Stereoscopic Shutter Glasses

Cory Francis Klein, William Lee, William Lorenzo Swank, Michael Yang

Contents

1. INTRODUCTION AND MOTIVATION

For this project we will be creating a pair of stereoscopic shutter glasses to view a ray-traced image on a computer monitor in pseduo-3D. This product will enhance a user's intuition about a scene or graphical data by allowing the user to view the data in pseudo-3D. NVidia already produces similar technology, but NVidia's solution requires costly hardware available only from NVidia. Our goal is to design shutter glasses that will be compatible with "run-of-the-mill" graphic cards and monitors with refresh rates as low as 60 Hz. This will bring three-dimensional display technology to users at a low price point by allowing users to utilize their existing hardware [4].

The project is comprised of two main components: shutter glasses and the display software. This divides the project up rather neatly into discrete hardware and software components. However, there must be great care given to interface issues between the software and the glasses. Timing data must make it from the software to the glasses quickly enough to allow the glasses to synchronize with changing screen content.

There are several different ways to present 3D graphics using computers and displays. The proposed method will work by making use of the eye's natural ability for stereoscopic vision, specifically by displaying separate images of a 2D surface to each eye. The shutter glasses will block the light to one eye as the image for the other eye is displayed on the screen, then it will switch, blocking light to the other eye while displaying a second image. This cycle will repeat at frequencies as high as 60 Hz. The user's brain will fuse the two images creating depth perception.

As indicated above, NVidia currently produces a similar package that works very well and delivers a clear 3D image to the viewer. This package typically requires users to buy new hardware and a new display. At a minimum, an NVIDIA GeForce 8000 graphics card and a monitor with a 120Hz refresh rate are required. NVidia's glasses alone cost $200. The user's plight is made more difficult by the DirectX 10 requirement. Since DirectX 10 is only available on Windows Vista, most users also face a significant hardware and software upgrade cost [3].

Our project will use a ray-tracer instead of a traditional Z-buffer-based rasterizer to produce both a series of still images and a movie to be viewed with the glasses. The rational for using a ray-tracer is rather simple: we would like to use this opportunity to explore some of the different visual effects that are offered best in a ray tracer. Participating media, specular rendering, thin-lens cameras, distribution ray-tracing, and other Monte Carlo methods have not been well-explored in stereo vision. While the purpose of this project is not to explore the problems, benefits, and unanticipated effects of Monte Carlo systems that arise in the special case of stereo vision, it will position us to be able to study any potential phenomena at a later date.

In order to keep the ray-traced scene in sync with the glasses we will implement clock synchronization between the two, with a phase-locked loop (PLL) circuit to correct for clock drift. We anticipate that much of the debugging will surround the issue of timing because there are so many factors that affect it (e.g. the glasses, ray tracer, monitor, and CPU/GPU capabilities). A PLL is a analog circuit which locks the phase of the output AC signal to the input signal. Using a PLL will help us achieve the goal of keeping the output signal from the computer synchronized with the LCD shutters. The PLL circuit contains a voltage control oscillator, a phase detector, and a loop filter. The phase detector compares the input and output signals to track the phase difference between them. The loop-filter transforms the difference signal into a DC voltage. The voltage-controlled oscillator captures the difference and modifies the frequency of the output signal without phase shift. This technique can help us synchronize the shutters with the monitor output. This will allow us to make sure the glasses are excatly synchronized with the output on the screen, preventing any flickering or ghosting.

The LCDs used for the shutters require an optimal 25 volts DC to go from completely opaque to completely transparent. Since USB only runs at 5V DC, this power must be amplified. Since the LCDs don't require any current, but just need a differential voltage, we can ladder multiple DC charge pumps to create the needed voltage difference.

Because the glasses and the ray tracer are independent from one another they will be developed in parallel. We plan to complete the dual projects within five months. An additional two months will be required to interface the products, debug them, and create a polished user experience.

2. IMPLEMENTATAION

Since this is a group project, the emphasis will be on design and construction of the 3D shutter glasses. As we can see from fig 1, there are four parts involved. The first part is the computer. There are two programs required when the shutter glasses are operating. Specifically, the one program generates a ray-traced scene and displays the scene on the monitor, and another program communicates through USB to the shutter glasses. The second part is an electronic oscillator and PLL (phase locked loop) circuit. The electronic oscillator and PLL circuit are components that control the oscillation of the shutter glasses. The third part is the LCD shutter glasses themselves. This pair of glasses shutter each eye at certain frequency to present a pseudo-3D image. The final component is the monitor with a ray-traced scene shown on it. With these components functioning correctly, the user can see a 3D image.

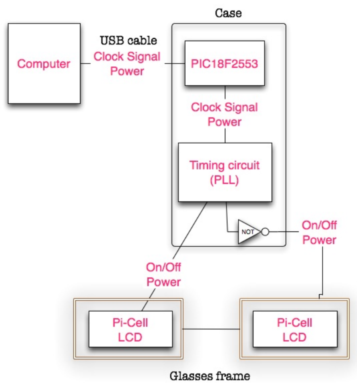

Figure 1. System block diagram

The circuit flow starts with a timing signal from the computer. The timing signal goes through the USB cable and is captured by a USB adapter circuit. Then the signal is restore by the PLL circuit. Finally, the PLL determines the On/ Off signals to drive the LCD voltages, making the LCD shutters transparent or completely dark.

2.1 Glasses

The glasses are made up of two parts: the Pi-Cell LCDs and the control circuitry. The part that the user will wear on their head will merely consist with two Pi-Cell LCDs mounted in an existing pair of glasses frames with wires extending out the back that will provide the voltage to the LCDs. The circuitry will consist of a board that controls the glasses. It will require DC power amplification, a USB controller, and some logical circuitry to handle interpret the commands from the computer and create an oscillating signal that corrects itself with the clock of the computer using a phase locked loop.

Pi-Cell LCDs will be used because of their ability to change from transparent to opaque at very high speeds. Traditional LCDs could be used, but the left and right eye images would become much more blurred, likely causing head-ache inducing images on the screen. The Pi-Cell LCDs have a special characteristic due to the alignment of the cells in the crystals [2].

Obtain Components (2 weeks, 5/9/09˜5/24/09) Order Pi-Cell LCDs. Find suitable lens frame. This will be done before the beginning of the semester by Michael Yang.

Test Components (2 weeks, 6/1/09˜6/15/09) Verify suitability and usability of components. Check maximum shutter speed for the Pi-Cell LCDs. This will be done before the beginning of the semester by William Lee.

Schematic Design and Verification (2 weeks, 6/1/09˜6/15/09) Complete and verify the design for all of the hardware components, laying out how the USB controller interfaces with the circuit board and the LCDs. This will be done by William Lee.

Implementation from Schematic (15 weeks 7/1/09~9/30/09) Actually building and testing the hardware components. Build the circuit board and put it together with the USB controller, LCDs, and frames to create the actual glasses. This will be done by William Lee and Lorenzo Swank

Initial Visibility Testing (milestone 9/16/09) Develop and construct a demo that will demonstrate correct shutter operation. This will be done by William Lee and Lorenzo Swank.

Shutter Vision Testing (milestone 10/16/09) This milestone will consist of a demonstration that the shutter glasses are working at a given frequency. The USB driver will not be completed at this point, so a very slow shutter rate will be used with the computer and glasses just to perform a rudimentary demonstration of functionality and concept. This will be done by William Lee and Lorenzo Swank.

2.2 Software

There are two main components to the ray tracing software. The main engine will take a file describing a scene, and create an array of pixels the desired size to show on the screen representing the ray traced image of the described scene. Then there will be a separate part that takes care of the 3D vision aspect by maintaining camera angle, movement, and eye separation. This part will also interface with the driver to make sure the timing of the glasses matches the framerate of the ray traced image.

Implement Basic Ray Tracer (4 week 7/01/09~8/1/09) Create the code supporting dual cameras in a ray tracer. This will be done before the beginning of the semester. This will be done by Cory Klein and Lorenzo Swank.

Stereo Buffering (10 weeks 8/01/09~10/16/09) Using GLUT, implement stereo output on a standard monitor/LCD [1]. This will be done by Cory Klein and Lorenzo Swank.

Stereo Buffering Demo (milestone 10/16/09) The purpose of this milestone is to demonstrate correct output and sychronization with the glasses for a single stereo image. This will be done by Cory Klein and Lorenzo Swank.

Stereo Video Playback (8 weeks 10/16/09~12/16/09) Using GLUT, implement stereo output on a standard monitor/LCD for a pre-rendered video. This will be done by Cory Klein and Lorenzo Swank.

Stereo Video Playback Demo (milestone 12/16/09) The purpose of this milestone is to demonstrate correct output and sychronization with the glasses for a stereo video file. This will be done by Cory Klein and Lorenzo Swank.

2.3 Driver

The driver is in charge of communicating the oscillation period and phase to the glasses controller. The driver interface has become simplified by the discovery of the USB Bit Whacker. This device removes the need to write a low level USB driver as it automatically mounts itself on the given system as a serial COM port. Control of the glasses is reduced to simple communication over a COM port, an easy task in most operating systems.

Obtain Components (2 weeks 5/09/09~5/24/09) Order the USB Bit Whacker. This will be done before the beginning of the semester. This will be done by Michael Yang.

USB Connection (6 weeks 7/16/09~9/01/09) Establish a rudimentary connection between the USB Bit Whacker and a personal computer. Verify that we can connect to the controller and communicate with it manually on the serial port. This will be done by the entire team.

Simple Communications Demo (2 weeks 9/01/09~9/16/09) Transfer some data from the computer to an external USB device over the serial connection. Different from the last step as the communication will be handled directly by the computer through software. The demonstration will be software control of LEDs on an external USB device. This will be done by Michael Yang.

Send The Clock (4 weeks 9/16/09~10/16/09) Simply send the clock in a manner recognized by the glasses. This will be done by the entire team.

2.4 Documentation

Write User's Guide (8 weeks 10/01/09~12/16/09) This will be done by Michael Yang and Cory Klein.

Write Final Report (10 weeks 9/16/09~12/16/09) This will be done by William Lee and Lorenzo Swank.

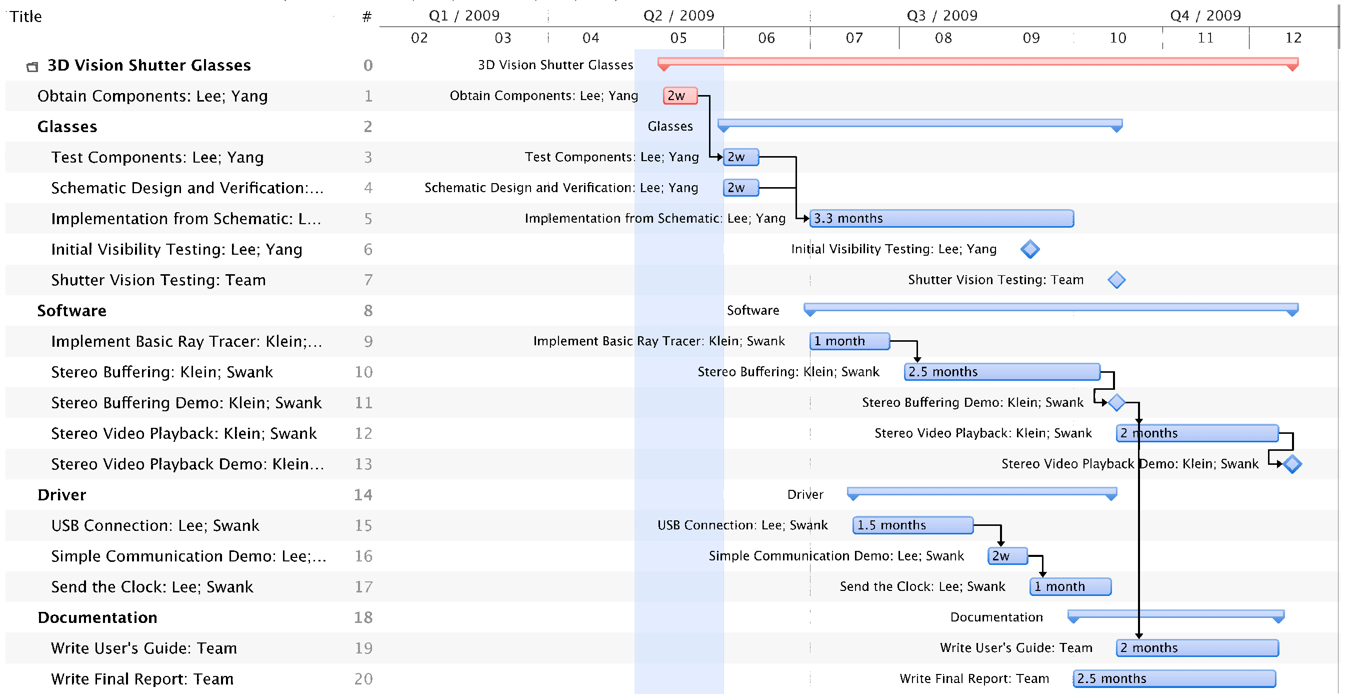

Figure 2. Gantt chart

2.4.1 Milestones

Major milestones are shown in 2 as diamond shapes and signify major advances in progress. In addition to these major milestones, there are the following intermediate milestones representing less monumental advances in the project.

6/30 Demonstrate transparency vs voltage on PI-cell LCDs. Also demonstrate manual serial communication via Bit Whacker.

7/21 Demonstrate alternating signal from components. Demo left and right camera renderings from OpenGL.

8/12 Demonstrate alternating and variable signal for glasses. Left and right camera renderings from ray tracer.

11/16 Full stereo buffer of ray traced image demo at low frequency (less than 5 Hz).

2.4.2 Testing and Verification

Because most of the project is very modular, testing of separate parts as they become completed is a simple task. Communication with the Bit Whacker can be tested using HyperTerminal on a Windows machine, the ray tracer and it's double buffer can be tested as a standalone unit by reducing frequency. The Pi-cell LCDs can be tested using an oscilloscope and a voltage provider with a resistor to limit current. The most difficult part will be putting the working parts together and debugging the interface until they work together in unity. This testing will be performed at first using a very low frequency on both ray tracer and shutter glasses. USB/serial communication will be tested by using an oscilloscope on the glasses end to test how accurate the signal is getting transferred from the computer to the glasses.

3. INTERFACES

The interfaces in our project are comprised of a USB-glasses interface, a software-USB interface, and a display-software interface.

3.1 USB-Glasses Interface

The controller running the glasses will require information directing it to use the proper flashing frequency and also to correct any time drift between the internal clock and the clock determining the frequency on the monitor. Because the visualization software will run on a standard PC, the most readily accessible communication medium between glasses and computer is the Universal Serial Bus, or USB.

USB requires that a specific protocol be followed for communication. In order to follow this protocol, the hardware on the glasses must have a dedicated controller for communicating with the USB. This will also perform the beneficial task of encapsulating the communication and simplifying the CPU on the glasses. To the glasses CPU, receiving timing information will be simplified to reading from memory shared by the USB controller.

Fortunately, a special device has been chosen to simplify the USB communication interface. A device called a USB Bit Whacker can be used. The Bit Whacker has its own USB driver that opens a new serial port that is routed to the USB microcontroller. Communication with the microcontroller is then performed over a serial COM port using custom commands given in the documentation for the Bit Whacker. The greatest advantage of this USB controller is that it can be programmed without any special hardware or tools.

The USB controller will consist of a serial receiving and decoding unit. It will be capable of interpreting information specific to the project and storing it in the appropriate place in memory. In order to confirm correct settings and perform clock synchronization with the PC, the controller will also be able to read from shared memory and send back commands following the USB protocol.

3.2 Display-Software Interface

The primary function of the ray tracer, however, is not to synchronize with the glasses, but to display ray traced 3D viewable images on the screen with real time rendering. In order to create the best possible image a computer and display combination can produce, it is necessary to know certain information about the hardware that the ray tracer is running on.

This interface will exist as purely a software piece, interfacing with the computer or OS it is running on in order to find the monitor refresh rate and any other necessary processor information. Because there will be a left eye and a right eye image being rendered, double buffering will be required [1]. Discovering information about the host computer and adjusting the quality of the visual image will be a one-time calculation during the run-time of the process. Once the capabilities of the computer are gauged, the ray tracer will modify the quality of the image by changing the image resolution, framerate, ray branching, and ray depth.

3.3 Software-USB Interface

The ray tracer will determine the appropriate frequency for the glasses to operate at, however, this requires that there be a communication medium for the ray tracing software to interact with the glasses. As the glasses communicate over the USB interface, a driver for the glasses will need to be written for this communication to be possible. This driver will be used in order to synchronize the glasses' clock and send the appropriate frequency for flashing.

Since the Bit Whacker creates a simulated serial COM port, the driver that we need to create will not be a USB driver, but a serial driver. This makes the implementation of this interface much easier, as all we need to do is get a handle to the correct COM port and send and receive predefined commands. There is a great amount of documentation for this kind of driver for Windows, which is the platform that will be supported.

4. RISK ASSESSMENT

4.1 Hardware Risk Assessment

Pi-cell LCD Risk Level: Low. A vendor has already been located. Pi-cell LCD's have simple circuitry, with only two states, on and off.

Controller Risk Level: Low. We need a controller that is small enough to mount on your head, can be reprogrammable, and is capable of supporting the USB interface. One team member is an electrical engineer, however, thus keeping this risk from being very high. A solution that does not require an FPGA or more complicated circuitry is being explored.

USB Risk Level: Low. None of the group members have experience with the Universal Serial Bus standard. This is mitigated by using a project board that handles USB drivers, providing a simple interface.

4.2 Software Risk Assessment

Determine framerate Risk Level: Medium. The framerate will depend on specific computer properties such as monitor refresh rates and graphics card capabilities. Finding out these values for specific computers internally may prove difficult.

Dual-buffering and timing Risk Level: Medium-Low. The ray tracer will need two separate cameras, and we need to find an efficient way to dual buffer these separate images and switch between these buffers at a given frequency.

Create scene Risk Level: Low. We need a scene with some content we can demo, but much content is available free online. Custom content may require a graphical editor to be created quickly.

Optimization Risk Level: Low. The faster the ray tracer, the better the image will look. However, in the worst case, this part does not fail, it only produces an image of lesser than desirable quality. The amount of extra time we have left over will determine the level of graphical optimization we may achieve.

Complexity levels Risk Level: Low. There will be the ability within the ray tracer to select a complexity level to automatically scale the image quality in terms of resolution, ray depth, and support for various materials. This is just a matter of taking the time to implement this functionality, as in ray tracing it is easy to scale the complexity of the image rendering.

5. BILL OF MATERIALS

Component	Vendor	Name of Contact	Telephone/Email
Pi-Cell LCD	Liquid Crystal Technologies	Tim	440-232-8590
USB Development Board	SparkFun	N/A	303-284-0979
USB Cable	MonoPrice	Karen Mu	support@monoprice.com
Frames for glasses	Walmart	N/A	(801) 484-7311
Charge Pump	Microchip		408-961-6444
Misc. circuit components	University ECE Stockroom	Omar Shami	

Component	Part Number	Lead Time	Unit Cost	Quantity
Pi-Cell LCD	N/A	3-4 Weeks	\$15.00	4
USB Development Board	PIC18F2553	1-2 Weeks	\$24.95	2
USB Cable	2099	1-2Weeks	\$1.73	2
Frames for glasses	N/A	1 day	\$5.00	2
Charge Pump	TC7660	1-2Weeks	\$0.79	2
Misc. circuit components	Various	None	~\$20.00	N/A

Table 1. Bill of Materials

REFERENCES

1. Blaine Hodge, “OpenGL GLUT Tutorial,” [Online document], 2009 April 9, Available HTTP: http://www.nullterminator.net/glut.html
2. Mingxia Gu, “The World of Liquid Crystal Display,” [Online document], 2009 April 9, Available HTTP: http://www.personal.kent.edu/~mgu/LCD/pi-cell.htm
3. NVidia, “3D Vision,” [Online document], 2009 April 9, Available HTTP: http://www.nvidia.com/object/GeForce_3D_Vision_Main.html
4. NVidia, “3D Vision Requirements,” [Online document], 2009 April 9, Available HTTP: http://www.nvidia.com/object/GeForce_3D_Vision_Requirements.html