6.111 Final Project Proposal
Danh (Danny) Vo and Weijie (Jeff) Yuan
November 2, 2006

1. Project Overview

This proposal details the design of a vertical-scrolling shooting game, in the style of the popular arcade game Galaxian. The player controls a spaceship that is situated at the bottom of the game screen, and attempt to destroy a horde of alien invaders at the top half of the screen by shooting projectile at them. Occasionally, some of the aliens will fall toward the player's ship in a kamikaze attack. The objective for the player is to destroy all aliens on the screen.

The 6.111 labkit will be used for the implementation of the core game logic. Specifically, the video module is required for its VGA connection to the LCD monitor. Also BRAM and ZBT RAM will be used to store game states, and processed inputs.

The user interface will allow the player to control the ship using hand movements in front of a video camera. The images of the user's hand will be passed into the labkit memory, and by comparing successive frames, the relative motion of the hand can be detected. Thus, the ship on the screen will mirror the player's hand movement from left to right. This style of controlling is somewhat unique, and should be an interesting exercise to implement. Alternatively, a keyboard connected to the PS/2 port of the labkit can also be used as a controller. When certain keys on the keyboard are pressed, they will be detected, and passed to the game logic.

2. Module Specifications

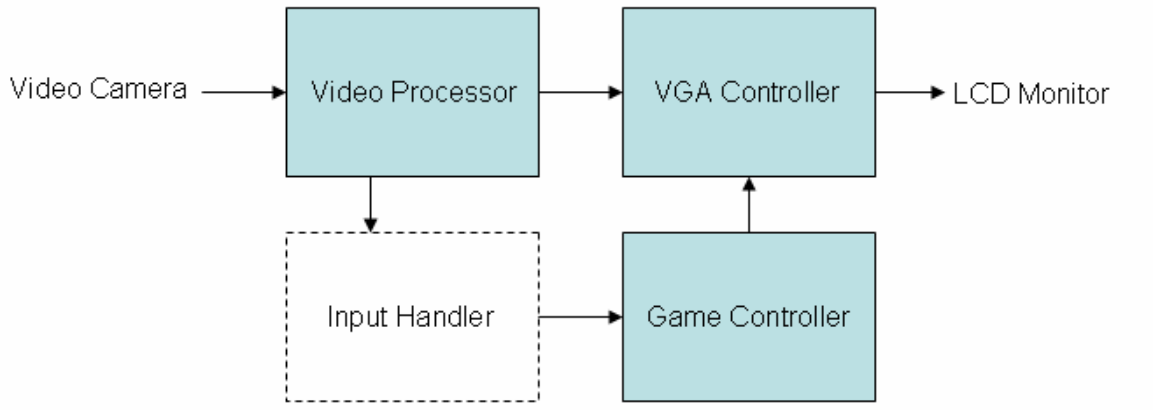

Figure 1: The four main building blocks of the project are shown above. The functionality of each is discussed in the paragraphs below. The Game Controller block is also shown in more detail in figure 2.

The implementation of the project is broken up into four major blocks. Each of the blocks handles a different set of functionality. The biggest and most important of these

blocks is the Game Controller, which handles the core game logic. Below is a description of each block:

2.1 Video processing module:
This module takes in a stream of pixels from the video camera in the YCrCb color space and stores them in the ZBT RAM. From the stored image in memory, it tries to detect the location of the player's hand by using the difference in color intensity between the hand and the background. Because noise is introduced by the synchronization between the video camera and the labkit, the hand's location will be computed as an average of over four successive frames.

2.2 Input handling
This module is used to switch between keyboard controller and video processed hand-movement controller. In principle, only a multiplexer is needed to handle the switching between the two control signals. However, this module will also contain some debugging features, such as displaying the video feed from the camera, and showing the center of mass of the detected hand. This will allow for easier testing of the video processing functionality.

2.3 VGA controller
This module read the pixel value stored in the BRAM frame buffer (section 2.4.6) and output it as a 24 bit RGB signal to the LCD monitor.

2.4 Game controller

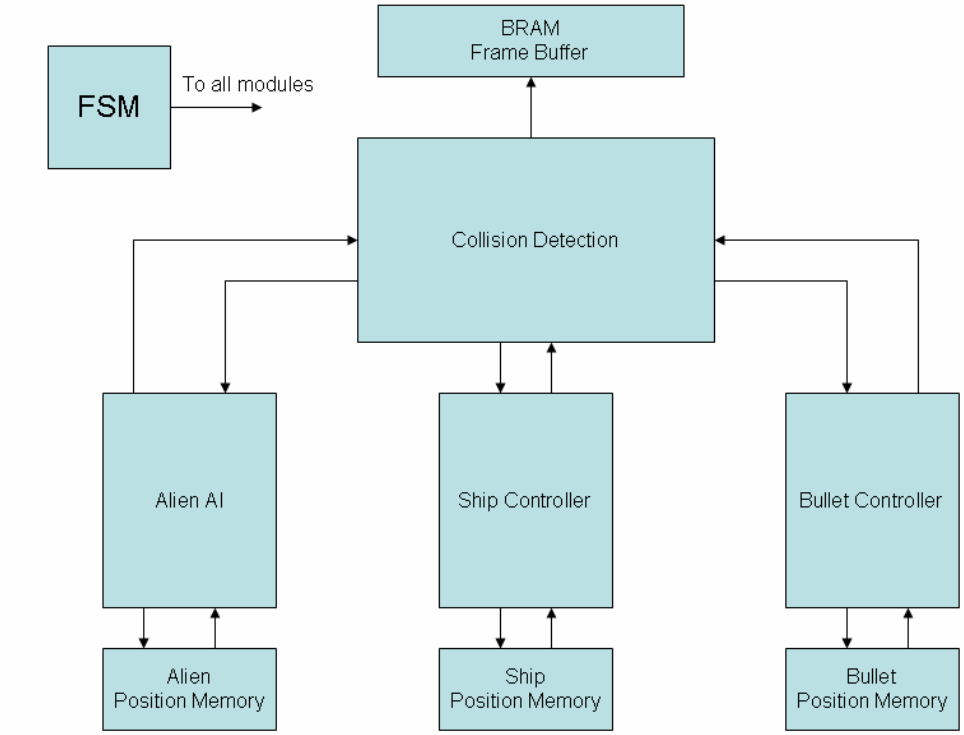

Figure 2: Overview of the modules in the Game Controller block.

This block is the largest and most important part of the project. It is responsible for generating the game logic and processing the player inputs. The game controller is actually made up of several modules and memory units:

2.4.1 FSM:
The Finite State Machine is the heart of the game controller. It specifies which mode the game is currently in. For example, whether the game has started, ended, or currently in progress. The FSM will also handle states within the game; for example, when a certain number of aliens have been killed by the player, it could indicate a new state where the rest of the aliens will be behave more aggressively. The detailed functionality and the exact number of states have not yet been mapped out. But conceptually, it is clear that the FSM needs to be connected to every module in the game controller block since it is the main "decision maker."

2.4.2 Collision Detector:
This module detects when objects in the game will collide into one another, according to their relative position and velocities. There are generally two types of collision that will be detected: alien into ship (ship will be destroyed), and bullet into alien (alien will be obliterated). Using the inputs from the Alien AI, Ship Controller, and Alien Bullet Controller modules, the Collision Detector will determine if any collision will take place in the next frame. It outputs this to the FSM, which can determine if a state transition is necessary. It also outputs to the three modules mentioned above, allowing them to update the list of existing objects in their respective memories.

2.4.3 Alien AI
The alien is capable of two behavior patterns: either staying at the top half of the screen, moving slightly from side to side, or diving down in kamikaze fashion at the player. This module handles the behavior pattern of each alien sprite on the screen. It also interfaces with a memory which will store the locations of each alien object in the game.

2.4.4 Ship Controller
This module serves as a controller to the memory unit which holds the ship's position. This is relatively simple, as there is only one ship in the game. The ship is also controlled directly by player inputs, so this module will only need to respond to player commands, which are taken in as input.

2.4.5 Bullet controller
Similar to the ship controller module above, this module controls the memory unit holding the bullet's position. In this game, we restrict the number of bullets that can appear on the screen at any one time to be one. This is consistent with the gameplay in the original Galaxian game, and also decreases some complexity in calculating the collision.

2.4.6 BRAM / Frame Buffer

This BRAM is used as a frame buffer to store the pixel values that is to be outputted at every location on the monitor. The VGA Controller module connects to the BRAM and reads pixel value and outputs to the monitor. The workflow for the video output to the monitor can be seen as follows: the Ship Controller, the Alien AI and the Bullet Controller modules will output their pixel color and screen location to the Collision Detector module. The Collision Detector module will write this pixel color to the ZBT RAM with an address that is a combination of the pixel's vertical and horizontal position. Two memories will be used for this, one for reading and the other for writing; a small module will control the swapping between the two.

3. Anticipated Challenges and Potential Issues

One possible problem in this project is the availability of ZBT RAM. Because the video processing block needs some amount of ZBT RAM to stores images, it is unknown whether enough will be available for the game logic to store and retrieve sprites. As more detailed plans emerge in the coming week, adjustments may need to be made in order to work around this issue.

4. Testing

Individual modules will be rigorously tested by simulation to ensure that they work according to specification. Certain modules, such as the video processor, will have to be tested manually by loading the bit files onto the labkit and seeing the result. Other modules, mainly related to the core game logic could be tested using logic analyzer. After completing unit testing, the main blocks will be connected and tested as a unit. Finally, the entire project will be integrated and tested with a logical analyzer. Of course, the main test for the project will be to load it onto the labkit and manually testing its functionality.

5. Work Division

Danny will implement the video processing, hand movement recognition, and the VGA display interface. Jeff will work on main modules of the core game. This division of labor is still tentative, and is subject to change as more details of each module are fleshed out.