Design Specification for Augmented Reality Engine

Erin Jamroz, Selah-Mae Ross, Matthew Burke, Thomas Freeman, David Greene

October 2, 2012

Introduction

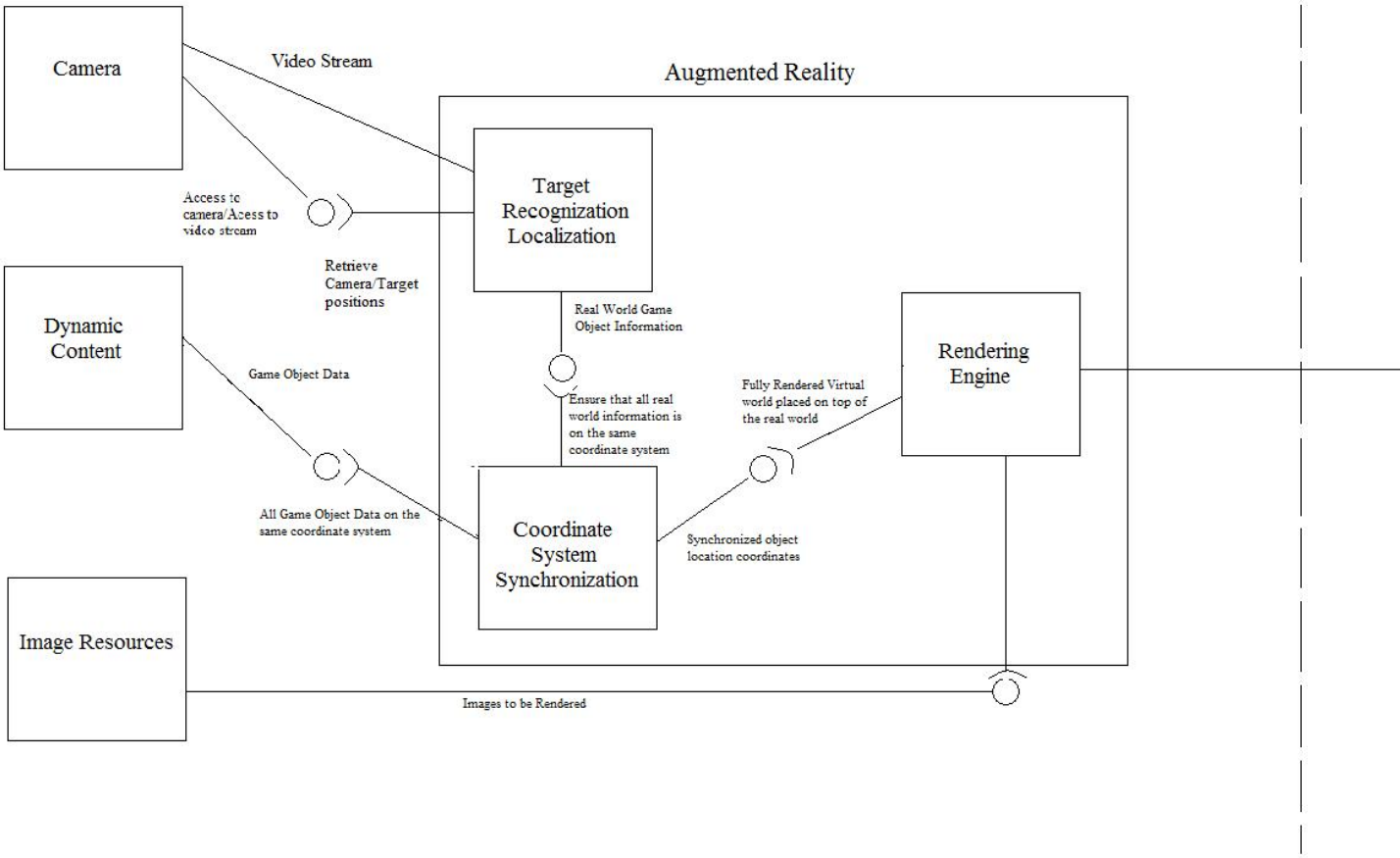
Our firm has been tasked with the creation of a game that incorporates somewhat unique input and output methods that utilize both motion capture as well as augmented reality. The game itself will be of a platforming genre, where a motion controlled puppet attempts to survive in a fixed environment for as long as possible while being assaulted by users of mobile devices. Our specific team is tasked with the creation of the augmented reality system, that is to be implemented on these phones.

This game will serve 4-7 users, or players, who will be split into two distinct groups. The first group, consisting of 3 players, will interact with the system via motion capture, posing and gesturing while in front of a Microsoft Kinect. The other group, will consist of 1-4 players, provides the system with input and receives output via Android smart phones. These devices will observe a game setting by looking at a specified region, and will be able to rotate and move around this region to attain different viewpoints. The core objective of our team lies in creating a working system that can correctly determine the position of the game world, and work in conjunction with other teams to achieve the full effect of a 3d game world portrayed in 'real' space.

In order to anchor the game world into the real world, we need localization. This will be accomplished through the use of a large printed 'AR target' that will establish where the game world will be drawn. The goal of this printed target is to provide a basis for our team's software to determine the android phones position and angle relative to the target. Our team's software will gather this positional information, and then render the game scene from the perspective of each individual observer to be displayed on their mobile devices.

Architecture Design

The Augmented Reality architecture design details the required modules of the AR team's product. Information provided as input by other teams (as depicted below: Image Resources/Dynamic Content/Camera) will be processed through the AR modules in order to display a superimposed 3D scene on the user's phone. The first module that is accessed is the Target Recognition and Localization module. This module requires real world data provided by the camera and mobile group, and will retrieve the camera and target positions in order to determine where their positions are in relation to the real world. After accessing the first module, the Coordinate system Synchronization (or CSS) is next. This module requires the information that was processed by the first module, as well as data from the web server and phone in order to merge real world information with what is provided by the game. The final module that the information will pass is the Rendering Engine. This module receives information from the CSS and the Image Resource data bank in order to create the 3D world imposed on the real world. This scene is then sent to the mobile group to have displayed on the phone's screen.



Overview of Architectural Design

Summary of Interfaces

Required Interfaces:

- The phone application needs to initiate the AR module.
- The Target Recognition and Localization module requires video feed from the camera.
- Image resources for each object to be displayed
    - Specifically, the AR team will need a *Wavefront OBJ* file and a corresponding .jpg or .png image file with the object's texture.
    - The OBJ file will be converted to a .h header file and these resources will be loaded and stored in the image resource data bank, which will be loaded onto the phone at installation.
    - The above will allow the AR team to display static objects, research continues on what will be needed for animation.
- Regularly updated location information for all objects in the game.

Provided Interfaces:

- Augmented video feed will be provided to the phone's display.
- Location information for the phone and all detected AR targets.

Module Design

Target Recognition/Localization:

Description:

This module will be taking information from the mobile devices camera directly. Information picked up via the camera will then be used to accurately identify AR targets which then will be used to calculate both the cameras position and the targets position in a 3D coordinate space. After calculating relative phone and target positions this information should then be passed off to the Coordinate System Synchronization module.

Purpose:

This module functions as an interface for between the Coordinate Systems Synchronization (CSS) module and the mobile devices optical hardware. Both the phones relative position and the AR targets positions in the game are calculated here so that they may be passed off to the CSS module in a meaningful format.

Provided interface(s):

1. Ability to retrieve relative camera and target positions in a to be determined 3D coordinate system.

Required interface(s):

1. Access to the mobile devices camera OR access to the video stream from the mobile devices camera.

Coordinate System Synchronization (CSS):

Description:

This module will receive data from the Target Recognition/Localization module and from both the web server and the mobile host device (the device this module is running on). Objects and events from these different sources will then be merged together to align on a universal coordinate system overlaid on target data. After all objects, images, and characters are assigned their proper positions in the 3D coordinate system they will be ready to send off the the next module in the "pipeline."

Purpose:

The goal of this module is to convert all of the game world objects (characters, the terrain, etc) and make sure that they all match a universally approved coordinate system. Additionally, this module will merge the "real world" related game objects and ensure that they also match the approved coordinate system. Furthermore,

this module will also take any object creation/destruction/modification commands from the host mobile device, and ensure that this phone data is properly synced with the web server and Target Recognition Location module. This module's final objective is to push the now merged object data to the Rendering Module.

Provided interface(s):

1. Access to all incoming game data which are all on the same coordinate system.

Required interface(s):

1. Access to the Target Recognition/Location output. (The "real world" game object information.)
2. Access to the host device's button presses or other triggers. (Button presses and other events that occur on the host phone that modify the game world.)
3. Access to the web server to receive gameobject data from the machine hosting the game and other mobile devices.

Rendering Engine

Description:

This module will receive information from both the CSS module and Image Resource data bank. It will overlay image and texture data from the data bank onto the appropriate objects in the virtual world based upon positional data obtained by the CSS module.

Purpose:

This module is responsible for making sure that the world is rendered properly. All textures are overlayed on the appropriate objects. Without this module, the virtual world would only be a collection of points in three-dimensional space. This module gives those coordinates their unique look and feel based on the image data created by the graphics design group.

Provided Interface:

1. A fully rendered virtual world properly overlaid on real world image data.

Required Interface:

1. Access to Image Resources data bank
2. Properly synchronized object location coordinates

Other Design Views

In order for our model to function re require an image resource databank that will house all of the game models and objects on the mobile device. When a game object is received from either the host device or the web server, the object ID will be used to find the appropriate image or object in the Image Resource Databank. This image data will be passed off to other modules in order to properly render a frame. This databank will contain all image textures and "wireframes" for static content such as the level backgrounds. The purpose of this

database is to house game object's models and textures, and give other modules access to these files.

Design Rationale

The architecture we have designed involves the sequential manipulation and synthesis of inputs from several sources. We start with a camera image that needs to be processed to determine the viewers angle on the scene. Then the location of every object in the game world needs to be modified to reflect this particular viewpoint. Finally, we must draw these objects on top of the image of the real world in order to produce our final output.

This method greatly reflects the 'pipe and filter' design pattern, as we have three key modules, or 'filters', in which we engage in sequential processes, each relying on the previous, in order to produce our final output. Each of these tasks needs to be completed in a fairly strict sequential order, and so the only way to deviate from this design is to combine or divide steps. Splitting any of these modules further would likely create unnecessary complexity, and combining these tasks would be unwise as each is fairly distinct and would have few resources in common.

These modules will respectively take and process an image of the real world, determine correct perspective, and then render the correct perspective of the game world, all of which are tasks laid out in our requirements.

Implementation Notes

In order to implement this architecture design, the Augmented Reality group has determined that their primary source of coding will be done in C++ and Android Java, using the compiler Eclipse and the API provided by Vuforia. Each module is handling the pieces of rendering an overall scene, and the library provided by Vuforia is integrated with OpenGL to assist in the process of creating 3D graphical scenes. Although the extent of Vuforia's capabilities are still a mystery to the AR team, our beginning work with Vuforia has yielded results that allow us to expect that most of our code in order to implement these modules will be done with the library provided by the SDK.

As of the latest communiques between the mobile phone team and the AR team, we are considering implementing the AR module as a Fragment within the larger Android application.

In agreeance with the CS240 class, our version control will be GitHub in order to access necessary information on other groups as needed, as well as provide an informative wiki about the AR team's progress, or potential issues that may arise as we progress through the project. For documents such as the Design Document or the Requirements Specification, the AR team has found that google docs provides a simple and easy to modify workspace. Any other information will be provided on GitHub and made available to the rest of the project's teams.

Glossary

Image Resource Data Bank: This will be a collection of image resource files of all objects to be displayed in the game. It will be stored on the phone.
Vuforia™: is an augmented reality software development kit for mobile devices. It is produced by Qualcomm and facilitates the development of AR applications.
Vuforia™ Jargon:

- **Image Target:** Images that the Vuforia SDK can detect and track. Any image that has certain characteristics and has been processed by the Vuforia Target Management System may be used as an Image Target. Image Targets have robust tracking and are resilient even when partially occluded.
- **Frame Marker:** these are simplified AR targets compared to Image Targets. The Vuforia SDK identifies Frame Markers through a unique id coded into a binary pattern of black and white squares around their edge. The entire
- **Multi-Target:** A collection of Image Targets arranged on a three-dimensional box. Since the relative locations of the targets is known, the SDK can anticipate their locations and include augmentations even before new targets are acquired.
- **Virtual Button:** Virtual buttons can be included in Image Targets. These buttons are "pressed" when they are occluded, at which point, some corresponding action can be executed.
- **Target Management System:** A web based system that allows developers to upload custom images to be used as Image Targets.

References

"Vuforia Developer Guide," Qualcomm, Version 2.1.18, last accessed 10/1/2012, https://ar.qualcomm.at/developer_guide

“Vuforia Augmented Reality SDK,” Wikipedia, last updated 9/8/2012, last accessed 10/1/2012, http://en.wikipedia.org/wiki/Vuforia_Augmented_Reality_SDK