**BILKENT UNIVERSITY**

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

# CS491 – SENIOR DESIGN PROJECT

# PROJECT *inlight*

# ANALYSIS REPORT

**3.10.2014**

**Ahmet Yavuz Karacabey**

**Latif Uysal**

**Yiğit Özen**

**Süleyman Kılınç**

# TABLE OF CONTENTS

## 1. INTRODUCTION

Our project aims to simulate light effects on various materials in augmented reality environment, creating a convincingly realistic representation of what that material would look like in reality. The application of this project is primarily targeted at tablet computers. More specifically the application is intended for decoration / furniture industry.

In order to be able to simulate the light, the light sources in the environment of the simulation first needs to be found and identified. The designated means of detecting these light sources is the built-in front facing camera that is to be found on a great deal of the modern tablet computers. However, the front facing camera has a relatively narrow field of view, typically around 50 degrees from side to side and 80 degrees from top to bottom. In this case, it may not be possible for the camera to detect all of the light sources that are illuminating the material to be simulated on the tablet screen. Using an aftermarket fisheye lens on the front facing camera, it is possible to widen its field of view up to 150 degrees side to side and 170 degrees top to bottom. This offers a practical and elegant solution to camera's field of view problem.

The captured photographs through this fisheye lens needs to be taken using a certain method called HDR (High Dynamic Range) imaging to capture the entirety of the light sources within the field of view. For example in a typical scenery with a single real light source such as a light bulb, the material is illuminated not only by the light bulb but also the reflective surface/object present in the environment as well. In this sense, using HDR imaging yields images with greater amount of lighting detail. Then this image will be used to designate light sources in Augmented Reality environment.

Currently, we are planning to use shader-based OpenGL for programming certain portions of the project. Additionally, Unity's rendering engine along with certain other features are intended to be used.

As a final product, the end user will be able to simulate a set of different materials of various colors on the tablet screen as if the user is holding the actual material in their hand instead of the tablet.

## 2. RELATED WORK

We were unable to find any similar applications to inlight during the research. However, there are several academic papers on the methods we are planning to use that were published in the last decade, since the seamless integration of synthetic objects into real images is one of the hot topics in computer graphics area.

Environment mapping was proposed for rendering ideal mirror surfaces with a perfect specular reflection. To include diffuse reflections in the environment map, irradiance information needs to be considered. Brute-force technique is the traditional method of calculating irradiance. However, it is impossible implement it in real-time because of its computational complexity.

To decrease the number of instructions and computational complexity, J. Kautz et. al. proposed an irradiance environment map [1]. Their method that is aimed at real-time applications generates the irradiance map in advance by prefiltering the environment map. This prefiltered map stored the radiance of light reflected towards the viewing direction, which is computed by weighting the incoming light from all directions. By fast prefiltering and GPU programming, they decreased precomputation time.

R. Ramamoorthi et. al. showed that using 9 spherical harmonics coefficients in frequency space when storing a compressed environment map is enough to make irradiance calculations with lower than 1% error [2]. Using this approximation allowed them to include complex materials under environmental illumination with little precomputation.

J. Kautz et. al. presented a method rending system for VR applications [3] and G. King introduced a technique replicating the above methods on the GPU [4].

The researches mentioned so far have dealt mainly with static situations in a fixed illumination environment. V. Havran et. al. proposed an interactive system to handle HDR image-based lighting captured in dynamic, real world conditions, with complex shadows and arbitrary reflectance models [5].

J. Kim et. al. proposed a method for modifying the environment map on the GPU in real-time to generate photo-realistic rendered images at interactive frame rates in dynamic environment [6]. Their system is the most similar one to our project. We will try to implement their technique with some modifications for our use-cases.

## 3. PROPOSED SYSTEM

Image Based Lighting (IBL) is a rendering method that involves capturing a still image of the environment where the object to be rendered resides and using this image to overlay certain effects such as illuminated textures. This technique, as the name suggests, is used primarily for rendering a detailed lighting in real-world environment. For this reason, specialized equipment is often used for to capture the scenery. In our project, we are intending to use front facing camera of the tablet computer coupled with a fisheye lens. The research we have done so far indicated that fisheye lenses do yield a significant increase over the field of view of the camera and can readily be used for real time rendering.

The later process of casting the light onto the object is mapping the light sources captured by the camera accordingly. This process is generally called environmental mapping. Environmental mapping is a popular IBL technique that allows the modeling of a reflective surface by making use of precomputed textures. There are certain different approaches as to how an environmental map can be implemented. These include sphere map, cube map and so on. The majority of contemporary efforts on IBL techniques rely almost exclusively on cube mapping as sphere mapping has become rather obsolete. Incorporation of cube map into IBL, according to a number of different experiments and papers, is likely to be the most accurate method to cast lighting and illumination on objects to be rendered. We will use the 9 spherical harmonics coefficients method for storing the irradiance map as it mentioned in the related work chapter.

We will use GPU programming to modify the irradiance map in real-time in a similar way as J. Kim et. al. proposed.

High Dynamic Range (HDR) imaging is widely used photographing technique that makes it possible to captures lighting details in a scene in a more complete fashion. In order to achieve this, the typical method is to combine 3 images with different exposure values (-1, 0, +1) of the exact same scene. This is also the method we are planning to use in our project. By combining together these images with different exposure values, it is possible to obtain one image where what would appear to be well lighted and illuminated in each of the 3 images. In fact, HDR imaging is a popular technique in rendering graphics. During our research we came across several different articles and experiments on image based lighting as well as environmental mapping that makes use of HDR images captured [7][8]. The key point in choosing HDR images in any case is obtaining a means of interpolating lighting settings in rendering environment.

The details of system-user interaction will be explained in the user interface chapter.

## 4. GLOSSARY

**User**: The end user of the application.

**Material**: A 2D asset that is rendered by the system according to the dynamic lighting of the environment.

**Object**: A 3D asset that is rendered by the system according to the dynamic lighting of the environment.

**Catalog**: A list of materials or objects that are presented to User for choosing to render.

**Rendering**: The process of generating an image after lighting, shading etc. as a verb, or the result of the process as a noun.

# 5. REQUIREMENTS

## 5.1. Functional Requirements

- The user will be able to select an object/material from the catalog to render and display.
- The user will be able to see the object as if it exists physically in the scene.
- The user will be able to alter the location of the object by moving the device.
- The user will be able to alter the texture of the object.
- The user will be able to save the rendering.
- The user will be able to load and display a previously saved rendering.
- The admin will be able to add materials to the database.
- The admin will be able to delete materials from the database.
- The admin will be able to modify the materials in the database.

## 5.2. Non-Functional Requirements

- The rendering of the object should be photorealistic.
- The computation process for light sources should be low latency so as to make rendering real time.
- The program should be available as an application on a tablet computer.
- The user interface should be intuitive and easy to use.

## 5.3. Pseudo Requirements

- OpenGL's cube map texturing extension will be used for rendering.
- The mobile application will be developed for Android operating system with Java.
- The camera on the tablet will be used for capturing the environment image.
- High Dynamic Range Imagery will be used for greater realism.
- Irradiance environment map will be used to store targeted environment's lighting information.

- Fish-eye lens will be used to enhance the angle which built-in camera of the device is able the capture.
- Anti-glare film may be used for glare reduction on the tablet screen.

## 6. USE CASES

**Use Case: SearchMaterial**

**Actor:** User

**Entry Conditions:** The device is connected to material/object database.

**Exit Conditions:** One or several materials/objects with their categories are listed on the screen.

**Flow of Events:**

**1.** User enters complete or partial name of the material/object or category.

**2.** User taps onto search button.

**3. Inlight** will change the catalog according to the search results.

**Quality Requirements:** Search process should not take too long.

**Use Case: ChooseMaterial**

**Actor:** User

**Entry Conditions:**

**T**he device is connected to material/object database.

Material/object catalog is not empty.

**Exit Conditions:** Material/object is loaded and ready for rendering process.

**Flow of Events:**

**1.** User selects the material/object that will be used for rendering listed in **SearchMaterial** use case.

**2. Inlight** loads the material/object for rendering and shows it in display screen.

**Quality Requirements:** Loading the material/object should not take more than few seconds.

**Use Case: StartRendering**

**Actor:** User

**Entry Conditions:** A material/object is loaded to the device and ready for rendering.

**Exit Conditions:** Rendered material/object will be displayed.

**Flow of Events:**

**1.** Users taps onto display screen after after a material/object is loaded to start rendering.

**2. Inlight** renders the material/object according to the light environment around device's location.

**Quality Requirements:**

Rendering should be performed in real time.

Rendered object should give the user a realistic feel as if the object is in the same location as the device.

**Use Case: SaveRendering**

**Actor:** User

**Entry Conditions:** Inlight is currently rendering a material.

**Exit Conditions:** Rendering is saved for later display.

**Flow of Events:**

**1.** User taps onto save rendering button.

**2. Inlight** displays a textbox for rendering name input from user.

**3.** User enters the name of rendering.

**4. Inlight** saves rendering with that name into device local.

**Quality requirements:**

**Use Case: LoadRendering**

**Actor:** User

**Entry Conditions:**

**Exit Conditinos:** A rendering stored in the device is displayed to user,

**OR** An error message is prompted to user indicating there is no rendering found.

**Flow of Events:**

**1.** User taps onto load rendering button.

**2. Inlight** displays the list of rendering stored in the device.

**3.** User taps onto a rendering.

**4. Inlight** displays the chosen rendering on full screen.

**Use Case: AddMaterials**

**Actor:** Admin

**Entry Conditions:** Admin's device should have internet access.

Exit Conditions: One or more materials are added to material database.

**Flow of Events:**

1. Admin activates add materials function in his device.

2. **Inlight** displays a form to admin with fields related to materials.

3. Admin fills out the form and submits it.

4. **Inlight** adds the specified material to material database.

**Use Case: ModifyMaterial**

**Actor:** Admin

**Entry Conditions:**

Admin's device should have internet access.

At least one material should exist in the database.

**Exit Conditions:** Material specified by admin is modified

**OR** Material specified by form parameter does not exist. An error message is prompted to admin.

**Flow of Events:**

**1.** Admin activates modify materials function in his device.

**2. Inlight** displays a form to admin with search parameters.

**3.** Admin fills out the form and submits it.

**4.** If there is a match in material database, **Inlight** displays another form with attributes of the matched material.

**5.** Admin modifies some attributes and submits the form.

**6. Inlight** modifies material's record in the database.

**Use Case: DeleteMaterial**

**Actor:** Admin

**Entry Conditions:**

Admin's device should have internet access.

At least one material should exist in the database.

**Exit Conditions:**

**Flow of Events:**

**1.** Admin activates delete material function in his device.

**2. Inlight** displays a form to admin with search parameters.

**3.** Admin fills out the form and submits it.

**4.** If there is a match in material database, **Inlight** displays another form with attributes of the matched material.

**5.** Admin may confirm or deny the delete operation.

**6.** If admin confirms the operation, Inlight deletes the material from material database.

## 7. USER INTERFACE

The application will be customized for each subject and distributed separately. For example, an app will be exclusively for Tepe Mobilya (furniture brand) products, and another for Ege Seramik (ceramics brand) products.

Opening up the application will directly bring the catalog to the screen. In this screen, the products/materials will be divided into categories/series. The categories will be stacked vertically, and the materials in a category will be listed horizontally to achieve efficient use of the space and easy discovery of assets. Both the category list and material lists will be scrollable without paginations.

On top of the screen there will be a search box for searching both categories and materials by names. There also will be a list of colors, displayed by circles filled with each color. Tapping onto a circle will filter the materials displaying only those that have the selected color.

Tapping onto a material will bring the rendering output. Full screen will be used to render the material in real time. The front camera will have been used to capture the environment for lighting and the resulting rendering will be displayed on the screen without any user input. Screen gestures such as long pressing and swiping will be used for returning to the catalog or saving the current rendering.

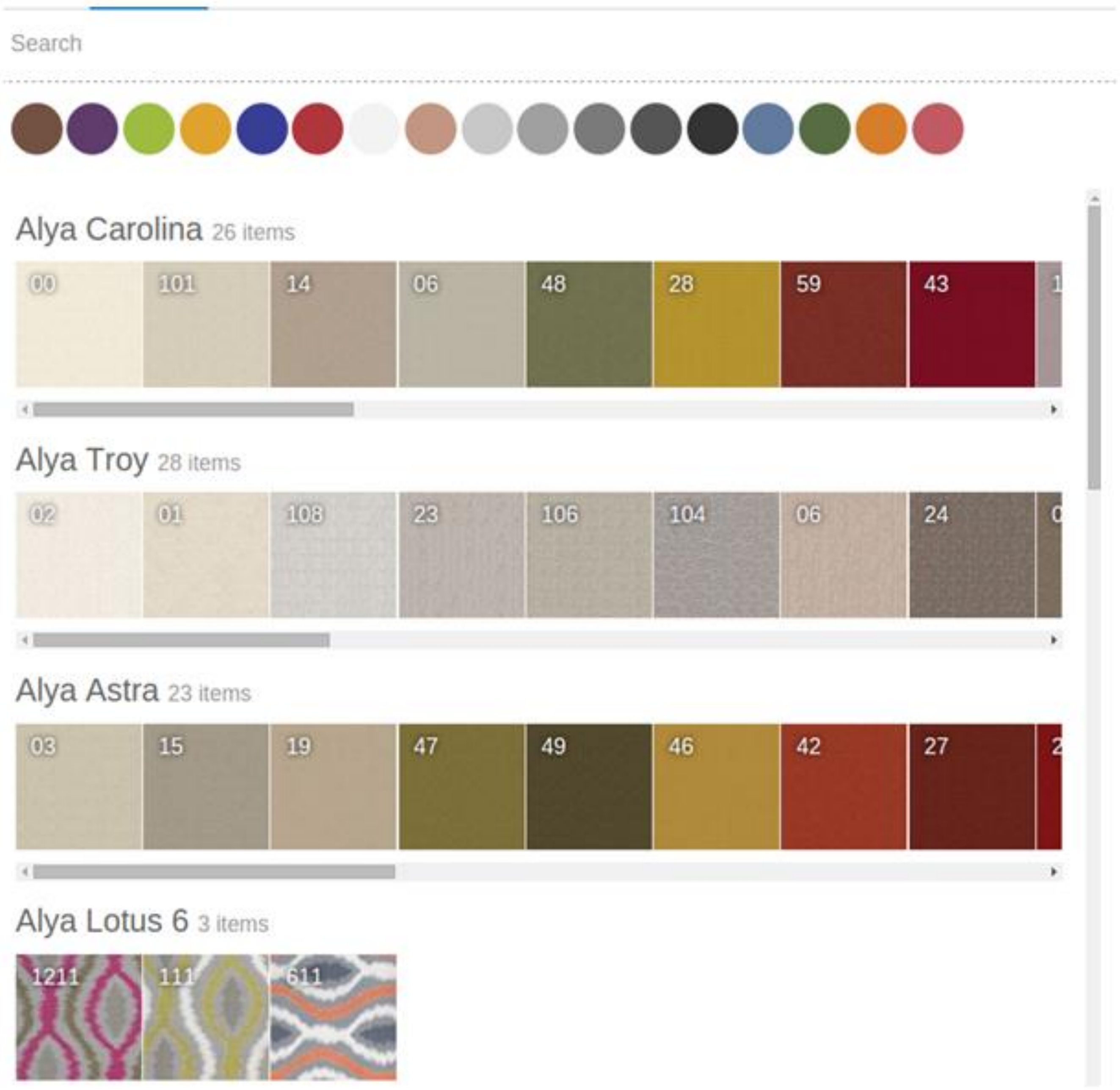

*Figure 1: Catalog View*

## 8. REFERENCES

[1] Kautz, J., Vazquez, P., Heidrich, W., Seidel, H.P.: A unified approach to prefiltered environment maps. In: EuroGraphics Rendering Workshop, pp. 185–196 (2000).

[2] Ramamoorthi, R., Hanrahan, P.: An efficient representation for irradiance environment maps. In: Proc. of SIGGRAPH, pp. 497–500 (2001).

[3] Kautz, J., Daubert, K., Seidel, H.P.: Advanced environment mapping in VR applications. Computers & Graphics 28, 99–104 (2004).

[4] King, G.: Real-time computation of dynamic irradiance environment maps; GPU Germs, vol. 2(4), pp. 98–105. Addison-Wesley Professional, London, UK (2005).

[5] Havran, V., Smyk, M., Myszkowski, K., Seidel, H.P.: Interactive system for dynamic scene lighting using captured video environment maps. In: Proc. of Eurographics Symposium on Rendering, pp. 31–42 (2005).

[6] Kim J., Hwang Y., Hong H. Using Irradiance Environment Map on GPU for Real-Time Composition. 2007.

[7] Palomo C. Real-time High Dynamic Range Image-based Lighting. 2008.

[8] Hajisharify S., Kronanderz J., Miandjix E., Unger J. Real-time image based lighting with streaming HDR-light probe sequences. SIGRAD (2012).