Developing an Augmented Reality Racing Game

Ohan Oda Levi J. Lister Sean White Steven Feiner

Department of Computer Science
Columbia University
New York, NY 10027

{ohan, ljl2116, swhite, feiner}@cs.columbia.edu

Figure 1. Augmented reality racing game. (a) User controlling virtual car on tracked physical gameboard. (b) Car collides with virtual signpost. (Images in all figures are captured live during game play.)

ABSTRACT

Augmented reality (AR) makes it possible to create games in which virtual objects are overlaid on the real world, and real objects are tracked and used to control virtual ones. We describe the development of an AR racing game created by modifying an existing racing game, using an AR infrastructure that we developed for use with the XNA game development platform. In our game, the driver wears a tracked video see-through head-worn display, and controls the car with a passive tangible controller. Other players can participate by manipulating waypoints that the car must pass and obstacles with which the car can collide. We discuss our AR infrastructure, which supports the creation of AR applications and games in a managed code environment, the user interface we developed for the AR racing game, the game's software and hardware architecture, and feedback and observations from early demonstrations.

Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques;* H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies, interaction styles*; K.8.0 [**Personal Computing**]: General—*Games*

General Terms

Human Factors, Design, Experimentation

Keywords

Augmented Reality, Mixed Reality, Gaming, Virtual Reality, Wearable Computer, Tangible Interaction, XNA

The Second International Conference on Intelligent Technologies for Interactive Entertainment (*ICST INTETAIN '08*). January 8–10, 2008, Cancun, Mexico. Copyright 2008 ICST. ISBN 978-963-9799-13-4.

1. INTRODUCTION

Most current computer games are locked within the confines of a computer, console, or hand-held display. The display acts as a small window onto the virtual game world, separating the player from the surrounding physical environment. User interaction is similarly restricted to a relatively small set of devices. Even systems that break this mold, such as the Nintendo Wii [32], are limited to what its controllers' buttons and sensors can support. In contrast, Augmented Reality (AR) [11] makes it possible to combine the physical and the virtual by displaying virtual images and sounds overlaid on the physical world, and can allow players to interact by manipulating existing real world objects. AR enables a new class of games that move game play into the physical envi-

ronment (e.g., [5, 40, 41]). AR can make the physical environment an integral part of the game, supporting multi-player experiences, enabling direct spatial interaction, and maintaining the real-world context of game play. One way of advancing AR gaming is to explore new development tools and techniques for game interaction.

In this paper, we present a novel multi-player AR racing game (Figure 1). We created our game by extensively modifying the existing XNA Racing Game [10], using an AR gaming platform that we are currently developing. We begin by discussing previous work in AR gaming. Next, we describe our goals for developing our AR racing game. We then introduce the architecture for the game, including its software, user interface, and hardware, describe the game play, and discuss preliminary user feedback. We finish by presenting our conclusions and ongoing and future work.

2. PREVIOUS WORK

2.1 AR Gaming

Some of the earliest research in AR games was conducted by Ohshima and colleagues to demonstrate advances in underlying hardware and software infrastructure, by creating, from scratch, multiplayer air hockey [36] and first-person shooter [35] games. In 2000, Piekarski and colleagues [1, 38, 41] extended the existing desktop game, Quake, into ARQuake, a first-person outdoor AR game. Movement of the player in physical space moves the point of view of the character in the AR game. In ARQuake, the player uses a two-button hand-held device to fire the weapon at a target at the center of the view seen through a head-worn display. Matysczok and colleagues [25] developed AR-Bowling, where players throw virtual bowling balls with tracked pinch gloves that detect hand gestures. The speed and spin of the virtual bowling ball vary according to a player's hand and finger movement. In all these systems, wired controllers, either held or worn, are used as interaction devices.

Knoerlein and colleagues [19] implemented an AR ping-pong game with a haptic interface in which the players receive force feedback through optically tracked ping-pong racket handles attached to SensAble PHANTOM haptic devices when hitting the virtual ping-pong ball. In contrast, AR Chinese Checkers [8] utilizes optically-tracked fiducial markers (printed black and white patterns whose pose and identity can be determined automatically) as input devices, in conjunction with attached wireless physical buttons for precise selection tasks. Barakonyi and colleagues [3] developed MonkeyBridge, in which players position similar fiducial markers to control virtual bridge components to construct a path along which virtual monkeys can travel. Cheok and colleagues [5] created an AR adventure game with a room-sized tracked area in which a player walks around and collects treasures to defeat a witch; a tracked hand-held wand is used to explore the game space.

AR games have also been implemented in tabletop environments with either front or rear projection. For example, Magerkurth and colleagues [23, 24] developed AR KnightMage, a rear-projected board-based role-playing adventure game. Physical game elements, such as dice and avatars, affect the virtual game world when moved and placed on top of the projected display. Lee and colleagues [21] developed a table-top card battle game using front projection. Optically tracked markers are attached below the cards, and the cards are visible to a camera underneath a transparent glass table.

As computers increased in power during the past few years, researchers have ported AR infrastructures to hand-held devices that are less costly, simpler, and lighter than earlier wearable AR systems. Wagner and colleagues [42] demonstrated an AR train game using a camera equipped PDA device. A miniature physical railroad track on a flat surface is surrounded by fiducial markers for tracking. Multiple players can each interact through a PDA. Game play involves switching the track direction at intersections to avoid collisions among several trains. Mobile-phone–based AR games [37] are also being actively investigated as an affordable AR system with multiplayer capabilities [4, 9, 15, 16].

We have been developing Goblin XNA [33], an infrastructure for creating AR games in particular, and AR applications in general, on top of an existing non-AR game development environment, XNA [29]. As a test case for our infrastructure, which emphasizes tracking and rendering for video see-through augmented reality, we decided to extend one of the "starter kit" games [30] already available for XNA. We selected the XNA Racing Game starter kit because it provided the richest set of assets at the time we chose it.

Wilson [43] has independently modified part of the XNA Racing Game, with the very different goal of demonstrating the utility of a prototype real-time depth camera. In his work, an overhead depth camera captures the height field and color image of real 3D objects on a table beneath the camera, which are combined to create an interactive textured mesh. Free-running cars drive over the terrain represented by the mesh, which can change dynamically as users move the objects. The cars are projected onto the physical table and objects by an overhead projector, while a separate display shows a fully synthesized view of the textured mesh and cars from an arbitrary viewpoint.

2.2 The Original XNA Racing Game

The XNA Racing Game starter kit is a typical racing game with a predefined track and environment. The single player tries to complete three laps as quickly as possible without driving off the track. The player is prompted first to select a track from among a small set of tracks with varying levels of difficulty, and next to select one of several possible cars. The player then accelerates, decelerates, steers left or right, and moves forward or backward with either keyboard/mouse presses or an Xbox 360 controller.

During the game, sound effects are triggered by events such as acceleration, braking, and collisions. A replay of the best lap performance is displayed using a phantom racing car every time the driver starts a new lap. From the second lap on, a time difference compared against the best performance is displayed when passing each waypoint. Finally, the ranking based on the previously played record is shown to the player after she completes three laps.

3. AR RACING GAME

In designing our game, we decided to support multiplayer interaction between a driver and additional players who can modify the game environment. To emphasize the interaction with the physical world and remove the artificial boundaries of the game world,

we eliminated the original predefined environment, including the track. Instead of being restricted to the track, the driver can drive the car anywhere she wants on the ground plane, which is defined by a planar array of optical markers forming a gameboard of essentially arbitrary size (covering several tabletops in our largest version). Virtual objects that act as physical obstacles are overlaid on the gameboard and can be attached to separate movable markers that can be manipulated by additional players.

We then redefined the completion of a lap to require the driver to pass through a series of designated waypoints in sequence, potentially slowed by collisions with obstacles. The driver views the game through a head-worn display tracked by an attached camera, and additional players watch stationary displays associated with other cameras. The head-worn display and additional cameras can be positioned and oriented arbitrarily in the environment, provided they have an adequate view of the gameboard.

Rather than relying on conventional wired or wireless game controllers, or special-purpose driving controllers, we decided to use optical marker tracking to track an otherwise passive physical artifact, as described below.

4. SYSTEM ARCHITECTURE

To create the AR Racing Game, we combined the XNA Racing game with our AR Gaming Platform (Goblin XNA), an optical marker tracking system (ARTag [13]) a physics engine (BulletX [44]), and a network message passing library (Lidgren [22]).

As depicted in Figure 2, our game uses three machines: a Controller Machine, a Driver Machine, and a God's Eye Machine. The Controller Machine captures an image from a camera and processes the image to detect the optical marker configuration of the driving controller and interpret its pose. It then sends this information to the Driver Machine over the (wireless) network. The Driver Machine uses a physics simulator to model the interaction of the car with the environment based on the driving controller input obtained from the Controller Machine. The Driver Machine renders the view from its physical camera (on the driver's head-worn display) and combines it with the view of the rendered virtual environment, as seen from a virtual camera whose position and orientation are derived by the optical marker tracking system, based on the physical camera's view of the gameboard. Note that the physics simulator can also cause objects other than the car to move in the virtual environment.

The God's Eye Machine receives, from the Driver Machine, virtual objects' positions and orientations; sound events; and game status data, including current car speed, lap number, number of waypoints passed, and time. It tracks the pose of the ground plane relative to its camera, renders the virtual models in their proper locations, and displays the game status data. To avoid audible synchronization delays, we currently play audio only on the God's Eye Machine, which is connected to a speaker system. (Were we to use headphones instead, we would play audio on both the Driver Machine and God's Eye Machine.)

To support multiple drivers, the Controller Machine camera must be able to see the additional drivers' controllers. Alternatively, additional cameras can be attached to the Controller Machine, or additional Controller Machines with their own cameras can be added. Each additional driver uses a Driver Machine similar to that of the original driver, with one important distinction. An additional Driver Machine does not run its own physics simulator, but instead relies on the single copy of the physics simulator on the original Driver Machine (as does the God's Eye Machine). While this simple approach is not intended to scale well to support large numbers of additional drivers, it works for the intimate environment of our shared physical game space.

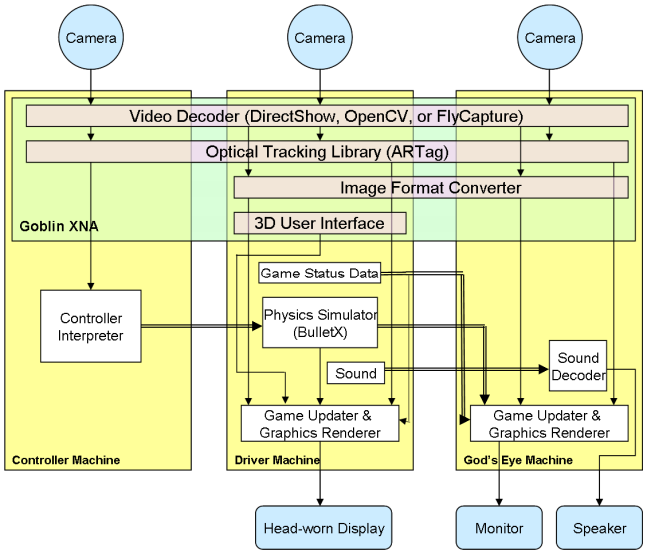

Figure 2. System architecture.

4.1 XNA and XNA Racing Game Starter Kit

XNA unites multiple game development application program interfaces into a common managed code [28] framework based on a managed version of DirectX [27]. It is designed with an extensive set of class libraries specific to game development exercises. XNA managed code has its execution overseen by a runtime-aware infrastructure that makes possible a range of development advantages and safety guarantees.

The XNA Racing Game starter kit is written in C# and contains many of the components we needed, including game logic, simple car collision detection, movement control, shaders, landscape models, shadow mapping, and audio files. We removed large portions of the game that we did not need, and modified many other parts, retaining approximately half of the code. Major changes included replacing the predefined track and landscape with the real world background image captured live from the cameras, changing the first-person view to a third-person view similar to that of real remote-controlled model racing cars, replacing keyboard and mouse input with fiducial marker-based controller input, placing landscape objects on the fiducial marker tracked ground plane, modifying the collision detection mechanism and adding the landscape objects to the physics simulation, adding networking capabilities for controller inputs and god's eye views, and modifying the basic game play.

4.2 Managed AR Gaming Platform

AR games require a range of functionality, including accurate 6DOF tracking, real-time video capture at interactive frame rates, user interaction techniques and devices, physics simulation, networking, and combined rendering of real and virtual worlds.

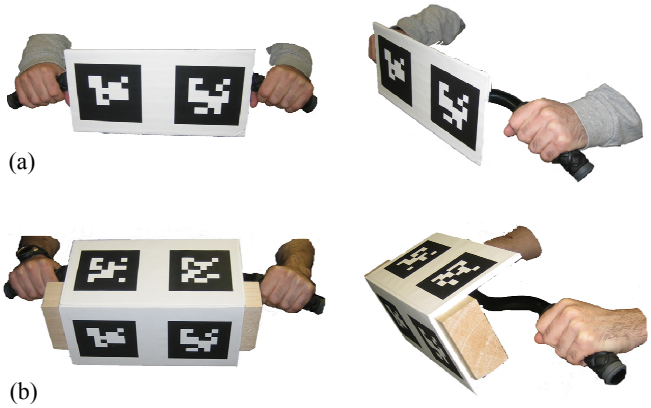

Figure 3. (a) Initial driving controller. (b) Modified driving controller.

While other researchers have developed infrastructures to support the development of AR applications, typically using OpenGL and C++ (e.g., Ohlenburg and colleagues [34]), we have been building Goblin XNA to support AR games in a managed environment. Goblin XNA is also being used to teach a course on 3D user interface design, emphasizing AR interaction techniques [6], where the use of C# makes the system easily accessible to students already familiar with Java or C++.

Goblin XNA provides interfaces to several video capture libraries (DirectShow [26], OpenCV [17], and FlyCapture [39]) to handle a wide range of cameras. Images captured using these libraries have different image formats, and Goblin XNA automatically converts these formats to a unified format that is used to pass images to vision-based tracking systems (e.g., ARTag) and to render the physical world. Goblin XNA also supports a set of 2D and 3D interaction techniques, and facilities for incorporating external 2D GUIs through texture-mapping.

4.3 Physics Engine

The original XNA Racing Game had only a very simple collision detection mechanism between the car and the guardrails on the predefined tracks. To make possible a better physical simulation, including rigid body dynamics and efficient 3D collision detection, we use BulletX, a fully managed 3D physics engine that is being ported from the original Bullet library [7]. We use BulletX to support collision detection between the car and other objects, and the effects of collisions. We also utilize BulletX's collision detection capability to handle interaction with 3D widgets. Unfortunately, BulletX is not completely ported at this time, and does not yet provide the support for triangle mesh collision detection and vehicle physics that is part of Bullet. Thus, all collision detection is performed using bounding box approximations. Because vehicle physics is not available, we enhanced the race car dynamics by imposing ground and air friction.

4.4 Networking

Since XNA does not currently include any network communication functionality, we integrated a reliable UDP network library. We chose the Lidgren open source C# library [22] to provide a simple way of transferring data between machines with a small amount of overhead. Lidgren supports flexible message delivery methods, message fragmentation, message coalescing, and connection statistics.

4.5 Tracking

We require accurate 6DOF tracking to establish a ground plane on which the car, environment objects, and waypoints interact, and to track the controller. Portability is also an important concern for our game in order to be able to play it anywhere.

Many tracking technologies that have been applied to AR [2] would not be suitable for our application. For example, GPS is a highly portable position-tracking technology, but it does not work well indoors, is too coarse to support registration with nearby objects, and must be combined with other technologies to support orientation. A number of systems track position and orientation by attaching active components, such as diodes and electromagnetic sensors, to tracked objects, but we wanted to avoid this to limit size, weight, and cost. Therefore, we decided to use ARTag [13], a computer-vision–based fiducial marker tracking package that is highly portable and can provide millimeter-level accuracy. We chose ARTag over ARToolKit [18], which had been developed earlier and used more extensively, because, in our experience, ARTag performs better in a wider range of lighting conditions, has a lower rate of false marker detection, and is more robust to partial marker occlusions [12].

Figure 1 shows an array of fiducial markers used for tracking our ground plane. We chose to combine markers of different size to achieve accurate tracking when the camera is close to the ground plane, as well as when it is further away.

5. USER INTERFACE

One of the main benefits of AR gaming is the ability to manipulate 3D objects in the environment directly to make possible innovative and intuitive user interface [20]. This capability can greatly enhance the 3D game experience. In this section, we describe the user interface for the driver and the non-driver players.

5.1 Driver

We constructed a passive tangible interaction device to function as the game controller in the spirit, for example, of Fiala's magic mirror device [14]. The initial version of our driving controller (Figure 3a) consisted of a fiducial marker array mounted on a piece of foam core board, which is rigidly fixed to a pair of bicycle handlebars. (We chose the mixed metaphor of a bicycle controller for a racing car because we found the handlebars to make a more comfortable unattached controller than a steering wheel.) Turning is accomplished by rotating the controller roughly parallel to the ground plane, while the car is accelerated or decelerated by tilting the controller forward or backwards.

Initial observations and informal user feedback made it clear that the controller did not have a sufficiently large pitch angle range relative to the Controller Machine camera. Therefore, we added another fiducial marker array to the controller, perpendicular to the first array (Figure 3b), significantly improving the range and accuracy of tracked pitch. Both of these marker arrays need to be visible from the controller camera at the start of the game to register their initial pitch angles. Optical marker tracking made it possible for these changes to be accomplished quickly and inexpensively.

Figure 4. Moving a waypoint attached to a marker array.

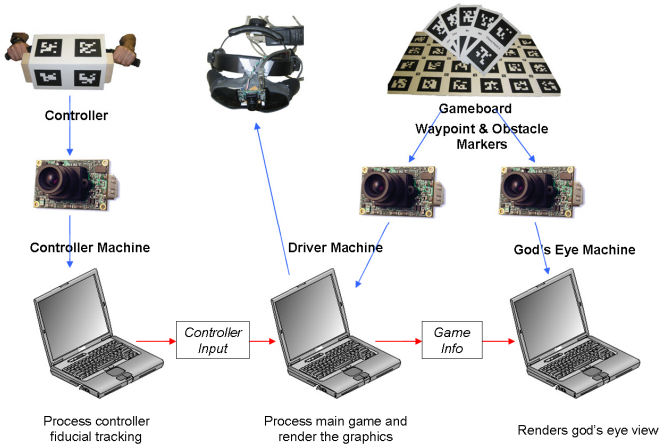

Figure 5. Hardware configuration.

5.2 Non-Driver Players

Unlike traditional console games, in which a limited number of players have access to controllers through which they can participate in the game, marker tracking makes it much easier to support additional players who can manipulate game objects. To take advantage of this, we attached environment objects to additional marker arrays. This enables observers to become non-driver players, who can work alone or together to assist or hinder the driver by dynamically modifying the environment during game play. These additional players view the game and the effect they have on it through the God's Eye camera.

Thus, there are four types of environment objects in the scene: pre-positioned dynamic objects, player-manipulated waypoints, static waypoints, and player-manipulated static objects. *Pre-positioned dynamic objects* are associated with the game board and act as obstacles in the scene. They can physically interact with the racing car and other virtual objects (except for the waypoints) and move in accordance with the physics simulation. For example, the road sign shown in Figure 1(b) is a pre-positioned dynamic object, which is knocked over as the car bumps onto it.

Player-manipulated waypoints, such as the one shown in Figure 4, are each associated with a fiducial array mounted on a separate card and act as pass points that must be visited to complete a lap. Their positions can be modified at any time during game play, and they do not physically interact with any other virtual objects including the racing car.

Static waypoints, such as the one shown in Figure 1(a), are each associated with a pre-set position on the gameboard, and can be used together with or as replacements for player-manipulated waypoints. For example, a complete set of static waypoints may be used when the driver wishes to play against a pre-set course.

Finally, *player-manipulated static objects* are associated with individual fiducial arrays, like the player-manipulated waypoints shown in Figure 4, but act as obstacles. Their positions can be changed like the player-manipulated waypoints, and they can physically interact with the racing car and other virtual objects in the scene. However, they do not move in response to collision events, and thus remain rigidly fixed to their arrays.

6. HARDWARE PLATFORM

A typical hardware configuration (Figure 5) that we have used for this game consists of three laptops (Controller, Driver, and God's Eye Machines), a video see-through head-worn display, three cameras, the driving controller (Figure 3), the ground plane marker array, and a set of smaller marker arrays.

We use a Sony LDI-D100B 800×600 resolution, color, head-worn display, on which we mounted a Point Grey FireFly MV camera to capture the driver's view for 6DOF fiducial tracking and biocular (i.e., two-eye, non-stereo) video see-through AR. Both are connected to the Driver Machine, which is a Sony VAIO VGN-SZ480 computer running Windows Vista with NVIDIA GeForce Go 7400 graphics card, 2GB RAM, and a 2.0 GHz Intel Core 2 Duo CPU, mounted on a small, lightweight backpack frame. The loaded backpack weighs well under 10 lbs. For our Controller Machine, another Point Grey FireFly MV camera is connected to an Apple Macbook running Windows XP with 1GB RAM and a 2GHz Intel Core Duo CPU. The God's Eye Machine uses a Point Grey DragonFly 2 camera connected to a Dell XPS M1710 computer running Windows XP with NVIDIA GeForce Go 7950 GTX, 2GB RAM, and a 2.33 GHz Intel Core 2 Duo CPU. All machines are interconnected wirelessly through an IEEE 802.11g network.

ARTag requires a relatively high-resolution image and high frame rate to achieve accurate dynamic position and orientation tracking. It is possible to use low-end capture devices such as web cameras, but we have found that accuracy suffers, resulting in jitter in both static and dynamic scenes, and complete loss of tracking in fast moving scenes. Therefore, we chose FireWire cameras that provide 640×480 resolution at frame rates of up to 60 Hz and use better lenses, providing significantly better image quality than commodity web cameras. While the pixel resolution is the same or less than that of high-end web-cameras, the captured image quality is significantly better. We use color cameras for the driver and god's eye cameras, since they provide a crucial part of the displayed imagery. However, we use a grayscale camera to track the controller, whose image is not displayed. In comparison to a single-chip color camera of the same nominal pixel resolution, not having an array of dyed pixels results in greater effective spatial resolution, decreasing the width in pixels at which markers can be detected [12].

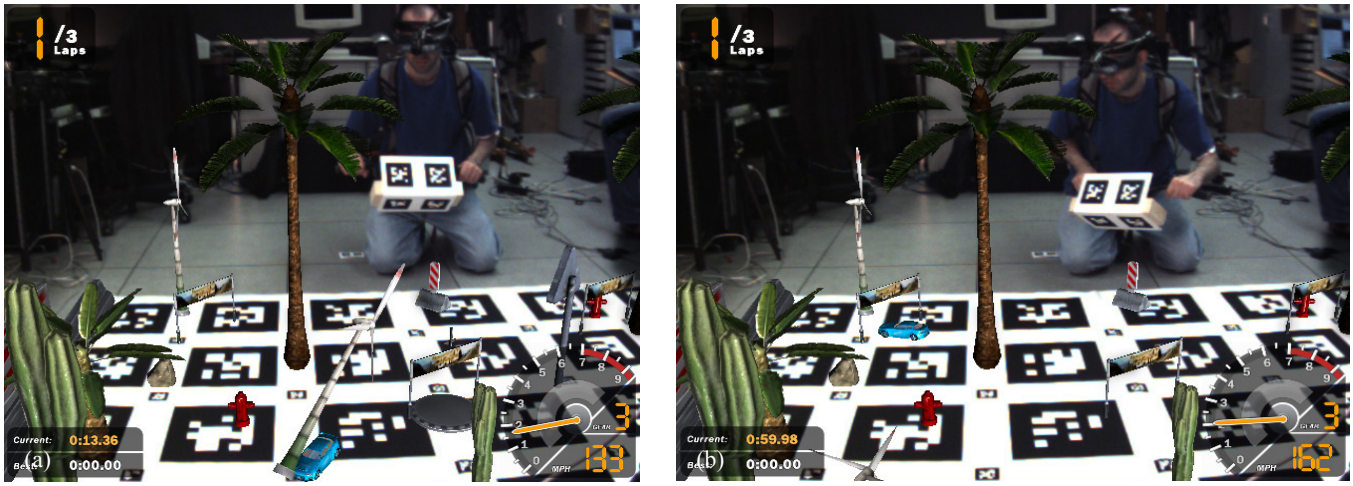

Figure 6. Game play. (a) Car collides with a windmill and knocks it down. (b) Car passes a waypoint.

We have used gameboards spanning a wide range of sizes. For public demonstrations and the images in this paper, an 80" wide × 60" deep gameboard array is mounted on a set of foam core boards to maintain a relatively flat surface across the array, which is lit with a pair of Lowell RIFA-LITE LC44 250W Softbox lights. Our more portable gameboards have been as small as 10-2/3" wide × 8" deep. Smaller gameboards are useful for debugging purposes. Note that no changes to the software are needed to account for the actual size of the physical gameboard, since all virtual imagery scales with the gameboard fiducials. (On the other hand, the controller orientation alone affects game play, so a large-scale controller can be used with a small-scale gameboard.)

7. GAME PLAY

The driver first puts on the head-worn display and wireless backpack, and holds the driving controller. Next, she is prompted to select whether to start the game, change display options, view credits, or obtain help through a 3D user interface whose 3D buttons hover directly in front of her. Selection is accomplished by pointing at the appropriate button with a fiducial array for three seconds. Once the driver chooses to start the game, a count down is played, and the car emerges near the center of the gameboard.

The driver's goal is to finish three laps as quickly as possible, as in the original game. The driver is expected to traverse waypoints in a specific sequence in order to complete a lap. The next waypoint in the sequence is indicated by an animated arrow hovering above it, and a large round marker below it. As shown in Figure 6, the difficulty of the game for the driver depends on the number, size, and layout of the objects in the scene, compounded by how the additional players place the player-manipulated waypoints and obstacles (if any) on the gameboard. The car can fall off the gameboard if it goes over the boundary of the printed array when the array is not placed on a larger surface, such as the floor. When this occurs, the car reemerges near the center of the gameboard. In order to enhance realism, the ground plane is used for occlusion, so anything that falls off from the ground plane will be partially or fully occluded by it.

8. FEEDBACK AND OBSERVATIONS

We received substantial informal feedback during a demo at the 2007 Microsoft Research Faculty Summit [31] of an early version of the system, both directly from player comments and indirectly through observation of player performance and behavior. Before each driver started the game, we gave them brief verbal instructions on how to play. Most of the approximately twenty drivers who tried the system clearly enjoyed the AR experience of the game play. However, more than half of them remarked about playability issues due to an overly difficult course layout or lack of responsiveness of the driving controller. Responsiveness issues were in part caused by delays in the early version of the networking code used in the demo. In addition, the controller software at that time did not support the ability to reverse the car, so once the player was stuck on an obstacle, they could not drive their way out of it. Adding the ability to put the car in reverse has improved game play significantly.

We observed that some players had difficulty in steering the car from the fixed exocentric viewpoint typical of a remote-control car system, rather than a viewpoint that moved with the car (whether inside or outside) that is typical of a real car or classical racing game. Several players suggested that we supplement the AR views corresponding to the driver's camera and god's eye camera with a fully synthesized VR view inside or attached to the car. While we intend to provide this additional view for use at the driver's discretion, we are also looking at control interfaces from existing remote control car systems.

We also observed that, in general, it can be difficult for a player to understand the extent of the playing field in AR. In contrast, a conventional computer display provides a clearly defined boundary between the virtual and the real. To address this potential confusion, we optionally add in visible virtual walls at the edges of the gameboard that prevent players from driving off the end of the world, examples of which are shown at the left edges of Figures 4 and 6(a–b). However, when the gameboard is placed on top of a larger surface, we often leave off the walls, allowing the car to drive off the gameboard (and push obstacles off it, too), which will be tracked as long as a sufficiently large portion of the gameboard is visible to the camera.

9. CONCLUSIONS AND FUTURE WORK

We have described our experiences implementing an AR racing game that explores tangible input devices using vision tracking and non-driver player interaction. Our AR infrastructure made it relatively easy to expand an existing non-AR game to become an AR game incorporating an external physics engine and networking support. The infrastructure supported video capture and compositing, 6DOF vision tracking using the ARTag library, and 3D user interaction techniques. We are continuing to develop the game, both to explore AR gaming further and to test our infrastructure as we extend it. One significant advantage of building on top of XNA is the set of powerful development tools that it leverages, which made it especially easy and fast to experimentally modify system behavior, even during demo sessions.

Rather than providing support for physics purely at the level of an individual game, we are currently extending Goblin XNA to include an interface that will make it easier to incorporate external physics engines. While the current version of XNA does not support networking, the next version will, so we will need to determine what additional networking support, if any, to include.

10. ACKNOWLEDGMENTS

We thank Wei Tang for helping us develop the AR Racing Game; Mike Sorvillo for helping us incorporate ARTag; and Steve Henderson and Anette Kapri for discussion and feedback. This research is funded by generous gifts from Microsoft Research and NVIDIA. The original XNA Racing Game was developed by Benjamin Nitschke.

11. REFERENCES

[1] Avery, B., Piekarski, W., Warren, J., and Thomas, B. H., "Evaluation of user satisfaction and learnability for outdoor augmented reality gaming," in *Proc. 7th Australasian User interface conference*, Hobart, Australia, vol.50, 17-24, 2006.

[2] Azuma, R. T., "A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments,* vol.6, no.4, 355-385, August 1997.

[3] Barakonyi, I., Weilguny, M., Psik, T., and Schmalstieg, D., "MonkeyBridge: autonomous agents in augmented reality games," in *Proc. ACM SIGCHI International Conference on Advances in computer entertainment technology*, Valencia, Spain, 172-175, 2005.

[4] Cheok, A. D., Sreekumar, A., Lei, C., and Thang, L. N., "Capture the Flag: Mixed-Reality Social Gaming with Smart Phones," vol.5, no.2, 62-69, 2006.

[5] Cheok, A. D., Yang, X., Ying, Z. Z., Billinghurst, M., and Kato, H., "Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing," *Personal and Ubiquitous Computing,* vol.6, no.5-6, 430-442, 2002.

[6] COMS W4172, 3D User Interface Design, http://www.cs.columbia.edu/graphics/courses/csw4172

[7] ContinuousPhysics, Bullet, http://www.continuousphysics.com/Bullet/

[8] Cooper, N., Keatley, A., Dahlquist, M., Mann, S., Slay, H., Zucco, J., Smith, R., and Thomas, B. H., "Augmented Reality Chinese Checkers," in *Proc. 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, Singapore, 117-126, 2004.

[9] Crabtree, A., Benford, S., Rodden, T., Greenhalgh, C., Flintham, M., Anastasi, R., Drozd, A., Adams, M., Row-Farr, J., Tandavanitj, N., and Steed, A., "Orchestrating a mixed reality game 'on the ground'," in *Proc. SIGCHI conference on Human factors in computing systems*, Vienna, Austria, 391-398, 2004.

[10] exDream, XNA Racing Game, http://www.xnaracinggame.com/

[11] Feiner, S., "Augmented Reality: A New Way of Seeing," *Scientific American,* vol.286, no.4, 34-41, April 2002.

[12] Fiala, M., ARTag Rev1: Marker Detection, http://www.artag.net/rev1.html

[13] Fiala, M., "ARTag, a fiducial marker system using digital techniques," *Proc. CVPR 2005,* 590-596, 2005.

[14] Fiala, M., "Magic Mirror System with Hand-held and Wearable Augmentations," in *Virtual Reality Conference '07*, 251-254, 2007

[15] Hakkarainen, M. and Woodward, C., "SymBall: camera driven table tennis for mobile phones," in *Proc. ACM SIGCHI International Conference on Advances in computer entertainment technology*, Valencia, Spain, 391-392, 2005.

[16] Henrysson, A., Billinghurst, M., and Ollila, M., "AR tennis," in *ACM SIGGRAPH Emerging technologies*, Boston, Massachusetts, 1, 2006.

[17] Intel, "OpenCV: Open Source Computer Vision Library,"

[18] Kato, H. and Billinghurst, M., "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *2nd IEEE and ACM International Workshop on Augmented Reality '99*, San Francisco, CA, 85-94, 1999

[19] Knoerlein, B., Gbor, S., and Harders, M., "Visuo-haptic collaborative augmented reality ping-pong," in *Proc. international conference on Advances in computer entertainment technology*, Salzburg, Austria, 91-94, 2007.

[20] Lee, G. A., Nelles, C., Billinghurst, M., and Kim, G. J., "Immersive Authoring of Tangible Augmented Reality Applications," in *3rd IEEE and ACM International Symposium on Mixed and Augmented Reality*, vol.00, 172-181, 2004.

[21] Lee, W., Lee, W., and Lee, J., "TARBoard: Tangible Augmented Reality System for Table-top Game Environment," in *PerGames*, Munich, Germany, 2005. http://www.ipsi.fraunhofer.de/ambiente/pergames2005/

[22] Lidgren, Lidgren Library Network, http://code.google.com/p/lidgren-library-network/

[23] Magerkurth, C., Memisoglu, M., Engelke, T., and Streitz, N., "Towards the next generation of tabletop gaming experiences," in *Proc. Graphics Interface 2004*, London, Ontario, Canada, 73-80, 2004.

[24] Magerkurth, C., Stenzel, R., Streitz, N., and Neuhold, E., "A Multimodal Interaction Framework for Pervasive Game Applications," in *Artificial Intelligence in Mobile System*, Seattle, USA, 1-8, 2003

[25] Matysczok, C., Radkowski, R., and Berssenbruegge, J., "ARbowling: immersive and realistic game play in real environments using augmented reality," in *Proc. ACM SIGCHI International Conference on Advances in computer entertainment technology*, Singapore, 269-276, 2004.

[26] Microsoft, DirectShow, http://msdn2.microsoft.com/en-us/library/ms783323.aspx

[27] Microsoft, DirectX, http://msdn2.microsoft.com/en-us/xna/aa937781.aspx

[28] Microsoft, Managed Code, http://msdn2.microsoft.com/en-us/library/bb278146.aspx
[29] Microsoft, XNA, http://www.xna.com/
[30] Microsoft, XNA Starter Kits, http://creators.xna.com/Education/StarterKits.aspx
[31] Microsoft Research Faculty Summit, http://research.microsoft.com/workshops/FS2007/
[32] Nintendo, Wii, http://wii.nintendo.com/
[33] Oda, O. and Feiner, S., Goblin XNA, http://www1.cs.columbia.edu/graphics/projects/goblin/goblinXNA.htm
[34] Ohlenburg, J., Herbst, I., Lindt, I., Frhlich, T., and Broll, W., "The MORGAN framework: enabling dynamic multi-user AR and VR projects," in *Proc. ACM symposium on Virtual reality software and technology*, Hong Kong, 166-169, 2004.
[35] Ohshima, T., Sato, K., Yamamoto, H., and Tamura, H., "RV-Border Guards: A Multi-player Mixed Reality Entertainment," in *Transactions of the Virtual Reality Society of Japan*, vol.4, 699-705, 1999.
[36] Ohshima, T., Satoh, K., Yamamoto, H., and Tamura, H., "AR2 Hockey," in *ACM SIGGRAPH 98 Conference abstracts and applications*, Orlando, Florida, United States, 110, 1998.
[37] Paelke, V., Reimann, C., and Stichling, D., "Foot-based mobile interaction with games," in *Proc. ACM SIGCHI International Conference on Advances in computer entertainment technology*, Singapore, 321-324, 2004.
[38] Piekarski, W. and Thomas, B., "ARQuake: the outdoor augmented reality gaming system," *Communications of the ACM* vol.45, no.1, 36-38, 2002.
[39] PointGreyResearch, FlyCapture, http://www.ptgrey.com/products/pgrflycapture/index.asp
[40] Schmalstieg, D., "Augmented Reality Techniques in Games," in *Proc. 4th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 176-177, 2005.
[41] Thomas, B., Close, B., Donoghue, J., Squires, J., Bondi, P. d., Morris, M., and Piekarski, W., "ARQuake: An Outdoor/Indoor Augmented Reality First Person Application," in *Proc. 4th IEEE International Symposium on Wearable Computers*, 139, 2000.
[42] Wagner, D., Pintaric, T., Ledermann, F., and Schmalstieg, D., "Towards Massively Multi-User Augmented Reality on Handheld Devices," in *Proc. 3rd International Conference on Pervasive Computing*, Munich, Germany 208-219, 2005.
[43] Wilson, A., Depth-sensor based XNA racing game, http://channel9.msdn.com/ShowPost.aspx?PostID=290587, mms://wm.microsoft.com/ms/evnet/RB_TechFest_4_s_ch9.wmv
[44] XnaDev, BulletX, http://www.codeplex.com/xnadevru/Wiki/View.aspx?title=Managed%20Bullet%20Physics%20Library