Outdoor Augmented Reality n-th Game Editing Suite

by

Benjamin Arthur Schmeckpeper

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007
[June 2007]

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2007

Certified by 5/25/07
Eric Klopfer
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Outdoor Augmented Reality n-th Game Editing Suite

by

Benjamin Arthur Schmeckpeper

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2007, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

OARnGES was developed to be a comprehensive Augmented Reality game development tool. Augmented Reality provides players a view of the real world augmented with virtual characters and information. Augmented Reality has proved to be a valuable educational tool, but bridging the gap between experimental game engines and in-class deployment has proved difficult. OARnGES attempts to close that gap by removing the technical barriers of game development. The Augmented Reality engine framework is still in flux and OARnGES provides a stabilizing layer between the developer and the platform.

Thesis Supervisor: Eric Klopfer
Title: Associate Professor

Acknowledgments

I would like to thank Professor Klopfer for hiring me as an RA despite my lack of hands-on experience.

Thanks to Judy Perry for her tremendous project management abilities and for keeping this project on track and moving forward.

And most of all, thanks to my fiancée Kathryn for letting me take two years to finish a degree that was supposed to take me one.

Contents

List of Figures

Chapter 1

Introduction

1.1 Explanation of Augmented Reality

Augmented Reality (AR) immerses players in the real, physical world and provides auxiliary data, such as virtual characters, which augment the player's view of the world. By moving through the physical world to interact with this virtual world players are drawn more deeply into AR games. The AR platform allows game designers to present a wide variety of data to players and also customize the ways in which players access that data. Players can be presented with a lot of information at once, forcing them to prioritize their information gathering, or players can be presented with limited information, requiring in-game critical thinking to unlock or discover further information. The Augmented Reality paradigm provides instructors with a captive audience and an incredibly flexible platform.

Implementations of Outdoor Augmented Reality games involving middle and high school students have been "effective at fostering collaboration" as students probed each other for new information [3]. Students have reacted positively to the Outdoor AR platform, excited that it gave them "a chance to go out of school to learn math and English" [2]. The features of the platform observed to captivate the most students were the game's narrative, the ability to adopt a role and solving codes to access critical game content [2]. The extent of students' collaboration and engagement with the platform varies with the style of game and extent of the narrative. Games which

include multiple roles and present each role with customized content have "enhanced the potential for larger scale collaborative learning" [3].

1.2 Goals for OARnGES

The Outdoor Augmented Reality n-th Game Editing Suite (OARnGES) attempts to simplify the creation of Augmented Reality games. Harnessing the full power of the AR platform can be a long and technical process, which may seem daunting to teachers otherwise willing to use AR games in their classrooms. The utility of the AR platform is dependent upon the quality of games developed for the platform and developing game files in a text editor is not a viable option. Even a basic AR game requires a fairly complex Xml document to completely specify all the necessary settings for the AR Engine to function. A game development tool is needed to ease the development of engaging, instructive AR games [4].

A useful tool will hide the majority of the AR engine's technical details from the game designer, and also providd multiple views of game data and assist in game development, beyond simple content creation. OARnGES is more than an Xml authoring tool, its functionality enables game designers to develop richer OAR games. As the size and scope of a game grows it becomes increasingly difficult to account for all game content and ensure players are presented with a consistent narrative. OARnGES is offered as a simple to use tool which gives game developers multiple, meaningful views of their game data.

1.3 Design and Implementation

The Outdoor AR platform is being developed to enable educational research and OARnGES has been strongly influenced by the wants and needs of the educational researches utilizing it. Teams at both Harvard University and the University of Wisconsin-Madison have developed multiple augmented reality games using successive versions of OARnGES and providing constant feedback. This situation has presented

unique challenges, as the research needs of the Harvard and UW teams dictated the evolution of OARnGES.

OARnGES has been designed and implemented with a strict separation between the front-end GUI and the back-end data structures. This decoupling has allowed for rapid feature development as well as isolating the impact of errors and reducing the time required to fix bugs. This design strategy has proved useful in OARnGES development environment, enabling a rapid release schedule and enough flexibility to respond to both feature requests and bug reports from the end users relying upon OARnGES as a research tool.

The user is presented with a view of game data which closely resembles the end result displayed on a player's PDA. However, OARnGES was designed to allow for the future development of alternate views to enable alternate game development workflows. While the current implementation of OARnGES has focused on perfecting just a single view, ideas for alternate views are presented in the future work section of this thesis.

1.4 Results

Through a series of releases spanning approximately 14 months OARnGES has matured into a stable product and an important tool for Outdoor AR game development. Multiple OAR games have been developed using OARnGES and approximately 200 students have played those games. OARnGES has met many of its stated goals, decoupling game data from user interface and has been adapted to the requests of its core users, game developers. However, the difficulty of developing a useful and intuitive UI was underestimated and OARnGES currently lacks the multiple development views which the decoupling was intended to allow. Ideas for future views, along with other development areas, are presented in the Future Work section of this thesis.

A more detailed description of Augmented Reality is presented in Chapter 2 and the initial goals of OARnGES are described in Chapter 3. Chapter 4 discusses the design of OARnGES and Chapter 5 details how those designs were implemented.

Chapter 6 describes the results of OARnGES development and Chapter 7 outlines possible future development of OARnGES.

Chapter 2

Outdoor Augmented Reality Overview

The Teacher Education Program at MIT has developed an augmented reality game designed to teach math and science literacy skills to middle school students [1]. This game is played on a Pocket PC (PPC) and uses GPS technology to correlate your real world location to your location in the game, overlaying your position in the real world on top of a map displayed on the PPC. The map also displays objects and people who exist in the virtual world and with whom the students can interact. When playing the game, the student moves around in the real world to interact with virtual Items and virtual characters (referred to as non-playing character, or NPCs.)

The Outdoor AR engine, which runs on a student's PPC, allows games to be played anywhere around the globe when given correct latitude and longitude. Initial work on OAR began with the development of the game Environmental Detectives, which is played on the MIT campus. The game has been updated with successive changes to the OAR engine, renamed the Summerbridge Game and remains the game of choice to demonstrate Outdoor Augmented Reality at MIT. Players are assigned one of three roles, Engineer, Environmental Scientist or Reporter, and dispatched to discover the reason two MIT students have recently become ill.

Players are presented with a bird's eye view of the MIT campus with NPCs and Items scattered throughout. The focus of the Summerbridge Game is information

gathering and players are able to glean different types of information from different characters. For instance, visiting the NPC Andrew Davis, who is walking his dog along the river, triggers an old drain pipe up river. Inspecting this new Item reveals that foul smelling water is trickling into the Charles River. The Engineer is able to inspect a variety of sampling stations along the river but unable to understand their readings until visiting the NPC Eileen Chang, who works for the Charles River Watershed Association. Players spend their time canvassing the campus conducting interviews and gathering evidence, then reconvene at the end of the game to compare data and formulate theories.

Because of the limitations of GPS, OAR games are mainly conducted out of doors, but through the use of Gates parts of the game can be played indoors. When students visit a gate, which is typically placed near a building on the map, they are instructed to enter the building and find a clue or other information posted somewhere inside that building. After finding the posted information, the student is asked to enter a code which then unlocks an NPC or Item which is only available through that gate. For example, the designer of the game could wish to have the students interview an NPC in that character's office. To accomplish this, the game designer would place a Gate on the map where the NPC's office building sits and classify the NPC as "gate-accessible." On the day of the game, the designer may post a puzzle or a math problem on the office door which the students must first solve before interviewing the NPC. The students then enter the solution to the problem into their PPC, and if they entered the correct solution they are then presented with the interview for the NPC. In the Summerbridge Game players wishing to interview one of the sick students must enter MIT's Student Center in order to speak with him. To ensure players physically enter the building, a sheet of paper is posted inside with the code required to interview Carl Samson printed upon it. Once the player enters that code they are able to view a video of Carl explaining why he believes he got sick.

The game allows students to perform a variety of virtual experiments, such as sampling ground water or using a Geiger counter to detect radiation. Using the results of these experiments, along with information gained from Items and NPCs, the

students can formulate hypotheses to answer some larger question, such as who caused a chemical spill which contaminated the ground water around MIT. The game designer creates a variety of spills throughout the virtual world, giving the students something to sample. Before placing spills on the map, the game designer must first create at least one gradient. A gradient can be thought of as a type of spill, and individual spills of the same gradient share common characteristics, such as the sampling method for the spill and the accuracy of that sampling method. Once gradients are created, spills can then be added, by specifying two foci and the width of the spill. Spills are not displayed on the map as the students play the game, but through sampling at different locations the students can begin to form an idea about the intensity and location of the spill(s). In the Summerbridge Game the Environmental Scientist carries both a Geiger Counter to measure radition, particularly useful at MIT's nuclear reactor, and a soil sample kit to test for contamination in the ground.

Beyond merely placing objects on the game map, there are many features of the virtual world which the game designer needs to configure. Some of these features define small aspects of the game like specifying the image file to be used as the map image and the GPS coordinates of the corners of the map, but other features dictate larger aspects of game play. One of these larger features is the creation of a variety of roles for students to assume at the start of the game. The use of multiple roles allows, for instance, a group of students to play the game as scientists while another group plays the game as reporters. The addition of roles adds flexibility to the game, because all game objects can be configured as visible or invisible to individual roles. Furthermore, the information associated with NPCs and Items can be customized on a per-role basis. This allows for more realistic game play, as students receive information specifically customized for the role they are pretending to embody. Also, details about the length of the game, both real and imaginary, can be set by the game designer. The game can be divided into any number of time switches, which partition the real time length of the game. Information associated with NPCs and Items can also be customized by time switch, which allows the state of the virtual world to change over time.

Games typically occur outside and cover a large area, which makes centralization of the game very impractical. Items and NPCs are able to trigger other game objects, which results in previously unseen game objects becoming visible on a student's map after the student views the information related to an NPC or an Item. For instance, if an NPC recommends that the student interview a colleague, then the use of triggers would allow the NPC's colleague will appear on the student's map after the interview. However, this situation introduces non-determinism into the game as changes to one student's view of the world are not propagated amongst the rest of the students and by the end of the game each student arrives at a slightly different view of the world than their peers. While there is no central state maintained during the course of the game, students are able to share information using the IR transmitter on their PPC. Interviews, documents and other data collected over the course of the game can be beamed between PPCs, which allows students to learn about different areas of the virtual world even if they are not able to cover the whole map during each time switch.

Chapter 3

OARnGES Goals

The overarching aim of OARnGES is to make the process of designing games accessible to even the least technical users. As development of the Outdoor AR (OAR) platform continues, its feature set remains in flux and the platform undergoes frequent changes. OARnGES enables game designers to ignore this volatility and simply develop games. Additionally, OARnGES aims to alleviate Xml authoring difficulties, enable varied development styles and provide tools to aid game development, while handling the challenges of the platform's flexibility.

An OAR game is defined by an Xml document following a very specific format. Any variance from this format, or any malformed Xml, will render the entire game unplayable. Creating valid Xml by hand is a tedious process, as all tags must match and attention must be paid to proper casing of tags. OARnGES is able to automate the creation of these documents, and as such manages to abstract away Xml authoring details and difficulties. OARnGES provides much quicker and easier Xml authoring and the development of OARnGES in conjunction with the OAR platform allows OARnGES to readily incorporate the Xml definitions of new game features and remain consistent with new requirements.

The process of designing an OAR game is very nebulous and personal. As no standardized process for game development exists, OARnGES should enable multiple development strategies. OARnGES was designed to accommodate game development, not to enforce a specific game development workflow. To this end, OARnGES

decouples the game data from it's graphical representation in order to allow rapid changes to the GUI as well as enabling the development of alternate views of game data. The development of OARnGES began early in the life of the OAR platform and OARnGES needed to be flexible enough to respond to both changes in the platform and new strategies for developing OAR games. As educational researchers continue to investigate the OAR platform they will certainly develop new methodologies for designing games which will place new demands upon OARnGES. Separation of the internal representation of game data from the view of this game data OARnGES presents to the user allows changes to OARnGES' UI without the possibility of disturbing its data model. This not only allows simple tweaks of OARnGES' UI, but also the development of alternate views of game data.

Finally, OARnGES aims to aid game developers with tools that go beyond simple content creation. These tools will aid collaborative development, speed the deployment of games to PDAs and verify all references to external content. Collaborative development of OAR games often reaches a snag when attempting to work on a shared game document with external content, such as images or video, because of differences in the absolute pathnames on different machines. OARnGES recognizes the benefits of collaboration and will provide built in tools to quickly convert filenames and overcome difficulties in sharing game documents. Although OARnGES attempts to faithfully recreate the game play experience through it's user interface, some aspects of the game simply need to be tested on a PDA. Developers need to ensure that colors are drawn correctly, images fit the smaller screen size, etc. Aware of this need, OARnGES attempts to speed testing and also assist in content organization with a few included deployment tools. A development environment like OARnGES will allow users to copy all necessary game files to the PDA with a single click, giving the game developer a no-hassle deployment option. If the developer does not wish to immediately copy game files to the PDA, OARnGES will create a locally stored "package" containing all files necessary for future deployment. Furthermore, since references to all external content are stored in the current game, OARnGES is able to quickly verify those references and report to the developer whether any game files

are missing. This scan occurs automatically before packaging and deployment and is also available to the user on-demand. Perhaps the most useful tool allows the user to quickly and easily obtain both an aerial map of the game's location and exact GPS coordinates for the game space. This pre-existing tool has been developed independently of OARnGES, but will seamlessly integrated into newer versions of the suite.

The very things which make the Outdoor AR platform an appealing educational tool, it's malleability and customizability, present a number of obstacles which OARnGES must address in order to be an effective game design tool. OARnGES is forced to present the user with the wide range of options available to them, but at the same time should not overwhelm the user with myriad choices. Internally, OARnGES needs to store these customizations and provide ready access to them. Many customizations are allowed on a per-role and per-time switch basis, resulting in OARnGES needing to track role-times-time switches pieces of information for each customizability. Finally, because the platform is still under development OARnGES needs to be flexible enough to accommodate new features and also changes to existing features, i.e. settings or data once thought to be fixed becomes customizable.

Chapter 4

OARnGES Design Strategies

4.1 Overall Design

OARnGES essentially consists of two parts, the back end data structures which store game data and the front end user interface which allows the game designer to manipulate game data. Throughout the design process these two pieces of OARnGES were willfully and intentionally kept separate. This decoupling allows tuning of the user interface design without disturbing the underlying data structures, and the internal workings of the data structures can be easily modified without affecting the user interface. Future work on OARnGES can develop completely new user interfaces which rely entirely upon OARnGES' existing back end. The user will be able to seamlessly toggle between alternate interfaces built upon the same data structures and receive different views of the same game data.

4.2 Back End Design

OARnGES has been designed using an object-oriented approach, to encapsulate data and achieve the desired decoupling between game data and user interface. All the common features of game elements which players interact with (NPCs, Items, Gates, etc.) appear in a common `GameObject` class. `GameObjects` are stored, along with other game data, in a central `Game` object. The `Game` object provides access to stored

data and contains most of the general game logic as well.

4.2.1 Classes

Each type of game element extends the `GameObject` base class to incorporate their element-specific data. This allows the game logic to operate on collections of objects using their shared base type and only cast to a specific object type when necessary. This also eases development by centralizing common data and logic so that changes need only occur in one location. When the OAR platform expands and new game object types are invented, their addition to OARnGES is simplified by this common base class.

Other ancillary classes exist which encapsulate the remaining game data, such as the introduction content displayed at the start of the game and the different roles available to players. Finally, a few classes are needed to provide consistency across the game. For example, when a player visits an NPC they are able to view an "Interview" for that person, and when they visit an Item they are able to view "Information." Both Interviews and Item Information behave in the same way and have the same properties, so one class is used to store this data for both NPCs and Items. It is left to OARnGES' user interface to clarify whether a class represents an Interview or Item Information.

4.2.2 Storage

The most essential function of OARnGES is the storage of all game data represented by the above classes, and OARnGES' back end makes this data available via a clean and simple interface. The power of the OAR platform is it's customizability; nearly all aspects of the game, especially it's content, can be defined on a per-role and per-time switch basis. While any one game may only employ a small portion of these options, OARnGES needs to accommodate their full utilization. This customizability has led to the development of dually-indexed data stores. These are essentially two-dimensional arrays, except they are indexed by an integer (the time switch) on one

dimension and an object (the role) on the other dimension.

4.3 UI Design

OARnGES user interface has been designed to continually provide the game developer with a map based view of the game they're editing. OARnGES uses the game map as its background and allows the developer to position game objects directly on the map. OARnGES draws all game objects on the map, allowing the developer to quickly judge their relative positioning.

4.3.1 Map Based Design

The map centric UI of OARnGES has been designed to echo the appearance of the game on the handheld. Many aspects of the UI follow a "What You See Is What You Get" (WYSIWG) strategy. Game objects are drawn on the map using the same shape and color as they will when playing the game on a PDA. OARnGES allows the user to see how the game will appear to different roles at different times by filtering which objects are drawn on the map. OARnGES' WYSIWYG feature aids game design by providing developers with a representative view of the game world. Constantly updated feedback helps users envision their game throughout the development process and fosters an interative design process. The form factor of Windows Mobile devices plays a vital role in the effectiveness of game content and OARnGES' WYSIWYG interface strives to reflect the device's constraints.

The map view also speeds the process of porting an existing OAR game to a new location. After loading a new map, all the game designer needs to do is relocate game objects. The existing game objects preserve all their content, they only need to be placed in accessible locations on the new map.

4.3.2 Access To Game Data

The amount of customizability of the OAR platform presents a sizable hurdle for OARnGES. There is a need to allow the user to configure everything while at the same time not overwhelm the user with options. OARnGES gives the user full power to manipulate every setting but limits the amount of information presented to the user at any given time. This means that for fields which the user can customize for each time switch and each role, OARnGES only allows the user to view a single time switch-role combination at any time. This allows OARnGES to reuse UI elements, replacing data when the user chooses a new time switch-role combination.

OARnGES also attempts to make all information readily available, requiring just a few clicks to access any particular piece of game data. A "flat" layout allows the game developer to quickly add game content and not waste time clicking through multiple menus. This strategy is not without it's problems, however. A flat design requires OARnGES to sort game information into a large number of categories and the game designer must remember where a specific piece of data is stored. OARnGES attempts to overcome this issue by intelligently sorting information (related features are stored together,) using an open layout (instead of hiding options in drop down menus,) and plenty of icons.

4.4 Overcoming Decoupling

Although the separation between the two sections of OARnGES gives us many advantages, it requires extra effort for effective communication between data structures and user interface. Furthermore, OARnGES' data structures have not been designed to support undo operations, which forces the UI to employ temporary storage variables to track changes until the user chooses to apply them.

Chapter 5

OARnGES Implementation

Both sections of OARnGES have been implemented in Microsoft's C# language, making use of the .NET runtime (version 2.0) [5]. This language choice limits OARnGES to platforms which support .NET, which is predominantly just the Windows platform. Mono, an open source effort, is attempting to spread to .NET platform to Linux, Solaris, Mac OS X and Unix, but OARnGES has not been tested on any of those systems [6]. However, the benefits of using the .NET runtime sped the implementation process and overcame the runtime's limited portability.

5.1 Back End Implementation

The back end data structures form the backbone of OARnGES; they need to manage all game content without losing or distorting it. The amount of content included in any one game is not so large as to require the implementation of any advanced data structures, which allowed all effort in implementation to focus upon correctness and ease of use.

5.1.1 GameObject Class

The `GameObject` class encapsulates all features common to objects which the player interacts with during game play (NPCs, Items, Gates, Spills, etc.) A `GameObject`

contains such properties as Name, Location, Description and path to the object's Image file. The `GameObject` class contains a static variable which serves as a count of all `GameObjects` created, and the constructor assigns each `GameObject` a unique id based off of this count. The `GameObject` class is declared to be abstract, which prevents its instantiation. `GameObject` exists simply as a template, used to consolidate shared features and code into one central location. This strategy provides much of the same functionality of an `Interface`, specifying the properties and methods of any class which inherits from it. While an `Interface` merely defines methods, abstract classes are able to fully implement those methods, which means derived classes all share common code and changes to the abstract class are automatically propagated to all subclasses.

There are multiple subclasses of `GameObject`, as each type of object in the game derives from `GameObject`. Each subclass provides its own constructor which sets type-specific data after calling the `GameObject` constructor. For some objects, such as NPCs, this is merely a formality, the constructor is only used to set a few default values. Other objects differ greatly from the `GameObject` base class and their constructors becomes more important. For instance, a `Spill` inherits from `GameObject`, but needs extra data, such as its radius and two focus points, which the `GameObject` class is unable to store and must be initialized in the `Spill` constructor.

Instantiation of `GameObjects` occurs via two flavors of constructors. `GameObject` and its subclasses can be created from their basic data, passed as distinct parameters to the constructor. This is the standard behavior for creating brand new `GameObjects`. A `GameObject` is created from the minimum necessary data: name and location; that object is later fleshed out with auxiliary content, such as description, image and visibility information. Alternatively, objects can be created from their Xml definitions, often loaded from a saved game file. This strategy creates a distributed environment for parsing an OAR game file. The only section of OARnGES which knows how to parse an NPC's Xml definition is the `NPC` object itself. The main parsing algorithm merely needs to identify the type of object each `XmlElement` defines and then dispatch that element to instantiate the necessary `GameObject`.

The `GameObject` class contains an abstract property `Type`, which all derived classes are forced to override. This strategy allows all subclasses of `GameObject` to identify their type, even when stored as a generic `GameObject` object. The `GameObject` class also contains the abstract method `ToXML`, which all derived classes again must override. When implemented, this method creates an `XmlElement` containing a full definition of the `GameObject`. The exact format of this `XmlElement` varies by object type, which prevents the consolidation of any Xml creation in the `GameObject` class. However, since each subclass of `GameObject` is required to implement this method, OARnGES is able to simply iterate over a collection of `GameObjects`, oblivious to their specific types, call `ToXML` on each object and generate full Xml definitions of all objects in the game.

5.1.2 Game Class

The core of OARnGES' back end is the `Game` class. This class stores all game objects and provides a clean interface for accessing these objects. The `Game` class enforces constraints placed upon the game by the OAR Engine, such as preventing the creation of objects with the same name. Because the `Game` object contains references to all game content, including all `GameObjects`, the enforcement of these constraints is very straightforward.

The `Game` class contains a separate Collection for each type of `GameObject`. This allows the `Game` class to easily separate NPCs from Items from Gates, etc. These collections make use of C#'s generics feature, which means each Collection only accepts it's intended subclass of `GameObject`, an NPC can never be added to a Collection of Items. This feature also relieves the `Game` class from needing to recast an object when accessed via the Collection. Because each Collection knows which type of object it stores it is able to return that exact type, instead of the more general Object type as it would without using the C#'s generic feature. `GameObjects` are added to the game via the `AddGameObject` method, which examines the `Type` of the `GameObject` and places it in the correct Collection.

Xml Creation

The `Game` class governs the creation of an Xml game file, stitching together the `XmlElements` created by the various `GameObjects`. Although no formal schema exists to define the format of this game file, development of OARnGES has coincided with the development of the OAR platform and this process has proved beneficial in debugging errors in both the Xml specification and the Xml files generated by OARnGES. The creation of an Xml document is greatly simplified by the classes included in C#'s System.Xml namespace. The `Game` object creates the top-level element and adds background data, such as the number of time switches and the roles in the game, then simply iterates through all `GameObjects` and appends their Xml definitions. The existing Xml specification does not completely lend itself to an object oriented approach and after appending all objects to the top level element, `Game` must iterate through `NPCs` and `Items` and append Interview and Information data separately. This added complexity could be removed through the alteration of the Xml specification.

Xml Parsing

An existing OAR game file is parsed in a distributed manner, each back end object knows how to instantiate itself from its own Xml definition and the `Game` object merely needs to know how to identify different elements. When the `Game` constructor receives a path to an Xml game file, the class begins the process of parsing the Xml in order to instantiate both itself and all `GameObjects` in the game. It begins by comparing the version number of OARnGES which created the file with its own version number and throws an Exception if they do not match. This allows the user interface to warn the user before trying to open a possibly unsupported game file. The constructor accepts a Boolean value to indicate whether or not to throw the Exception. If told not to throw the Exception, the `Game` class does its best to load the game. The `Game` class parses the background information which it stores itself and then simply iterates over the elements defining `GameObjects`, using them to instantiate new objects and

adding those objects to its Collections.

Access to Game Objects

The `Game` class exposes multiple overloaded methods to access stored `GameObjects`. The use of multiple methods allows the results to be filtered by role and/or time switch, if desired. This results in four separate methods:

- `GetAllGameObjects(void)`
- `GetAllGameObjects(Role)`
- `GetAllGameObjects(int)`
- `GetAllGameObjects(Role, int)`

Calling `GetAllGameObjects` with no arguments retrieves all `GameObjects` without any filtering, but passing a specific Role, time switch or both will return only those `GameObjects` which would be visible to that Role and/or at that time switch. The user interface uses this feature to give the game developer control over which objects are drawn on the map. All four methods simply provide public access to a single private method `_GetAllGameObjects`, which requires both a Role and time switch as parameters. This method iterates over all stored objects and returns those which are visible to the specified role and at the specified time switch. Each `GameObject` contains its own logic to determine its visibility relative to a given Role or time switch.

5.1.3 ObjInformation Class

Preparing interaction with virtual objects during game play requires OARnGES to maintain several types of content: paged text presented to the player to read, paths to audio and video files which the player can view and listen to, and also paths to documents (HTML files and images) which the player can inspect. The `ObjInformation` class encapsulates all this disparate data into a single entity for OARnGES to operate upon. This class provides a clean interfacing for manipulating pages of text, adding

and removing documents, audio and video files along with storing the time switch when this data will be shown and the Role which this data will be shown to. In addition, `ObjInformation` contains a deep cloning method, which creates a brand new `ObjInformation` object containing copies of, not shared references to, the original object's data. This proves very useful when the user wishes to copy the content for one Role at one time switch across multiple roles and/or multiple time switches.

ObjInformation makes use of a private class, `PagedText`, which manages the text presented to the player when visiting a `GameObject`. Although `PagedText` is nothing more than a simple array of strings, its public methods greatly simplify the usage of multiple pages of text. For instance, while the internal array of strings begins at the 0th index, `PagedText` makes that page publicly available as page number one. This coincides very cleanly with a game developer's expectation and precludes the UI from doing messy index calculations. The `PagedText` class also cleanly handles the addition and deletion of pages of text, isolating these operations prevents the objects interacting with `PagedText` from needing to concern themselves with maintenance of the `PagedText`. The `ObjInformation` class further simplifies this interaction, exposing just two methods, one to access a specific page of text and another to set a specific page's text.

5.2 User Interface Implementation

Although OARnGES has been designed and implemented to accommodate multiple, interchangeable user interfaces, only one interface has been fully developed at this point in OARnGES' development. This interface provides the developer with an overview of the state of the entire game from the vantage point of a game player.

5.2.1 Main Form and Map Panel

OARnGES' Map View interface presents the user with their games map as a canvas to place `GameObjects` upon. Many of OARnGES' features on this creen were intended to mimic a graphics program, where the game map cts as the user's canvas and the

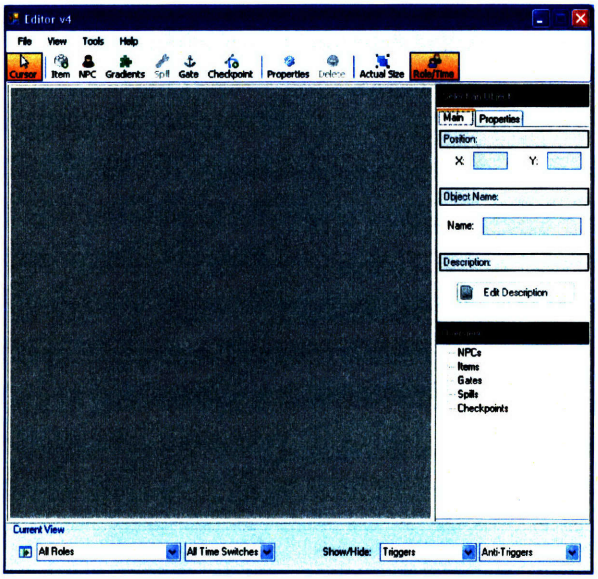

Figure 5-1: OARnGES' Main Form and Game Map View.

type of `GameObject` to place on the ap is chosen similar to choosing the type of brush to draw with. The ser is presented with a toolbar (Figure 5-2) above the canvas, hich allows them to select their "brush," an NPC, Item, Gate, etc. hen placing an NPC on the map, for instance, OARnGES draws the efault NPC shape at the point of the cursor (Figure 5-3). Once the NPC has been ositioned where the developer desires, a click places the NPC and pens a window to customize the new object's features.

Figure 5-2: Main Form's Toolbar.

Once a new `GameObject` has been added to the game, it appears on the map, with the color and shape specified when created. The Main Form gives the user the ability to filter which objects appear on the map to get a view of the game map at a certain time switch and to a particular role. Also, OARnGES provides the user with the ability to visualize triggers (Figure 5-4), drawing arrows from an object to those which it will trigger or anti-trigger. Since triggered objects are typically invisible prior to their triggering, OARnGES draws an empty square in the location where the object will appear, to visually discriminate between visible objects and objects which

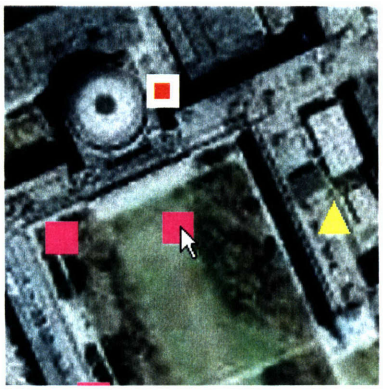

Figure 5-3: Placing an object in OARnGES.

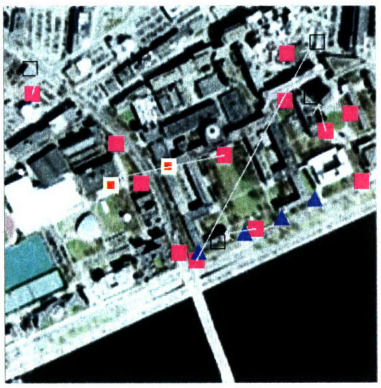

Figure 5-4: Visualizing Triggers in OARnGES.

appear via triggering.

5.2.2 Game Object Properties Form

The Game Object Properties form allows the game designer to configure all aspects of a `GameObject`. A single form contains fields to manipulate every feature of every subclass of `GameObject`, taking advantage of the many similarities between these subtypes. This not only allows OARnGES to reuse code, but the game developer is always presented with a consistent interface. The Game Object Properties Form constructor requires either the `GameObject` to be edited or the type of `GameObject` to be created. The constructor then displays the applicable features for that object type, while hiding the unsupported features. The Game Object Properties form allows the developer to control all properties of the object, from its name and location to which

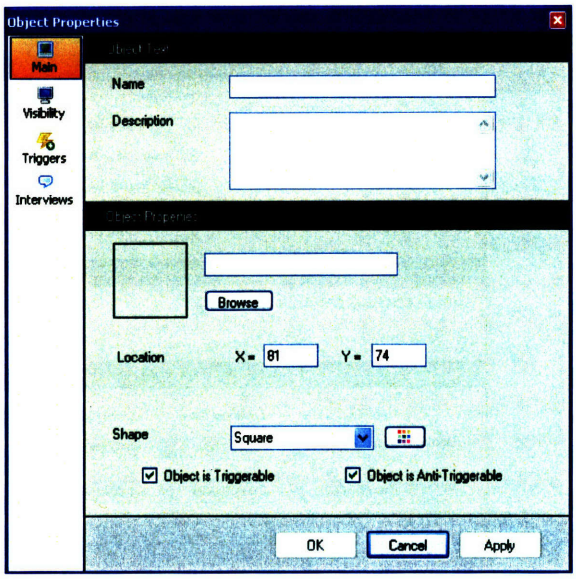

Figure 5-5: OARnGES' Game Object Properties Form.

roles can interact with it to what game objects are triggered after the current object is visited.

The form is forced to create many temporary storage variables to avoid committing changes to the back end data structures before the user decides to keep the changes. This allows the user to cancel the form without affecting the state of the game. Once the user does decided to keep the changes, the Game Object Properties Form signals the Main Form, which sets every property of the object being created or modified. The Main Form obtains the data through a series of public variables exposed by the Game Object Properties Form.

5.2.3 Properties Form

The Properties Form shares the look and feel of the Game Object Properties Form, but has much simpler behavior, since it avoids the dynamic display feature of the Game Object Properties Form. The Properties Form layout is static, as opposed to the Game Object Properties form, because the global settings available to the developer do not vary between games. These settings include the path to the map image, the GPS coordinates of the map, the number of time switches in the game

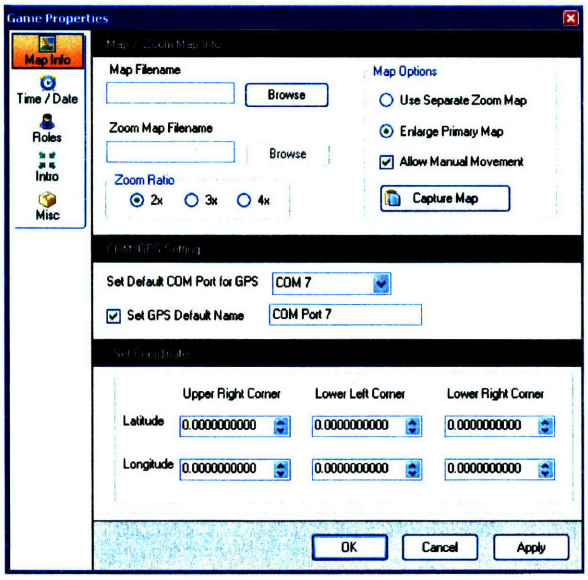

Figure 5-6: OARnGES' Game Properties Form.

and the types of Roles in the game. Similarly to the Game Object Properties Form, the Properties Form must utilize temporary storage to avoid committing changes too early and Main Form accesses data via a set of public variables.

5.3 Development Timeline

Initial work on OARnGES began with prototyping in March 2006 in preparation for a meeting between all teams working on the OAR platform. The map-centric UI was presented to the Harvard and UW-Madison researchers who would be OARnGES' initial target audience. The UI design was tweaked slightly addressing feedback from that initial meeting and some of OARnGES' goals were reprioritized. The back end of OARnGES was largely stabilized by early April 2006 and a functioning beta was release on April 17. The first major deadline for OARnGES occurred in July 2006. The team at UW-Madison needed a stable release to develop games in anticipation of a workshop demonstrating the capabilities of the OAR platform. Through a process of interim releases and bug fixes a feature complete version of OARnGES was obtained in late June and successfully generated OAR games for both the UW-Madison workshop and a workshop run at Harvard University.

The experiences of building games both in anticipation of the workshops and as part of the workshops largely influenced the future direction of OARnGES. Most of the developer tools were added in the fall of 2006 and by the winter OARnGES had largely been stabilized and few features have been added since then. Beginning in early 2007 a UROP at MIT has worked on both beautifying the UI and adding a "quick view" panel to show the main attributes of a selected object.

Chapter 6

Results

Throughout the 15 months of OARnGES' development, its various releases have been used to create, update and maintain approximately a dozen Augmented Reality games. These games have been designed and implemented by multiple developers, some built collaboratively and others built by a single developer. The evolution of OARnGES has been greatly influenced by the development of these various games.

The current design and implementation of OARnGES largely matches its initial concept, either a testament to its original design or an indication of underdevelopment.

Researchers at Harvard and UW-Madison expressed a desire to add new types of game objects to the OAR platform following the implementation of a few OAR games during the summer of 2006. The most sought after addition was an object which would give the developer greater control over the flow of the game. Further discussion of this issue resulted in the development of a game object which simply triggers and anti-triggers other objects, provided the player correctly enters a code. This object, named a Checkpoint, allows the game designer to manipulate which objects are available to the player and possibly require the player to solve a puzzle or answer a question in order to unlock new objects. Alternatively, the Checkpoint can be used to ensure a large class of students remain in the same general vicinity. In this scenario, players interact with available objects until arriving at the Checkpoint. With no further objects to interact with, the player remains at the Checkpoint until given a code to unlock new objects. This strategy allows the teacher to hold the class

at specific points until stragglers catch up.

The addition of Checkpoints to OARnGES was quite straightforward and accomplished in just a few days. The back end `Checkpoint` class naturally inherited most of it's properties from the `GameObject` class, with only slight modifications required to accommodate the code feature. The majority of the implementation time was spent updating the user interface to accommodate the new object and its attributes. While many UI elements were able to be reused due to the common `GameObject` base class, features such as conditional triggering required some redesign. However, the effort devoted to the early design of OARnGES' data structures proved beneficial, not only by speeding the development process but also in thinking about ways to expand the Outdoor AR platform.

Although OARnGES does not currently provide multiple views of game data, the separation between game data and user interface has proven beneficial. An undergraduate student was able to leverage the decoupled data structures to quickly and cleanly embed data about a selected `GameObject` in the Main Form. After laying out the UI elements, all that is required to add this functionality is to populate those elements when the selected object changes. This is easily accomplished by piggybacking on the existing object selection code. Similarly, the Main Form was extended to include a simple tree listing all `GameObject`s on the map. The implementation of this feature was also aided by the clean interface to OARnGES' back end data structures.

While OARnGES does a good job of separating back end data from its user interface, the map centric design has not proven conducive to complete game development. Many users of OARnGES design games and assemble content outside of OARnGES, then merely use OARnGES as a tool to create an Xml game file from that pre-existing content. Clearly, OARnGES has failed to provide useful functionality in guiding the development of an OAR game from concept through implementation. The lack of a standard game design process at the early stages of OARnGES' design resulted in OARnGES positing a certain design style and developing around that style. OARnGES has yet to evolve beyond that initial design style and incorporate lessons learned from 15 months of OAR game development.

Chapter 7

Future Work

Although OARnGES currently provides all the functionality needed to develop OAR games, there still remains many further areas of development. For instance, the current map-centric user interface could certainly be refined through usability testing and alternate views could be developed to aid game design. Also, the .NET Framework continues to grow and further development of OARnGES can harness the new functionality of the framework. Finally, development of the OAR platform continues and OARnGES will need to accommodate new game features.

7.1 User Interface

7.1.1 Existing Design

The current user interface for OARnGES allows the developer to view the entire game state and modify all features of game data. However, the number of features which game developers are allowed to modify has continued to grow since work began on OARnGES. The UI has not handled this expansion gracefully, many new features and settings have been forced into an existing template which struggles to contain them. The current UI would benefit greatly from a redesigned approach to editing `GameObjects` and game content. Also, as the number of OAR game developers continues to grow, conducting usability tests of the UI becomes a feasible idea.

7.1.2 Future Designs

Currently, OARnGES restricts game developers to a universal view of their game state. This places strict limitations upon their ability to view interactions between `GameObjects` and does not give a sense of the game's narrative. Many OAR games are designed to guide the player through a defined story, moving from location to location along a well defined path. Future UI designs of OARnGES could provide the game developer with alternative views of game data which are better suited to developing these types of games.

Storyboarding

For instance, OARnGES could enable game designers to develop game content through a storyboard style interface. This interface would focus on giving the developer a clean view of the information each object will present to players during the game. Because OARnGES currently restricts the developer to seeing this content one Role and time switch at a time, a storyboarding interface could present multiple Interviews at a time, allowing for easy comparison. Many OAR developers already develop games in this style, but OARnGES' limitations force them to develop game content outside of OARnGES to allow this side-by-side comparison. The largest obstacle to the development of this interface is the amount of variability allowed by the platform. Three dimensions of information must be accommodated, game object, Role and time switch. The challenge will be in both helping the user to select a slice of this cube to view, and also to alert the user to which slice they are currently viewing. Once this content has been created, OARnGES' map centric UI can then assist the developer in positioning objects on the map.

Game Paths

OARnGES' map centric UI attempts to show the developer how the game will appear at a certain time to a certain role, but the triggering feature of the OAR platform causes a players view of the game to change as they interact with different objects.

At the early stages of OARnGES' development triggers were not an integral part of game play and this shortcoming of the UI was not a significant issue. Over time, however, triggers and anti-triggers became widely used to guide a player's movement through the game, and the usefulness of OARnGES' UI declined. An additional view of game data could allow game designers to observe how interaction with game objects changes a player's view of the world. The game designer would be presented with a view of all objects visible at the beginning of the game. Choosing a visible object would then update the view as necessary, showing the newly triggered objects and hiding the anti-triggered objects. The interface would also show the "path" followed to arrive at the current game state. The user could then step backwards through this history and branch off along a different path to arrive at an alternate game state.

Junior Editor

OARnGES has been developed to enable game developers to control every facet of OAR games. As a result, OARnGES' learning curve is very steep, a large number of OAR game concepts must be understood to successfully create an interesting and engaging game. This creates a formidable barrier to entry for both OARnGES and the OAR platform. A wizard style game editor would make a nice addition to OARnGES, to guide new users through the process of creating an OAR game, while hiding much of the complexity from the user. This "junior" editor would present the user with only those features which are required to create a workable OAR game, and automatically flesh out the remaining features with default values. Guiding the developer through the game design process step by step, this junior editor would not only introduce the OAR platform and concepts, but also show the user sound game design strategies and workflows. Once users new to OARnGES have become familiar with OAR games through the junior editor, they can use the main OARnGES interface to enhance their initial game or develop a more complex game from the beginning. Ideally, this junior editor would be straightforward enough for students to develop their own games as part of an OAR curriculum. Enabling students to create their own content and develop their own game would expand the educational benefits of the OAR platform.

7.2 .NET Framework

Although the .NET Framework is a state of the art technology developed by Microsoft, OARnGES is limited by the Windows Forms API used by the framework. Microsoft is posed to release a brand new graphical user interface API known as Windows Presentation Framework (WPF.) By upgrading to WPF OARnGES will be better positioned to remain technologically relevant. In April of 2007 Microsoft announced its new Silverlight project, aimed at enabling the development of rich internet applications. Silverlight is a cross-browser, cross-platform technology which allows users to launch a .NET application from their web browser [7]. This is a powerful paradigm from which OARnGES would greatly benefit. Developed with Silverlight, OARnGES can be delivered across the Internet, instead of distributed as a compiled program. This not only gives greatly flexibility to OARnGES release schedule, but will also enable game designers to work with OARnGES from any computer with an Internet connection. A Silverlight based version of OARnGES would no longer require the Windows platform, removing a barrier to entry of the OAR platform.

7.3 Data Structures

A more general game object would give game designers a much more flexible platform on which to develop games. This evolution would require much work on the OAR game engine, along with a rethinking of both OARnGES' front end and back end designs. Currently the OAR platform restricts game developers to well defined subclasses of the general `GameObject` class, limiting their creativity. A more powerful platform would allow designers to develop their own subclasses, mixing and matching features and also specifying in game look and feel for each new class. This would require a much more powerful and more flexible design for OARnGES to accommodate the greater level of customizability allowed by this new platform. However, allowing game developers to build custom objects will enable much richer narratives and give developers much more control over the appearance of their games. For example, a de-

veloper could create a Witness class, which the player would be allowed to interview, and a Suspect class, which the player would get to interrogate.

This scenario would also foster more cross-game collaboration and development than the current OAR paradigm. Currently, the effort spent developing game characters is tied very closely to the narrative of that game, outside of that context much of a character's content loses its relevance. In contrast, a general characteristics of a new class of game objects developed for one game may easily fit the narrative of a different game. The development of a shared repository for these new objects, browsable through OARnGES, would help foster cross-game collaboration and encourage richer OAR game development.

Appendix A

OARnGES User Interface Manual

A typical game design process using Editor3 is detailed below.

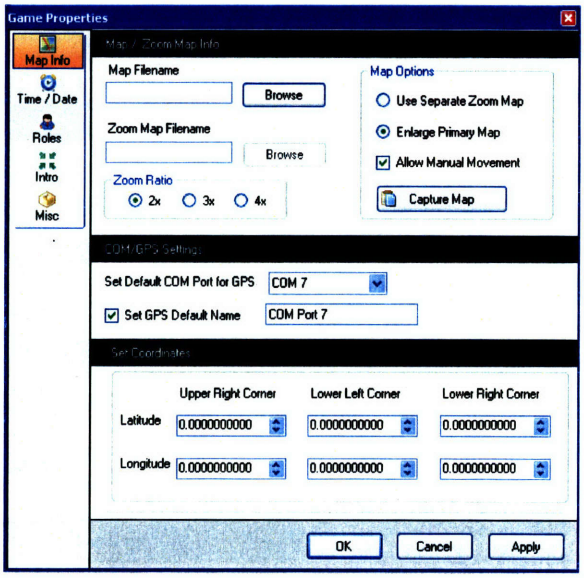

Figure A-1: OARnGES' Game Properties Form.

With the Cursor button selected, click Properties to view the game properties form. Here the user can browse for a map file and specify your zoom map. Alteratively, they can launch the MapTool and grab a map file and GPS coordinates automatically. The buttons on the left side of the form switch panes, allowing the user to specify information about the Time and Date, to create Roles and write Introduction text.

Once all background information has been specified, the next step is to place game

objects on the map. Using the game object properties form you can specify all the information related to that object. The form has a similar look for all objects but the specific fields are tailored to the features of a given object.

Game Object Background Pane (Figure A-2)

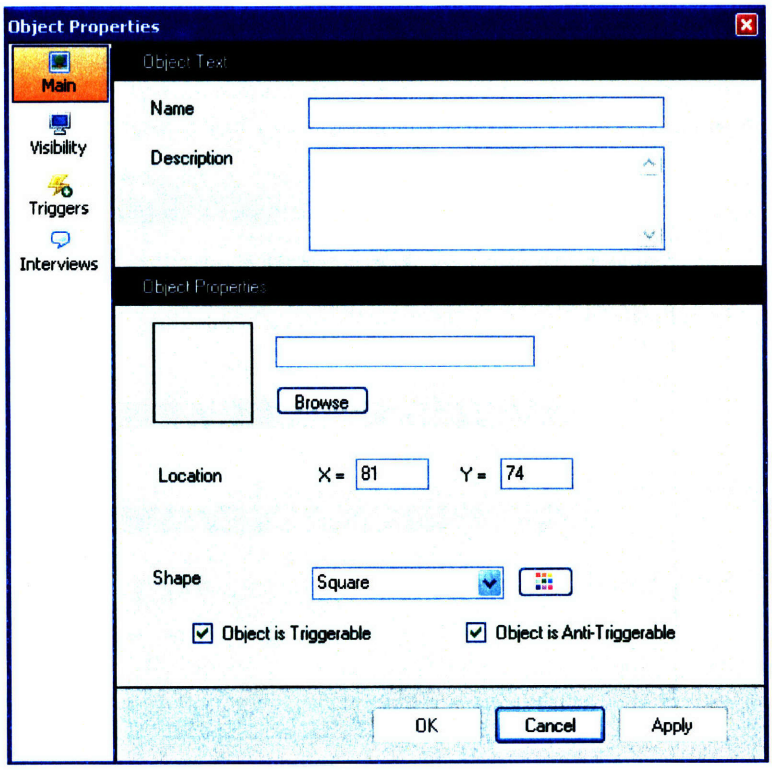

Figure A-2: OARnGES' Game Object Background Pane

Used to give the object a name, description, picture and to set other background information about the object. This information will be displayed to the user when they interact with the game object during the game. The object's location can also be adjusted manually, specifying its exact (x,y) coordinates, on the 200x200 game map. Both the color and shape of all GameObjects can be set on this pane; these settings will carry over to the game on the handheld.

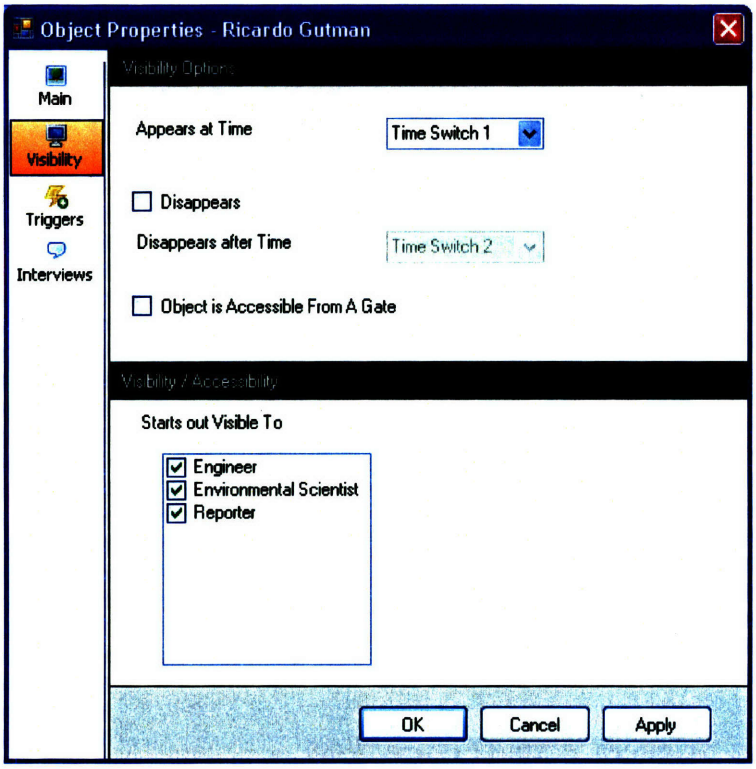

Figure A-3: OARnGES' Game Object Visibility Pane

Game Object Visibility Pane (Figure A-3)

This pane allows the user to set the time when the object will first appear and specify whether the object will ever disappear. If the object does disappear, the last time switch when the object will be on the map must be specified. If the object is set "accessible from a gate," it will not be drawn on the map, it must be associated with a gate. Finally, using this pane sets the roles which the object is initially visible to. The object will only be drawn on the map for the checked roles, and the object will not be seen by the other roles until triggered. An object which begins as not visible to a certain role can be triggered by a different object and become visible to a certain role. For Items, you can also specify accessibility in the same way. Roles which can see an Item but not access it are able to interact with the Item, see its name, description and picture, but cannot see the associated Information.

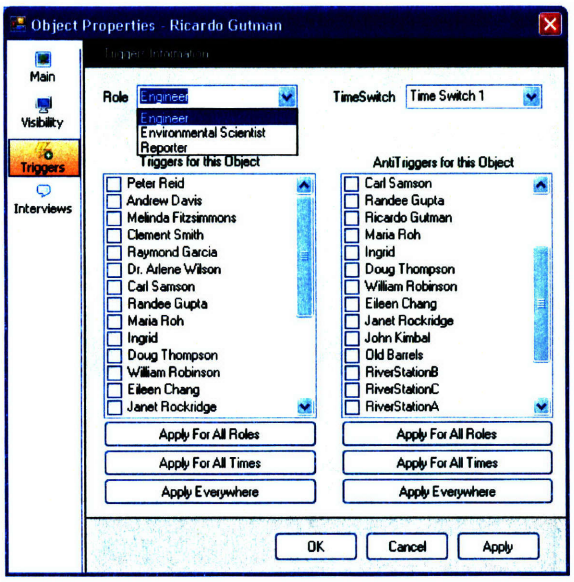

Figure A-4: OARnGES' Game Object Triggers Pane

Game Object Triggers Pane (Figure A-4)

Used to specify the objects which will be triggered or anti-triggered by the current object. After the player interviews an NPC or inspects an Item, the triggered objects will appear on the game map and the anti-triggered objects will be removed from the game map. To set an object to be triggered or anti-triggered, simply check the checkbox next to it. The objects to be triggered and anti-triggered can be specified by role and time, using the drop down menus at the top of the pane to choose which Role and time combination to set triggers for. To quickly apply the current set of Triggers or Anti-Triggers use the Apply For All Roles / Apply For All Times / Apply Everywhere buttons. Apply For All Roles will give the current set of Triggers or Anti-Triggers to each role at the current time. Apply For All Times will give the current set of Triggers or Anti-Triggers to the current Role at all times. Apply Everywhere will give the current set of Triggers or Anti-Triggers to every role at every time.

Item Information Pane (Figure A-5)

Used to compose the text which the player will see when inspecting the Item in the game. Multiple pages of text can be created by changing the "Information Length

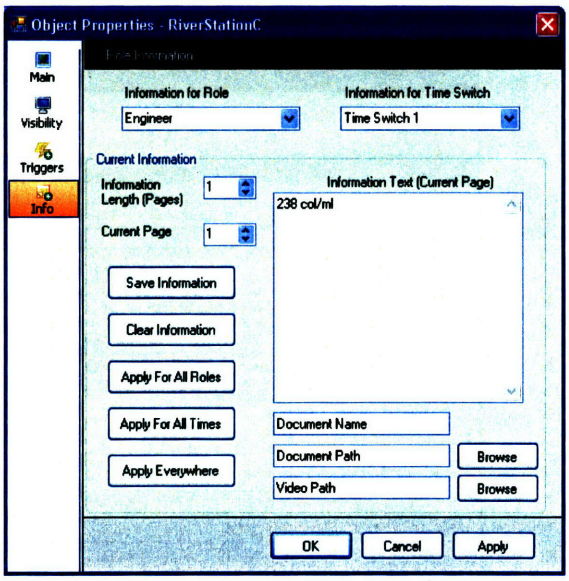

Figure A-5: OARnGES' Item Information Pane

(Pages)" property and the "Current Page" property is used to control which page is being edited. Just as with Triggers, the Information returned by an Item can be customized to the Role viewing it and when it is being viewed. The Item Information pane uses a similar interface to the Triggers Pane, you specify a Role and time combination using drop down menus at the top of the pane and can use the Apply For All Roles / Apply For All Times / Apply Everywhere buttons to quickly replicate Information among roles and times. Each Information can specify a document to give the player, which can be a web page, an image, an audio file or a video file. If a document is added to an Item Information, it must also have a Document Name, which must be unique across the game. An audio or video file can also be added to the Item Information and the player will be able to listen to or view this file as they inspect the Item. The Clear Information button will reset all the data, but only for the current Item Information. To clear all data, clear the current Information and then use the Apply Everywhere button.

NPC Interview Pane (Figure A-6)

Used to compose the text which the player will see when interviewing the NPC in the game. Multiple pages of text can be created by changing the "Interview

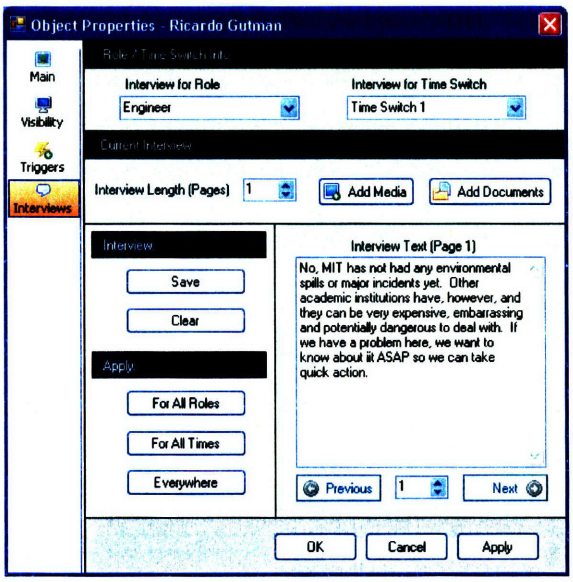

Figure A-6: OARnGES' NPC Interview Pane

Length (Pages)" property and the "Current Page" property is used to control which page is being edited. Just as with Triggers, the Interview returned by an NPC can be customized to the Role viewing it and when it is being viewed. The NPC Interview pane uses a similar interface to the Triggers Pane, you specify a Role and time combination using drop down menus at the top of the pane and can use the Apply For All Roles / Apply For All Times / Apply Everywhere buttons to quickly replicate Interviews among roles and times. Each Interview can specify a document to give the player, which can be a web page, an image, an audio file or a video file. If a document is added to an Interview, it must also have a Document Name, which must be unique across the game. An audio or video file can also be added to the Interview and the player will be able to listen to or view this file as they interview the NPC. The Clear Interview button will reset all the data, but only for the current Interview. To clear all data, clear the current Interview and then use the Apply Everywhere button.

Gate Information Pane (Figure A-7)

Used to put Gate Accessible objects behind the Gate whose properties you are editing. To place an object behind the Gate, simply check the object's checkbox. To edit the

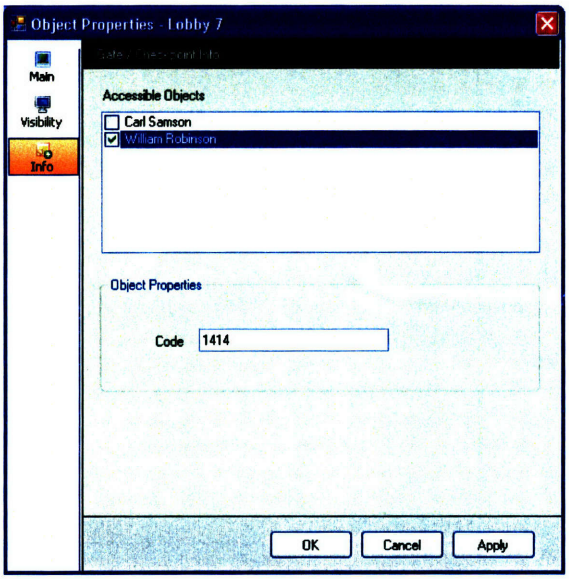

Figure A-7: OARnGES' Gate Information Form

code for an object behind the Gate, click the object's name and the name will then be highlighted and the code textbox will contain the current code for that object.

Spill Information Pane (Figure A-8)

Used to specify properties of the Spill being created. Spill Dropoff can be set as either Exponential or Linear (at the current time neither setting affects gameplay.) Spills are drawn as ellipses around two given foci, and the minor axis of the ellipse is set via the Radius property. The major axis is then determined by the minor axis and the distance between the foci. Finally, the intensity of the spill can be set, on a relative scale from 0 to 100.

Gradients Form (Figure A-9)

Used to create and edit Gradients in the game. Gradients do not have many properties, just a name, the amount of time it takes to sample the gradient and the roles which are able to access this gradient. Spills will have the same accessibility rules as the gradients which they belong to (i.e. either a role has a Geiger counter and can measure all Radiation spills, or they don't have a Geiger counter and have no way of measuring any Radiation spills.) The Display Color setting only affects the way spills

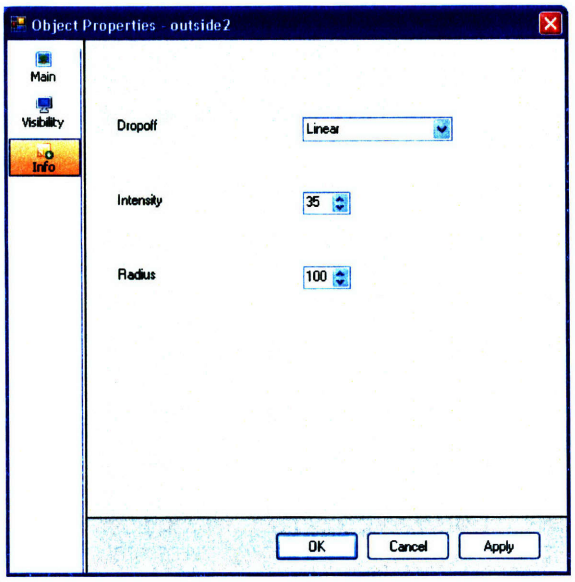

Figure A-8: OARnGES' Spill Information Pane

are drawn on the Editor screen, the color setting will not carry over to the handheld. To edit an existing Gradient, selecting it from the table at the bottom of the form will populate the fields with its information. Finally, each gradient can have multiple sample types, or different methods of sampling which the player can choose from during the game. The "Add Sample" "Edit Sample" and "Delete Sample" buttons allow the user to create and edit these. The "Edit Sample" and "Delete Sample" buttons will affect the highlighted sample, while "Add Sample" will create a new sample.

Sample Type Form (Figure A-10)

This form is used to create and edit a Sample Type associated with a specific Gradient. There is currently no way to copy Sample Types between Gradients. Each Sample Type has only four properties, a description, the length of time (in seconds) it takes to process a sample of this type, the accuracy of the sample type and which roles can use the sample type.

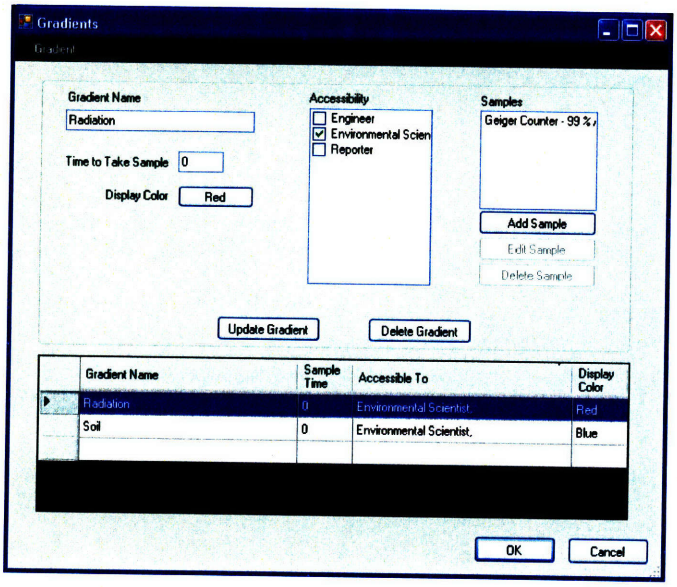

Figure A-9: OARnGES' Gradients Form

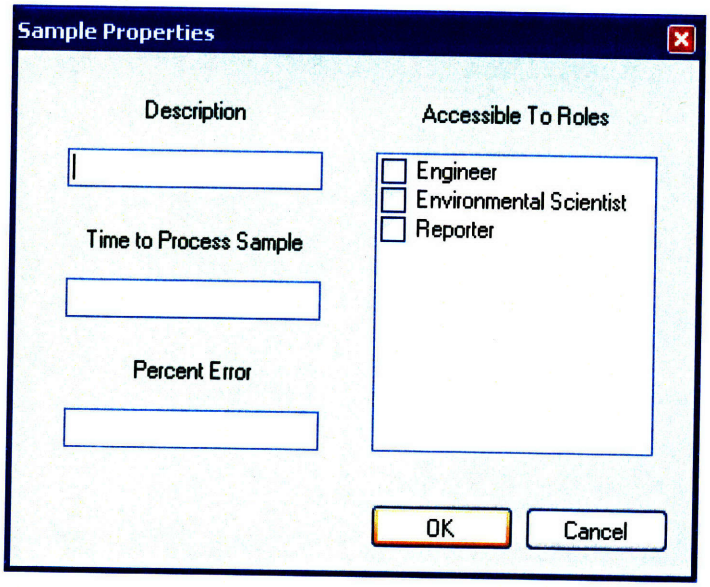

Figure A-10: OARnGES' Sample Type Form.

Bibliography

[1] Priscilla Cheung. Charles River City: An Educational Augmented Reality Simulation Pocket PC Game. Master's thesis, Massachusetts Institute of Technology, 2003.

[2] Matt Dunleavy. Harp evaluation report. Evaluation of Fall 2006 Implementation of OAR game at Harvard.

[3] Eric Klopfer, Judy Perry, Kurt Squire, and Ming-Fong Jan. Collaborative learning through augmented reality role playing. In *CSCL '05: Proceedings of th 2005 conference on Computer support for collaborative learning*, pages 311–315. International Society of the Learning Sciences, 2005.

[4] Eric Klopfer, Kurt Squire, and Henry Jenkins. Environmental Detectives: PDAs as a Window into a Virtual Simulated World. In *Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education*, pages 95–98, 2002.

[5] Mickey Williams. *Microsoft Visual C# .NET.* Microsoft Press, 2002.

[6] http://www.mono-project.com.

[7] http://www.microsoft.com/silverlight.