6.111 Final Project Proposal
Team: Hui Ying Wen, Michael Price, Joyce Chen

Realtime Light-Saber Generator

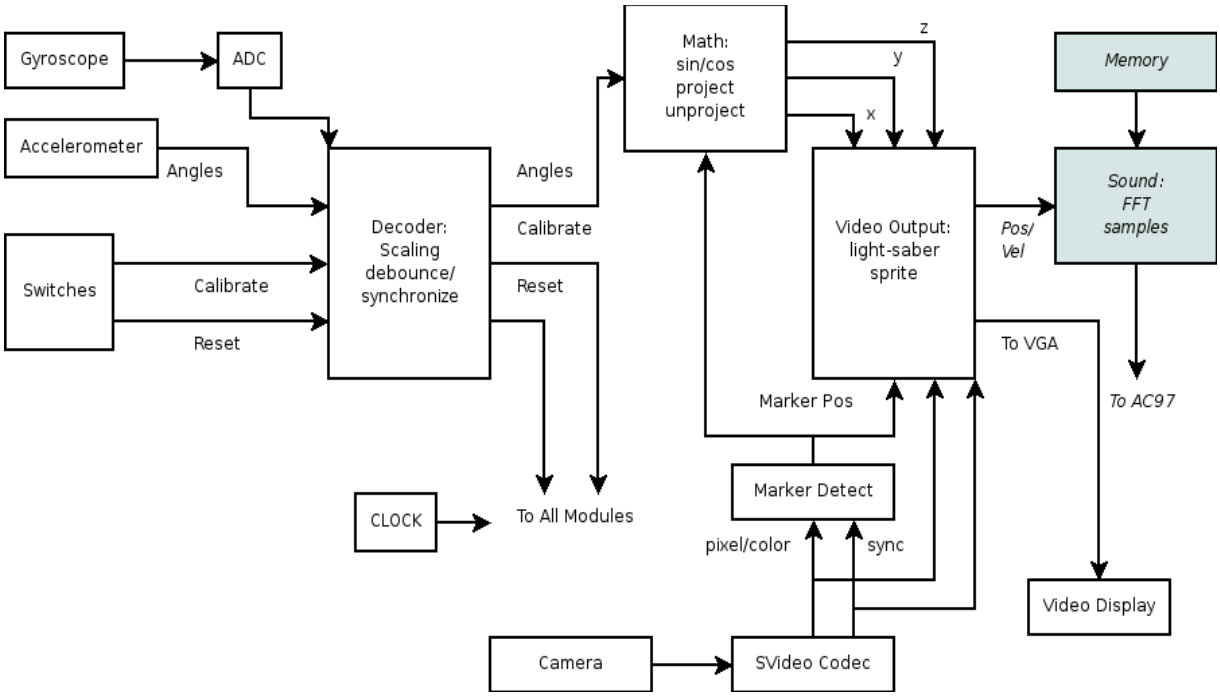
BASIC FUNCTIONALITY

The "Realtime Light-Saber Generator" will display live video of a person such that he appears to hold a light-saber in one hand. The projected light-saber changes length and direction in accordance with his hand motion.

The baseline system will utilize a physical handle, containing a colored marker, an accelerometer, and a gyroscope, held in the person's hand. The origin of the light-saber is determined by detecting the colored marker or light from video input (detecting the colored marker). Matrix transformations will be performed on input values from the accelerometer and gyroscope to determine the pixel dimensions and direction of the projected light-saber. Using these dimensions, the pixels composing the light-saber will be printed over the live video input, which will be directly output to a monitor.

Additional features may include sound based on light-saber movement, varying saber width based on distance from camera, tapering saber width based on tilt, and shading for basic three-dimensional realism effects.

BLOCK DIAGRAM



DESCRIPTIONS OF MODULES

Math (implementation by Michael Price)

Inputs:
"clock" - clock input
"reset" - resets module to default values
"phi_d," "theta_d" - gyro angular velocities
"x_a," "y_a," "z_a" - accelerometer coordinates
"x_s," "y_s" - screen coordinates of marker

Outputs:
"x[3:0]," "y[3:0]," "z[3:0]" - global coordinates of light-saber boundaries
"x_s[3:0]," "y_s[3:0]" - screen coordinates of light-saber boundaries

State variables:
(Matrices are 4x4 to include homogeneous coordinates)
- [phi, theta]: integrated gyro angles
- saber[3:0][3:0]: transformation from local lightsaber frame to global frame
- tran_global[3:0][3:0]: transformation matrix from screen frame to global frame
- tran_screen[3:0][3:0]: transformation matrix from global frame to screen frame

Module Description:
The math module is designed to take input from the inertial sensors and video input module, computing the parameters of a local coordinate frame based on the rotation of the physical light-saber. A set of fixed coordinates representing the shape of the light-saber will be rotated by the angles measured by the gyroscope, then translated by the distances measured by the accelerometer. These points will then be displayed on the screen.

There are two important coordinate frames with which to represent spatial positions: the global frame and the screen frame. The global frame is a typical [x,y,z] space with the z-axis pointing up, the x-axis pointing towards the camera, and the y-axis pointing to the right. The screen frame is an [x,y,z] space where x- and y-coordinates between 0 and 1 map to pixel positions within the screen. In the screen frame, the z-coordinate does not matter. A linear transformation between the two frames corrects the perspective issue: faraway objects appear smaller than nearby objects. We will invent an appropriate fixed-point scale for coordinates, so that the system retains adequate precision while not clipping coordinates on the order of a few meters.

This module will have to compute three coordinate transformations, stored as 4x4 matrices. The first, saber, is used to compute global coordinates based on the pitch, yaw, and absolute position estimates of the light-saber. The shape of the lightsaber in its local frame is a rectangle, with each point being a column of a matrix:

```
S_l = [ x1 x2 x2 x1
        y1 y1 y2 y2
        0  0  0  0
        1  1  1  1  ]
```

The transformation matrix, saber, includes the coefficients of the spherical-to-cartesian transformation for the current angle estimates. Multiplying S_l on the left by saber produces global coordinates.

The math module takes the result of that transformation and multiplies the points on the left by tran_screen. This creates a set of 4 points in screen coordinates, which are the outputs of the module: the array nets x, y, and z. These values are passed to the video output module, which superimposes the color of the lightsaber within the quadrilateral enclosed by those points.

A practical implementation of the math module will require several instantiations of a 4x4 fixed-point matrix multiplication module, as well as CoreGen lookup tables for the sin(:) and cos(:) functions. Other arithmetic logic will update the angle estimates and the individual elements of the three transformation matrices. The total number of state variables (required memory) is 50, or 900 bits if 18-bit fixed point values are used.

The update rate of this module will need to be significantly faster than the 60 Hz video frame rate, to ensure a minimum of numerical errors in the angle and position estimates. A local clock frequency of around 1 KHz may be appropriate.

The module will populate three matrices and perform three matrix multiplications per clock cycle. The corresponding computational requirements are approximately:
- 2 sine and cosine lookups per clock
- 150 multiplication operations per clock
- 100 addition or subtraction operations per clock

Video Output (implementation by Hui Ying Wen)

Inputs:
"clock" – clock input
"reset" – resets module to default state
"hcount" – horizontal pixel count
"vcount" – vertical pixel count
"hsync" – horizontal sync signal
"vsync" – vertical sync signal
"saber_xbase" – horizontal pixel location of light-saber base
"saber_ybase" – vertical pixel location of light-saber base
"saber_x" – fixed-point scaled length (from 0 to 1) along x-axis of light-saber
"saber_y" – fixed-point scaled length (from 0 to 1) along y-axis of light-saber

Outputs:
"pixel" – Y'CbCr value of current pixel to be drawn. Value is "0" if pixel does not need to be drawn.

Module Description:

This module acts as a "sprite," drawing pixels over the live video output as needed.

The input "saber_x" and "saber_y" values will be scaled to the current screen resolution to give x-axis and y-axis pixel lengths. In addition, a constant width in pixels will be assigned to the light saber. The value of output "pixel" is assigned according to whether the current values of inputs "hcount" and "vcount" land within these saber dimension values.

Complexity: Most likely, fixed-point multiplication and division will be performed on 11-bit "saber_x" 10-bit "saber_y," and width values (around 10 bits). Other than individual registers, the algorithm will not require significant RAM memory.

Testing: Test values of "saber_xbase," "saber_ybase," "saber_x," and "saber_y" will be input to the module, and the output will be viewed on the monitor.

Marker Detection/Video Input (implementation by Joyce Chen)

Inputs:
"S-video" - live video input
"clock" - clock input
"reset" - resets module to default values

Outputs:
"marker_x" - horizontal pixel location of marker center of mass
"marker_y" - vertical pixel location of marker center of mass

The Marker Detection module will process S-video input from the camera to detect the position of the marker in the images. The marker will be detected by its distinctive color in the raw image data. The position of the marker will be the marker's center of mass. The module will find the center of mass of the marker by summing over the x and y pixel coordinates of pixels with the distinctive marker color and dividing by the total number of marker-pixels in the frame.

Let the image width be w and the height be h.

Estimation of memory needed:
Marker Detection module needs to keep track of 3 numbers:

(1) total number of marker-colored pixels n
(2) sum of x coordinates of marker-colored pixels X_m
(3) sum of y coordinates of marker-colored pixels Y_m

n has an upper bound of the total number of pixels on the screen: $w*h$.
X_m has upper bound $n*w$.
Y_m has upper bound $n*h$.

If the image is 1024 pixels wide and 768 pixels high, we will need 20 bits for n, 30 bits for X_m, and 30 bits for Y_m.

Estimation of arithmetic operations:
For every marker-colored pixel, n will be incremented by 1, and two addition operations will be performed to update X_m and Y_m. Incrementing n by 1 is negligible. The two addition operations will operate in parallel. Each addition operation will add a 20 bit number and a 10 bit number.

For every image frame, one division operation needs to be performed to find center of mass coordinate x, another to find center of mass coordinate y. These two divisions will operate in parallel.
Each division operation will divide a 30 bit coordinate value by a 20 bit value. The output of the

division will be a 10 bit value within the boundaries of the screen. The division operation for each frame will be the greatest source of delay for the Marker Detection module.

Testing Marker Detection/Video Input Module:
S-video decoding will be tested by displaying video input from camera on the screen.
To test the marker detection module, simple test images will be created where the center of mass of the marker-colored pixels is known.

Sound

This module is an additional feature to be implemented after the successful operation of the baseline system. Velocity will be calculated using the outputs of the math module, which will be used to vary the amplitude of a recorded sound clip. Stereo sound can be implemented according to saber direction along the x-direction.

INTERFACING OF EXTERNAL COMPONENTS

(implementation by team)

The light-saber machine will need to read inputs from inertial sensors (an accelerometer and a gyroscope) in addition to computing the on-screen marker position. Interfacing those two chips to the lab kit will require external electronics. With the right inertial sensor chips, we may avoid the need for a separate analog-digital conversion (ADC) chip.

The user of the machine will hold a cylindrical light-saber handle (without the beam). The handle will be about 2" in diameter and 8" long. A brightly colored ball, perhaps illuminated by an LED, will be attached to the end of the marker so that it is equally visible from any direction. There will be a small electronics module at the opposite end of the handle, including an accelerometer, gyroscope, and interface circuitry. The electronics module will be connected by a long tether to the FPGA labkit.

We will use Analog Devices micromachined inertial sensor chips, acquired through their sample program. In particular, we are interested in the following devices:
- ADXL213 PWM-output, 2-axis accelerometer
- ADIS16100 digital-output rate gyroscope (2)

In addition to the MEMS chips, the electronics module will contain a voltage regulator filtering a +12V rail from the labkit down to +5V for its local power supply. Simple transistor buffers will be used to drive the long wires connected to the labkit, using output signals from the chips. The tether will include a total of 9 wires:
- Power [2]: +12V and ground
- Accelerometer output [2]: X and Y axes
- Gyroscope outputs [5]: Yaw and pitch axes (single serial clock; serial data in and out for each axis)

The tether does not need to carry high-frequency signals; the communication clock frequencies can be as low as 10 KHz as specified in the ADIS16100 datasheet. We may construct a tether from Cat5 twisted-pair wires, connected to the power supply outputs and user I/O connectors on the labkit.