Immersive 3D World

Christine Konicki and Mikhail Rudoy
6.111 Final Project Proposal
13 November 2015

I Overview

Virtual reality has become a new and exciting realm for research and product development. This past July, OnePlus became the first company to launch a product using VR, and others are slowly incorporating VR into what they build. Our final project aims to create an audio-visual VR simulation using as input the turning motion(s) of the user's head and a game controller. In this game/simulation, the user will be able to walk through a virtual world using the controller. The turning of the user's head will be used to trigger a "looking around" effect, where looking to the right will cause the contents of the screen to move left, as if the user were actually in the virtual world looking around.

The game/simulation will consist primarily of a virtual 3D world constructed with a polygon model. The graphics system will allow the player to translate and rotate her in-world perspective. This motion in the virtual world will mostly be at the hands of the player through a video game controller—just like for a normal video game. In addition, the player will be wearing a headphone set with an attached 3-axis gyroscope to easily pick up rotation values in multiple directions for the player's head. The readings will be used to adjust the player's perspective in the 3D world; as the player looks to the right, for example, the world will "rotate left" bringing virtual objects that are further to the right into view. The headset to which the gyroscope is attached will let the user hear the sound of footsteps as she walks around in the world.

II Design

In this section, we will give a general overview of the modules comprising our project. We will also provide some insight into our design choices.

Terms

The following are some terms which will be useful to understand the rest of this proposal. They are standard buzzwords used in graphics design and dynamics.

- **Camera**: the player's vantage point into the game, the eye within the virtual world that sees the virtual world
- **Yaw**: the rotation of a rigid body about the vertical z-axis (e.g. the user turning her head left and right)
- **Pitch**: the rotation of a rigid body about its "wingspan" (e.g. the user turning her head up and down)

Design Decisions

At its highest level, the project is designed to be a game logic module with a graphics engine generating a 3D world for VGA display. The person playing the game constantly provides new readings from the gyroscope (i.e. by turning her head); the orientation data from the gyroscope is transformed into the yaw and pitch of the user's head which are then sent to the game logic module to adjust the camera rotation. The player also provides input via the controller (i.e. from movement of the joystick), and this controller data is sent to the game logic module to adjust the camera position and rotation. The final module generates audio (more specifically the sound of footsteps) and is triggered by the game logic module.

The game logic and graphics FSM module is in charge of displaying the video output and coordinating the other modules. It will operate ideally at 60 frames per second in order to keep display changes fluid, but a frame rate of 30 frames per second is an acceptable minimum in the event that 60 is too high. The virtual camera position and rotation coordinates will be stored in register memory while the graphics data for the screen will be stored in a frame buffer in memory. The module will contain VGA logic as well so that the FPGA can output a display of the 3D world (as seen from the virtual camera) to a monitor. In addition, this module will periodically trigger the audio module according to the game state.

The gyroscope interface module generates yaw and pitch values of the user's head using readings from the gyroscope. This requires reading in all three rotational coordinates for the gyroscope and transforming them into the coordinate system of the user's head. As a result, we expect the gyroscope to be useable in any orientation with only a short setup stage to configure the constants for the change of coordinates. The yaw and pitch will then be sent to the FSM so that the virtual camera can be updated. We plan to allow a few weeks for the gyroscope IC to arrive. The coordinate transformation system can be implemented in the meantime.

The game controller interface module allows us to connect the controller to the FPGA. It will read in coordinates indicating the positions of the joysticks on the controller and use them to control translation and rotation in the world. The coordinate data will be sent directly to the FSM so that the virtual camera position can be updated. We will start off by building up the software for this in the FSM and use some of the buttons on the FPGA in place of a game controller. This way we can allow time for whatever game controller we choose to arrive, as it has to be ordered.

Finally, the audio module will generate the sound of footsteps. The game FSM module will count up the total "distance" traveled in the virtual world after a certain reference point. After a certain amount of distance (most likely the game's equivalent of of a single pace), a trigger signal will be sent to the audio module to indicate that a footstep sound needs to be generated. After the sound is produced, we will deactivate the trigger and start the FSM's count over at a new reference point. Since this module is not difficult to implement, it will most likely be completed early.

System Block Diagram

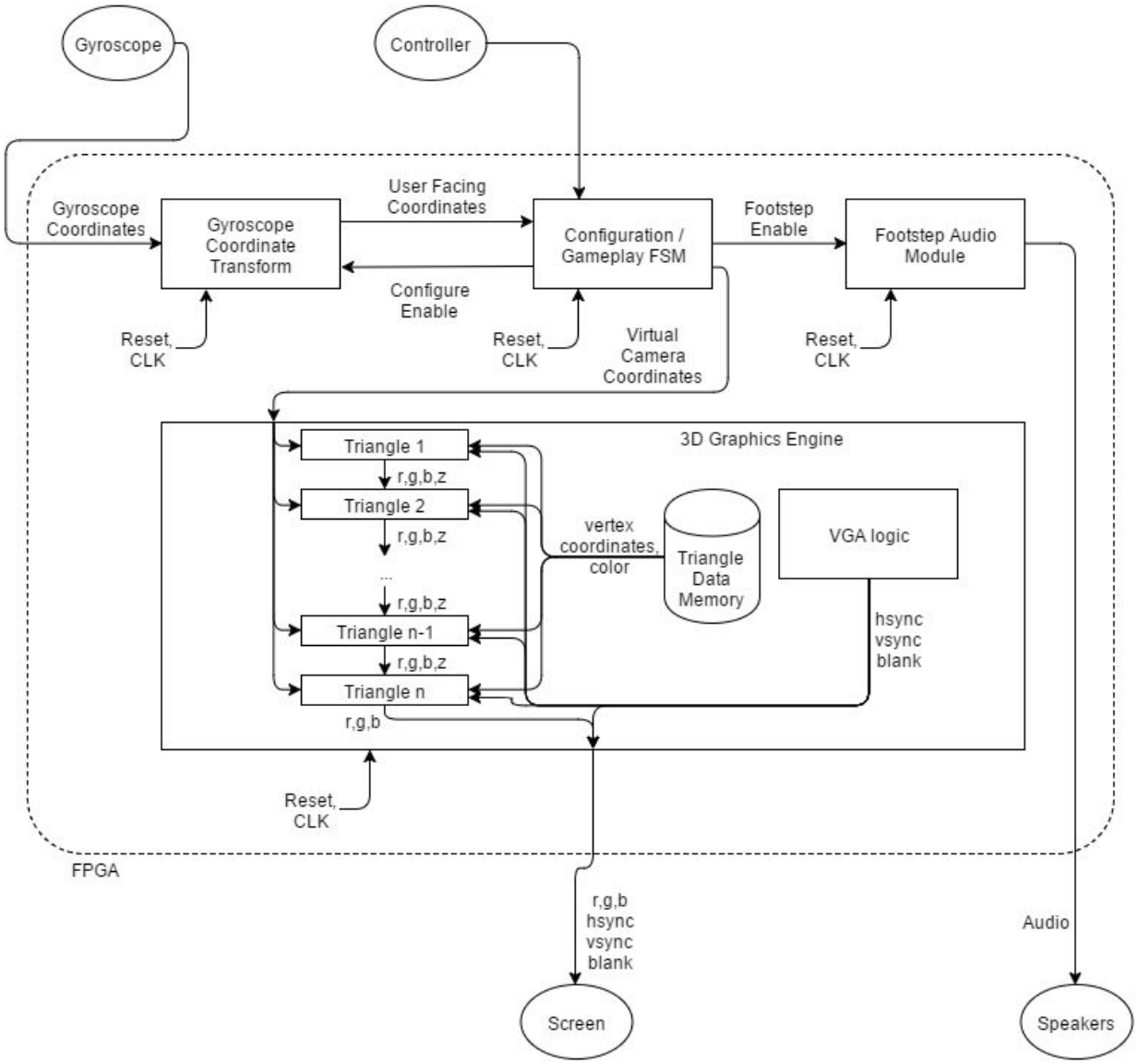

Figure 1: A block diagram showing all the modules of our system. The hardware peripherals are shown as ovals external to the FPGA.

III Implementation

3D Graphics Engine and Game FSM Block

The entire 3D world can be decomposed into a series of triangles as shown in the graphics engine of Figure 1. The vertex coordinates in 3-space and color of each triangle (all of which will be static) will be stored in memory. The triangle modules will each use the coordinates of the virtual camera to generate pixel values as viewed from the perspective of the camera. Each triangle module will output the current color of the pixel in question with the depth into the

screen at which that color occurs. The triangle modules will be in series, and each triangle will have as input the closest color so far at the pixel in question (and its depth); each triangle module's outputs will equal its inputs unless the triangle is closer at the pixel in question than the closest color before that, in which case the values are adjusted accordingly. The VGA logic included in the FPGA will take the final color values outputted by the final triangle and display them on the screen. The colors will be represented as red, green, and blue signals as is customary for VGA. Despite the series arrangement of the triangle modules, most of the computation for the triangles can occur in parallel. As a result, our design are using a triangles in parallel and pixels in series design. We chose to use this type of design instead of the alternative—triangles in series and pixels in parallel—because computing over all pixels in parallel would require having dedicated hardware for each pixel: far more hardware than can fit in the FPGA.

The FSM block will be used to first configure the game for gyroscope readings and then to control game logic (i.e. camera coordinates). During the configuration step, the screen will display instructions telling the user to look at the left of the screen, then the right of the screen, then the top, and then the bottom. These various configuration instructions will each have their own configure-enable signal, which are generated from within the FSM and sent to the gyroscope interface module.

Footstep Audio Block

This module will generate the sound of footsteps for the user to hear through the headphones. As soon as the user maneuvers the joystick on the game controller, a timer in the game FSM will start counting up from 0 at a speed proportional to the walking speed (i.e. in units of "virtual world distance") until it reaches about half a second (assuming the joystick is still being driven). At this point, a trigger signal will be sent from the FSM module to the audio module. The audio module will then generate a square wave with a low frequency (100 - 150 Hz) to be played from the speaker output on the FPGA for between a tenth and a quarter of a second. The duration of the square wave will also be timed. Once the square wave stops playing, the first counter will start again, counting up from 0 until it reaches about half a second. If the user chooses to let go of the joystick at any point, the timer will be reset to 0 and not start counting up again until the user uses the joystick again. The periodic sounding of the low-frequency square wave is meant to mimic the sound of footsteps as the user traverses the 3D world. The half a second between "footsteps" is meant to mimic a reasonable walking pace, as it would take someone about that much time to complete a single step if they were walking at a comfortable speed.

Gyroscope Interface and Transformation Block

The gyroscope IC chip attached to the user's headphones will output analog signals indicating its change in position in 3-space. We will first pass these three signals through analog-to-digital converters (ADCs) so that they can be read by the FPGA. The newly digital coordinates will then be sent through a coordinate transformation block that will use (some combination of) the four screen position reference points calculated in the configuration stage of the FSM to transform the coordinates mathematically and determine the pitch and yaw of the user's head. This way,

no matter how the chip is tilted (according to the whim of the user wearing the headphones), the turning of the user's head can still be responded to appropriately. The new coordinates will be sent to the game FSM so that the camera can be updated.

Controller Interface Block

The game controller PCB will output signals indicating the position of the joystick (which has a full circle of motion). The signals (presumably digital), will then be sent to the FSM so that the camera can be updated and the display of the world can be changed. The joystick allows the user to walk forward, backward, left, and right in the world. Walking in the world will cause the camera position to move left, right, into, or out of the screen (or more generally to rotate in that plane).

Levels of Completion

The most basic level of completion consists of having the graphics engine, game logic, and gyroscope interface working. The next level adds logic for the game controller but uses some of the FPGA buttons in place of the joystick. The next level adds the audio module. The final, most complete level adds the interface for the game controller itself.

IV Timeline

The figure below is a chart showing the proposed task schedule of this project. Red tasks indicate work to be done by Mikhail, blue tasks indicate work to be done by Christine, and green tasks indicate a combined effort. The following are the steps mentioned in the chart:

1. Implement the graphics engine for the game world and interface with VGA. Order all necessary parts.
2. Implement logic module to control the game world using the gyroscope and controller (just software at this point in order to buy time for when the necessary parts are acquired).
3. Implement audio module (without controller, again just software).
4. Interface gyroscope and controller with audio module and game logic module.
5. Test integration of all modules
6. Additional time for testing and/or leeway in the event of setbacks
7. The project demo and checkoff occur during the last week of the semester.

	Week of 11/2	Week of 11/9	Week of 11/16	Week of 11/23	Week of 11/30	Week of 12/7
1. Graphics and VGA interface, order parts						
2. Gyroscope and controller game logic						

3. Audio module SW						
4. Interface gyroscope and controller HW						
5. Test integration of modules						
6. Buffer week for testing and/or catch up						
7. Demo and Checkoff						

V Testing

For basic testing of each module, we will write Verilog test benches and run them using ModelSim. More expansive test benches will be written in order to test the integration of the modules, making sure that everything still works as we put modules together. The VGA display will be tested by displaying the current state of the game world from the camera. The construction of the game world will increase in complexity, so naturally the nature of the VGA testing methods will as well. The hex display on the FPGA will be used for testing and debugging to track states and register values; this will allow us to confirm that the gyroscope and controller readings are being processed correctly. Finally, we will test the complete functionality of the gyroscope, the game controller, and their use in game play.

VI Resources

This project requires a few parts not already available to us. We will need a gyroscopic IC to wire to a breadboard and a couple of ADCs so that it can be interfaced with the labkit. We will also need a pair of headphones with a relatively wide headband. The breadboards supplied to us in our labkits are too large to be attached to the headband of the headphones, so we need a breadboard that is small enough for this. The last part we will require is a game controller, preferably one with a joystick.

VII Conclusion

In conclusion, our immersive virtual reality system is technically complex—particularly in its graphics engine—yet reasonable to implement using an FPGA. We expect building this project to be very educational. We will learn about the complexity of implementing three dimensional angular and translational coordinate transformations in a digital system. We will also learn about using a game controller as a hardware peripheral. Throughout this project, we will be applying the concepts learned in the course in a new and interesting context.