Carlos Cuevas & Cody Winkleblack
6.111 Project Checklist
16 November 2018

Laser Cyclops

Project Checklist

Contents

Basic Level

- Goals
    - Basic implementation of the laser tracking system functioning properly.
        - All hardware components operational, with the target being followed by the laser, and subsequent laser firing as soon as it determines the target location.
    - The VGA should display the feed from the camera, so that the user is able to visualize the movement of the target.
- Major Modules
    - interpreter
        - Taking the input provided by the NTSC camera, a pixel is given in the format YCbCr. This module must then translate this pixel into the format RGB. Since there will be 30 bits generated after this translation, this must be cut down to 18 bits using the most significant bits from each color value so that two pixel values can be inserted into each ZBT-memory address.
    - zbtMemory
        - Should be saving the pixel that is coming in from the interpreter module into an address on ZBT memory. Two pixels per address should be saved. With a given address, as requested by the displayManager module, the respective two pixels in that address will be outputted.
    - displayManager
        - Responsible for sending a pixel to the VGA monitor for every (hsync, vsync) value. It shall extract two pixels for every address request sent to the *zbtMemory* module.
    - ledFinder
        - This module is tasked with determining the center coordinates of the target at every frame, using the pixels that goes through it.
    - arctan
        - A pre-written Verilog module found in ISE, this module will take in the two shorter sides of the triangle wishing to compute an angle for (in radians). For this project, the inputs will be a coordinate (x or y) and depth (z), which will be manually given via the switches on the Labkit.
    - servoH | servoV
        - Given the radian angle for a plane, this module will translate this angle to the appropriate pulse required by the servo. This pulse must be sent every 20ms, which is what the input *pulse_20ms* is for. However, should ledFound be low (in which the target is not found), the module should

ignore the angle input and move the servos to rest (pi/2 radians), using the neutral pulse value.

- servoClock
    - A clock that uses the FPGA base clock and a counter in order to send out a pulse every 20ms.

Intermediate Level

- Goals
  - Implement Chroma Keying, wherein the system would be able to lock onto a color hue that is specified by the user.
  - An image will be imposed over the target and displayed on the VGA output (e.g. the Death Star or someone's face).
  - Using a green screen, an additional background will be displayed behind the target. This would allow us to use various landscapes (e.g. outer space, a war zone, etc.).
  - A firing sound should be generated every time the laser fires at the target. This should be implemented on the go using a siren and counters.
- Major Modules
  - displayMonitor
    - This module will be rewritten in order to retrieve the target coordinates from *ledFinder* and be able to place an image centered over it, in addition to what it was doing already. It would also have to display the cross-hairs generated from the appropriate module.
  - ledFinder
    - Would have to take into account a range of RGB values, based on the hue generated from *chromaKey*. Extensively, there should be a method to manually input a hue on the Labkit.
  - chromaKey
    - Responsible for converting the RGB value located in the center of the cross-hairs. It would extract the coordinates for the cross-hairs and recover the pixel at the respective address for the RGB value; it would thereafter convert this value to a hue (to encompass a range of RGB values, rather than a specific one).
  - pewPew
    - A sound module that will will initially play a simple sound repeatedly when the laser is firing. Counters will be used to play this snip, reminiscent of the 1970s star wars blaster rifle, on the go. The Wilhelm Scream will also be played after the laser is fired.

Advanced Level

- Goals
  - Add more sophisticated sounds into flash memory that will play on queue.
  - A specific hue will be selected by the user using cross-hairs that are controlled via the Labkit buttons and displayed on the monitor.
    - From this, the distinct object that has the same hue as the cross-hair target will be aimed at by the laser.
- Major Modules
  - music
    - Instead of playing a short sound byte, a song will be played. This will be the Star Wars Cantina Theme, upon firing the laser it will be turned off. This will be stored in the flash memory.
  - chromaKey
    - This module will be expanded so that it will take the cross-hair target as input, thereby converting the RGB value of that pixel to a hue.
    - From this hue, the laser will be able to fire at whatever distinct object we lock on with the cross-hairs.
  - crosshairs
    - Must add a vertical and horizontal line that will aid *chromaKey* in selecting a RGB value to be used in identifying the target. These cross-hairs can be moved via buttons on the Labkit.
    - Extensively, the RGB value for the pixel that it's locked on can be shown on the hex-display.