The DynaDOOM Visualization Agent: A Handheld Interface for Live Action Gaming

Gita Sukthankar
Cambridge Research Laboratory
HP Labs
One Cambridge Center
Cambridge, MA 02142 USA
gita.sukthankar@hp.com

ABSTRACT

This paper presents the DynaDOOM Visualization Agent, a system for providing first-person visualization of live-action multiplayer games. Participants carry a handheld computer instrumented with location sensing devices and a wireless 802.11b connection. A smooth trajectory of each player's motion is automatically generated from position estimates using the Dynadraw algorithm, and translated into commands that are fed to unmodified first-person shooter game clients. Spectators can view the action in a 3-D virtual environment that mirrors the real setting, either from the perspective of one of the players or from the viewpoint of a virtual spectator.

Keywords

context awareness, location detection, pervasive computing, embedded systems, intelligent interfaces

1. INTRODUCTION

Our goal is to create visually compelling interfaces through which spectators can follow the activities of participants in a live-action indoor multiplayer game. The DynaDOOM Visualization Agent provides a first-person perspective of each player's actions in a virtual world that mirrors the physical gaming environment. Each player is equipped with a handheld computer instrumented with location sensing devices and a wireless 802.11b connection. Location Agents monitor players as they move through the real world and convert noisy position data into smooth trajectories augmented with orientation information. Visualization Agents translate these into appropriate first-person shooter actions and generate synthetic keyboard and mouse events. Each Visualization Agent interacts with an unmodified multiplayer game client (such as DOOM), enabling our system to be used with any standard first-person shooter application. Spectators can watch the game either from the perspective of one of the players, or from the viewpoint of a non-participating character in the virtual environment. The visualization complements raw video feeds provided by any handheld cameras carried by participants. The DynaDOOM Visualization Agent was first publicly demonstrated at MIT in February 2002 to visualize the movements of players participating in an indoor version of Capture-the-Flag [1]. Although the system was designed for use in a game setting, we believe that it can be extended for augmented reality applications.

Figure 1: Each participant in the CTF game is equipped with the hardware shown above. Left: the iPAQ H3650 handheld and BackPAQ interface sleeve. The BackPAQ provides several I/O options including an accelerometer, camera and 802.11 wireless communication. Right: the Cricket listener (indoor location sensor) provides an estimate of the user's global position.

Although most game engines are engineered to appeal to the popular audience, they have been successfully used both as test beds for autonomous agent research [7, 8, 9] and as part of virtual reality systems [6]. Here we attempt to give the players limited access to a virtual world through a handheld gateway, without encumbering them with head mounted displays and backpack computers [10].

2. HANDHELD PLATFORM

The user interacts with the virtual environment through a handheld interface running on an iPAQ H3650, a commercially available PDA, using the freely-distributed Linux Familiar distribution [2, 4] (see Figure 1). By using Linux as our embedded systems platform, we leverage open source code originally written for desktop sys-

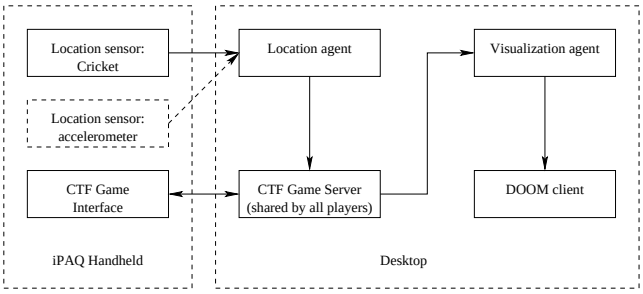

Figure 2: This diagram shows the architecture of each participant's agent in the DynaDOOM visualization system. Information about the iPAQ handheld's location is sent (via wireless) to the location agent and integrated into the CTF game server. Successive, noisy global position estimates are translated into first-person DOOM movement commands and sent to the Visualization Agent. All of the CTF participants move and interact in the DOOM world shown in Figure 4.

tems, such as XWindows. Linux also offers full multi-tasking and networking functionality that is needed for pervasive computing applications. Development tools for most languages are available on the Linux iPAQ; this project uses components developed in C, Java and Python.

The iPAQ handheld is augmented with a *BackPAQ*, a prototype expansion pack developed at the Cambridge Research Lab [5]. The BackPAQ hardware provides a variety of useful I/O devices for pervasive computing applications: a camera, an accelerometer and two PCMCIA slots for wireless networking and additional storage devices (see Figure 1). iPAQ handhelds with BackPAQ attachments running the Familiar Linux distribution have also been adopted by other research programs, such as MIT's pervasive computing effort, Project Oxygen [13].

3. SYSTEM ARCHITECTURE

Figure 2 summarizes the system architecture. Each participant's state is maintained by a pair of agents: the *Location Agent* and the *Visualization Agent* (detailed below). Since computational resources on the handheld computer are limited, both of these agents run on desktop machines, on data provided by components running on the iPAQ. The *CTF Game Server* maintains the global game state and communicates with each participant's *Location Agent* and his/her iPAQ game interface. This interface is a lightweight Java GUI (see Figure 3) that supports player-to-player communication, shooting, and a 2-D overhead view of the game. The CTF Game Server also provides personalized game state to each player's *Visualization Agent*. This agent converts the information into DOOM actions that are fed (as synthetic keyboard and mouse events) to an unmodified DOOM client. Spectators view a 3-D first-person perspective of the game on displays projected from the desktop computers (see Figure 4). The following subsections present selected aspects of the system in greater detail.

3.1 Location Agent

Each handheld is equipped with two hardware sensors for detecting location information: a Cricket listener and the BackPAQ accelerometer. Raw sensor information is published on two daemons (`cricketd` and `acceld`) running on the handheld. The *Location Agent* acquires raw sensor data from the handheld and derives an

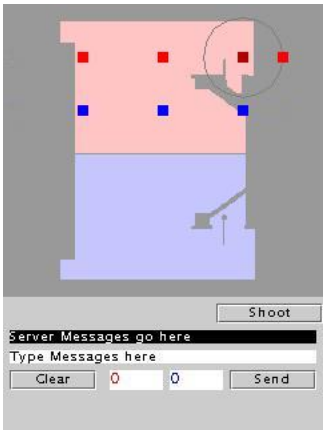

Figure 3: This is a screenshot of the CTF interface on the iPAQ handheld. In addition to serving as a gun and walkie-talkie, the application provides a virtual aerial view of the CTF battlefield. Locations of team-mates and known enemies are updated in real-time. A first-person view of the game environment is simultaneously rendered in the DOOM engine and made available to spectators on a projected display (see Figure 4).

estimate of the handheld's global position.[1]

The Cricket location support system [11] was developed to be a low-cost decentralized solution for devices to obtain their location in indoor environments where GPS is non-functional. A listener device is attached via serial cable to the iPAQ, and the indoor environment is instrumented with Cricket beacons at known locations. Beacons emit RF/ultrasound "chirps" that are detected by the listener device; a daemon on the iPAQ publishes distance estimates to each beacon, derived from ultrasound time-of-flight information. By triangulating readings from multiple beacons, the Location Agent derives an estimate of the player's current global position.

Before the game, beacons were taped onto the ceiling in the game area. Since the beacons are completely self-contained, requiring neither network or power, they can be positioned anywhere in the environment. For the scenario, we chose an arbitrary x-y coordinate system and tabulated each beacon's coordinates in a file used by the Location Agent. Changes in the coordinate frame only require modifying the configuration file without physically relocating the beacons; the same beacons can support many applications, each with its own separate coordinate system.

Since the Cricket system was primarily designed for pinpointing the locations of quasi-stationary objects, it does not provide reliable data while the listener device is moving. In our application, where the participants are rarely standing still, using raw Cricket data results in an unacceptably-high positioning error. Furthermore, the standard Cricket system does not provide any orientation information[2] Our solution to both of these shortcomings is discussed in Section 3.2.

[1]Since the BackPAQ prototype currently provides only 2-D accelerometer data, our implementation relies solely on the Cricket system for position estimates. Accelerometer information will be integrated into the next version of our system.

[2]The new Cricket compass system [12] employs listener devices with multiple receivers and should provide an estimate of the device's heading.

3.2 Visualization Agent

Each participant's Visualization Agent has the task of taking positional information calculated by the Location Agent and computing a smooth trajectory (including orientations) that can be expressed as DOOM commands (keystrokes and mouse events).

As discussed above, the raw global position data presents two challenges. First, the location sensors do not provide any orientation information. Given the absence of actual data, the Visualization Agent makes the reasonable assumption that the player is facing in the direction of motion. Second, the position estimates are corrupted by noise; simply connecting successive estimates from the location sensors results in an avatar that blips randomly around the virtual world (even when the player is stationary in the real world). We explored several filtering approaches to this problem and selected a second-order dynamic filter motivated by Dynadraw [3] (a drawing tool that refines users' noisy pen strokes using a second order dynamic model). Rather than treating incoming location estimates as new absolute positions, each new data point exerts a force on the avatar's current position according to following mass/spring/damper model:

$$\begin{aligned} \ddot{\mathbf{x}}_t &= \kappa||\mathbf{x}_{t-1} - \mathbf{p}_t||^2 \\ \dot{\mathbf{x}}_t &= (1-\beta)(\dot{\mathbf{x}}_{t-1} + \ddot{\mathbf{x}}_t \Delta t) \\ \mathbf{x_t} &= \mathbf{x}_{t-1} + \dot{\mathbf{x}}_t \Delta t \end{aligned}$$

where Δt is the time interval between successive measurements, $\mathbf{p}_t$ is the location measurement received at time t, and $\mathbf{x}_t$ is the current (smoothed) estimate of player position. The control parameters, κ (normalized spring constant) and β (damping coefficient) were tuned until the avatar's motion in the virtual world seemed smooth without appearing too sluggish.

The DOOM actions are generated as follows. A change in orientation is mapped to the "turn left" and "turn right" keys. The mapping between turn angle and keypress duration was determined empirically (by measuring the time it took the DOOM avatar to rotate through a series of known angles). Similarly, the scaling factors used to convert between distances in the real world and keypress interval for forward motion were determined by measuring the time it took for the DOOM avatar to move in a straight line between known locations in the virtual world. Synthetic keyboard and mouse events are generated using the Java `Robot` interface, and sent to the gaming client. For additional control, the user can press buttons on the iPAQ handheld to generate specific synthetic events in the virtual world. This enables users to perform actions that cannot be perceived by the location sensing alone (e.g., firing the gun or opening a door). The same interface can also be used to manually tweak the avatar's movement in the virtual world, if necessary. The Visualization Agent must be co-located with the DOOM client due to security restrictions on synthetic events, whereas the Location Agents can be distributed across as many computers as are available.

3.3 CTF Game System

In addition to the location and visualization agents described above, each player was equipped with a 2-D user interface that allowed him/her to interact with the CTF (Capture-The-Flag) game system. The CTF game server monitored all state specific to the Capture-the-Flag scenario: player state, game time, score, territory boundaries, flag position. Players accrued points for their team by moving the enemy's flag (a specially designated Linux iPAQ instrumented with a Cricket listener) into their own territory. While in enemy territory, the player was vulnerable to being tagged by players of the opposing team. Due to the difficulty in distinguishing orientation with this version of the Cricket system, any player of the opposing team within a certain radius of the shooting character was tagged. Once tagged, players were relegated to a "virtual penalty box" during which time their CTF client was disabled, preventing them from interacting with the game server. Although player movement was tracked and displayed in the virtual DOOM world, actions in the DOOM world (e.g., firing gun, opening doors) had no effect on the state of the CTF game.

3.4 DOOM Client

Our current implementation uses `lxdoom` as its visualization engine. `lxdoom` is a multiplayer version of the DOOM first person shooter game originally released by Id software in 1993. Although its graphics are not as impressive as some of the newer first-person shooter games (e.g., Quake III, Unreal Tournament), `lxdoom` does offer several benefits including freely available source code, the existence of a stable iPAQ Linux port, freely available level editing tools and reliable support for networked multiplayer gaming. Figure 4 shows the DynaDOOM Visualization Agent connected to an `lxdoom` client.

Note that DynaDOOM Visualization Agent is completely compatible with any multiplayer gaming system since the Visualization Agent interacts with the game client exactly as a human player would—through keyboard and mouse actions.

4. DISCUSSION

An important question for designers of such environments is how to partition the game between the virtual world and the physical space. On one extreme, the virtual world could bear little resemblance to the player's physical environment. In this case, the player would be required to make movements in the real world while constantly monitoring the effects of those actions in the virtual world (e.g., using a handheld display). As the real and virtual worlds diverge, the mapping between the player's actions and their virtual consequences becomes less intuitive. Ultimately, this leads to an abstract interface, similar to moving in the virtual world using keyboard actions. At the other extreme, the virtual world could be an accurate mirror of the real world. Since physical actions map closely to virtual actions, this allows players to play the game without monitoring the virtual world. However, accurate localization and interpretation of player actions is critical since any divergence between real and virtual worlds becomes very disorienting to participants.

In our implementation we decided to use the Visualization Agent solely to mirror the physical world to the spectators; however, it could easily be adapted to fit other paradigms of virtual world interaction. For some game settings, it might be interesting to have the players accomplish tasks in a virtual world setting before being able to move on to some other stage of the physical game; this type of interface could be well suited for a treasure hunt or adventure game motif.

5. CONCLUSION

This paper presents a system for visualizing real-world multiplayer games in a virtual environment. The current implementation of our system has been successfully demonstrated at MIT for a Capture-the-Flag game application.

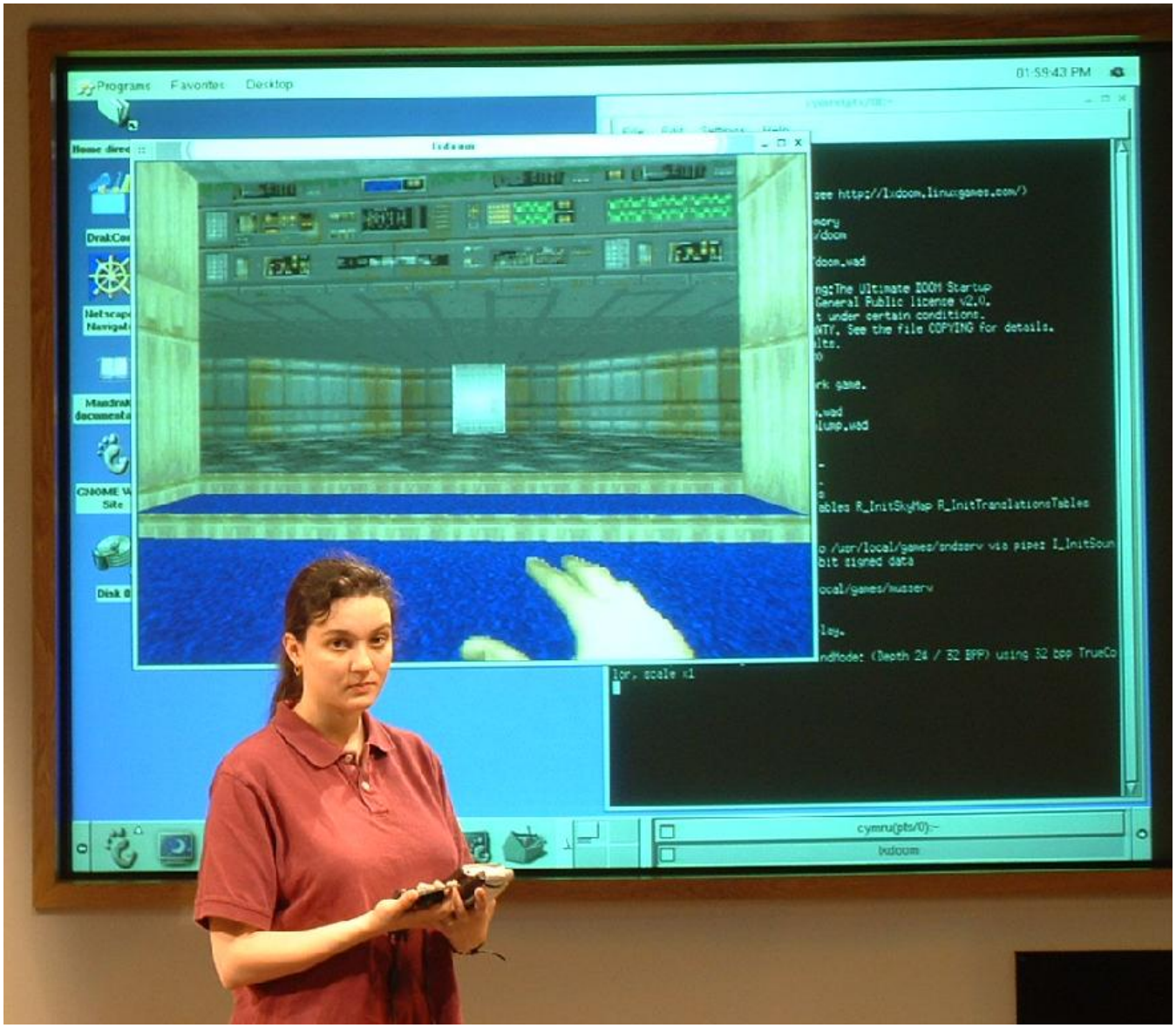

Figure 4: This photo shows a CTF participant with the DynaDOOM Visualization Agent projected in the background. Each user's position in the room is estimated using the Cricket location system and appropriate DOOM commands are generated to move the user's avatar in the virtual world.

We plan to extend the DynaDOOM Visualization Agent in the following ways. First, we will improve the Location Agent by exploiting hardware advances in the BackPAQ and Cricket systems: 3-D accelerometer input will be combined with CricketCompass [12] location and orientation data using a Kalman filter. Second, we wish to expand the player's interface to include an augmented reality display using the BackPAQ camera. The goal is to annotate the camera images with useful game information (e.g., locations of enemy players, or best route to the target area). Finally, we will extend the Visualization Agent to incorporate information about the state of the virtual world, either by interpreting the output of the DOOM client, or through separate interactions with the DOOM server. With these enhancements, the DynaDOOM Visualization Agent will transform the live-action gaming experience.

6. ACKNOWLEDGMENTS

We would like to thank the following research groups for their valuable assistance: Open Handhelds developers, the MIT Capture-the-Flag team and the MIT Cricket Project. Thanks also to Frank Bomba, Jamey Hicks, Ken Steele, Larry Rudolph and Rahul Sukthankar for helpful advice and feedback.

7. REFERENCES

[1] A. Champaneria, C. Music, M. Bourget, S. Garg, G. Sukthankar, and P. Ho. Capture-the-Flag using iPAQs, January 2002. `http://www.cs.cmu.edu/~gitars/CTF/`.

[2] Compaq Research – Handhelds Group. `http://www.handhelds.org/`.

[3] P. Haeberli. Dynadraw: A dynamic drawing technique, 1989. `http://www.sgi.com/grafica/dyna/`.

[4] C. Halsall. Linux on an iPAQ, 2001. `http://linux.oreillynet.com/lpt/a/linux/2001/06/01/linux_ipaq.html`.

[5] J. Hicks. Compaq Research: Mercury Project Home Page. `http://crl.research.compaq.com/projects/mercury/`.

[6] J. Jacobson and Z. Hwang. Unreal Tournament for immersive interactive theater. *Communications of the ACM*, 45(1), 2002.

[7] G. Kaminka, M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. Marshall, A. Scholer, and S. Tejada. Gamebots: A flexible test bed for multiagent team research. *Communications of the ACM*, 45(1), 2002.

[8] J. Laird. Research in human-level AI using computer games. *Communications of the ACM*, 45(1), 2002.

[9] M. Lewis and J. Jacobson. Game engines in scientific research. *Communications of the ACM*, 45(1), 2002.

[10] W. Piekarski and B. Thomas. ARQuake: The outdoor augmented reality gaming system. *Communications of the ACM*, 45(1), 2002.

[11] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket location-support system. In *Proceedings of ACM MOBICOM*, 2000.

[12] N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. The Cricket Compass for context-aware mobile applications. In *Proceedings of ACM MOBICOM*, 2001.

[13] Project Oxygen Home Page. `http://oxygen.lcs.mit.edu/`.