Design Document for
Better Graphics For A Robotics Grasping GUI

Shady Robots

Group 12

Justin Bibler

Matthew Huang

Daniel Goh

CS461: Senior Software Engineering Project

Fall 2016

December 1, 2016

Abstract: Our customer is using a simulation to create visuals that are used for online data collection. This simulation is using outdated libraries which result in outdated graphics. Design definitions outlined in this document will be used to accomplish our customer's request. The request being to update the simulation's graphics with warm cool shaders, shadows and silhouettes.

Keywords: OpenInventor, OpenGL, OpenRave, shaders, warm cool shaders, silhouettes, shadows, robotic simulation, geometry, visualization, render, vertex lines

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to elaborate on the design and logic of how we will be implementing our requirements.

1.2 Scope

The scope of this document is solely about how new features will be implemented and how those implementations will be tested.

1.3 Context

Currently, our client is using visualizations, of a robot hand grasping objects, to collect data online. These visualizations are created from a simulation program (OpenRave); the data collected is used to create a model of the human grasp. However, the current graphics in the simulation are outdated. It is hard to see and understand the shapes and contact points represented in the scene.
The context of this document is focused primarily on the visuals of the project.

1.4 Summary

In clauses 1 to 3, Matthew Huang outlines the design of Gooch shading, silhouettes, and run-time analysis with CodeXL respectively. In clauses 4 to 6, Justin Bibler outlines the design of shadow volumes, performance benchmarks using FRAPS [1], and methods of code maintainability. In clauses 7 and 8, Daniel Goh outlines the methods and guidelines to create the online survey, and methods to analyze and visualize the collected data.

REFERENCES

[1] Fraps, “Fraps.” [Online]. Available: http://www.fraps.com/

[2] F. D. M. Pizka, “How to effectively define and measure maintainability.” [Online]. Available: http://www.itestra.de/fileadmin/Redaktion/Documents/07testradefineandmeasuremaintainability.pdf

[3] Google, “Google forms.” [Online]. Available: https://www.google.com/forms/about/

[4] S. Monkey, “Writing good survey questions.” [Online]. Available: https://www.surveymonkey.com/mp/writing-survey-questions/

[5] Google, “Google sheets.” [Online]. Available: https://www.google.com/sheets/about/

[6] S. Monkey, “Making sense of the numbers: When to embrace relativity, and when to ignore it.” [Online]. Available: https://www.surveymonkey.com/blog/2012/06/28/making-sense-numbers-when-embrace-relativity-when-ignore-it/

2 GLOSSARY

FRAPS

FPS

Frequency

3 REQUIREMENT: SHADOWS

By Justin Bibler

3.1 Software Design Description

Our client requires us to add shadows into the simulation's graphics. The addition of shadows will improve the overall graphics of the simulation, which will help a user better understand where an object is within a 3D space.

3.2 Design Stakeholders

Justin Bibler, Matthew Huang, and Daniel Goh

3.3 Design Concerns

All stakeholder share similar design concerns.
Shadows design concerns:

1) Will our shadow implementations have a major impact on FPS.
2) Will our shadows be too pixelated.
3) Will users be able to better understand where an object is because of our shadow implementation.

3.4 Design Views

3.4.1 Interface Design

This section is dedicated to talking about the graphical quality of our shadow implementation.

3.4.1.1 Design Concerns

Interface design concerns:

1) Will users be able to better understand where an object is because of our shadow implementations.
2) Will our shadows be too pixelated.

3.4.1.2 Design Elements

Name

Shadow Interface.

Type

Visual, or Graphic.

Purpose

Shadows with the simulation exist to better help users understand an objects location within a 3D space.

Relationship

The quality of the visuals rely heavily on the design discussed later within the Algorithmic implementation of shadows.

Interface attribute

Depending on the implementation in the algorithm viewpoint, every 3D model within the simulation should have a shadow. Whether or not the shadow is visible is dependent on if there is a surface that the shadow volume collides with. These shadows should have minimal pixelation. In addition, the shadows should be relatively close in shape to their original models, as to not confuse what shadow is projected out from what object. All of this is done so that the users will easily be able to understand what shadow is related to what object, and to receive a better understanding of the 3D environment in the simulation.

3.4.2 Algorithm Viewpoint

The idea of the shadow volume algorithm is to extend the silhouettes of objects that are in contact with light sources into volumes. And whenever another object falls within the shadow objects volume, it is rendered in a different way. [**?**]

3.4.2.1 Design Concerns

Algorithm design concerns:

1) Will the algorithm be costly to the simulation's FPS.

3.4.2.2 Design Elements

Name

Algorithmic implementation of shadows.

Type

Algorithm, or method.

Relationship

Shaders, and the Shadow Interface.

Design Constraints

The implementation of shadows will most likely rely on the knowledge we gain from implementing warm cool shaders into the simulation. That knowledge will allow us to better understand how the simulation is being rendered, making it easier for us to adjust and change certain shaders. Another constraint that we have is that there is a relatively small amount of documentation related to OpenRave and OpenInventor. This makes it difficult to find exactly where we need to implement shaders to get our desired results.

Processing Attribute

The bases of the algorithm will be using the depth fail test. This will be done by first rendering all the objects in the scene into the depth buffer. From here, we create shadow volumes for each object based on the objects silhouette projected into infinity relative to where the light is striking the object. Then we render the volume into the stencil buffer based off the depth fail test.

Depth fail test:

1) If the depth test fails while rendering back facing polygons of the shadow volume, we increment the stencil buffer.
2) If the depth test fails while rendering the front facing polygons of the shadow volume, we decrement the stencil buffer.
3) Nothing happens if the test passes, or the stencil test fails.

This depth fail test is conducted for each shadow volume. Once the tests have concluded, the scene will be rendered into the viewport. However, only the objects, or parts of objects, that have a stencil buffer value of zero will be rendered without alterations. [**?**]

4 Requirement: Measuring the Simulation's FPS

By Justin Bibler

4.1 Software Design Description

Our project requires us to measure and analyze the runtime and FPS of our altered simulation. This is done so that we can ensure that our additions fulfill our performance metrics: the simulation must maintain an average 30 FPS. If we are unable to reach this metric our implementations will be considered incorrect, and will require revisions.

4.2 Design Stakeholders

Justin Bibler, Matthew Huang, and Daniel Goh

4.3 Design Concerns

All stakeholders share similar design concerns.
Requirement design concerns:

1) Failing to meet our target FPS and runtime will result in revisions of our implementations.
2) Will our measurements be accurate for the majority of computers.
3) Software used for measuring may cost money.

4.4 Design Views

4.4.1 Information Viewpoint

The purpose of this viewpoint is describe how information will be gathered, and how it will be analyzed. Our team will be using an external tool to obtain data related to the simulation's FPS. FRAPS will be used to examine the FPS and export the information into a file to be analyzed.

4.4.1.1 Design Concern

Interface viewpoint concerns:

1) FRAPS is paid software.
2) Will our measurements be accurate for the majority of computers.

4.4.1.2 Design Elements

Name

Method of analyzing and collecting FPS data.

Type

Method.

Purpose

Data collected will be analyzed to ensure that our implementations into the simulation meet our requirements.

Data attribute

FRAPS will be used to capture data related to FPS, this will be done by using the benchmarking tools that come with the software. We will run this test for one minute ten times, on five different sample simulations with our updated graphics. This data will be exported into a file, from here we will analyze the data by finding the mean and median across all executions. If these values are within a certain standard deviation of 30 FPS, our implementation will be considered correct.

Author

FRAPS is owned by Beepa.[1]

4.5 Rational

Even though FRAPS is a paid software I've decided that it be best if we use it. It's ability to export data into a file allows our group to easily obtain and analyze data. Furthermore, our concern related to the validity of the data stems from the fact that our group members own above average computers. Thus, testing on our computers would most likely be inaccurate relative to the average computer. To circumvent this problem we plan to do most of our testing on the computers on the OSU campus.

5 Requirement: Code Maintainability

By Justin Bibler

5.1 Software Design Description

The goal behind code maintainability is to increase the cost effectiveness of changes being made to a system. [2] Our group has made this a requirement because our client wishes to be able to easily change, or add into our implementations. The primary logic behind this design is the use of good programming practices such as documentation, and commenting, but also the use of external resources which are described within the viewpoint below.

5.2 Design Stakeholders

Justin Bibler, Matthew Huang, and Daniel Goh

5.3 Design Concerns

All stakeholders share similar design concerns.
Requirement design concerns:

1) Cost of static code debugging tools.
2) Lack of documentation for preexisting code.
3) Lack of comments for preexisting code.

5.4 Design Views

5.4.1 Resource Viewpoint

External resources will be used to help us maintain and upkeep our code quality. The first resource we will be using, which within today's standard is almost a necessity, will be GitHub. GitHub is a repository that is used for code version control. Secondly, we will be using Cppcheck. Cppcheck is a static code analysis tool that examines code for bugs that a compiler may not catch.

5.4.1.1 Design Concern

Interface viewpoint concerns:

1) Using Cppcheck may bring about problems within preexisting code not written by us.

5.4.1.2 Design Elements

Entities

Both GitHub, and Cppcheck are free to use by the public

Name

Resources for Code Maintainability.

Type

External resource.

Function

GitHub will be used for version control of the code. Cppcheck will be used to assist in finding and fixing bugs within our code implementations.

Resource attribute

GitHub is important because it will allow us to have easy access to past versions of code. Using GitHub is like a safety net that will allow us to be free with how we edit the code, because in the case that we create a massive new problem; it is very easy for us to pull an earlier version of the code to work on. Along with this GitHub also has a built in wiki that will be used to record and document our new implementations and functions within the code. Allowing for easy reference for ourselves and our client.

Cppcheck will be used to examine our new implementations into the code. Using this tool will allow us to quickly find and eliminate bugs within the code. Few amount of bugs within code will allow us to make changing and creating new implementations safer and easier.

Author

GitHub is owned by GitHub, Inc. Cppcheck is open source.

5.5 Rational

The reason Cppcheck was decided upon was because it is open source and free, other static analysis tools cost a significant amount of money. Which makes the Cppcheck option the best for our group. Also, the design behind ”Code Maintainability” can be improved by the use of the resources I talked about. However, a large part of it will rely more on how well our group documents the improvements we make through resources such as GitHub. It’ll be important while we go forward to document both how we implemented our improvements, and how to use our improvements. This is because, as mentioned in the Requirement concerns, we are working off preexisting code. Thus, our client will need some sort of good reference of how to navigate the code, and how to make proper adjustments.

6 Requirement: Creation of Online Survey

By Daniel Goh

6.1 Software Design Description

This requirement is established to allow participants to provide their input to determine if the team's visual enhancement to the robotic simulation has improved from its initial state. This will be fulfilled by running online surveys. Google Forms is the selected platform to run the online survey.

6.2 Design Stakeholders

Justin Bibler, Matthew Huang, and Daniel Goh

6.3 Design Views

6.3.1 Interface Viewpoint

6.3.1.1 Design Concern

Google Forms is the selected service that will be utilized to carry out the online survey. The design concern for the online survey will include if participants will be able to understand the questions of the online survey and select their choice of better visuals.

6.3.1.2 Design Elements

Name

Survey Interface Understandability

Type

System used to assess resulting visuals for the robot grasping simulation.

Purpose

This element exists to help guide the creation of an online survey that is easily understood and usable by the participants.

Function

Google Forms [3] is a matured survey platform that allows unlimited number of questions and the function to attach images to the created survey. Participants will be able to select a picture or answer based on the questions presented. The questions will cover the aesthetic aspects between pictures, the presence of shadows between pictures, and identification of object relativity or position within the picture.

Interface

Methods of interaction provided by Google Forms include:

- Allowing participants to view attached images for a question
- Allowing participants to select a single answer or image for a question
- Allowing participants to select multiple answers or images for a question (Check PLES)
- Participants responses will be stored as a data file within Google Forms and exportable to Google Sheets(.csv files, comma separated value files)

Users should not be overwhelmed by information in the interface while taking the survey. Survey questions should ask the user for a specific opinion (e.g. questions asking about presence of shadows should only focus on asking about shadows and nothing more) at a time and ideally have no more than one sentence [4]. Survey questions should be designed to be balanced and not biased. For example, stating the left visual is the improved visual in the question while asking the user to select the better visual is a form of bias.

7 Requirement: Analysis and Visualization of Collected Data

By Daniel Goh

7.1 Software Design Description

The data collected from the online survey will require analysis to determine if the team's visual enhancement to the robotic simulation has improved from its initial state. The data will need to be organized and analyzable by the stakeholders. Google Sheets will be used to analyze and visualize the collected data.

7.2 Design Stakeholders

Justin Bibler, Matthew Huang, and Daniel Goh

7.3 Design Views

7.3.1 Information Viewpoint

7.3.1.1 Design Concerns

Google Sheets [5] is the selected platform to parse and visualize the data collected from the online survey. The design concern includes if the data collected can be analyzed and visualized meaningfully to reflect the collective participants responses.

7.3.1.2 Design Elements

Name

Data Visualization

Type

System for data parsing and visualization that supports Google Forms

Purpose

The element exists to help guide the use of Google Sheets to create data visualizations.

Data Attribute

As Google Forms and Google Sheets are under one ecosystem, Google Sheets have access to parse the data collected directly into Google Forms. The resulting data content will contain individual responses on a single spreadsheet. Questions asked will be represented on the first horizontal row. Participant's responses are time stamped and is represented as a row for each individual. The participant's responses (answer to each questions) are recorded below the questions.

TABLE 1
Spreadsheet example with data

Respondent	Time stamp	Question 1	Question 2	...	Question N
1	11/12/2016 20:28:39	Choice 1	Choice 2	...	Choice 2
2	11/13/2016 1:55:37	Choice 1	Choice 2	...	Choice 1
3	11/13/2016 11:56:02	Choice 2	Choice 2	...	Choice 2
4	11/15/2016 23:52:12	Choice 1	Choice 2	...	Choice 1
5	11/18/2016 15:51:32	Choice 1	Choice 1	...	Choice 2

The selected data analysis method that will be used to determine the project's success will be based on frequency among all respondents [6]. An example model that represents the frequency method is as follow:

$$\frac{NumberOfChoice1OccurrenceForQuestion1}{TotalRespondents} = x\% \tag{1}$$

In this example, Choice 1 for Question 1 is the visuals with enhanced shadows implementation. x will need to be more than 79 in order for shadow implementations to be considered a success.
The data collected will be converted into visualizations using the charting functions provided in Google Sheets. For ease of understandability and quick visualization, the pie chart charting option will be used to display the collective responses.

8 Conclusion

In conclusion, this design document will be used as a plan of action as we go forward with our implementation. This document is not set in stone; it may change as needed as we begin making changes to the code. However, it is important to note that our requirements will not change; only our approach to tackle the requirments may change.