script. Instead, both the head and eyes follow the mouse pointer of the user. The result is that the avatar is always looking toward whatever spot on the display screen where you last left the mouse pointer.

3.2 Camera Control and Object Animation

Although scripted avatar animations are not supported in SL, scripted object (anything in the world that is not an avatar) movement is. One motivation for designing CLEM around the theme of a jungle tour guide was the idea that the guide could lead the tour while sitting in a vehicle and we could write a script to animate the vehicle. To make this work we needed camera control functions so that the view of the scene followed the movement of the vehicle, and we had to create an animation script for the vehicle.

Camera Control. The standard camera view of a scene in SL is either a third-person camera view from directly behind and above a user's avatar or a first-person camera view from the eye-point of the avatar. For both CARA and CLEM we wanted to control the camera so as to present the world from the viewpoint of the user instead of an avatar-controlled view. To help create this effect we needed fluid camera movements that allowed the user to *virtually* look around or move around a space. While there are scripted functions in the built-in scripting language (LSL) for instantaneous camera position changes, there are no built-in interpolation functions to provide fluid camera movement from one point of interest to the next. We needed to create our own interpolation functions from the basic commands provided by LSL.

For our camera algorithm to work we needed to change both the position and rotation of the camera. The incomplete implementation of the camera API in LSL made scripting difficult. For example, while there is a *get* camera rotation function,

there is no *set* camera rotation function. We used the built-in functions for the positional movement of the camera and found a "camera focus offset" to work around the lack of a camera "set" rotation function.

Since no interpolation functions existed we wrote an algorithm that took the starting point and target point of the camera and divided the movement up into a set number of "frames", each frame being an incremental step toward the target position and rotation. Because the movement function calls *snap* the camera into its new position, the result was camera movement that could not be smoothed enough so the eye could not distinguish the individual frames, resulting in usable but slightly jerky camera movements.

Vehicle Animation. To overcome the limitations on scripted avatar movements for CLEM we built our story line so that the tour guide rode in a vehicle. SL allows modeled objects to be scripted for movement; however, like the camera movement functions, scripted object movement in SL is limited, with no interpolation functions for smooth, continuous object movement. This meant that the vehicle's movement appeared jerky. Further, positional and rotational functions were separate function calls, meaning there was no way to fluidly turn corners as a real vehicle would. Instead, the movement script would incrementally change the position and rotation of the vehicle separately, both calls being distinguishable to the observer.

3.3 Program Control, Voice Recognition and Lip-Synching

Program control for both CARA and CLEM was handled by simple finite state machines. CARA carried on a limited conversation with the user and then showed them a video about a current Clemson School of Computing research project. The state machine used three main utilities behind the scenes to accomplish its function: a timer, a listener, and a current state. The states controlled context-specific actions for each state, normally followed by a prompt to the user and an initiation of the listener. From there, the listener would take over, listening for specific recognized text (again, depending on the current state). While the listener waits for a given input, a timer runs concurrently in order to control what happens when Cara does not receive input (when it is assumed that the user did not say anything). CLEM was developed using a similar logic structure as CARA, but was more complex, since Clem would give the user choices and the number of interactions increased.

LSL scripts may be used in order to start animations, play sounds, and do various other things inside SL. For our CARA prototype we used LSL for our entire logic script, maintaining a state machine for all control logic inside the SL script. To solve some of the architectural issues we encountered in the implementation of CARA we moved the logic script outside of SL and into a C# script for CLEM.

LSL scripts are the only method to enact changes inside the virtual world. The SL viewer program (used to connect a computer to the SL environment) limited us to using a keyboard emulator to send commands to LSL scripts through hidden text chat channels. SL has a main chat channel, and then any number of other channels, which you can specify to chat on. Every scripted object we built received commands on a particular channel; for instance, the doors to a room would swing open based on hearing the word "open" on the channel they were listening on.

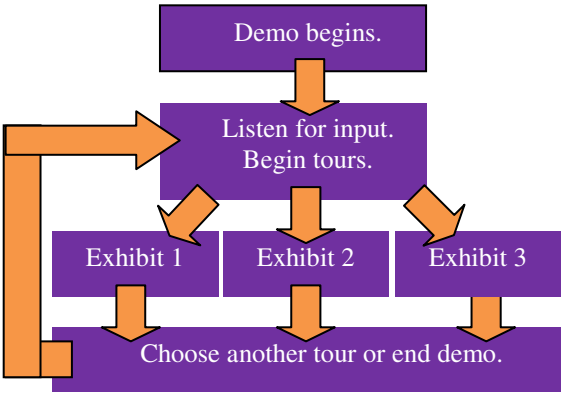

Fig. 3. Basic state machine for CLEM

For CARA, timing the sequence of events was straight forward except for animations and audio. Animations and sounds trigger, but don't block and have no way of communicating that they are finished. We had to time these and build wait times into our scripts. Audio files and animations also had to load from the SL server into the SL viewer, which took time and changed the wait time needed by the script, meaning that the timing was always uncertain, depending on the speed and traffic on the network connection. There is a flag that can be set to trigger a preload of an audio file before they are needed, but either the flag just does not work, or the amount of audio we wanted to preload was too much for the SL viewer's cache.

For CLEM, we not only had to build wait times into the C# script for the sounds and animations, but there was an additional lag between the input of the commands into the chat channel and the listeners on our scripted objects being fired. Because our state machine was outside SL, this lag was a major difficulty in timing our scripts for animation and movies. CLEM's voice, however, was being generated by the C# script, so there was no lag on his voice.

SL does not provide a voice recognition capability. For CARA we used Dragon Naturally Speaking and for CLEM we switched to Microsoft's SAPI. In both cases we implemented a keyword-based structure that listened for certain words only, and sent commands to SL based on what keywords were recognized. The CARA system sent only what it heard, maintaining no state, while the CLEM implementation could send more detailed information because both the state machine and voice recognition were outside of SL.

At the time of our project, SL used a proprietary tool, VIVOX, for both voice chat and lip-synching. The system can be set up to lip-synch audio coming through a SL voice chat cannel. For CARA, we prerecorded voice audio files and loaded them into SL. Our script would trigger the audio, which was then fed through a microphone on that computer through voice chat to trigger the avatar to lip-synch. CLEM used text-to-speech instead of prerecorded audio files, using a C# SAPI script. This was also fed through the voice chat for lip-synch. We did this in both cases by cabling our audio

speakers directly into a microphone. Using this setup, we had to turn the lip-synching off with our script every time the avatar finished speaking so that the avatar wouldn't lip-synch all the audio sources in the virtual world.

3.4 Modeling and Streaming Media

Working in the SL environment does provide free and straight forward 3D modeling tools for building content for the virtual world that an agent inhabits. In addition, many SL users build content that they give away to other users for free or sale to other users for a modest price. As a result we used free assets to build a very rich environment for our CLEM world, including trees, plants, birds, animals, and textures. Screen shots are shown in figure 4.

Fig. 4. Screen shots of the CLEM environment

Since we had video content to illustrate many of the projects that we wanted to feature on our tour we planned to imbed this content into the virtual world that we had created. Second Life provides many useful controls when working with streaming media, including the capability to play a video, provided it is in the correct format, from any web address onto any object of any size or shape in the virtual world. However, making this work in the context of a demo that has numerous collaborators, timing, dynamic content, and multiple media locations was very complicated.

Collaboration was an issue because of the way SL handles permissions for who can modify the script controlling the media player. At present SL requires that the owner of the script is also an owner of that particular parcel of virtual real estate. The rest of the group could not have permissions to edit the script controlling the media player, and SL would prevent even the owner from making modifications once the object had been placed in the environment. If everyone became an owner it would also present a security issue as there are many things an owner can do that would jeopardize the integrity of the entire virtual world (especially if that person is a novice SL programmer).

Timing was an issue because each media playing object (in this case, a model of a rectangular projector screen) took a variable amount of time to load the content to be played. If the screens were not triggered early enough, the user would see the screens loading and possibly be subjected to a long wait before the content was actually viewable. Having the screens play dynamic content (for the CARA demo, the screen script selected a random media URL to play) was also a timing issue. Since the objects displaying media in SL are not aware of when their media is done playing,

event timings had to be recorded and maintained in their respective order. This allowed for the script to keep track of when the media was finished and, subsequently, when to proceed with the rest of the demo.

Having multiple locations playing media, such as in the CLEM demo, was an issue because of the global way SL controls media playing for the virtual island. The commands to play media given by one screen to itself also play media on any screen in the same virtual space. Parceling the virtual real estate into separate areas—each with a separate screen—proved ineffective. The user therefore had to be kept out of sight from the remaining screens while viewing the content of a particular exhibit.

4 Summary

The primary advantages of using SL as a development platform were the tool sets for modeling the environment and modifying the appearance of avatars. In addition, many objects such as trees, birds and water simulations, were available for free and did not have to be modeled in-house. Second Life also provided automatic lip syncing, simple animation tools and a scripting language that allowed us to program object behaviors such as the opening of the gates to enter the tour and the movement of the vehicle.

Lack of support for databases, text-to-speech, and voice recognition in the SL programming environment meant that we had to move these tasks outside of SL to a C# script and use SL chat channels to transfer data between SL and our C# script. The most surprising problem was the limited SL capacity for scripted animations. The scripting language can be used to program object movements other than avatars but the tools provided are inadequate for creating smooth animations.

Despite limitations the final product was completed on time and has been shown to potential students at regular Friday demo tours in the School of Computing for a year. Although the animations were coarse, the visual models were rich and inviting. To quote one visitor to the lab: “We have been on tours and seen project demos at some of the best universities on the East coast. Nowhere have we seen a system build by undergraduates that was this impressive!”

Acknowledgments. This research was supported, in part, by NSF Research Experiences for Undergraduates (REU) Site Grant CNS-0850695. Janelle Arita, Freddie Dunn III, Hugh Kinsey III, and Nichole Wilson helped with the programming and modeling of this project.

References

1. Stevens, A., Hernandez, J., Johnsen, K., Dickerson, R., Raij, A., Harrison, C., DiPietro, M., Allen, B., Ferdig, R., Foti, S., Jackson, J., Shin, M., Cendan, J., Watson, R., Duerson, M., Lok, B., Cohen, M., Wagner, P., Lind, D.: The Use of Virtual Patients to Teach Medical Students Communication Skills. The American Journal of Surgery 191(6), 806–811 (2005)
2. Ziemkiewicz, C., Ulinski, A., Zanbaka, C., Hardin, S., Hodges, L.F.: Interactive Digital Patient for Triage Nurse Training. In: Proceedings of HCI International: Virtual Reality (2005)

3. Bertrand, J., Babu, S., Polgreen, P., Segre, A.: Virtual agents based simulation for training healthcare workers in hand hygiene procedures. In: Safonova, A. (ed.) IVA 2010. LNCS, vol. 6356, pp. 125–131. Springer, Heidelberg (2010)
4. Babu, S., Suma, E., Barnes, T., Hodges, L.F.: Can Immersive Virtual Humans teach Social Conversational Protocols? In: Proceedings of the IEEE International Conference on Virtual Reality, pp. 215–218 (2007)
5. Swartout, W., Traum, D., Artstein, R., Noren, D., Debevec, P., Bronnenkant, K., Williams, J., Leuski, A., Narayanan, S., Piepol, D., Lane, C., Morie, J., Aggarwal, P., Liewer, M., Chiang, J.-Y., Gerten, J., Chu, S., White, K.: Ada and Grace: Toward Realistic and Engaging Virtual Museum Guides. In: Allbeck, J., Badler, N., Bickmore, T., Pelachaud, C., Safonova, A. (eds.) IVA 2010. LNCS (LNAI), vol. 6356, pp. 286–300. Springer, Heidelberg (2010)
6. Cutler, B.F., Daugherty, B., Babu, S., Hodges, L.F., Van Wallendael, L.: Creating Blind Photoarrays Using Virtual Human Technology: A Feasibility Test. Police Quarterly 12, 289–300 (2009)
7. Maya, http://usa.autodesk.com/adsk/servlet/pc/index?id=13577897&siteID=123112
8. DI-Guy, http://www.diguy.com/diguy/
9. Unity3D, 12, http://unity3d.com/
10. Blender, 13, http://www.blender.org/
11. Bainbrigde, W.: The Scientific Research Potential of Virtual Worlds. Science 317(5837), 472–476 (2007)
12. Second Life, http://www.secondlife.com/
13. Active Worlds, http://activeworlds.com/
14. World of Warcraft, http://us.battle.net/wow
15. Dickey, M.D.: Three-Dimensional Virtual Worlds and Distance Learning: Two Case Studies of Active Worlds as a Medium for Distance Education. British Journal of Education Technology 36(3), 439–451 (2005)
16. Boulon, M.N., Hetherington, L., Wheeler, S.: Second life: An overview of the potential of 3-D virtual worlds in medical and health education. Health Information Library J 24, 233–245 (2007)
17. Nardi, B., Harris, J.: Strangers and Friends: Collaborative Play in World of Warcraft. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (2006)