BILKENT UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

CS 492: SENIOR DESIGN PROJECT II

PROJECT *inlight*

LOW LEVEL DESIGN REPORT

16.02.2015

Yiğit Özen
A. Yavuz Karacabey
Latif Uysal
Süleyman Kılınç

Contents

1. Introduction

Recent developments in technology made it possible to combine virtual and real world together. As the definition suggest, this is called augmented reality. Augmented reality is used to create a wide range of applications and services for almost everyone today. These include head mounted displays that perform like a smart phone (Google Glass for example) or gaming devices such as the Oculus Rift that provides the users with stereoscopic view as well as motion tracking. However these examples merely cover a fraction of Augmented Reality applications today.

We are creating an Android application for photorealistic rendering of different materials on phone or tablet screen. The application will display fabric samples on the screen and replace the actual fabric samples given at furniture sellers. The target is to make the difference so seamless that the displayed material under the same lighting conditions will look greatly alike as the real one.

In order to achieve this, the device will be equipped with a fisheye lens which will allow it to scan a larger field of view to capture lighting information. Then the captured lighting information will be normalized through a profiling system which later then be used by the shader. The shader that is going to render the material is based on Cook-Torrance shading model. This model is particularly useful for simulating materials in an environment where directional lighting is present [1]. Then the rendered image will be displayed on the device screen in real time.

1.1. Trade-Offs

1.1.1. Performance vs. Quality

The application is targeted at mobile Android devices, so it is likely to be run on a very large variety of different devices and Android versions. For this reason, it is impossible to optimize the application for one particular setup of hardware and Android version. The application is

supposed to render the material in real time, so it has to be configured in a way that it will not overwhelm the device and cause delay in renders. This is directly related to rendering quality. Higher resolution renders with more accurate lighting effects and detail will burden the hardware with more computations. However, turning down the render quality more than necessary will produce renders with very poor and unrealistic visuals. Having to target multiple hardware sets and Android versions therefore imposes such a tradeoff on the application.

1.1.2. Practicality vs. Accuracy

The device is to be equipped with a fisheye lens in order to cover a larger field of view. The lens is not a vital part of the system, but it is necessary for capturing as much lighting information as possible. Standard issue front facing cameras in most devices cover around 60 to 70 degrees less than the fisheye lens. In such a case a directional light with a horizontal angle of 40 degrees will not be detected and will not be included in the final render. This is will lead to poor lighting accuracy and therefore unrealistically rendered materials [2]. On the other hand having to rely on a 3rd party equipment for accurate functioning of the application comes in unpractical due to having to calibrate and profile the camera prior to using. Additionally, the lens has to be taken care of and carried with the device.

1.1.3. Bandwidth vs. Storage

The application will retrieve image files for materials from a server. This will make it possible add, remove and update materials available for simulating independent of application itself. However, doing so will force the application to have an active internet connection in order to function. On the other hand, storing the image files locally on the device would be very cumbersome as there are likely to be gigabytes of data to be stored most of which won't be used at all. In order to establish a reasonable middle-ground between local storage and bandwidth consumption, the application relies on the server to download image files whereas the frequently used files such as camera profiling data will be stored locally on the device.

1.2. Engineering Standards

In the entirety of project reports UML (Unified Modeling Language) is used to demonstrate class organizations, scenarios and application features, as UML is a widely known and accepted modeling convention. Additionally, the reports present the references and citations in IEEE referencing format.

1.3. Definitions, Acronyms and Abbreviations

Fisheye Lens: A camera lens type that increases the coverage of the camera through optical manipulation.

Real-Time Process: A process that produces output at the same time of execution or with very minimal delay.

Fabric Sample: Piece of fabric that is given to customers at furniture shops to allow the customers to try out and see.

Photorealism: Rendering of computer generated images in a realistic fashion by making use of lighting in the environment.

Bump Mapping: A technique for creating 3D bumps on a surface by distorting the normal vectors on the surface.

AR: Augmented Reality.

2. Packages

2.1. Model Package

Model package encapsulates the state of application and raw data that is to be used by inlight. It includes user data, material data, and image data in different stages of capturing and rendering process. Model communicates directly with Controller package. It can receive queries from controller about the current state of rendering, user data. Model can update this data as dictated by Controller classes. Model does not communicate directly with GUI packages. Any state changes in the model is propagated to GUI by Controller.

2.2. Graphical User Interface Package

GUI package performs as the intermediary between the application and the user. It takes input from the user, and displays output to the user via device screen. It includes activity managing classes, which have event handler functions to achieve communication between application and user. GUI classes delegates user input to respective Controllers, and raw or rendered images to user. It allows controllers to lay out the display screen. GUI does not communicate with Model directly. All communication between GUI and Model classes are handled by corresponding Controller classes.

2.3. Controller Package

Controller package encapsulates the logic of inlight, and thus is the brain of our application. It communicates with both Model and GUI directly and handles all the communication between them. It performs the core functionality of Inlight. It manages camera attached to the device. It uses the images taken by camera for calibration and rendering. Controller performs each step of rendering and post-processing until the image is ready for output. It manipulates display screen, the data in Model and performs user authentication.

3. Class Interfaces

3.1. Complete Class Diagram

Complete class diagram of the system is given on the next page (Figure 1).

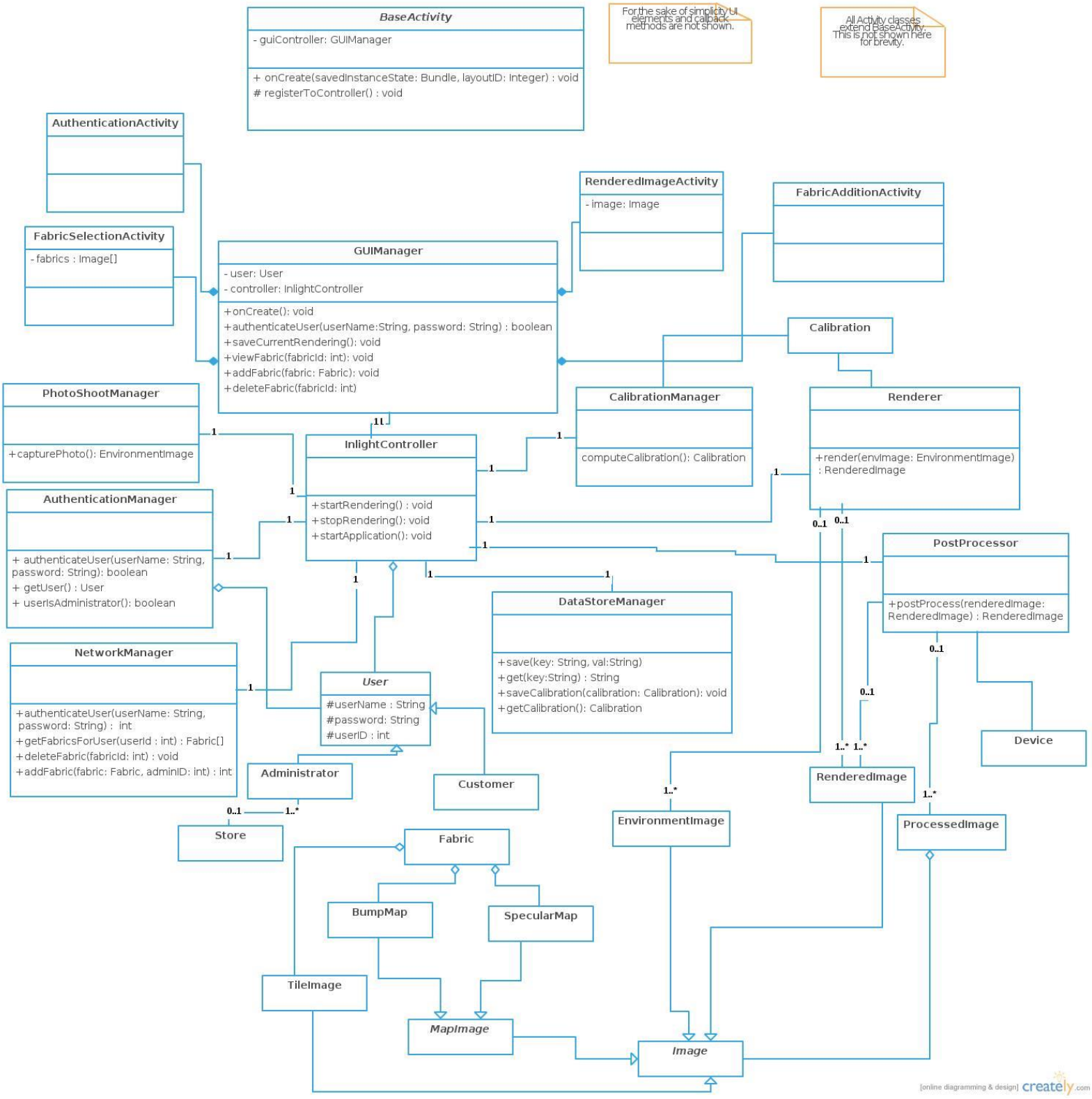

Figure 1: Complete Class Diagram

Figure 2: Complete Class Diagram

3.2. Package: com.inlight.controller

3.2.1. Class AuthenticationManager

Class Name	AuthenticationManager
Description	Responsible for authenticating users at the login stage. It communicates the credentials of the users to the server via NetworkManager. It also keeps the state of which user has logged in and whether s/he is an administrator.
Package	com.inlight.controller
Attributes	networkManager: NetworkManager user: User
Operations	authenticateUser(userName: String, password: String): boolean chekcs whether the user exists on the server database getUser() : User returns the current user userIsAdministrator(): boolean checks whether current user has administrator priviledges

3.2.2. Class NetworkManager

Class Name	NetworkManager
Description	Handles all kinds of communication with the server including user authentication, getting fabric details, fabric addition, and deletion.
Package	com.inlight.controller
Attributes	
Operations	authenticateUser(userName: String, password: String) : int Checks whether username and pasword are valid. getFabricsForUser(userId : int) : Fabric[] Returns fabrics available for a given user id. deleteFabric(fabricId: int) : void Deletes fabric. addFabric(fabric: Fabric, adminID: int) : int Adds fabric to the associated administrator's selection.

3.2.3. Class PhotoShootManager

Class Name	PhotoShootManager
Description	Responsible for periodically taking photos using device's camera which is enhanced by a fish eye lens for larger field of view. It acts an intermediary between Android classes and our application.
Package	com.inlight.controller
Attributes	
Operations	capturePhoto(): EnvironmentImage Takes the current photo of the environment and returns it.

3.2.4. Class DataStoreManager

Class Name	DataStoreManager
Description	Responsible for managing persistent storage on the client device. It abstracts away Android API and offers a convenient interface.
Package	com.inlight.controller
Attributes	
Operations	save(key: String, val:String) Saves given value corresponding the key. get(key:String) : String Returns the value associated with the provided key. saveCalibration(calibration: Calibration): void Saves the computed calibration values getCalibration(): Calibration Returns the saved calibration from memory

3.2.5. Class CalibrationManager

Class Name	CalibrationManager
Description	Responsible for device specific calibration. This class is only used once for each device at the time of the application setup.
Package	com.inlight.controller
Attributes	calibration: Calibration device: Device
Operations	computeCalibration(): Calibration

3.2.6. Class Renderer

Class Name	Renderer
Description	Responsible for realtime rendering of the fabric. It calculates the shading and illumination based on the current photo of the environment. The rendered image is sent to PostProcessor.
Package	com.inlight.controller
Attributes	fabric : Fabric
Operations	render(envImage: EnvironmentImage) : RenderedImage Renders the fabric according to the current environment image.

3.2.7. Class PostProcessor

Class Name	PostProcessor
Description	Post-Processor retrieves a rendered image from the Renderer and makes the last modifications on it, such as color profiling. The modifications done by the Post- Processor are generally device-specific .
Package	com.inlight.controller
Attributes	device: Device
Operations	postProcess(renderedImage: RenderedImage) : RenderedImage Applies the device specific modifications to the rendered image.

3.2.8. Class InlightController

Class Name	Inlight Controller
Description	This is the main controller class which manages the whole application lifecycle. It arranges the photoshoot frequency according to the capabilities of the device hardware. It selects an FPS at which all calculations can be concluded and runs the application at that FPS.
Package	com.inlight.controller
Attributes	
Operations	startRendering() : void Starts rendering loop stopRendering(): void Stops rendering startApplication(): void Starts the application

3.3. Package: com.inlight.model

3.3.1. Abstract Class User

Class Name	*User*
Description	This is the abstract class which defines a generic user. It has two subclasses, Administrator, and Customer.
Package	com.inlight.model
Attributes	userName : String password: String userID : int
Operations	

3.3.2. Class Administrator

Class Name	Administrator
Description	Represents store employees who has priviledges to add and delete fabrics and allow customers to see fabrics.
Package	com.inlight.model
Attributes	store : Store
Operations	

3.3.3. Class Customer

Class Name	Customer
Description	Represents a customer who can only see fabrics which are previously cleared for his viewing.
Package	com.inlight.model
Attributes	store : Store
Operations	

3.3.4. Class Store

Class Name	Store
Description	Represents a store where a group of fabrics belong to. Multiple Administrators can work in the same store.
Package	com.inlight.model
Attributes	name : String address: String phoneNumber : String
Operations	

3.3.5. Class Fabric

Class Name	Fabric
Description	Represents a fabric. It holds a collection of features which define a fabric such as its tile image.
Package	com.inlight.model
Attributes	tileImage : TileImage bumpMap: BumpMap specularMap: SpecularMap
Operations	

3.3.6. Class Device

Class Name	Device
Description	Represents a handheld computing device. It is assumed that the device runs on Android operating system.
Package	com.inlight.model
Attributes	screenSize: double androidVersion: String
Operations	

3.3.7. Abstract Class Image

Class Name	Image
Description	Parent class of various types of images that the application makes use of.
Package	com.inlight.model
Attributes	
Operations	

3.3.8. Abstract Class MapImage

Class Name	MapImage
Description	Abstract common properties of bump and specular maps
Package	com.inlight.model
Attributes	
Operations	

3.3.9. Class BumpMap

Class Name	BumpMap
Description	Map of surface normals of the fabric which simulate the bumps and wrinkles of the fabric.
Package	com.inlight.model
Attributes	
Operations	

3.3.10. Class SpecularMap

Class Name	SpecularMap
Description	Map of relectivity of the fabric surface which are used to define the surface's shininess and highlight color.
Package	com.inlight.model
Attributes	
Operations	

3.3.11. Class TileImage

Class Name	TileImage
Description	Represents the texture of the fabric surface.
Package	com.inlight.model
Attributes	
Operations	

3.3.12. Classs EnvironmentImage

Class Name	EnvironmentImage
Description	Represents the image of the environment captured by the device camera. This is a 2D image and needs to be transformed into 3D with respect to the calibration values.
Package	com.inlight.model
Attributes	
Operations	

3.3.13. Class Calibration

Class Name	Calibration
Description	Represents the precomputed transformation map which is used to transform environment image to a cube map.
Package	com.inlight.model
Attributes	
Operations	

3.3.14. Class RenderedImage

Class Name	RenderedImage
Description	The result of the real time rendering process. This simulates how the fabric looks under the current lighting conditions of the environment.
Package	com.inlight.model
Attributes	
Operations	

3.3.15. Class ProcessedImage

Class Name	ProcessedImage
Description	The output of the PostProcessor which is displayed on the screen. This is a modified version of the RenderedImage which is adjusted according to the device.
Package	com.inlight.model
Attributes	
Operations	

3.4. Package: com.inlight.gui

3.4.1. Class BaseActivity

Class Name	*BaseActivity*
Description	This is the abstract parent class of all other activity classes. Each activity class correnspponds to a different view, (a full screen window).
Package	com.inlight.gui
Attributes	guiController : GUIManager This is the reference to the GUIManager.
Operations	onCreate(savedInstanceState: Bundle, layoutID: Integer) : void Initializes the activity. *registerToController(): void* Registers this activity to the GUIManager

3.4.2. Class AuthenticationActivity

Class Name	AuthenticationActivity
Description	Shows the login form in which the users can enter the authentication info in order to use the application.
Package	com.inlight.gui
Attributes	
Operations	

3.4.3. Class FabricSelectionActivity

Class Name	FabricSelectionActivity
Description	Shows all available fabrics in a list and lets the user pick a fabric to view as rendered.
Package	com.inlight.gui
Attributes	fabrics: Image[]
Operations	

3.4.4. Class GUIManager

Class Name	GUIManager
Description	GUIManager controls other classes in the view package. It manages the interoperation of the view Activities. It also works as an intermediary between the view Activities and the main controller of the application.
Package	com.inlight.gui
Attributes	user: User inlightController: InlightController
Operations	onCreate() : void Entry point of the application, called at the initialization of the program. authenticateUser(userName:String, password: String) : boolean Delegates authentication to the AuthenticationManager. Returns true if the login information is correct. saveCurrentRendering(): void Saves the current rendered image of the fabric. viewFabric(fabricId: int): void Notifies the application controller of the selected fabric, and sets the RenderedImageActivity class in operation. addFabric(fabric: Fabric): void Adds a new fabric deleteFabric(fabricId: int) Requests deletion of the selected fabric from the server.

3.4.5. Class RenderedImageActivity

Class Name	RenderedImageActivity
Description	This is the heart of the application. It shows the real time rendered image of the selected fabric.
Package	com.inlight.gui
Attributes	fabric:Image
Operations	

3.4.6. Class FabricAdditionActivity

Class Name	FabricAdditionActivity
Description	Shows appropriate forms to add a new fabric by an administrator
Package	com.inlight.gui
Attributes	
Operations	

4. References

[1] "Writing a Cook-Torrance Surface Shader", Pixar Rendermen Documentation. Web. Retrieved Feb 2015.

[2] Yalin Xiong; Turkowski, K., "Creating image-based VR using a self-calibrating fisheye lens," Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on , vol., no., pp.237,243, 17-19 Jun 1997