Massachusetts Institute of Technology

Laser Pinball

6.111 Final Project Proposal, Draft 1

Pauline Varley, Jake Isenhart, and Weston Braun

10-30-2014

Project Overview

Our final project is to create a pinball arcade game on a FPGA that is displayed on a wall through a laser projector. This project is motivated by a desire to use technology for enjoyment rather than real purpose; if we don't capitalize on the opportunity to do a "just-for-fun" project with the resources we have at MIT, we think we're doing something wrong.

The pinball system will consist of a camera interface (used to allow the detection of physical game elements that will be placed on the display surface) and image recognition in order to create the game field, a physics engine to control the game play, and a vector graphics engine. The game will be displayed through a RGB laser projector which will be constructed for the project. We think that these design choices will result in the most entertaining and playable game.

We expect that most of the hang-ups in our project implementation will occur during the construction of the physical interface and laser display modules (discussed in depth in a later section); while we want our display to be laser-projected for the flashiness factor, handling the vector drawing and the limitations inherent to the galvos will be challenging. Additionally, describing the physics of the game may be more mathematically complex than we expect it to be and might require some cleverness in parallelization. Finally, we'll be using a Virtex 5 board rather than the Labkit, which gives us more flexibility to interface with external modules but may have somewhat of a learning curve in getting started.

System Diagram

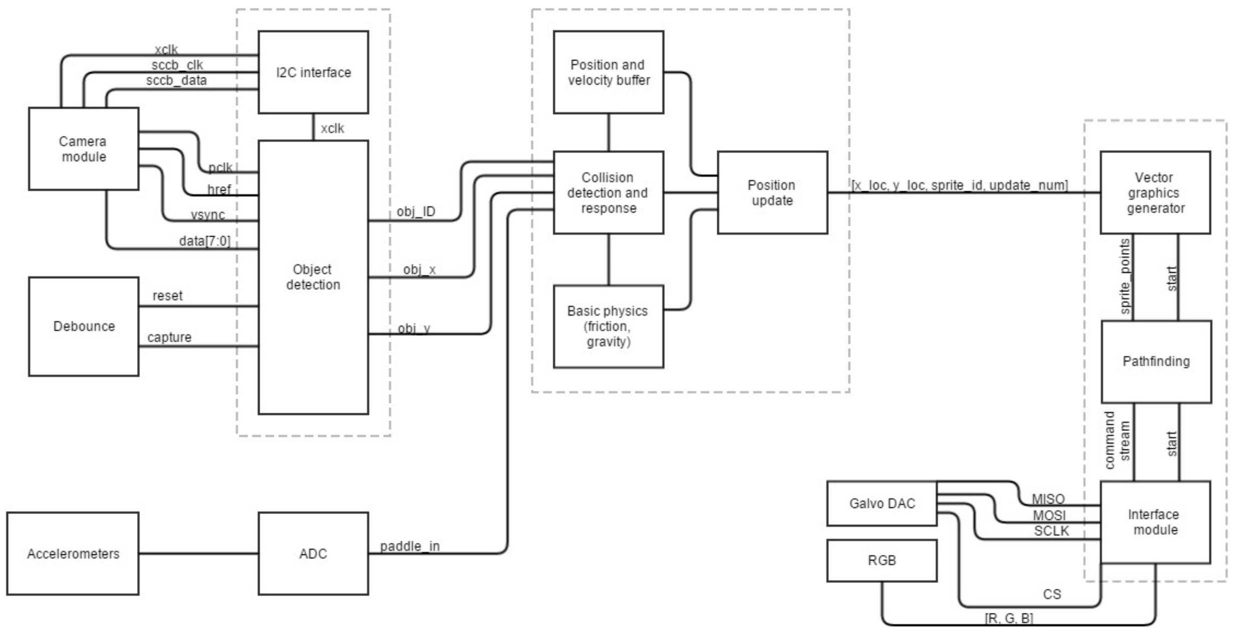

Figure 1: Full system diagram for Laser Pinball. Specific sections of the diagram will be discussed in more detail later.

System Design

Vision Processing

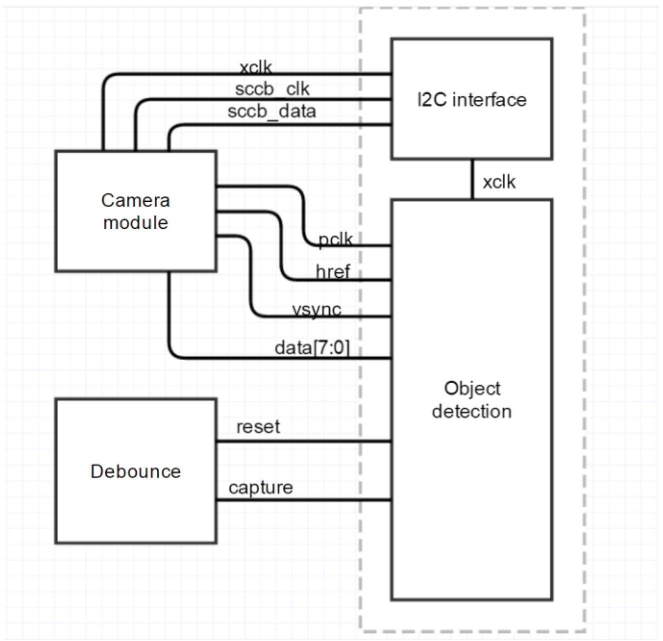

Figure 2: Camera interface and image psocessing module. Communication is done via both a parallel bus (for camera data) and an I2C-like interface.

In order to allow for a user-reconfigurable game board, the system will use a CMOS camera to capture an image of several movable colored objects, each of which will represent a game object. The user will be able to move these objects within the frame, then push a button to capture the game when ready. The camera will be an OV7670 (or similar), which can be used with I2C to control its internal DSP module and has an easy-to-read clocked parallel data interface that's extremely similar to VGA. Once the "capture" button is pressed, the vision processing module will read a single frame of data and look for the boundaries of the colored objects. The objects will be placed on a white background so that the color boundaries are clear; each object will also be rotationally symmetric for easier processing.[1]

The vision processing module will interface directly with the camera and will active on "capture" button press. It will buffer each pixel as it reads from the camera, then post-process the data to extract the center coordinates of each object, expressed as two 9-bit integers describing the x and y offsets into the frame. The color correspondence of each object will be fixed and estimated to within some error margin (to be determined during camera testing); the size of each object will be used to determine a scaling factor on the coordinates, and the edges of the white background will be used to determine coordinate offsets. The module will output a "ready" signal once it has parsed the board; at the request of the display module, it will then output in parallel the x and y coordinates of the requested object, each of which will be assigned an integer reference.

Physics and Game Engine

[1] If there is time, objects will be made rotationally asymmetric and the vision processing module will also extract rotation information.

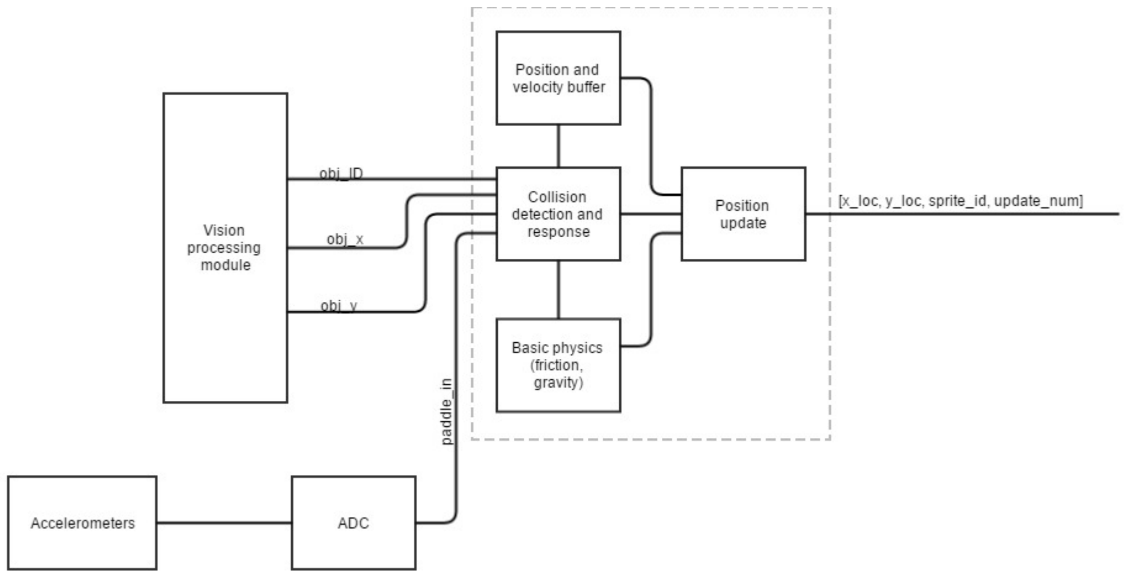
Position and velocity buffer
Vision processing module
obj_ID
obj_x
obj_y
Collision detection and response
Position update
[x_loc, y_loc, sprite_id, update_num]
Basic physics (friction, gravity)
paddle_in
Accelerometers
ADC

Figure 3: Physics and game engine, with connections to other modules and hardware inputs. This engine drives the game's behavior and its implementation will be predominantly composed of parallel mathematical calculations.

The physics engine will generate the coordinates of the ball based on its interactions with the game objects and pass those coordinates on to the vector graphics module for display. Each game object will have an associated behavior that affects how the ball will interact with it--for example, one object might slow the ball down on a rebound, while another might slingshot it. The physics module will keep a circular buffer of the ball's trajectory and use the values in this buffer to determine angle of approach, velocity, and acceleration when a collision is detected; the behavior of the ball after a collision will depend both on these values and on the specific physics of the object with which is collides.

Beyond collision behavior, the motion of the ball will be governed by simple acceleration and friction rules. We will store the ball's position and velocity in memory, and update each frame with simple arithmetic operations. If time allows, we may have multiple balls or other game objects that move -- to the laser projector, the process of updating won't be any different. The physics engine will update a frame buffer as fast as possible, and the laser will simply read frames out as needed. At this point, we plan to represent game objects as points -- the laser projector module will have a lookup table of sprites which it will use to draw game objects based on single coordinate an a defined vector path for that particular sprite.

The board will be configured before each game, with a poll to the vision processing module for a list of points and object types.

Vector Graphics Engine

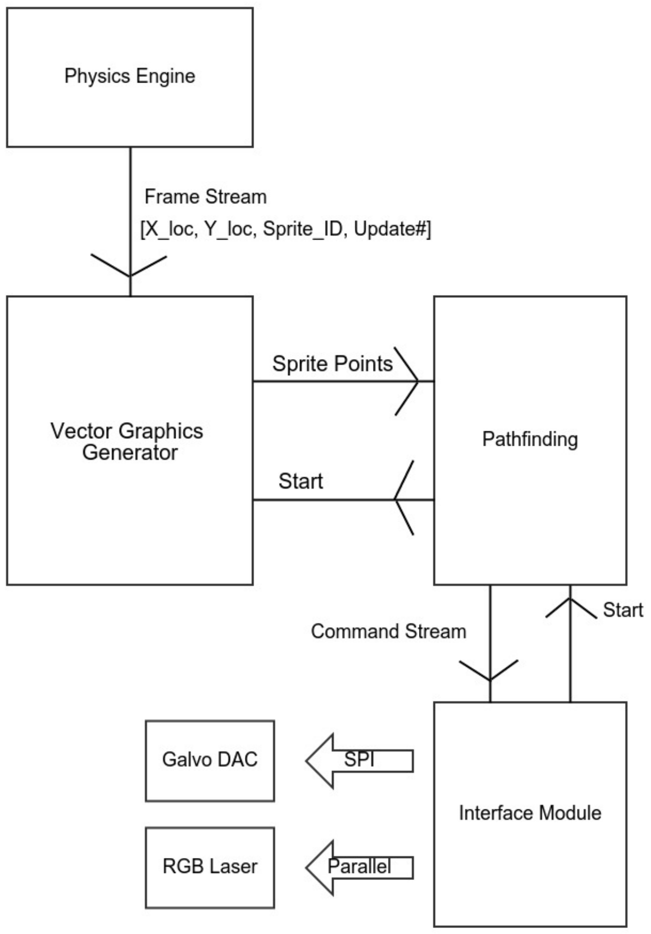

Figure 4: Physical interface, vector graphics, and pathfinding modules to drive the laser display.

Generating vector graphics for a laser projector is complicated by the fact that time for the laser projector to draw a frame is non constant. It is instead dependent on several physical constraints of the laser galvos and the number of segments in the frame. Additionally, there is a minimum frame rate constraint to prevent visual flicker.

To overcome these constraints, the vector graphics engine generates a new frame whenever the laser projector become idle. To ensure accurate graphics, the physics engine sends location data to the vector graphics rate at a rate much greater than the maximum effective frame rate. The vector graphics engine then captures an input frame whenever a new frame needs to be drawn.

The vector graphics engine is responsible for transforming the sprite IDs and location values from the graphics engine into coordinate locations for the pathfinding module. The vector graphics engine stores the sprite representations, which consists of bounding points, and a color value. The vector graphics engine uses these stored sprites and the information generated by the physics engine to generate a frame consisting of absolute coordinates that are ready to be plotted.

Due to the limitation of the galvos, the generation of frames is triggered by the proceeding pathfinding and physical interface modules.

Pathfinding

Generating the optimum path for the galvo is crucial for optimal image quality and high frame rate. This is required due to the limited travel speed between sprites and the non-ideal step response, which will distort sharp corners.

The pathfinding module translates the coordinate sets generated by the vector graphics engine value into an ordered set of coordinates suitable to be sent to the laser projector. The pathfinding module optimises the galvo path for the least amount of travel distance and add acceleration/deceleration for corners. The module creates a set of laser and galvo commands which are queued up as a frame to send to the laser projector. This data is send to the physical interface module once it is done plotting the previous frame.

Laser Display

The physical interface module controls the galvos and RGB laser that comprise the laser projector. Commands from the pathfinding module are queued and executed at a regular rate to drive the projector. When a frame of commands is completed, the next is requested by the pathfinding module.

The RGB laser used for the project has 1 bit resolution per color, and is controlled over a 3 bit parallel interface. The galvos have 12 bit resolution and are controlled over a SPI interface.

User interface

Given the time, the user input to the game will be supplied via a pair of accelerometers attached to gloves. Controlling the flippers and potentially other game objects with gestures should be relatively simple to implement via polling and limited signal processing by the FPGA through a pair of ADCs. Alternatively, the flippers can be controlled very easily through debounced buttons.

Resources

Below is a list of the physical resources we will need outside of what the course can provide.

1. Virtex 5 development board
2. CMOS camera (OV7670)
3. RGB laser
4. Galvos
5. Accelerometers
6. 10-bit ADCs
7. Power supplies for galvos and laser
8. Physical housing for galvos and laser

All of these components are either in hand or on order as of the writing of this document.

Weekly timeline

By color: Jake (green), Weston (red), Pauline (blue), associated RGB combinations

	10/27	11/3	11/10	11/17	11/24	12/1	12/8
Proposal and planning	⬜						
Source components	⬜	⬜					
Camera interface and object recognition		🟦	🟦	🟦			
Physics design and development		🟩	🟩	🟩	🟩		
Camera and game engine interface				🟦	🟦		
Galvo testing			🟥				
Game engine and display module interface				🟪	🟪		
Pathfinding		🟥	🟥	🟥	🟥		
Bonuses (calibration, sound effects, accelerometers)					⬜	⬜	⬜
Debugging and final touches					⬜	⬜	⬜