Virtual Rendering based Second Life Mobile Application to Control Ambient Media Services

A S M Mahfujur Rahman, Abdulmotaleb El Saddik
Discover Lab, University of Ottawa
800 King Edward Ave, Ottawa, Canada
+1 613 562 5800
{kafi, abed}@mcrlab.uottawa.ca

ABSTRACT
In this paper we propose the development details of a mobile client that allows virtual 3D avatar interaction and virtual 3D annotation control in Second Life. We established adaptation based virtual rendering of the Second Life client and encoded the real-time frames into video stream, which is suitable for mobile client rendering. Additionally, we re-mapped the touch-based interaction of the user and feed that to the Second Life client in a form of keyboard and mouse interactions. As a proof of concept, we annotated a virtual environment object in Second Life and linked that with a media service by UPnP [5]. Further, we captured the mobile interaction of the user and provided controller interface to change states of the media object through the virtual object interaction. We argue that by using the mobile Second Life virtual interface the user has a better look to monitor and control the home appliances. We present illustration of the prototype system and show its application in a smart environment setup.

Keywords
Virtual rendering, virtual home, Second Life, virtual world, virtual display, thin client

INTRODUCTION
Second Life is one of the most popular 3D virtual environments that acts as a medium of social interaction where people can build their virtual 3D home, customize and populate it with 3D furniture and other interactive devices. With the growing popularity of Second Life (SL) [13], users often design their virtual 3D homes mimicking that of their real smart homes. Smart homes are technologically augmented spaces where several interconnected devices; artifacts and other ambient services are available to support people. The lighting control service, ambient media service, and security service are few examples of services a smart home application may incorporate. With the decrease in the prices of electronic sensors and automation devices such as X10 [6], their usage to provide control and various entertainment facilities in smart spaces are becoming hugely popular. In order to efficiently control ambient media services inside a smart space, researchers have placed attention on the intuitive 3D GUI design and explored its usability issues [9][17]. Compared to WIMP like 2D interfaces, a 3D user interface such as Second Life can improve interaction with smart home [14, 11].

Natural interaction with devices frees people from working in a desktop-like setting and provides intuitiveness in accessing various services of interest [12]. For example, when reading a newspaper, a person might want to turn on the media player while sitting on the couch using his/her mobile device. Users need support to get access to smart home anywhere and anytime. However, currently the mobile devices lack the graphics horsepower to render SL’s 3D environment in a satisfactory way. SL uses gigabytes of textures, sounds and animation data that entail a huge amount of network download time and CPU processing time that can adversely affect frame rate, draw distance, render detail, and mostly the battery life. Therefore, in order to render SL in mobile device we adopt virtual display technique that incorporates device collaboration [19][16], where most computations are carried out remotely by leveraging remote computing resources, and screen updates are compressed and transmitted together with low-level display commands from servers to clients [18].

In the THINC [7] and the MobiDesk [8] architectures, virtual display buffer management and screen resizing were used for devices with different display resolutions. One of the closely related work was presented in ReDi [19] that proposed interactive virtual display system for ubiquitous devices using thin clients, through which users can efficiently leverage the local display capability and remote computing resources. However, unlike our approach additional portable kit was required to be attached with the display surface to provide a thin client with the said mobility option.
In this paper we propose the design and development of a mobile 3D virtual interface of Second Life [9] [17] in order to access smart home media services. Our contribution in this paper is two-fold. First, in order to bridge the gap between virtual and real, we present a Second Life virtual environment annotation based smart space automation system, where we provide web-service based architecture for controlling our previously developed ambient services [15]. Second, we devise a flexible system architecture for local and remote device communication by incorporating adaptive screen compression, interactive Region of Interest (ROI) control, and touch-based input processing. Our experiment shows that the proposed system can efficiently utilize the remote computing resources to render SL content and leverage that to access the ambient services.

The remainder of this paper is organized as the following. At first, in Section 2 we illustrate the proposed visual adaption scheme to render SL content in ubiquitous devices. Further in Section 3 we describe Second Life virtual annotation based ambient service interaction. At the end we provide conclusion of the paper in Section 4 and state some possible future work directions.

**Event based Adaptive Rendering**

As depicted in Figure 1, in order to access SL content in mobile screen, we incorporated device collaboration scheme [18], where the Second Life virtual environment rendering took place at a remote desktop computer. Additionally we utilized the computing resources of that same to encode the Region of Interest (ROI) of the Second Life client window in H.263 format with RTP packetization support. The libavformat library of FFmpeg [2] options for video container muxing and transcoding of the multimedia files. The compressed and encoded screen updates were streamed to the mobile client by using real time streaming protocol (RTSP) [3]. The mobile client were able access the encoded content by listening to the incoming UDP [4] packets with the help of a custom streaming application that was developed from VideoView library [1].

The ROI window size of the desktop Second Life was kept at a constant \( R = [640 \times 480] \) sized rectangular block. The open source ffmpeg encoder scaled the screen updates into \( 320 \times 240 \) H.263 video frames. In the desktop remote computer, RTSP was used to start the streaming. It provided us an SDP [10] profile that contained the description of the media type, format, and other associated properties. We designed our encoder system to manage an RTCP \(^1\) connection and a RTP \(^2\) encapsulation of the video channel. However, we did not implement the audio channel transcoder, as our focus was to interact and control smart home appliances only. While RTP carried the video stream, RTCP was used to monitor transmission statistics and the quality of service (QoS) parameters of the media stream. The RTP packets were streamed over UDP and we avoided the overhead of retransmitting lost packets.

The mobile client listened to the client_port of the RTSP stream and rendered the incoming media streams. RTSP had the overhead of requiring multiple requests before the playback can begin. However, the mobile client pipelined many of these requests and sent them over a single TCP connection. As discussed currently, the mobile client only supported video rendering without audio and performs only in landscape orientation (no accelerometer support). The video quality in thin-client based remote access is measured by the frame update rate as proposed by Yang et al [20].

---

2. RTP, wikipedia.org/wiki/Real-time_Transport_Protocol
We were able to render SL content in the mobile client at 6-12 FPS with 320x240 resolution support. The remote rendering of SL screen ROI in the thin mobile client is shown in Figure 2.

References and Citations
On the top layer of the streaming mobile client application we added interaction-capturing interface. The interface listened to the touch interactions of the user and reported that to the desktop Second Life renderer through a buffered input pipeline. The pipeline implemented as a queue, processed the input list in Fast-In-First-Out (LIFO) fashion. However, the recent mobile clients that we adopted only accepted touch-based interactions. Hence, we proceeded with the interaction mapping and used the Table 1 as reference in the input processing.

When a user tapped on the touch screen we mapped that touch as a single mouse click and packaged that information along with the screen coordinate position where the tap took place. By associating delay in the tap we identified long tap and double taps, the mapping scheme converted these interactions into a right click and a double click respectively. In order to map the avatar navigation keys, we used our previously developed motion path analysis of the touch surface [15] and obtained four swiping symbols, namely left, right, up, and down. These four symbols were later mapped with the left, right, up, down arrow keys of the keyboard respectively. By using the said swipe touch gestures, we made it possible to navigate SL 3D home and interact with the 3D objects. Furthermore, we encapsulated the packet data with event messages that SL 3D home and interact with the 3D objects. Furthermore, we encapsulated the packet data with event messages that included the screen coordinate position (S={x,y}), and keyboard character or mouse click type data.

Table 1: Interaction mappings to traditional WIMP controls to conform touch gestures

<table>
<thead>
<tr>
<th>ID</th>
<th>Touch Events</th>
<th>WIMP Mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left swipe</td>
<td>Left key</td>
</tr>
<tr>
<td>2</td>
<td>Right swipe</td>
<td>Right key</td>
</tr>
<tr>
<td>3</td>
<td>Up swipe</td>
<td>Up key</td>
</tr>
<tr>
<td>4</td>
<td>Down swipe</td>
<td>Down key</td>
</tr>
<tr>
<td>5</td>
<td>Tap</td>
<td>Single click</td>
</tr>
<tr>
<td>6</td>
<td>Double tap</td>
<td>Double click</td>
</tr>
<tr>
<td>7</td>
<td>Long tap</td>
<td>Right click</td>
</tr>
</tbody>
</table>

AMBIENT INTERACTION SCHEME
In order to communicate with ambient media service we created 3D virtual objects in Second Life, representing the service access points for various ambient services such as a lamp or a media player. Further, we annotated a 3D lamp, media object in Second Life and added web-service calls to it in its mouse click event. We describe the object annotation scheme in section 3.1 and the web-service based remote communication approach in Section 3.2. Later in section 3.3 we illustrate the real lamp/media player control mechanism.

Object Annotation
In our system we annotated Second Life 3D objects and specified the corresponding ambient service addresses in it. We incorporated lightON, lightOFF, playerON, playerOFF animation for the 3D lamp and media objects in SL. These animations were defined in the annotation file and associated with the ON or OFF service states of the real world lamp and player. The animation sequence for the 3D lamp object contained an animation unique identifier (UUID) that worked as a pointer to the animation file (BVH), where animation speed, duration etc. were defined. This file also specified the physical device specific data like UPnP port forwarding address, name, type etc. The physical lamp device was connected with the ambient services by using X10 or Wi-Fi connections. Similarly, in second life a 3D lamp object was annotated using the Second Life’s built-in script annotation mechanism3.

Interaction with media player application occurred implicitly, where the media selector service automatically invoked context-aware services through the mobile interaction device. In Second Life, the 3D media player was annotated to customize the mouse click event. The mouse click event was overloaded with the remote web-service call that communicated with the ambient media selector running in the same workstation.

Remote Communications
The Second Life provides both commercial and open source versions of its client that is termed as viewer. The viewer brings a wide range of web-service and communication handling APIs that can be leveraged to create listeners for events inside Second Life. Web service APIs that are supported in Second Life are a) Raw HTTP access, where the requests are initiated through the events written in LSL scripts, b) XmlHTTP access paradigm, here requests are initiated by external services and c) Email option provides full two-way communication, but with enforced sleep timers.

This web-service based remote communication architecture provided the option to incorporate real-virtual interactions without affecting the functionality of the Second Life communication system. Inside Second Life we adopted the RAW Https access interface to perform machine-to-machine http communication. Each LSL scripted object had a maximum of 2048 characters limit for the http responses. The llHTTPRequest runs entirely on the simulator running the script, hence, the messages communication was not impaired because of the overloaded central Second Life servers. By using the mobile interface, the user navigated the 3D Second Life home with his/her avatars. When the 3D lamp/player object was interacted by the user, the developed LSL script module captured the messages relating to the real device in the mouse click event. By

using the xmlHttp access mechanism we created an UPnP response object for the ambient media selector (AMS). AMS further captured the response object and parsed that to obtain service port address ID. Lastly, by decoding the port addresses ID the AMS then sent control signal to either turn the media service state to be ACTIVE or INACTIVE. This process is called “port forwarding” and works seamlessly in this situation. With this process shown in Figure 4, we mapped a port forwarding programmatically without user interaction.

The SL script utility that contained annotated port mappings on an UPnP-enabled router allowed flexible control (add/edit/delete) of mappings. The utility was conceptually broken into two pieces: a media service that performed the actual work, and the mobile UI that invoked the service.

**Interaction Controller**

In the ambient framework, we enabled the access of various media services and facilitated the control of ambient lighting using a smart phone interface. User’s media selection is played in VLC media player and ambient lighting is controlled using X10 home automation technology. To integrate mobile touch gesture with this framework, we first defined several gestures that can be mapped to specific operations required for accessing the available services. Recall, our proposed mobile gesture technique was based on touch motion path definition and hence it gave us the freedom to define customized gestures that are more natural in the given context and is easy for the user to remember.

**Figure 3. Sample LSL code that loads the XML data to process messages towards the communication channel.**

In a typical interaction scenario, the user navigated to the Second Life smart home by maneuvering the avatar with left, right, up, down touch based swipe gestures. The user could tap on a 3D lamp/player object to turn it ON inside the smart home and in turn the interaction-mapping algorithm would be generating a mouse click event. After receiving the remote interaction messages through scripts, the GUI events were generated in the remote Second Life client machine. Based on scaling method nS = R = {x,y}, where n is a constant, mobile screen to window ROI screen coordinates were calculated and mouse click event at the calculated R location was triggered. Afterwards, the event handler module of Second Life determined the particular event handling routine for the specified 3D lamp/player object. The handler then transferred UPnP response packet to the AMS in order to communicate with the actual home automation services.

**CONCLUSION**

In this paper we proposed a novel system to facilitate ubiquitous user interaction with ambient media services by using device adaption based mobile Second Life client system. The approach targeted to bridge the interaction gap between the virtual and real world media service selection mechanism in a mobile context. The developed system worked as a web-service and loosely coupled to the Second Life viewer. The animation and device control data were annotated in the 3D virtual object in Second Life. The 3D object representing a real device received inputs when interacted by a user and automatically responded with the state changes in the physical environment. The mobile client based pervasive access to the Second Life 3D environment is easy to use, and intuitive. The association of real lamp object to the virtual 3D object was natural and easy to understand. The proposed approach facilitated mobile use of digital devices by combining advanced display technologies, natural user input mechanisms, and remote high-performance computing resources with improved user accessibility.

In our future work we plan to evaluate users perception of the system in which we want to understand the ease of use, presented advantages, and intuitive factors of the prototype system. By using the motion path tool we plan to incorporate more gestures into the system.
We envision that these gestures would provide more control and support users with intuitive interactions capabilities. As in our system it is essentially easy to relate the real and the virtual 3D device representation we also want to explore the application in the elderly home control system.

REFERENCES