上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Software Engineering

PROJECT REPORT

Real-Time Facial Animation System

Group	Student I.D.	Name	E-mail
26	5140219190	庞博（Poli Pang）	244804587@qq.com
	5140219191	包伟铭（Weiming Bao）	wm_bao@sjtu.edu.cn
	5140219311	刘子凡（Zifan Liu）	601956270@qq.com

INSTRUCTOR: Bin Sheng

DURITION: 11/2016~1/2017

Introduction of the project

1 Purpose of the project

Facial animation has become a well-known and popular topic in fields of animated feature films and computer games. Besides, it can also be applied in quite a number of other areas, such as communication, education, scientific simulation, agent-based systems (i.e. online customer service representatives), etc. With recent advanced improvements in computational power of personal and mobile devices, facial animation has transitioned from appearing in pre-rendered content to being created at runtime.

The ultimate goal of our project is to accomplish a software that is capable of capturing the user's facial expressions with a camera, reconstructing the corresponding 3D facial models and generating 3D facial animations in real time.

2 Scope of the project

Facial performance capturing and tracking have been extensively studied in both computer graphics and vision.

For color and depth input, Weise et al. construct a user-specific blendshape model as a preprocessing stage and then fit the blendshape weights for each color and depth frame at run time. The resulting weights are then transferred to other 3D avatars for real time facial animation. Bouaziz et al. and Li et al. present solutions to further avoid the preprocessing by jointly optimizing the user-specific expression model and expression parameters at run time. Although these methods can capture the 3D facial performance in real time and are robust to illumination changes, they may fail for faces with large rotations or occlusions. Recently, Hsieh et al. propose a method which gives uninterrupted facial tracking even with large occlusions. However, it cannot recover motions of the occluded regions.

For color input, Cao et al. present a regression method for real-time 3D facial performance capture that requires user-specific training and calibration. Later, Cao et al. propose a general regressor that is learned from a public image dataset for reconstructing 3D facial shapes from video frames.

With recent advances in real-time facial tracking and performance capturing techniques, performance-driven facial animation has become available for many consumer-level real-time applications, such as telecommunications, computer games, training and other online interactions. For example, Snapchat, a chat application, has included a new layer of "fun" to their platform through Selfie Lenses. The new feature allows chatters to overlay on their selfie an animated selfie lenses that react to facial expression movements.

In our project, we use AAM (*Active Appearance Models*) to reconstruct 3D models from the input of camera, and output the 3D model in real time. Since the computing power of the GPU on a PC is limited, we use a server to do the computation. Therefore, in our project, we also have to establish a connection between clients and server.

3 Overview of the document

Part 2 describes the requirements of this project, including functional requirements, non-functional requirements and domain requirements.

Part 3 provides the design of the software. In this part, we describe the software model, its architecture design and object identification in detail.

Part 4 is about testing. We provide our test plan, test cases and test analysis.

Part 5 summarizes our whole project.

Part 6 is references.

DocNo: 001.C.1:1

Requirement Analysis Document

Version 2.0

Group Member:
庞博
包伟铭
刘子凡

Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-5	1.0	Finish the architecture of the document	Whole team
2016-11-11	1.0	Do assigned part respectively	Whole team
2017-01-2	2.0	Adjustment	刘子凡

Key Word
QuickFace
Functional
Non-functional

Abstract
This document is important for our QuickFace project, since it will define most of the requirements for this system. We describe not only functional but also non-functional requirement as well, and give a specific hardware and software environment, and other necessary information for our system.

Table of Contents

1 Introduction

1.1 Purpose

This is a requirement specification document. In this document, we will define most of the system requirements, so that all the develop team member can have a clear picture of the whole system. We define not only the functional requirement, but the non-functional requirement as well. This is the main document of this define phase. In this phase, we also have glossary document, use case specification document. We also offer a prototype.

1.2 Definition

- ✧ QuickFace: an efficient software capturing and recognizing human faces in real-time 2D video stream and reconstructing corresponding 3D models.
- ✧ Functional requirements: something that the system should do
- ✧ Non-functional requirements: all the remaining requirements which are not covered by the functional requirements

1.3 Reference

Bob Hughes and Mike Cotterell *Software Project Management*

2 System Overview

We separate our system QuickFace into two packages in our implementation, which is composed of a server-end application and a user-end software.
Main features of QuickFace:

a) Capture live camera images
b) Recognize human faces and reconstruct corresponding 3D models
c) Provide the interface to export the generated 3D models to improve compatibility
d) Enlightening applications like scene replacement

3 Detailed System Requirements

3.1 Functional Requirements

1. When a user launches the application, the User Interface must be displayed
1.1 The application must open the camera and display the image captured by the camera in real time.
1.2 The application must display the 3D model reconstructed from the video stream captured by the camera within acceptable temporal delay.
1.3 The application must provide an interface (i.e. a button) for users to export the current 3D model in common file formats for further potential applications.
1.3.1 When a user employs the interface (clicks the button), the application must save the current 3D model properly in the local storage.

1.4 The application must provide an interface (i.e. a button) for users to change the backgrounds of the displayed 3D models.
1.4.1 When a user employs the interface (clicks the button), the application must change the background of the displayed 3D model.

3.2 Non-functional Requirements

1. The software must establish an internet connection between the server-end application and user-end application.
1.1 There must be instant interactions between the two ends in form of image stream, instruction and file transmission.

2. The program in the server must use AAM to reconstruct a 3D facial model with identical poses and shapes to the original 2D image and transmit it to the user-end terminal.

3. When the application is doing the training and cannot provide the 3D model immediately, it must display a picture showing that the user should wait for a second

4. If any source is inaccessible, the application must inform the user and let the user choose the appropriate action
4.1 If the camera is inaccessible, remind the user to check the camera
4.2 If the server does not response in 10 seconds, display a message saying that the server is not available

3.3 Architecture Requirements

3.3.1 Hardware Requirements

3.3.1.1 Basic Requirements:

CPU:	Intel Core™ i5 1.3Ghz or equivalent
Main memory:	512MB

3.3.1.2 Recommended Requirements:

CPU:	Intel Core™ i5 2.1Ghz or equivalent
Main memory:	2GB

3.3.2 Environment Requirements

OS: Windows 7/8/10

3.4 User Interface Requirements

Details in 4.3.

4 Requirements Analysis Models

4.1 Functionality (behavioral)

4.1.1 Use Cases

This system is divided into five use cases: Start Camera, Save Object, Change Mode, Quit, Error.
Each use case is specified in the use case specification document. Please refer to each use case specification document to get detail information to know more about each use case.

4.1.2 Functional Analysis Model

Here is the view of use case model.

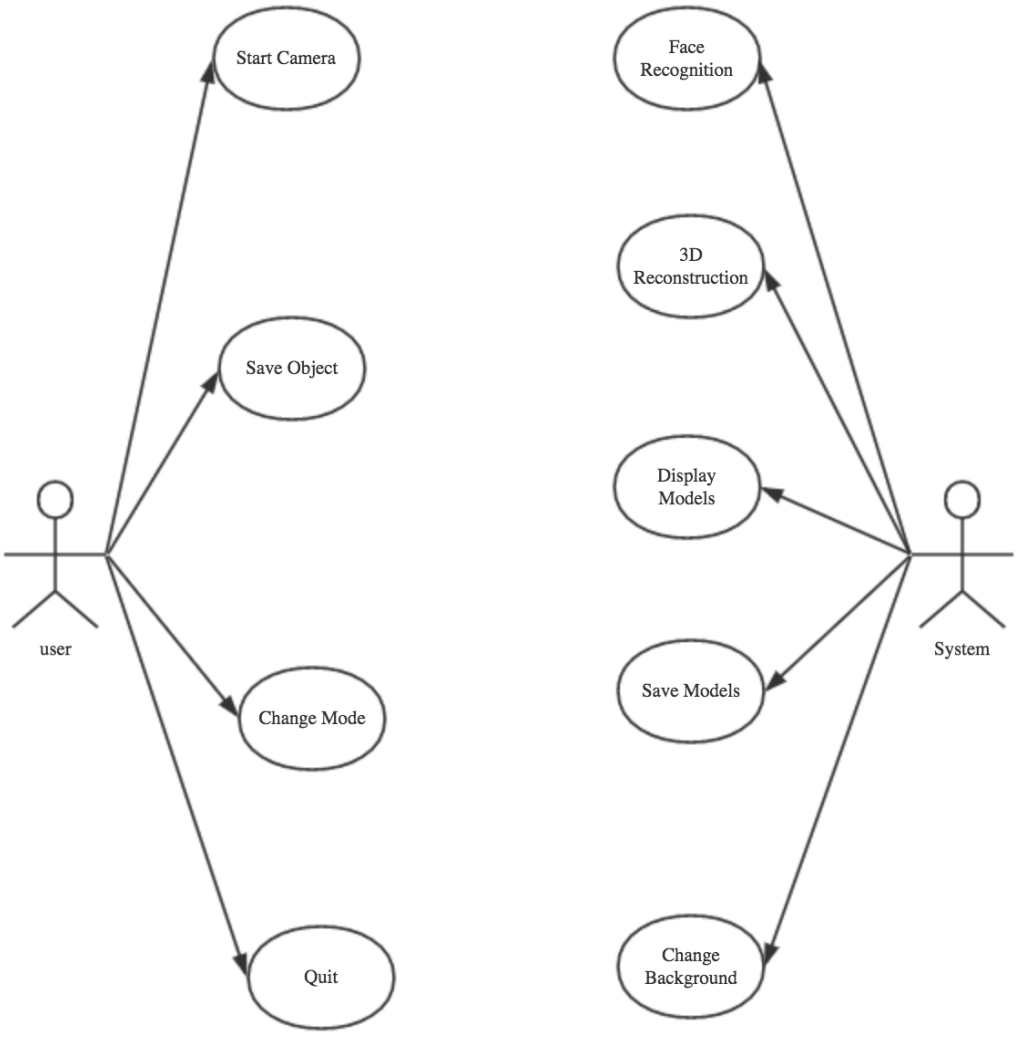


Figure 1. Use Case Diagram

4.2 Architecture & Bill of Materials

Seq	Item	Qty	Functional Description	Vendor/ Manufacturer	Product Name	Version	Part No.	Purchase Reuse Upgrade	Standard **(Y/N)**	CR #*	Remark
1	PC	High	Provide essential platform for the runtime of the software	Dell Inc.	Laptop	Ins14-7447	10001	R	N	2	
2	Server	High	Computing resources support	Unknown	Server	Unknown	10002	R	N	2	
3	Server-end Operating System	High	Platform for the server-end computing tasks	Ubuntu, America	Ubuntu	16.04 (LTS)	10003	R	N	3	
4	GPU	High	Hardware providing graphic computing resources	NVIDIA	Tesla Series GPU	Tesla K80	10004	R	N	2	
5	User-end Operating System	High	Platform to develop system	Microsoft, America	Windows 10	10 Professional	10005	U	N	3	
6	Office Tools	Middle	Software to edit documents, table, etc.	Microsoft, America	Microsoft Office	2016 Professional	10006	R	Y	1	
7	PDF Editor	High	Software to edit PDF documents	Adobe, America	Acrobat	Acrobat DC 2015	10007	R	N	2	

* If not a standard, indicates Standards Change Request #

4.3 User Interface

In the graphical UI, there are four buttons and two video frames:

- ✧ Start Camera: A button to open the camera and the connect with the server.
- ✧ Save Object: A button to save the general 3D model to the local device.
- ✧ Change Mode: A button to change the 3D Model's background
- ✧ Quit: A button to quit the software
- ✧ Capture Video: A video frame to show the 2D video stream captured by the webcam
- ✧ Reconstruction Model: A frame to show the reconstructed 3D model

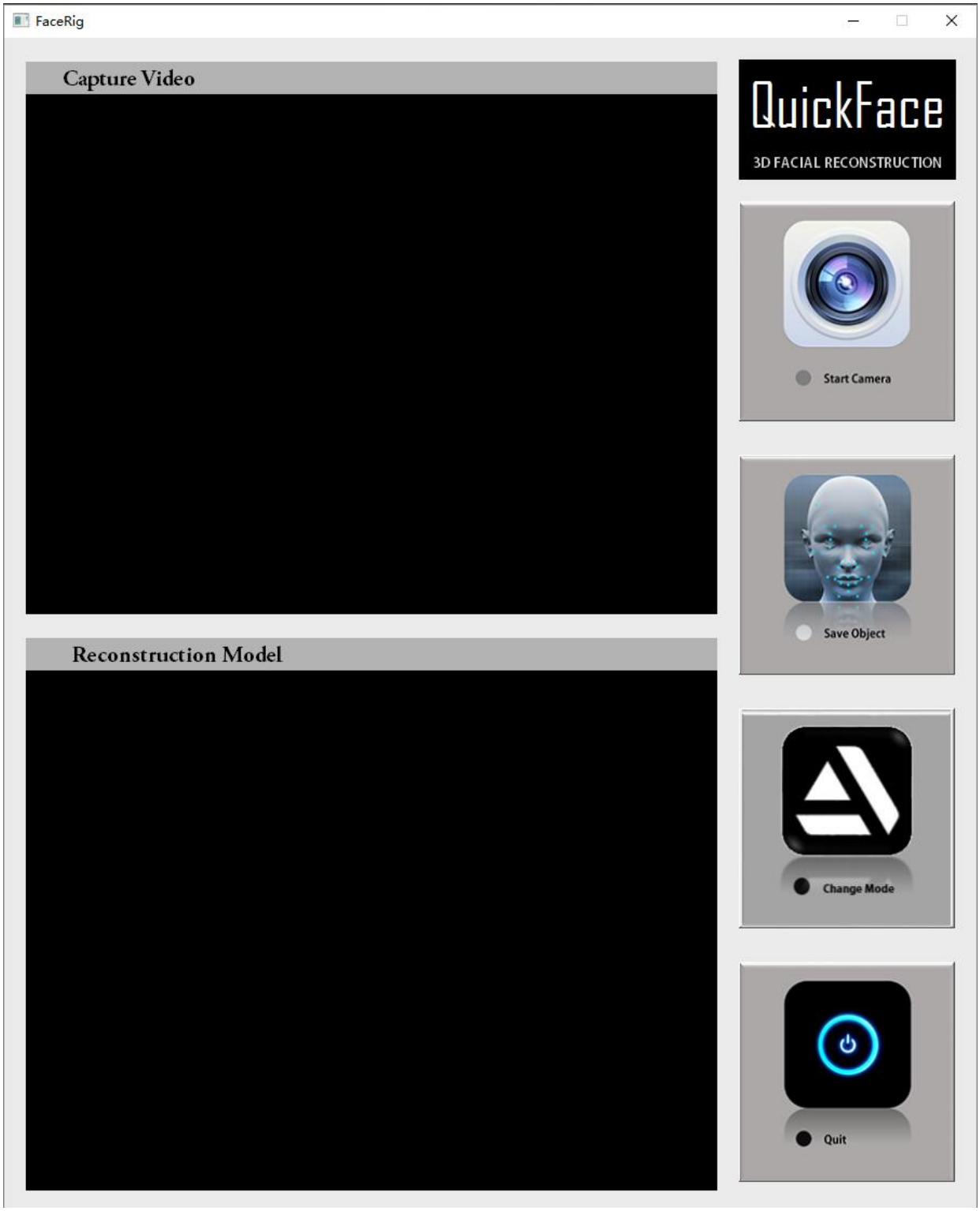


Figure 2. User Interface

5 Requirements Traceability

In phase of QuickFace developing, we will make three modules: Use-case Module, Analysis Module, and Design Module. So traceability exists among those modules.

In the UML diagram below, we describe the traceability as follows:

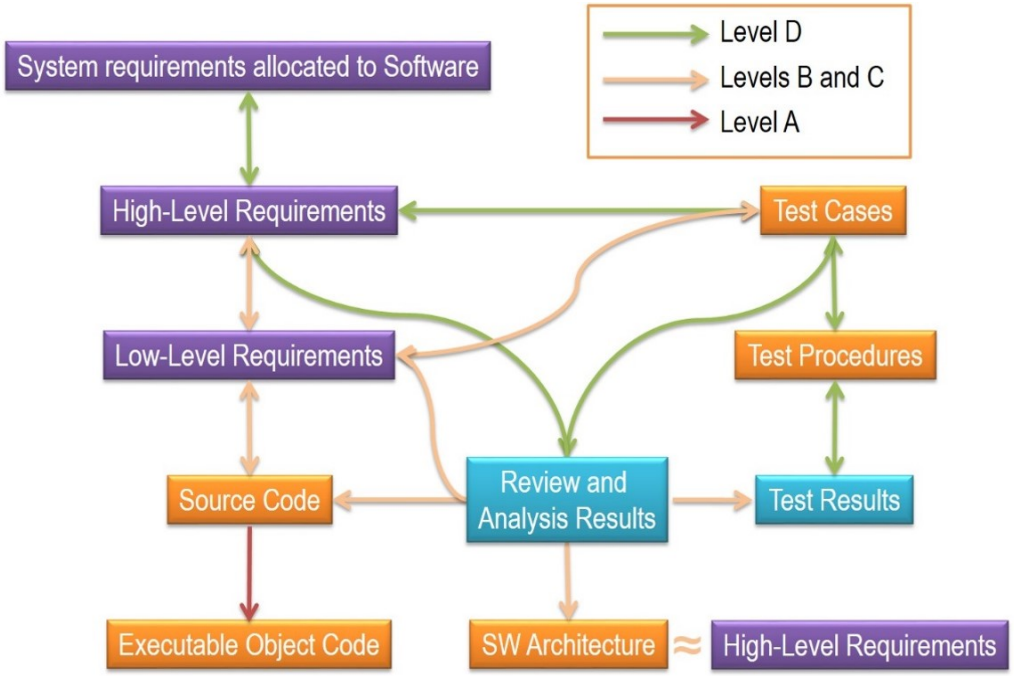


Figure 3. Requirements Traceability Diagram

DocNo: 001.D.1:1

Software Architecture Document

Version 2.0

Group Member:
庞博
包伟铭
刘子凡
Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-15	1.0	1st edition	刘子凡
2016-1-2	2.0	Final edition(Revised)	Us all

Key Word

QuickFace
Architecture
Design

Digest

This document is to describe the software architecture of QuickFace. It is the guideline of our QuickFace developing. In this document, we describe the detailed design ideas and readers can have a clear picture with reference to the design model.

Table of Contents

1 Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of QuickFace, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions, which have been made on the system.

1.2 Scope

This document should contain an overview of the architecture of QuickFace and explain how it should be modeled. Decisions in this document reflect how the system is modeled.

1.3 References

a) UseCaseSpecification_Start Camera.doc
b) UseCaseSpecification_Save Object.doc
c) UseCaseSpecification_Change Mode.doc
d) UseCaseSpecification_Quit.doc
e) UseCaseSpecification_Error.doc
f) SystemRequirementsDocument.doc

2 Architectural Representation

The architecture of the application is represented following the recommendations of the *Rational Unified Process* and the *Rational Architecture Practice* guidelines. And this document presents the architecture as a series of views:

a) Use Case View
b) Logical View
c) Process View
d) Implement View
e) Deploy View

3 Use-Case View

A description of the use-case view of the software architecture. The Use Case View is an important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. The functionality of QuickFace is captured in the use-case diagram below:

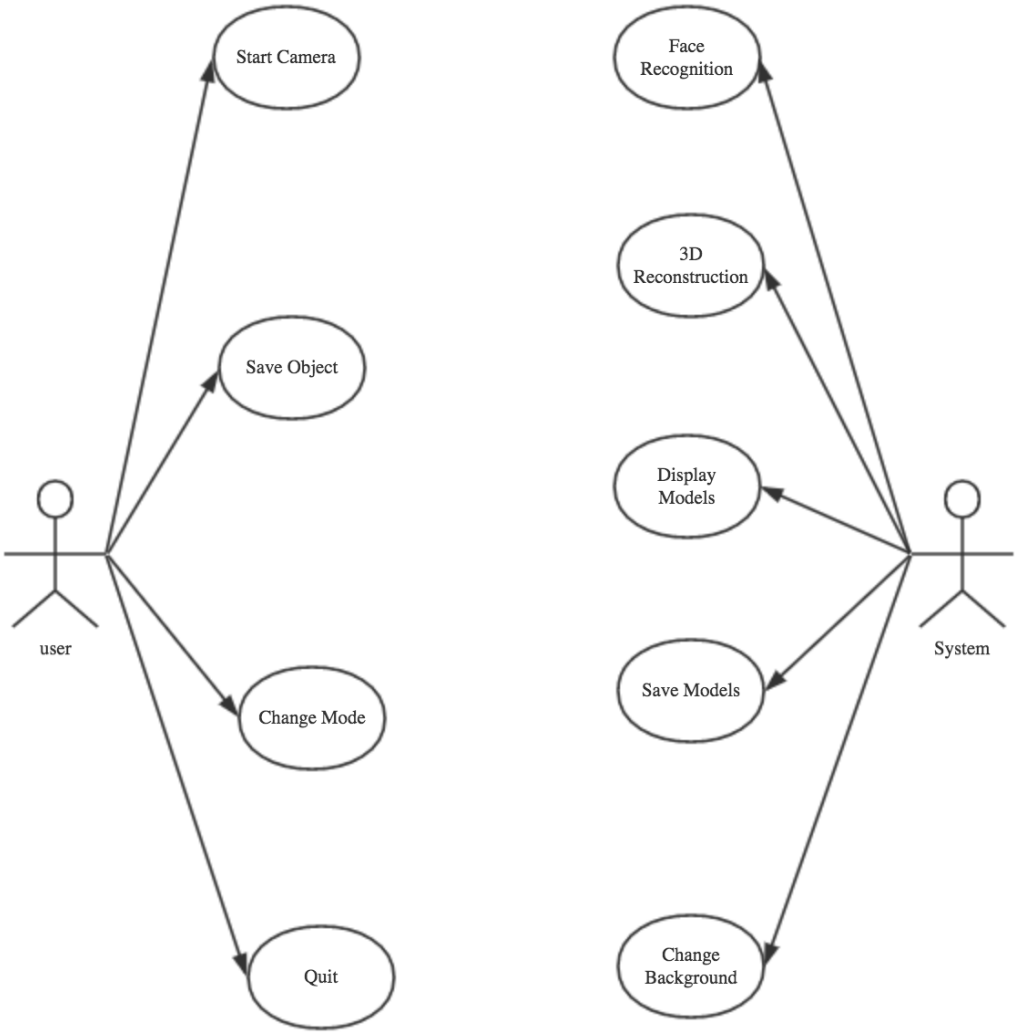


Figure 1. Use case diagram of QuickFace

All implemented use cases have an associated Use Case Specification document. References to these documents can be found in the same directory.

3.1 Architecturally-Significant Use Cases

The architecturally-significant use cases are those, that “exercise” the most critical parts of the system architecture and demonstrate the core system functionality. In this system, they are:

Start Camera: This use case allows users to input live facial images through the camera. When the application is launched and the camera is accessible, this use case starts.
Basic Scenarios:
a) The user uses the camera to input live facial images.

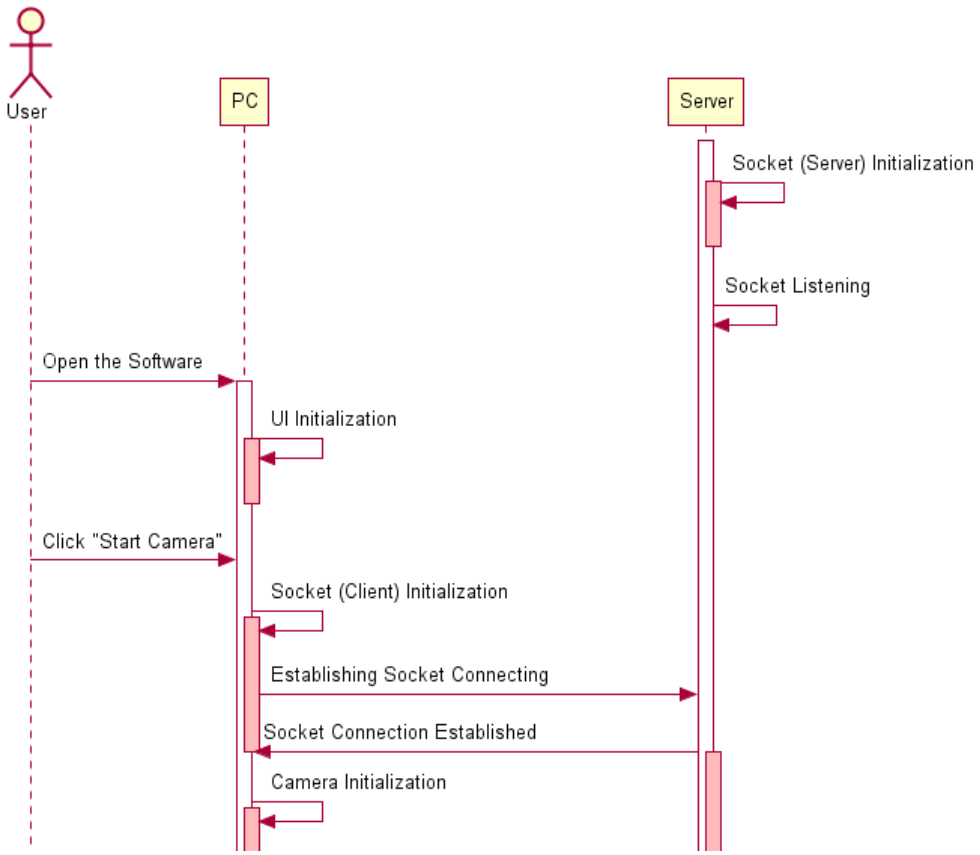
b) System receives the image and checks if the image contains sufficient facial information.

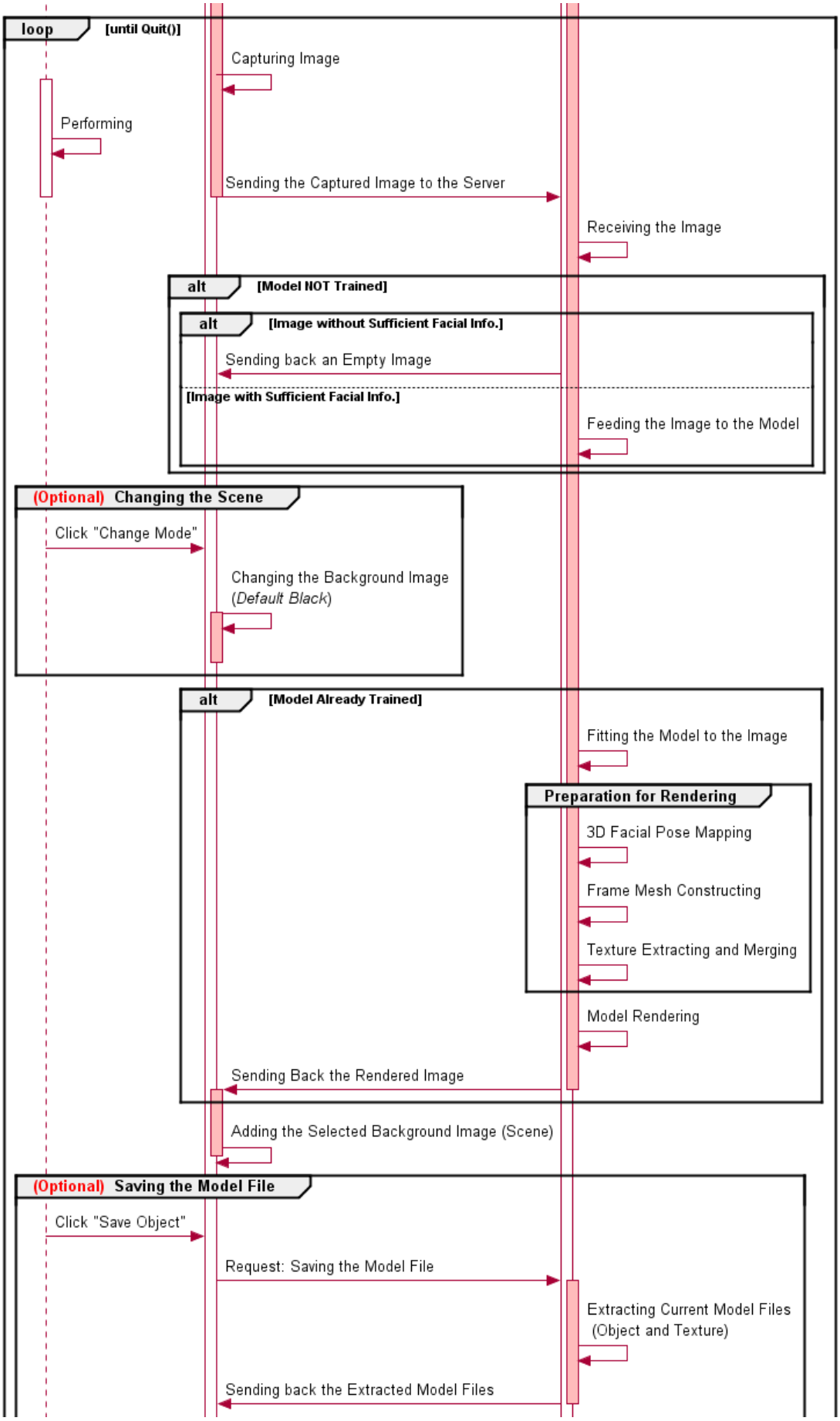
Save Object: This use case allows users to save the current reconstructed 3D facial model.
Basic Scenarios:
a) The user clicks the ‘Save Object’ button.
b) The software saves the current 3D model in a proper place in local storage.

Change Mode: This use case allows users to change the background image of the reconstructed 3D facial model.
Basic Scenarios:
c) The user clicks the ‘Change Mode’ button.
d) The software changes the background scene of the reconstructed 3D model.

Quit: This use case allows users to elegantly quit the software.
Basic Scenarios:
e) The user clicks the ‘Quit’ button.
f) The software quits with all process terminated properly.

The system architecture use cases are illustrated below as a whole in form of UML Sequence diagram.




loop
[until Quit()]
Capturing Image
Performing
Sending the Captured Image to the Server
Receiving the Image
alt
[Model NOT Trained]
alt
[Image without Sufficient Facial Info.]
Sending back an Empty Image
[Image with Sufficient Facial Info.]
Feeding the Image to the Model
(Optional) Changing the Scene
Click "Change Mode"
Changing the Background Image
(Default Black)
alt
[Model Already Trained]
Fitting the Model to the Image
Preparation for Rendering
3D Facial Pose Mapping
Frame Mesh Constructing
Texture Extracting and Merging
Model Rendering
Sending Back the Rendered Image
Adding the Selected Background Image (Scene)
(Optional) Saving the Model File
Click "Save Object"
Request: Saving the Model File
Extracting Current Model Files
(Object and Texture)
Sending back the Extracted Model Files

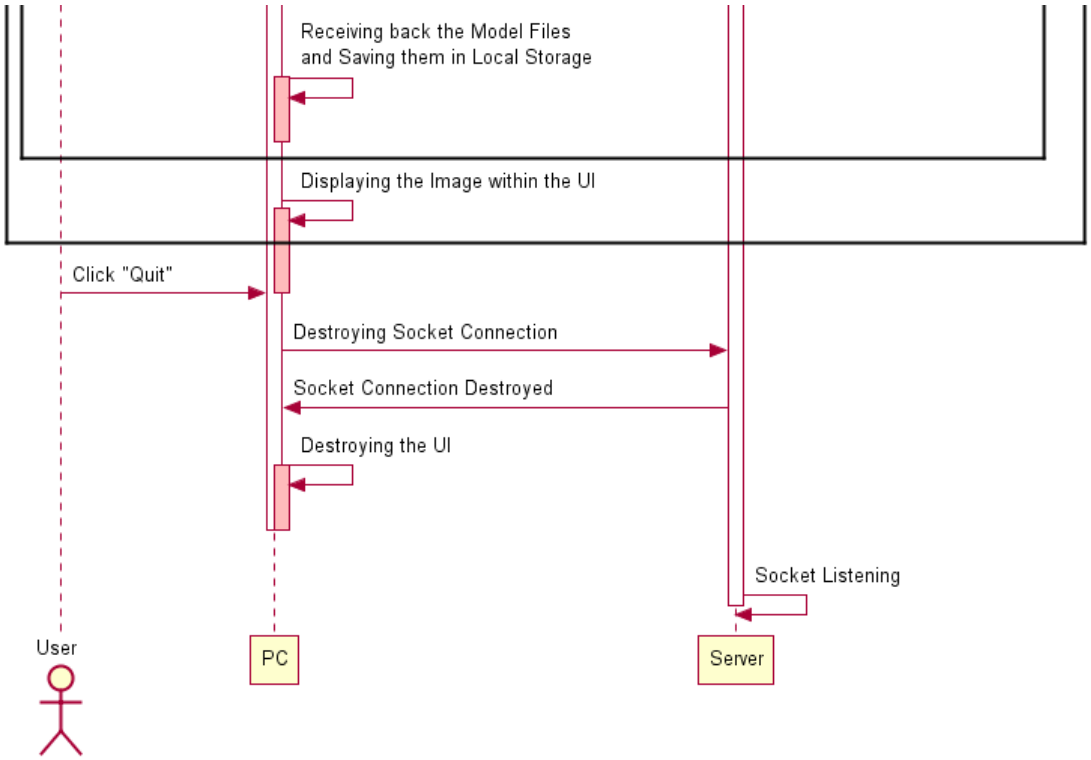


Figure 2. Sequence diagram of QuickFace

4 Logical View

This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. It describes the logical structure of the system. It starts from the overview of the architecture and then presents its key structure, behavioral elements and mechanisms.

4.1 Overview

There are three dominant structures in the application design model:

a) Logical decomposition of the system into three layers.
b) The structure of the use case realizations derived from model templates of architectural mechanisms.
c) The design mechanisms package contains a pre-designed solution to a common problem.

The high-level diagram of above is showed below:

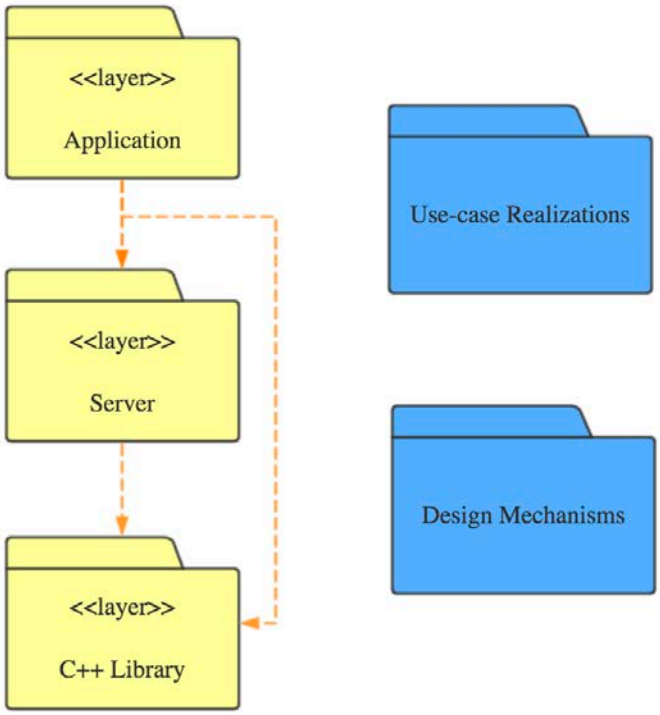


Figure 3. High-level layer diagram of QuickFace

The three layers are introduced below:

a) C++ library: Including some C++ Library functions we need to call in our programs.
b) Server: Functions which should be run on the server.
c) Application: This layer aims at special logics. It has a close relation to the presentation logics. The boundary classes are contained in this layer.

4.2 Architecturally-Significant Model Elements:

4.2.1 Application:

Socket: Establish internet connection between the user-end and the server-end applications.

4.2.2 Server:

QuickFace: Reconstruct 3D facial models from 2D video stream imput.

4.2.3 C++ library:

OpenCV, Boost and Eigen.

5 Process View

The process view deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behaviors of the system. The process view addresses concurrency, distribution, integrity, performance, and scalability.

The activity diagram is showed below:

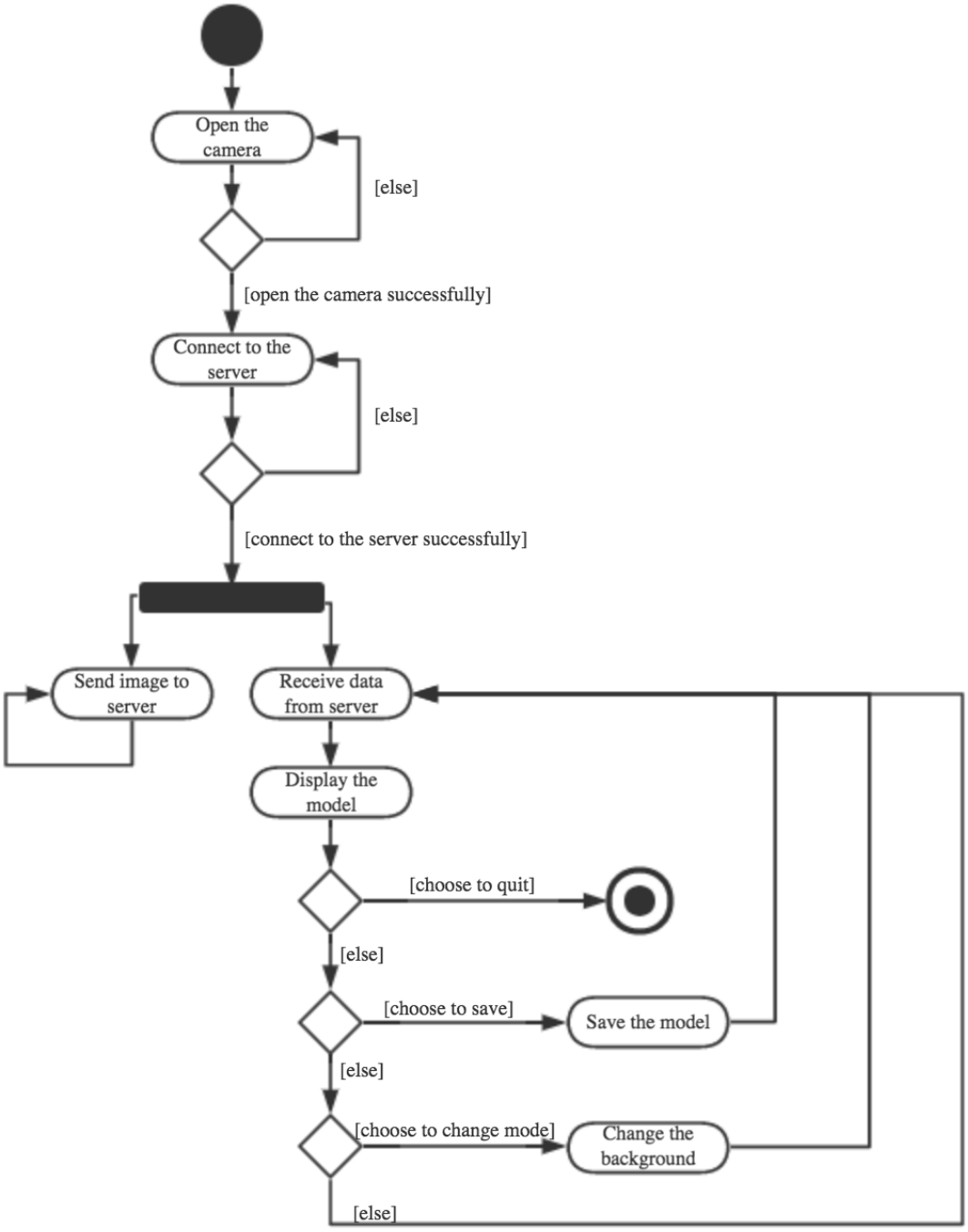


Figure 4. Activity diagram of QuickFace

6 Deployment View

The deployment view of a system shows the physical nodes on which the system executes and the assignment of the system processed to the nodes. Our application is deployed on PCs.

The diagram below shows the most typical deployment configuration used by the development team.

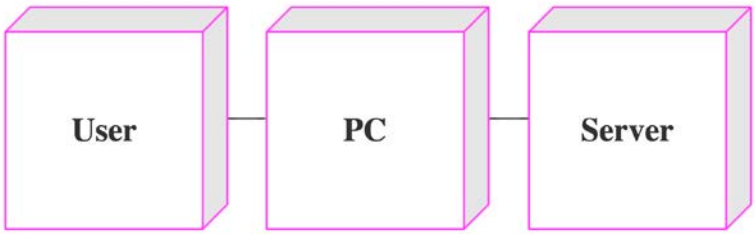


Figure 5. Deployment diagram of QuickFace

As to the User node, four operations are allowed including “Start Camera”, “Save Object”, “Change Mode” and “Quit”, which compose of the outmost level of the software architecture where users’ intents are launched.

As to the PC node, it composes of a graphic UI which integrates and responds to the users’ operation. Four buttons associated with corresponding operations and two video frames for the input raw video stream and the real-time reconstructed 3D facial model respectively are the user oriented components. It also interacts with the server via internet connection by launching instructions, sending and receiving data. The PC-end also catches and responds to the expected invalid operations and errors.

As to the Server node, it provides computing resources and serves to support the PC front end application. Computation-heavy tasks, such as facial information detection, 3D facial model fitting and constructing, facial texture extracting and graphic matrix rendering, are conducted on the server-end. It interacts with PCs via internet connection.

7 Implementation View

The development view illustrates a system from a programmer's perspective and is concerned with software management. This view is also known as the implementation view. It uses the UML Component Diagram to describe system components.

The component diagram is shown below:

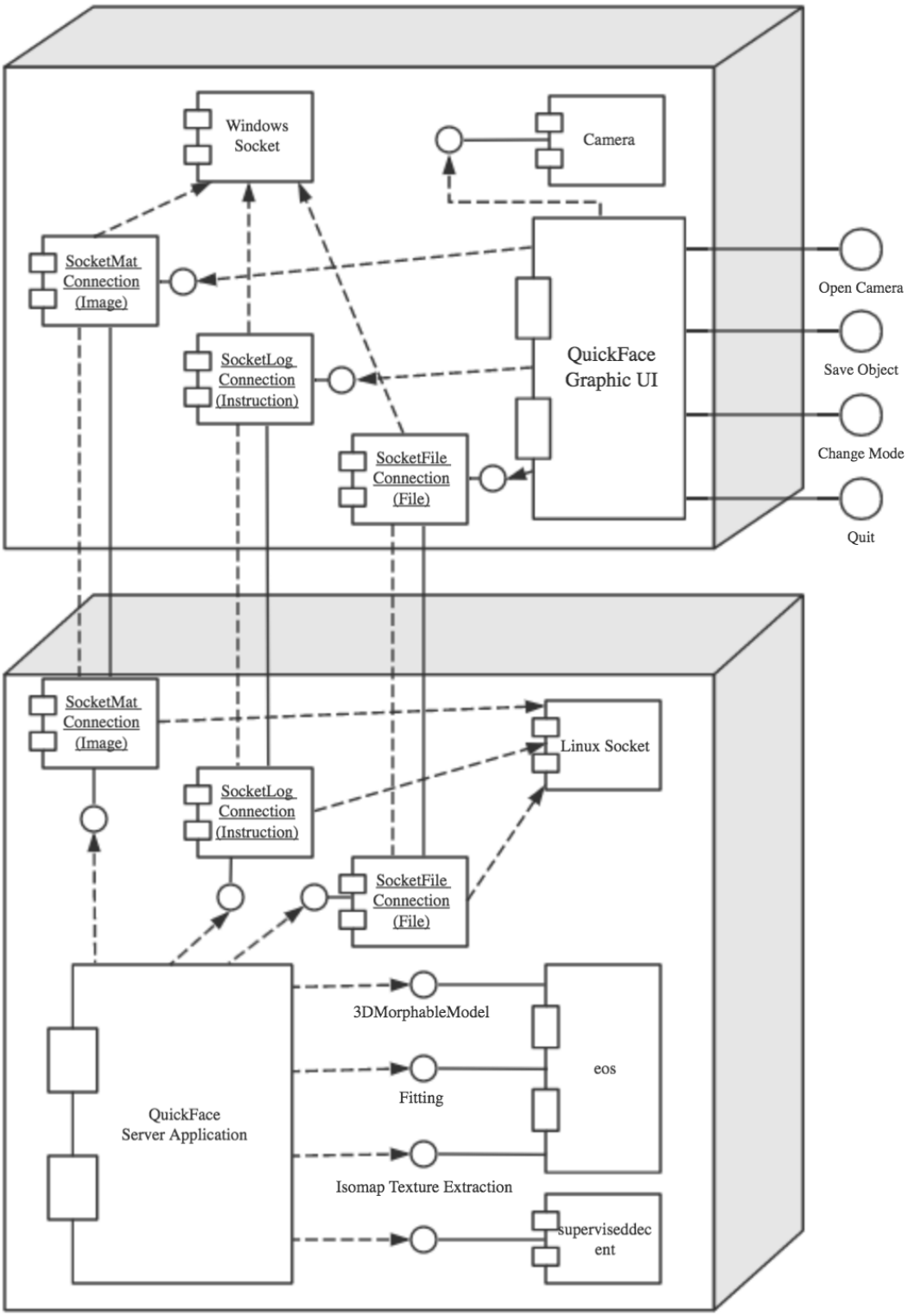


Figure 6. Component diagram of QuickFace

DocNo: 001.C.1:1

Start Camera
Use Case Specification
Vision 1.0

Group Member:
庞博
包伟铭
刘子凡
Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-15	1.0	Use case specification of Start Camera	刘子凡
2014-01-2	1.1	Unitize the format	刘子凡

Table of Contents

1. Definition

This is the requirement description for the Start Camera use case. Start Camera use case is for user to open the camera and start the real-time 3D animation.

2. Preconditions and Post Conditions

2.1 Preconditions

The user has opened this software.

2.2 Post Conditions

When the camera is open and the real-time animation is running, this use case is over.

3. Basic Scenarios

1) User press on the Start Camera button.
2) System opens the camera and start the real-time 3D animation.

4. Exceptions or Branches

System opens the camera and start the real-time animation when the camera is accessible. If there is no camera device accessible, the system will pop up an error message and remind the user to check his camera device.

5. Note

Null.

DocNo: 001.C.2:1

Save Object
Use Case Specification
Vision 1.0

Group Member:
庞博
包伟铭
刘子凡
Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-15	1.0	Use case specification of Save Object	刘子凡

Table of Contents

1. Definition

Save Object use case is for user to get the current 3D facial model of.

2. Preconditions and Post Conditions

2.1 Precondition:

The camera is opened and a 3D model is displayed on the screen.

2.2 Post Condition:

The software will save the current 3D model in a proper place.

3. Basic Scenarios

a) The user clicks the Save Object button.
b) The software saves the current 3D model in a proper place.

4. Exceptions or Branches

If the camera has not been opened or the 3D model has not been generated, the software will pop out an error message to tell the user that there is not object to save.

5. Note

Null.

DocNo: 001.C.3:1

Change Mode
Use Case Specification
Vision 1.0

Group Member:
庞博
包伟铭
刘子凡
Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-15	1.0	Use case specification of Change Mode	刘子凡

Table of Contents

1. Definition

This is the requirement description for the Change Mode use case. Change Mode use case is for user to change the background of the displayed 3D model.

2. Preconditions and Post Conditions

2.1 Preconditions

The camera is opened and a 3D model is displayed on the screen.

2.2 Post Conditions

The background of the displayed 3D model has been changed and this case ends.

3. Scenarios

a)User click the Change Mode button.
b)The system change the background of the 3D model displayed.

4. Exceptions or Branches

If the camera has not been opened or the 3D model has not been generated, the system will pop out an error message tells the user that the background cannot be changed until a 3D model is displayed.

5. Note

Null.

DocNo: 001.C.4:1

Quit
Use Case Specification
Vision 1.0

Group Member:
庞博
包伟铭
刘子凡

Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-15	1.0	Use case specification of Quit	庞博

Table of Contents

1. Definition

This is the requirement description for the Quit use case. Quit use case is used for users to quit the software.

2. Preconditions and Post Conditions

2.1 Preconditions

The system can be under any active condition during runtime.

2.2 Post Condition

The software disconnects with the server and quits.

3. Basic Scenarios

1) The user finishes the job, and the software can be under any condition.
2) Software quits and the processes terminate.

4. Exceptions or Branches

Null.

5. Note

After quit request sent to the software, it will disconnect with the server. The server will listen to other clients.

DocNo: 001.C.5:1

Error
Use Case Specification
Vision 1.0

Group Member:
庞博
包伟铭
刘子凡
Document Language:
English

Revision History

Date	Version	Description	Author
2016-11-15	1.0	Use case specification of Error	刘子凡

Table of Contents

1. Definition

This is the requirement description for the Error use case. Error use case is for system to throw an error message.

2. Preconditions

a)The camera is not accessible.
b)The server is not accessible.
c)The software is not successfully executed.

3. Post Conditions

The software goes to homepage.

4. Scenarios

1) The system throws an error message.
2) The system goes to homepage.

5. Exceptions or Branches

The system will give a warning and the user should try again.

6. Note

Null.

DocNo: 001.I.1:1

Test Analysis Report

Version 2.0

Group Member:
庞博
包伟铭
刘子凡

sDocument Language:
English

Revision History

Date	Version	Description	Author
2016-12-15	1.0	1st edition	刘子凡
2017-1-2	2.0	Final edition(Revised)	Whole team

Key Word

QuickFace
Test
Test case
Analysis Report

Digest

This document is to record the test phase. In this document, there is the test analysis report with the test cases and suggestions for the project.

Table of Contents

1 Introduction

1.1 Purpose

This document is our test analysis report for QuickFace, which illustrates the details of test result according to the test context, test scope, test standard design in the test plan document. This document will be the main reference for our testing. Therefore, the readers for this document are mainly the testers and the project manager of QuickFace.

1.2 Background

The system tested is named as “QuickFace”, short for quick generation of facial 3D models. The QuickFace is used for recognizing faces and reconstructing 3D models. The whole project began at November 1st. After requirement analysis, system designing, and coding. the next step is testing. After coding out the system and our testers master the testing knowledge and skills, we can do our test.

1.3 Defination

a) QuickFace: face recognition and reconstruction system
b) Black box testing: A test method, which testers only pay attention to input and output.
c) White box testing: A test method, which testers must know the inside instruction of test object.
d) Stub module: When taking unit testing and integration testing, the test object needs to call other unit, then stub module can take instead of the called unit.
e) Driven module: When taking unit testing and integration testing, the test object needs to make active by others, then driven module can take instead of the caller.
f) Test script: A small test program for testing to call unit or be called by unit.
g) equivalence partition: A test method in black box testing. It uses a set of values selected, instead of many input value, which are dealt with in the same way.
h) boundary designing: It is the extension of the equivalence partition, usually it is the boundary of equivalent class.
i) causation graph: When considering the relationship of each input, causation graph can show the combinations of all inputs and outputs.
j) Unit testing: Test on the smallest unit such as class in the software.
k) Integration testing: Test on the combination of several units to check if they can work together.
l) Regression testing: In integration testing, some integrations must be test again

to check if they can work with other integrations.

m) System testing: Compared with requirement definition, look for some parts which are not coincident with the requirement.

n) Stress testing: Test if the system can afford heavy using stress.

1.4 Reference

Software Testing Ron Patton
QuickFace Software Architecture Document

2 Test Overview

Function	Input	Output
Start Camera	A click on the ‘Start Camera’ button	System will try to establish socket connection with the server and open the camera on the device. If the socket connection is failed or the camera is not accessible, an alert window containing corresponding error messages will come out. If successful, then the system will display the image captured by the camera and the received reconstructed 3D model.
Save Object	A click on the ‘Save Object’ button	System will save the current model files in a proper place in local storage.
Change Mode	A click on the ‘Change Mode’ button	System will change the background of the displayed 3D model.
Quit	A click on the ‘Quit’ button	System quit elegantly with all connections destroyed and all processes and services terminated.

3 Test Result and Findings

3.1 Function Test

The tested functions covers all use cases designed.

All these functions run normally without any unhandled exceptions or unexpected crashes.

3.2 Testcase Result

3.2.1 Start Camera

Test Name	Action	Expect Result	Actual Result
Start Camera 1	Click the ‘Start Camera’ button	Program goes well.	Fit expectation.
Start Camera 2	Disable the camera and click the button	An error message comes out to tell the user that the camera is inaccessible.	Fit expectation.
Start Camera 3	Click the button when the server is not accessible	An error message comes out to tell the user that the server is inaccessible.	Fit expectation.

3.2.2 Save Object

Test Name	Action	Expect Result	Actual Result
Save Object 1	Click the ‘Save Object’ button when a 3D model is displayed on the screen	Program goes well and an object is saved in the proper place.	Fit expectation.
Save Object 2	Click the ‘Save Object’ button when the 3D model hasn’t been generated	An error message comes out to tell the user that the 3D model hasn’t been generated.	Fit expectation.

3.2.3 Change mode

Test Name	Action	Expect Result	Actual Result
Change Mode 1	Click the ‘Change Mode’ button when a 3D model is displayed on the screen	Program goes well and the background of the displayed 3D model is changed.	Fit expectation.
Change Mode 2	Click the ‘Change Mode’ button when the 3D model hasn’t been generated.	An error message comes out to tell the user that the 3D model hasn’t been generated.	Fit expectation.

3.2.4 Quit

Test Name	Action	Expect Result	Actual Result

Quit 1	Click the ‘Quit’ button	The software quits.	Fit expectation.

4 Analysis Abstracts

4.1 Capability

The Tested QuickFace have the following functions:

a) Open the camera after the user clicks the ‘Start Camera’ button

b) Display the image captured by the camera

c) Construct a 3D model and display it

d) Change the background of the displayed model after the user clicks the ‘Change Mode’ button

e) Quit the User-end application elegantly after the user clicks the ‘Quit’ button

4.2 Flaws and Limitations

4.2.1 Some existed problems

a) Efficiency
The application depends on a server to do the computation. When the network condition is poor, the performance will suffer.

b) Quality
Since the constructed 3D models only depend on 2D inputs, so sometimes the result is not very accurate, which highly rely on the quality of the first image with sufficient facial information.

4.2.2 Some unrealized functions

a) Face replacement
With the expressions captured, we can replace the face with another face, such as faces of some celebrities or some cartoon characters and compose the poses and shapes onto the mimic face.

b) Hair and ears
In our model, the faces have no hair or ears.

4.3 Suggestions

Here are some experiences we got during our tests

a) Add more annotations to the source list.
b) Make the interface more user-friendly

4.4 Evaluation

The QuickFace can run in a correct and efficiency way, and its basic functions do work according to its user’s guide.

5 Test Cost

No special cost is spent but developing human resources.

Conclusion

Our software QuickFace has a great performance to construct a 3D face model from a 2D video stream captured by webcam. And the software provides an interface to save the general 3D face model to the user's device for further uses. And there is an interface for the user to change the background of the 3D model.

The design style of our software is minimalist which is user friendly and easy to use for everyone. The graphical buttons are intuitionistic making it easy to understand the function of the button.

In order to achieve a great efficient, we separate the software to two ends—client end and server end, the main computations are in the server, so that the software has an excellent performance.

After several testing and debugging, the software has a great robustness, and the current version V1.0.0 is the first stable version.

We provide an installation package for Windows x86, so that it is easy to install our software without any system configuration.

Reference

[1] ALDRIAN, O., AND SMITH, W. A. P. 2013. Inverse rendering of faces with a 3D Morphable Model. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 5, 1080–1093.

[2] BLANZ, V., AND VETTER, T. 1999. A Morphable Model for the synthesis of 3D faces. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques(SIGGRAPH), ACM Press/Addison-Wesley Publishing Co., 187–194.

[3] CAO, C., BRADLEY, D., ZHOU, K., AND BEELER, T. 2015.Real-time high-fidelity facial performance capture. ACM Trans. Graph. 34, 4 (July), 46:1–46:9.

[4] HUBER, P., HU, G., TENA, R., MORTAZAVIAN, P., KOPPEN,W. P., CHRISTMAS, W., RATSCH ¨ , M., AND KITTLER, J. 2016.A multiresolution 3D Morphable Face Model and fitting framework. In International Conference on Computer Vision Theory and Applications (VISAPP).

[5] ICHIM, A. E., BOUAZIZ, S., AND PAULY, M. 2015. Dynamic 3Davatar creation from hand-held video input. ACM Trans. Graph.34, 4 (July), 45:1–45:14.

[6] MAIER, R., STUCKLER ¨ , J., AND CREMERS, D. 2015. Super resolution keyframe fusion for 3D modeling with high-quality textures. In 2015 International Conference on 3D Vision, 3DV2015, Lyon, France, October 19-22, 2015, IEEE, 536–544.

[7] SHEN, J., ZAFEIRIOU, S., CHRYSOS, G. G., KOSSAIFI, J., TZIMIROPOULOS,G., AND PANTIC, M. 2015. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In 2015 IEEE International Conference on Computer Vision Workshop, ICCV Workshops 2015, Santiago, Chile, December7-13, 2015, IEEE, 1003–1011.

[8] HUBER, P., CHRISTMAS, W., HILTON, A., KITTLER, J., & RÄTSCH, M. 2016. Real-time 3D face super-resolution from monocular in-the-wild videos. In ACM SIGGRAPH 2016 Posters (p. 67). ACM.

Thanks Prof. Sheng, T.A. Saleha Masood and Zhezhou Cheng for the careful guidance and helps.

We have benefited a lot from this project.

Bo Pang
Weiming Bao
Zifan Liu
2017.1.7