Daniel Hawkins
Anna Waldo

Asteroids

6.111 Final Project Proposal

For our final project, we plan to implement a game based on the classic arcade game *Asteroids*, in which the player controls a spaceship in the center of the screen that shoots at and attempts to destroy incoming asteroids. The player passes each level when all the asteroids in the level are destroyed. When the player first hits an asteroid, it will break into two smaller asteroids, and each of those will break into two smaller asteroids when they are hit; shooting these ¼-size asteroids destroys them. The movement of the spaceship will be controlled by an accelerometer that will be physically manipulated by the player. An additional button will control the spaceship's guns. The game will also feature sound outputs for different aspects of the game. This project shows off various aspects of digital systems that we have learned in class, as well as analog systems (through the accelerometer and sound outputs) in a fun and engaging manner that people who may be unfamiliar with the project details can still link to video games they are familiar with.

The major components of the Asteroids game are the accelerometer and button inputs, controlling the spaceship, detecting collisions, video output, and sound output. The analog signals from the buttons and the accelerometer will have to be converted into digital signals, which will then feed into the Game Logic module to change the angle of the spaceship. The button input will indicate whether or not the player is attempting to shoot, and the collision of bullets with asteroids will determine when the asteroids should be destroyed instead of continuing along their path. The button input may also indicate a game reset, which will restart the game. Using a debouncer similar to what we used in previous labs, we will create a clean signal from a button-press that will feed into the game. Various components from the game, such as shooter button presses and the destruction of an asteroid will determine the sound output. A block diagram of the game's modules is shown below in Figure 1.

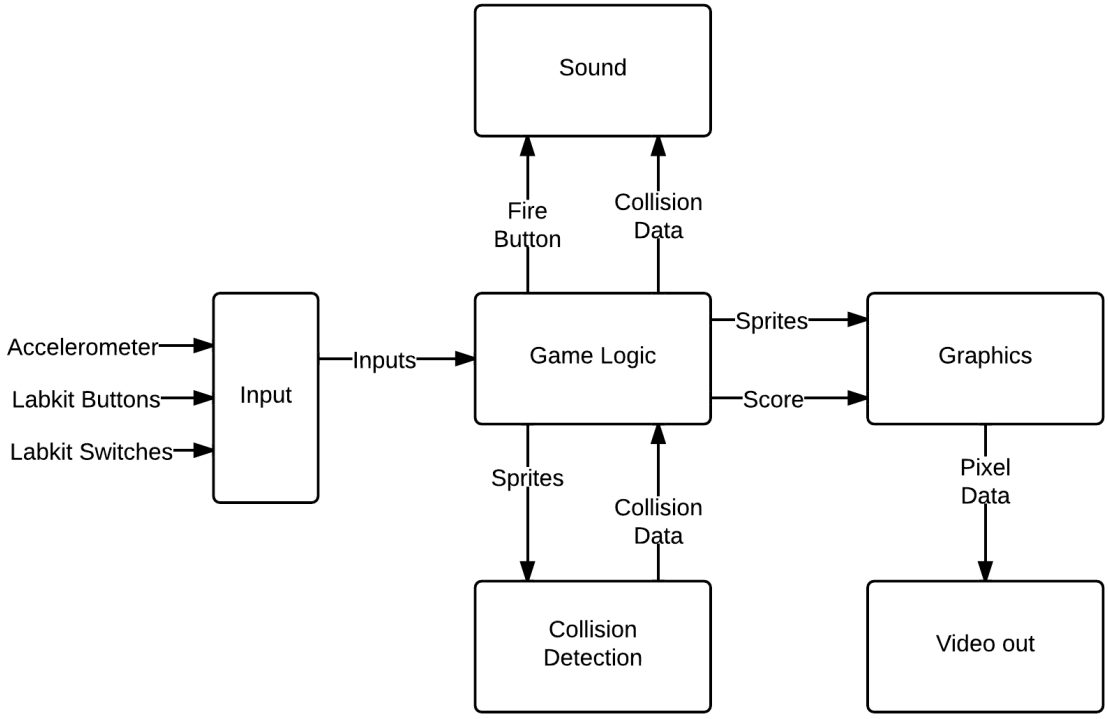

Figure 1 - Block diagram of major modules and their respective inputs and outputs for the Asteroids game implementation.

The accelerometer we will be using is a 3-axis analog output accelerometer. When the device accelerates along an axis, it outputs an analog signal. This signal will control the movement of the spaceship in the game. The primary function of the Input module is to take in the analog input from the accelerometer and convert it into a digital signal that will then be fed into the Game Logic module to control the spaceship's movements. In our preliminary tests, we will determine the most reliable means of obtaining a useful output from the accelerometer, whether that may be rotating the accelerometer or moving it along a single axis. Because of noise and unpredictable hand movements, we will need to keep a moving average of the signals from the accelerometer so that the least significant bits are not constantly changing back and forth and causing jittery spaceship motion. We will also construct a Wii-mote style encasing for the accelerometer itself for convenience on the user's part.

We will use debounced buttons to control the spaceship's guns and the reset signal. The debouncers in the Input module will be essentially the same as the ones we have used in the labs this semester. They will take in signals from the buttons we are using, create a clean signal, and output the clean signal to the game. Rather than using buttons on the labkit itself, we plan to use a separate handheld device with buttons on it.

The Game Logic module will take in clean inputs and update the state of the game, including the position, speed, and direction of all the objects on the screen. It will feed new object positions to the Collision Detection module, and upon collision of a

bullet and an asteroid, or the spaceship and an asteroid, it will update or reset the game's state accordingly. This module will also pass along signals to the Sound module to trigger appropriate sound output, and the game score and sprites for each of the on-screen objects will be passed to the Graphics module to make sure the screen accurately displays the state of the game.

The Collision Detection module will take in information about the type, size, and location of objects on the screen and produce an array of collision information based on the overlapping of bounding boxes. This information will be used to update the game's state and to trigger sounds for important events in the game.

The Graphics module will draw sprites for all the game objects and the user's score, based on information from the Game Logic module. The sprites will be chained together and will output one pixel at a time to the Video display module, similar to the design of the 6.111 Pong game lab.

The Sound module will output sound data, which we will store on a ROM, to the AC97. At various points during the game, such as starting up, shooting, destroying asteroids, and an asteroid-spaceship collision, the game will play an appropriate sound. The sound will have to first be converted to a COE file, which we can accomplish with the MATLAB script created by previous 6.111 student Yuta Kuboyama. Once we have all the necessary files, they will be stored on the ROM for the sound module to access when needed. Depending on which inputs the sound module gets, it will output the appropriate sounds.