COMPSCI 715
Advanced Computer Graphics

Writing about Design and Implementation

Today's Mission

1. How do you make well-informed design decisions?
2. How does a good design section look like?
3. How does a good implementation section look like?

Typical Research Paper Structure

1. **Introduction**: What is the research problem? Introduce and motivate it. Summarize your contributions.
2. **Related Work:** What have others done? How is it different? Cite, summarize other solutions & compare it with your own.
3. **Design:** Your solution. Describe it in enough detail so others can implement / replicate it. Software architecture (e.g. class diagram)? User interface (e.g. screen diagram)? Algorithms?
4. **Implementation:** How have you implemented your solution? Tools and technologies used? Implementation challenges?
5. **Evaluation:** Explain the methodology you used for evaluation. Present the results. Discuss them.
6. **Conclusion**: Summarize contributions. Point out future work.

What is Design?

How do you solve your research problem
or answer your research question?

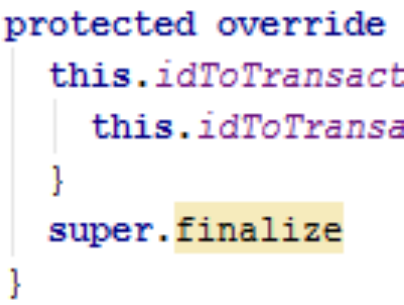
- Top-down: start with the overall idea, then drill down to the details
  - What are the overall ideas used to solve the problem?
    -> Reader should get a complete picture first
  - What are the contributions (the new bits)?
    -> This is most interesting so should be described in all detail
- Explore the design space of your project analytically
  - Scope: What are the possibilities? Alternatives?
  - Analyze: What are their advantages? Disadvantages? Limits? Sweet spots?
- Justify your design decisions with arguments, examples, experimental data, related work, heuristics, proofs...

Writing about Design (2-6 pages 2-column)

Describe your solution with details about contributions:

1. Create appropriate illustrations, e.g.
   user interface diagrams / screenshots, architecture diagrams, algorithm pseudocode, data models, formulas…
2. Start with an overview of the solution
   a. Add figure to outline the overall setup, data flow, etc.
   b. Describe the figure to start the section
3. Write subsections for all solution parts, esp. the contributions
   a. Describe the part (using appropriate illustrations) so that reader could implement it (reproducibility)
   b. Justify the design decisions by discussing the reasons, alternatives, advantages, disadvantages, …

Implementation



How did you build your system?

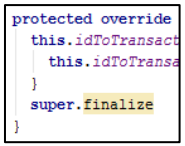
- Usually a comparatively small part (less than a page)
- What tools/technologies were used?
- Implementation challenges and how you solved them
  - What was hard? Why? Solutions?
  - Small code snippets for illustration
- Limitations: often your implementation is just a prototype
- How could you help others do the same? What information would have been useful for you when you started coding?

Design vs. Implementation

Design (Solutions)

- Abstract, high-level
- Mostly independent of particular technologies, and tools
- Could be applied in many different projects, implemented in many different systems
- Consists of the contributions

Implementation (Prototype)



- Concrete, low-level
- Build using particular technologies and tools
- Just one example of how the solutions (esp. the contributions) were applied
- Merely used to illustrate / demonstrate contributions

Good Implementation Example

https://www.cs.auckland.ac.nz/~lutteroth/publications/VanDykEtAl2012-GLDebug.pdf

GLDebug's probe was implemented using BuGLe (see Section 3.1) as a basis. The complexity and time requirements of implementing a debugger from scratch are significant. Using BuGLe as a basis greatly decreased the time required to develop the probe and implement the ability to capture OpenGL commands and state. However, BuGLe still had to be extended to meet the needs of GLDebug, e.g. with functionality for logging and sending information about OpenGL commands.

All communication between the probe and the controller is done through a single TCP connection. This allows the probe to run on the same system as the controller or on another system, as required. The communication is primarily initiated by the controller [...]

The data store was implemented using a temporal triple store [... which] makes it possible to access any of the previously stored OpenGL states [...]

The controller was implemented using Java, while the probe had to be coded in a lower-level language (in this case C) in order to be compiled into a shared library. This separation was helped by the fact that both components communicate over a remote interface based on TCP, as explained earlier. [...]

Exercise

Learn from good and bad example design sections

1. Read them (note: examples are shortened)
2. Identify the following parts:
   a. Solution overview
   b. Solution details
   c. Justifications of design decisions

3. Discuss:
   What are the good and bad points?
   How could it be improved?

Good Design Example 1

https://www.cs.auckland.ac.nz/~lutteroth/publications/VanDykEtAl2012-GLDebug.pdf

GLDebug is designed based on several high-level components, as shown in the architecture diagram in Figure 1. The OpenGL application is the program being debugged. It is executed on top of …

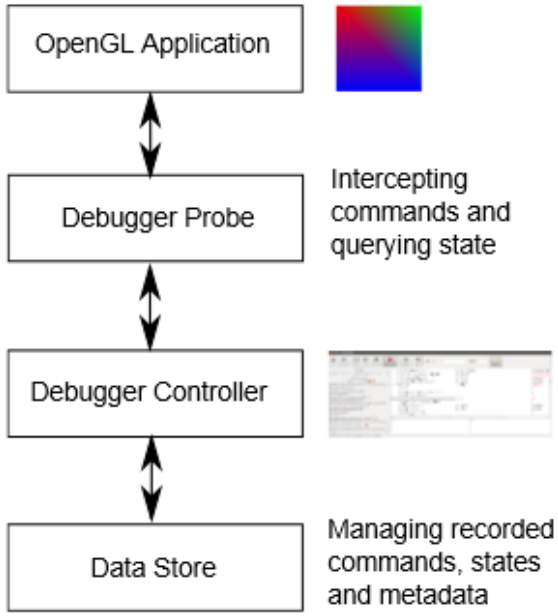


Figure 1: Architectural overview of GLDebug.

4.2 Controller

The controller is responsible for controlling the running OpenGL application and retrieving information about it through the probe […]

Figure 2 shows the controller GUI. The buttons at the top allow users to connect to a running probe and influence the control flow of the application being debugged, i.e. start, pause, stop and step through it. [...]

Note that the table on the right shows two graphics states, one in the left column and one in the right column. Differences in these two states are highlighted using color coding [...] This allows users to quickly compare two graphics states.

Good Design Example 2

https://www.cs.auckland.ac.nz/~lutteroth/publications/PenkarLutterothWeber2013-NavigatingHypertextWithGaze.pdf

When dwelling on a hyperlink, a confirm button is activated in the right margin. The user has to look at this button to click the hyperlink. A screenshot of the implementation is shown in Fig. 2 and the state machine in Fig. 3. This design tries to control inadvertent clicking by providing a confirmation step. The confirm button has a darkening border to show time progression during dwell.

Fig. 2. Single Confirm: User navigating to "UNESCO World Heritage Site"

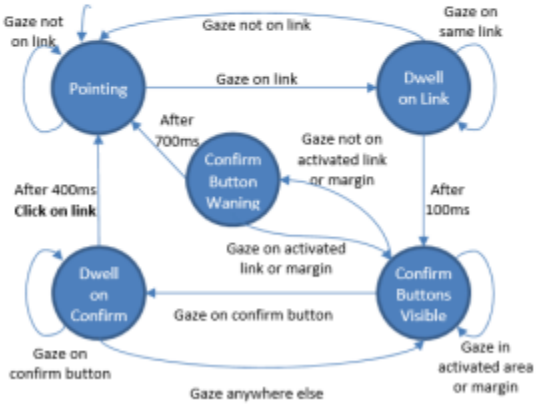


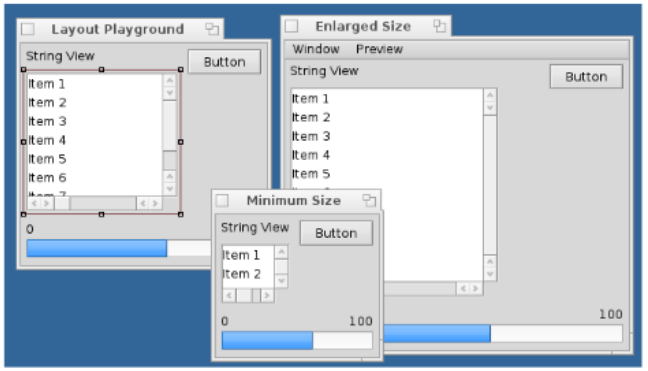
Fig. 3. State machine for Single Confirm click alternative

The initial design had the button appearing right next to the line with the hypertext being looked at, in order to minimize gaze travel. A light grey line extended from the button to the edge of the hyperlink to disambiguate among multiple links present in the same line. This was changed after the pilot, as outlined in Section 4.

[...] text was placed outside the buttons to avoid inadvertent clicking and a crosshair was added in the center of buttons to provide a visual anchor, in accordance with earlier findings [9]. More text was made visible in button labels since some links were not disambiguated easily by only the first five letters.

Good Design Example 3

https://www.cs.auckland.ac.nz/~lutteroth/publications/ZeidlerEtAl2013-AucklandLayoutEditor.pdf



PREVIEW OF RESIZE BEHAVIOR

[...] we introduce the idea of optimal resize previews. The goal of these previews is that the user is able to see the resize behavior of the layout quickly, using minimal screen real estate. Such an optimal resize preview shows the window both at the minimum size and at another size, with a clearly visible size difference relative to the construction window for every widget or constraint, so that the user can see whether the layout specification is correct.

The minimum layout size shows the layout in the most compact form. To determine this size, the hard constraint [...] One option for a larger size would be the preferred layout size. However, this is non-optimal, as this size may not significantly differ from the minimum size in all aspects. Thus, we show a second preview that is enlarged enough to enable the user to easily see all size changes relative to the construction window. The just noticeable difference between two simultaneously visible line segments is less than 5 percent of the length [21]. To ensure that our resize visualization is above this threshold, we [...]

Bad Design Example 1

We designed an interface with a 3D object to be modeled. The 3D object can be rendered as a solid object or wireframe (with or without Cull facing). We coded an algorithm for the subdivision of triangular meshes to render the required levels of details. We also implemented a brush for the different types of operations like Push, Pull/Inflate, Draw, Smooth and other functions on the object.

I decided to work with the mesh subdivision and mesh deformation part in the development of the tool. To implement the mesh Subdivision, I used loop Subdivision proposed by Charles Loop as it deals with the subdivision process of triangular meshes. From research I found out that loop subdivision is considered to be one of the best algorithms when one is working with adaptive subdivision of triangular meshes. I am implementing loop subdivision using JOGL. For Mesh Deformation, I will be using UpdateGeometry() class of JAVA 3D.

Bad Design Example 2

User interface design

MDI (multiple documents interface) is used in this project. When the program is running, by default it is shown like this. The user then can select options from the menu. When the Document Queues option from the View menu is selected, a child window is shown. This form is basically how it looks like after interface redesign and this project mainly deals with this form. When an envelope is selected, a window pops up showing the messages in the envelope. When an message is selected, another window pops up showing the message data in the message.

Database design

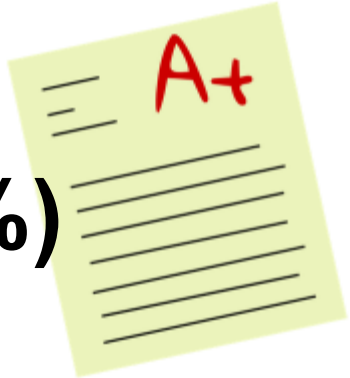
The diagram above shows the tables that are relevant to the Message Queue project. One Partner may have one or many inbound/outbound envelopes. One inbound envelope may have 0 or one or many inbound messages. One outbound envelope may have 0 or one or many outbound messages.

Bad Design Example 3

The software should offer tools to push, pull, smooth, grab, pinch or otherwise manipulate a digital object and make it look life like. 3D sculpting tool uses mesh based geometry, in which an object is represented by an interconnected surface mesh of triangles that can be pushed and pulled around. To do this we need to subdivide the meshes when the user grabs pushes or pulls the object. Loop subdivision is implemented for mesh subdivision. The GUI and other features like rotate zoom were designed based on the other tools present.

The collaborative feature should be such that when one user makes some modification to the model the operation done is sent to all the other users who are linked in the network. Once the user receives the operation he performs the same on his model. At the end of the day all the users should be able to communicate and edit it together.

Assignment: Write Design & Implementation Sections (2.5%)



Write a design and an implementation section for your project (~3-7 pages double-column)

- Individual submission, no group work, worth 2.5%
- Solutions can be hypothetical where necessary: imagine your project is over and was successful
- Be professional: try to imitate well-written papers
- Use LaTeX and BibTex, e.g. https://www.writelatex.com/

Submit PDF by Sunday 21/9 7pm to assignment dropbox: https://adb.auckland.ac.nz

All the best :-)