Using the Peppers Ghost Pyramid technique to create real-time holograms for a charades style game

Final Report for CS39440 Major Project

Author: Elliot Oram (elo9@aber.ac.uk)
Supervisor: Dr. Helen Miles (hem23@aber.ac.uk)

28th February 2017
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name .....

Date .....

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name .....

Date .....

Acknowledgements

I'd like to thank Dr. Helen Miles for her continued support throughout the duration of this project which included advise and guidance on the project context and implementation. In addition I would like to thank the members of the Aberystwyth University computer science department for their assistance in setting up the stall for my prototype demonstration at the 2017 Aberystwyth Science week.

Abstract

This report describes the process followed to develop a system to create real-time holograms using the Pepper's Ghost pyramid technique and an accompanying charades game. The system was first proposed to be used at Aberystwyth university's Science week in 2018, but is also suitable to be used at any appropriate outreach events. The Peppers ghost pyramid technique is an excellent tool to use in outreach as it is both simple to understand how it works, and creates an impactful display. In order to improve the outreach experience for participants, the charades game is designed to provide a way to interact with the holographic system.

This report will present the purpose, technologies and processes surrounding this system. The development process followed an adapted single person FDD plan driven methodology which will be discussed in detail below. The report will also mirror the methodology and present development tasks and their completion relative to software iterations which took place on a weekly basis. Furthermore, there will be a focus on the software quality and testing performed throughout the development of the process.

Finally, the report will conclude with a critical evaluation of the system itself, the implementation choices and the development processes and methodologies. This will include but not be limited to the programming languages and frameworks used, continuous integration, source control and testing tools, and prototyping and implantation testing.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

This section should discuss your preparation for the project, including background reading, your analysis of the problem and the process or method you have followed to help structure your work. It is likely that you will reuse part of your outline project specification, but at this point in the project you should have more to talk about.

Note:

- All of the sections and text in this example are for illustration purposes. The main Chapters are a good starting point, but the content and actual sections that you include are likely to be different.
- Look at the document on the Structure of the Final Report for additional guidance.

1.1 Background

1.1.1 Pepper's Ghost Pyramid

What was your background preparation for the project? What similar systems did you assess? What was your motivation and interest in this project?

The Pepper's Ghost technique was originally used for stage and theatre productions in the Victorian era to display holographic illusions to the audience. The technique, discovered by Dr. Henry Pepper, was first used in theatre in xxxx. The holograms being visible is reliant on lighting and the viewer being positioned correctly relative to a transparent surface. Figure 1 shows a basic example of how the technique produces holographic illusions. The object of interest is the focus of the lighting and, in this implementation, is out of the line of the sight of the audience. The light bounces from the object of interest and travels radially until it makes contact with the surrounding surfaces. The majority is absorbed by the surrounding black walls, but the remainder will reach the angled transparent plane. The plane is angled at a 45 degree offset to the object of interest. The plane manipulates the light particles, by refraction, to display them at a right angle to their angle of entry. The audience are positioned at 90 degrees from the object of interest and therefore, when the light is refracted, they see the object of interest appear in front of them.

This technique has been used in different some differing applications such as [arcade machine]

but it has had a more recent resurgence as an impactful display media in computer visualisation. Whilst the technique still uses Henry Pepper's original concept, it has been modified to display the hologram from multiple angles. This technique is now more commonly known as the Pepper's Ghost pyramid as it uses a transparent pyramid rather than a single plane. Figure 2 shows an example of a Pepper's Ghost pyramid. The pyramid is square based, made from a transparent material (such as perspex or clear acrylic) and is open at both the top and bottom. The technique for displaying holograms differs with the object of interest now being an image or video displayed on a screen. Furthermore, to create an illusion from all sides of the pyramid, four images are required (one for each side of the pyramid). This means that the 2D projection of the image can be seen from any side of the pyramid. Figure 3 shows this in more detail.

Whilst the pyramid implementation is an improvement on Pepper's original design it still suffers from the some of the same limitations of the original. The original design is reliant on the viewer looking directly at the transparent plane which makes it both vertically and horizontally intolerant for the viewer. The pyramid implementation means that the hologram is visible from four perspectives (in front, behind, left and right). Whilst this is an improvement, the hologram is not easily viewed from the edges of the pyramid and does not affect the vertical intolerance as viewing from above is not possible. Some concepts for a spherical design (instead of a pyramid) are proposed to resolve the horizontal intolerance completely.

1.1.2 Charades game

TODO

1.1.3 Motivation and Justification

This project is appealing due to the uniqueness of the technology being used to produce it. Whilst the charades game can be considered as a generic system, the creation of real-time holograms is not heavily developed. Many examples of video footage that can be used with the Pepper's Ghost Pyramid are readily available online, however there are far fewer implementation that do real time manipulation for this purpose.

Further to the project being interesting, it is designed with outreach events in mind. Already having an intended purpose means that it can be used as a helpful teaching tool for children to learn how basic physics and computer science can be used to create an interesting and engaging visual display. In addition, the charades game and real-time hologram system will help to further engage the audience with technique and visualise an impactful application of the technique.

1.2 Analysis

Taking into account the problem and what you learned from the background work, what was your analysis of the problem? How did your analysis help to decompose the problem into the main tasks that you would undertake? Were there alternative approaches? Why did you choose one approach compared to the alternatives?

There should be a clear statement of the objectives of the work, which you will evaluate at the end of the work.

In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was determined to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

As mentioned in the lectures, think about possible security issues for the project topic. Whilst these might not be relevant for all projects, do consider if there are relevant for your project. Where there are relevant security issues, discuss how they will this affect the work that you are doing. Carry forward this discussion into relevant areas for design, implementation and testing.

1.3 Process

The development of this project followed the Feature Driven Development (FDD) plan driven methodology. FDD is normally considered for larger projects as it provides a framework for distributed development. By dividing developers into smaller teams, FDD allows those teams to tackle features one at a time in parallel. Furthermore, the up front planning stage is generally indicative of projects that are more stable as, whilst it can be adapted throughout the process, the overall model is normally only added to, and the core architecture remains static.

The steps required to complete this project are well defined and therefore would be well suited to having up front design. Furthermore, FDD encourages continuous integration (CI) which offers a good way to produce a functional prototype at various stages of the project. CI will be a significant aid for the mid project review and potentially testing a prototype with users at the 2017 Aberystwyth University science week event.

1.3.1 Single Person FDD adaptation

To successfully use FDD for this project, several adaptations to the normal processes of the methodology will need to be made. The most notable is the abolition of the developer teams in favour of a single developer. This requires the developer to assume multiple roles throughout the development process and at each different stage.

1.3.1.1 Develop an Overall Model

Initially, a Domain Expert (Customer) is required to aide in the development of the overall model and feature creation. In this context, the project supervisor can fulfil this role and the developer shall act as both the Chief Programmer and Chief Architect. For this project, the Domain specific language is shared by both the Domain Expert and the developer.

1.3.1.2 Build a Feature list

The feature list produced will form the requirements list for the project. This will be generated from discussion between the developer and the Domain Expert and then when complete will be verified by both parties.

1.3.1.3 Plan by Feature

Steps such as establishing developer teams and scheduling developer teams time throughout the project no longer exist. In their place, the features can be given priorities that will aide in time scheduling and development order. Once the order is created, the features can be assigned to separate iterations.

1.3.1.4 Design by Feature

Once a feature is selected, that feature will be exhaustively designed taking into consideration the functions required to fulfil the feature. The project uses GitHub for version control and an issue will be created which corresponds to a feature. The first action in an issue will be to complete a design of the feature and update the overall design as required.

1.3.1.5 Implement by Feature

Features will be implemented in the same GitHub issue as the design work. The project is being developed using TDD and therefore, the test suite will be updated before any code is implemented. Once the tests are created, an implementation is then added and this must pass the tests to be acceptable. Whilst the tests can be run locally, there will also be a Jenkins service for continuous integration to run the full test suite before it can be merged with the master branch.

To allow for a code review/walk through, completed features are to be left as open pull requests for a day before reviewing the code to ensure good quality code is created for each feature.

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected.

The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation.

Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here.

How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor.

You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on "real users"? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the 'future conversations'. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library - The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [1]. The library is released using the Apache License [2]. This library was used without modification.

Appendix B

Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

Appendix C

Code Examples

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Final Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [3].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)
```

```
double ran2(long *idum)
{
  /*-------------------------------------------------*/
  /* Minimum Standard Random Number Generator        */
  /* Taken from Numerical recipies in C              */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity   */
  /* Always call with negative number to initialise  */
  /*-------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
    *idum += IM1;
  }
```

```
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```

Annotated Bibliography

[1] Apache Software Foundation, "Apache POI - the Java API for Microsoft Documents," http://poi.apache.org, 2014.

This is my annotation. I should add in a description here.

[2] ——, "Apache License, Version 2.0," http://www.apache.org/licenses/LICENSE-2.0, 2004.

This is my annotation. I should add in a description here.

[3] W. Press *et al.*, *Numerical recipes in C*. Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content.*

[4] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, "Don't touch me, I'm fine: Robot autonomy using an artificial innate immune system," in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

[5] H. M. Dee and D. C. Hogg, "Navigational strategies in behaviour modelling," *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

This is my annotation. I should add in a description here.

[6] Various, "Fail blog," http://www.failblog.org/, Aug. 2011, accessed August 2011.

This is my annotation. I should add in a description here.

[7] S. Duckworth, "A picture of a kitten at Hellifield Peel," http://www.geograph.org.uk/photo/640959, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.