Fpglappy Bird: A side-scrolling game

Wei Low, Nicholas McCoy, Julian Mendoza
6.111 Project Proposal Draft, Fall 2015

1 Overview

On February 10th, 2014, the creator of *Flappy Bird*, a popular side-scrolling game for mobile devices, removed the game from mobile application stores (Apple, Android, etc.). The reason for the removal was due to the game becoming "an addictive product" that had "become a problem" (Nyugen). Currently *Flappy Bird* is only available to those who downloaded the game before its deletion from application stores or through a fan-created website, flappybird.io.

This final project aims to implement this popular and exciting game in hardware. Previous implementations of this game, such as the original mobile app or fan-created online version, exist in software form. As a result, bringing this game into the hardware realm truly distinguishes our project from previous implementations of *Flappy Bird*. In the preexisting software-implemented versions of this game, a small bird must hop to avoid obstacles in the form of green pipes that scroll across the screen from right to left. The user taps the screen or clicks a button to make the bird jump up, but otherwise the bird is constantly falling. Due to the counter-intuitive control scheme, which required the user to remain vigilant, constantly monitoring the bird's path of movement during the game, *Flappy Bird* became renowned for its difficulty and infuriating controls.

Our goal is to implement our own version of the game on an FPGA (Nexys 4 DDR) and make the game more interesting and difficult by incorporating a vision-tracking element that requires the player to jump in order to control the on-screen bird protagonist. The added vision-tracking component will continually track the player's movements, resulting in greater emphasis on the player's attention on the game. By requiring the player to physically jump to control the bird on-screen, the player will receive an entirely different game experience from the original. Additionally, during game play, the player's face will be layered onto the bird sprite, allowing for a truly personalized experience.

Stretch goals of this project include implementing our project on two FPGAs connected via serial link to create a multiplayer experience and dynamic calculations that allow rotation of the bird sprite during jumps. The ideal outcome would be to deliver an aesthetically pleasing and physically engaging game that demonstrates successful implementation of the game integrated with the vision-tracking component.

2 Design

2.1 System Overview

The project can be visualized in four major blocks as shown in Figure 1: object tracking, gameplay logic, audio, and video display.

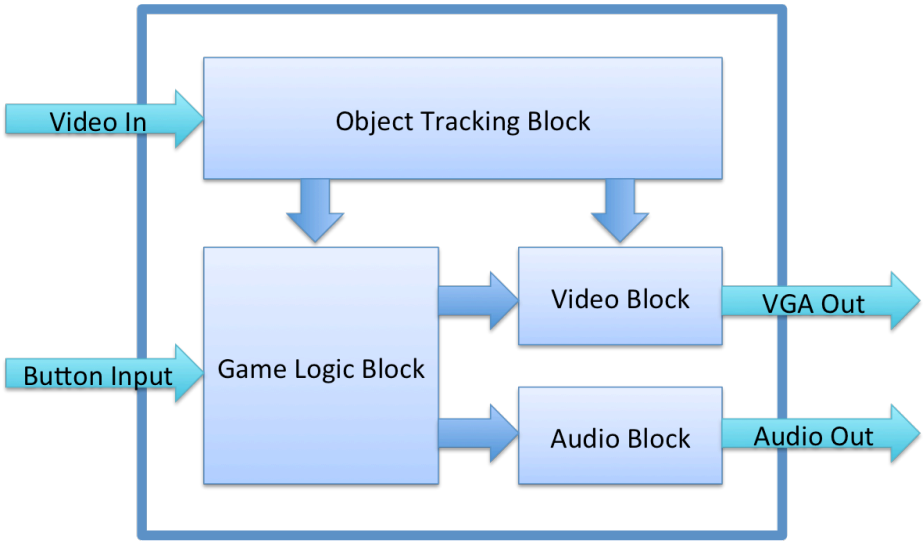

Figure 1: The high level design of the game

2.2 Design Decisions and Motivation

The primary motivation for this project is to make a well designed and physically engaging game that emphasizes and personalizes user experience. Completion of this project requires full functionality of every module. As a result, we require that each of the four critical blocks be independent of the others, allowing for streamlined parallel development and comprehensive testing of the modules. For object tracking purposes, the player is required to wear a “beak”, a brightly colored party hat over their nose and mouth, similar to a mask. The purpose of the physical “beak” is to provide an object of high contrast that allows us to easily determine the location of the player’s face (Figure 2 below).

Figure 2: “Beak” on player face

At the highest level, the project will track the player’s face during gameplay and output sound effects (Audio Out) and video graphics (VGA out). Depending on the player’s actions, game logic processing will change and update the sound and graphics. The vision/object processing block takes input from the OV7670 camera and processes the incoming video stream to determine the location of the “beak” and then sends the location coordinates to the game logic block. The game logic block uses an input button to start the game, and uses a combination of physics and saved previous states to determine when to jump. Additionally, the game logic block handles the location of new obstacles, and controls the movements of the bird. If the bird collides with an obstacle, it ends the game. The game logic then sends position data of objects, specifically the player face and obstacle location coordinates, to the video block, and also tells the audio block what sound effects to play. The video block takes input of object positions, converts them into image representations, and sends them over VGA. The audio block plays sound effects based on events determined by the game logic.

3 Implementation

3.1 Object Tracking Block (jmend)

The object tracking block contains three submodules: preprocessing, object identification, Kalman Filter (Figure 3 below).

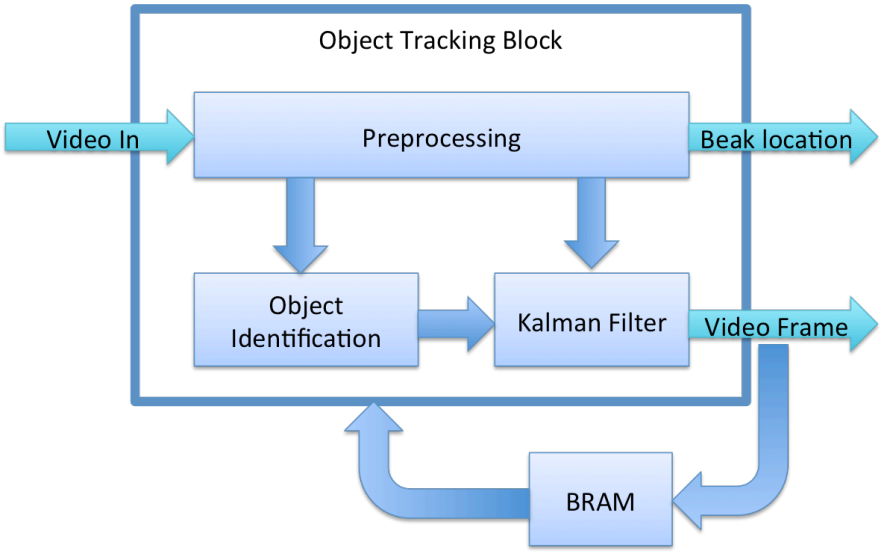

Figure 3: Object Tracking Block with submodules

To reduce data bandwidth issues that graphics processing creates, the object tracking block operates at 10 frames per second. Additionally, due to the large size of each color image frame from the OV7670 camera, we cannot store the original image in BRAM. Instead, we opt to perform preprocessing on the individual pixels from the incoming camera stream rather than storing the entire image frame and then do preprocessing, to significantly lower the amount of data stored in BRAM. During the preprocessing stage, we also store a copy of the image in greyscale in BRAM, so that we can later display the face of the player. After preprocessing, the object identification submodule searches for and identifies the bright and highly contrasting object of interest (the "beak" worn by the player) in the image. The center of mass of the object is calculated and its location passed through a Kalman filter to smooth out noise produced by the environment and camera reading.

3.2 Game Logic Block (weilow)

In total, there are three submodules for the gameplay: game state, physics and high score module (see below).

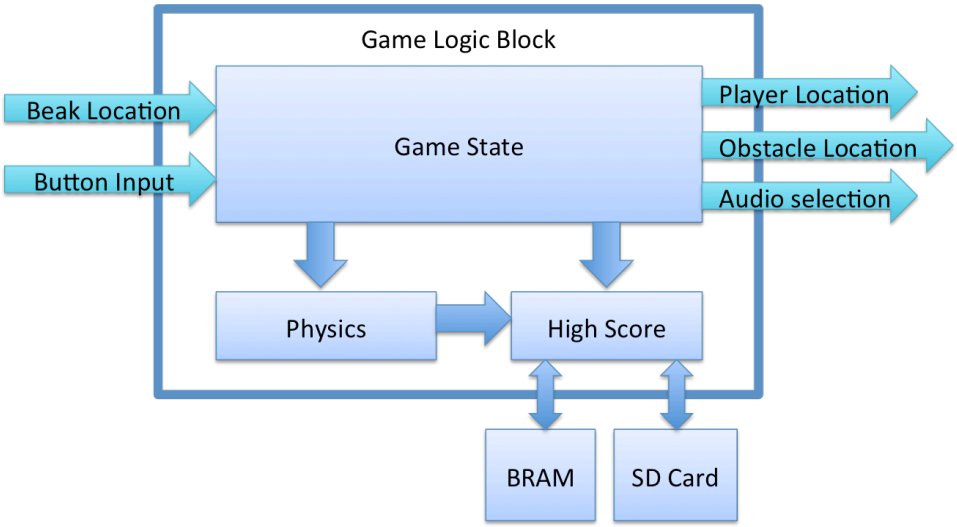

Figure 4: Game Logic Block with submodules

The gameplay will be similar to that of the original *Flappy Bird*. This game logic block takes input from the object tracking block on the coordinates of the beak to determine the velocity and acceleration of the bird. Additionally, the game logic block converts the beak location into player coordinates for the video block. This block also relies on previous player locations saved in memory to make the distinction of whether or not the sprite has bumped into an obstacle (a "pipe"). The physics submodule will handle sprite movement, such as jumping and falling with gravitational effects. The game state module will handle the different states within the game, which will be: START, PLAY, PAUSE, LOSS, HIGH_SCORE. Within the START state, the game's default screen will load, bearing the game title. Upon pressing the ENTER button, the PLAY state will begin. During the PLAY state, if the bird hits an obstacle, the game will transition to the LOSS state. After a specified amount of time, the game will transition from the LOSS state to the HIGH_SCORE state. In the HIGH_SCORE state, a leaderboard of the top ten scores will be displayed. If the player achieves a score eligible to be included in the top ten scores, he/she will have the option to input initials and the high score submodule will save his/her score on the SD Card.

3.3 Video Block (nmccoy)

The video block has two submodules, the sprite pipeline and VGA controller (see figure below). This block takes input from the game logic block on the player coordinates (location of the bird), obstacles (up to 3), total distance, and whether the game has ended or not. The coordinates of the player's face are needed to extract an image for the player face sprite from the BRAM. This block will use a sprite pipeline system, described in lab 3, where data is passed through blocks for each sprite to give a layering effect. The VGA controller submodule generates all signals/clocks needed for a VGA output, and passes the sprite pipeline data out to the display.

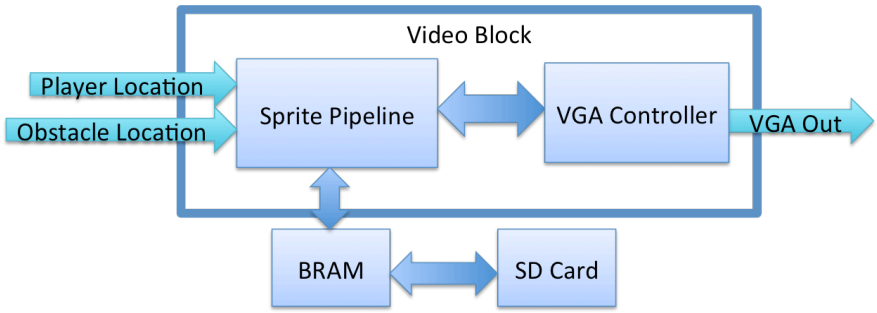

Figure 5: Video Block with submodules

3.4 Audio Block (nmccoy)

The audio block takes input from the game logic block on when to play certain sounds. It will have an input bit for playing each possible sound (jumping sound, crashing sound, background music). When prompted, the audio controller will load the respective sound from the SD card and output it over the audio DAC. In the case of the background music, the file will be looped continuously as long as it is enabled.

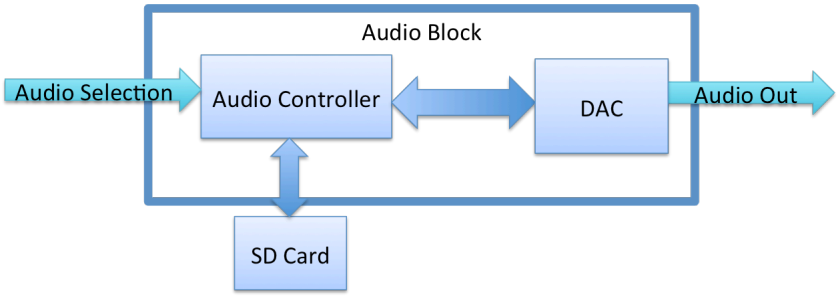

Figure 6: Audio Block with submodules

4 Schedule

Because there are three members working on this project, sections of the project can be completed in parallel. The GANTT chart below provides the proposed schedule for project completion. The green represents the Development stage, orange represents the Testing and Integration stage, and blue represents the Final Stage. Task assignment is done with labels of “All”, “J”, “N” or “W” which represent all team members, Julian, Nick and Wei respectively.

Task	11/1	11/8	11/15	11/22	11/29	12/06
Interface with FPGA	All					
Object Tracking Module	J	J	J			
Audio/Video Module	N	N				
Game Logic Module	W	W				
Preliminary Testing		All	All	J		
Integration: Game Logic, Audio Video			N, W			
Integration: Object Tracking			All	All		
Testing				All		
Buffer Time/Stretch Goals					All	
Demo/Final Presentation						All

Figure 3: GANTT Chart of Proposed Schedule

Below we elaborate on the various tasks within the proposed schedule in Figure 3.

- Interface with FPGA - Interfacing between the camera input, VGA output and Audio output will require attending tutorials and reading hardware documentation for the camera of choice, SD card and storage modules on the FPGA
- Object Tracking, Audio/Video, Game Logic Module - Done in parallel by the team
- Preliminary Testing - Testing of individual components within each of the respective modules will be done independently and throughout integration
- Integration of Game Logic and Audio/Video - The bulk of the integration will occur in week 3, to ensure states and variables invoked within game logic correspond to the correct video and audio outputs (Optional Modules will be implemented at this time for advanced game play)
- Integration of all Modules - Integration for all modules will occur over two weeks
- Testing - Testing of the system as a whole will occur in lab
- Buffer Time for Testing - Buffer week added for testing
- Demo/Final Checkoff - Week of demos, final check off, and paper.

5 Testing

Each of the four main modules can be tested separately, allowing for parallel development of the project. Testing for each submodule with the main modules requires writing a test bench in Verilog and running the simulation in Vivado. As implementation becomes a larger goal, test benches will also be created for combinations of submodules within blocks. The actual modules will be primarily tested in hardware, due to the utility of using VGA output for the object tracking, video display and gameplay blocks, as well as the sound output for the sound block.

5.1 Object Tracking Block

Object tracking testing can be split up into two large parts, Target Representation/Localization and Filtering/Data Association. Blob tracking can be tested either by passing in test video data and testing its detection/tracking capabilities. Filters can be testing using fake noisy motion tracking data and measuring its smoothing effect.

5.2 Game Logic Block

For the gameplay logic block, isolation from the rest of the modules can be accomplished by using the buttons and switches on the Nexys 4 to simulate different signals of events occurring.

5.3 Video and Sound Block

The video module can be built and tested without depending on the gameplay module. Testing would consist of using the VGA output to see if the proper items are displayed on the screen, as well as testing the different interactions between the objects on the screen and the outputs to the game logic.
The sound block is the simplest to separate from the rest of the modules. Testing will primarily occur on the Nexys 4 and include outputting different sounds depending on varied test input signals.

Conclusion

Implementing this game in hardware would be an enjoyable project for all of us. It pairs a fun final product with a challenging implementation, an ideal combination for a project in this class. The work is modular and divides well between three people, so we can optimize the division of labor. We have a schedule planned out, and all goals seem achievable. Overall, this project will help all of us gain experience working with hardware implementation and interfacing with the outside world.

Resources

Besides the provided Nexys 4 FPGA board, OV7670 camera, and lab station complete with logic analyzer, we will need the Github to regulate version control since three people will be working in parallel. Additionally, our object of high contrast will be in the form of a solid conical party hat acquired from Amazon.

Citations

Nguyen, Lan Anh. "Exclusive: Flappy Bird Creator Dong Nguyen Says App 'Gone Forever' Because It Was 'An Addictive Product." *Forbes*. Forbes Magazine, 11 Feb. 2014. Web. 10 Nov. 2015.