SNAPPA REFEREE

Project Proposal Draft

Matthew Orton | Juan De Jesus

1. OVERVIEW

Snappa is a drinking game played by two teams of two that sit on opposite ends of a long rectangular table. The teams take turns throwing a die to score points, until 7 points are reached for a win. The die must travel in an arc and clear a loosely defined "low line" the four players agree on before the game. If the die bounces on the table and goes off the opponent's end of the table (exiting through the side of the table does not score a point) without being caught, the shooting team scores a point. Our project, Snappa Referee, not only aims to eliminate the difficulty of deciding whether the die passed the height threshold by tracking the movement of the die, but also enhances the experience with added sound and visual effects, by allowing replays of any plays, and a scoreboard to keep track of points in this game.

To achieve these goals, we are planning on using an FPGA-based motion-sensor referee. For our project, we will be utilizing a tennis ball instead of a die for motion-tracking simplicity. The ball tracking system is the most vital part of our project, as it compares the ball to its background to determine its position in space. Our Referee can then utilize this video input information to determine whether the ball traveled high enough to cross the predetermined low line. The integration of other modules allows for replays and sounds, further enhancing the gaming experience. These modules are explained in detail below.

Stretch goals for this project revolve about being able to decide whether the die in a Snappa game actually went off the opponent's end of the table (point) or the side of the table (no point), This can be achieved by modifying the Referee module to consider where the ball falls, plus attaching motion-sensors on the edges of the table to eliminate contentious edge cases.

1. DESIGN

When choosing a final project, we wanted modules that were closely tied to the skills we learned from our labs while also requiring us to break into new problems. In addition, we wanted to make sure the project stayed fun to work on, so we chose to design around a game to keep everything light hearted at a high level. Snappa is a good game for this project because it is visually simple for an FPGA to be able to extract the relevant visual information and because there are ways in which an impartial observer could improve the quality of the game.

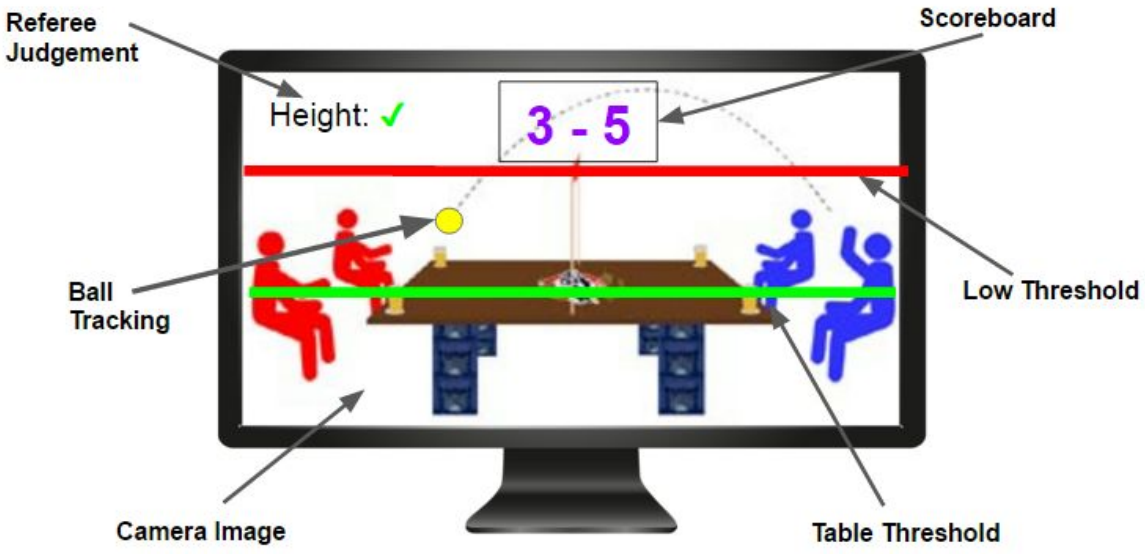

Figure 1: This is a basic representation of what our VGA output will look like.

2. 1. DESIGN CHOICES

One of the primary concerns in designing a Snappa Referee was to make sure that our Referee would neither be a drag to use, nor take away from player's autonomy. We achieved this in a variety of ways. One of these was by allowing the players to be the ultimate judges of the validity of a throw by updating the scoreboard manually.

Another way in which we made the Referee a more informative guide was with a replay option. Reducing the system's interference with the original gameplay was most difficult when designing the replay module. We wanted the FPGA to know when to start and stop recording a shot with as little player input as possible. Ideally, there would be no required player input, but we realized it would be easy to confuse the system into thinking the next shot was starting very shortly after the previous shot had ended. While this could be worked around with a delay, we decided the system would be more robust if players provide a button push to signal the start of a shot and the FPGA determines the end of a shot. This does not alter the flow of the game much because offensive players must wait for the defense to be ready before throwing anyways.

2. 2. ENHANCING GAMEPLAY

In a traditional game of snappa, the defensive team must call the shot low before the play has ended, so to simulate this, our system will play a sound clip signifying a low throw once the ball is moving down after not crossing the low line. An indicator corresponding to a low throw will also be displayed on the monitor along with an indicator corresponding to a throw going off

the side of the table, but no sound will be played to indicate side. This is in part because our current Hue ball-tracking technology cannot discern effectively whether a ball exits through the end or side of the table unless our stretch goal is met of making corner-case detectors. Because the defensive player will often have a better perspective than a side-view camera as to where the ball left the table, this is not a major concern for our project.

The primary output for this system will be a display of the camera input with additional information. The threshold lines (Background Recognition Block) and the scoreboard will always be displayed on top of the background. After a throw, the display will indicate whether or not the shot was high enough, and if the system thinks the ball left the table from a side edge or the back edge. Additionally, the ball will be highlighted on the display so it can be more easily seen by the players during replays or while a throw is happening.

2. IMPLEMENTATION

This section will explain the details of how our project will be implemented, including the modules which are summarized by Figure 2.

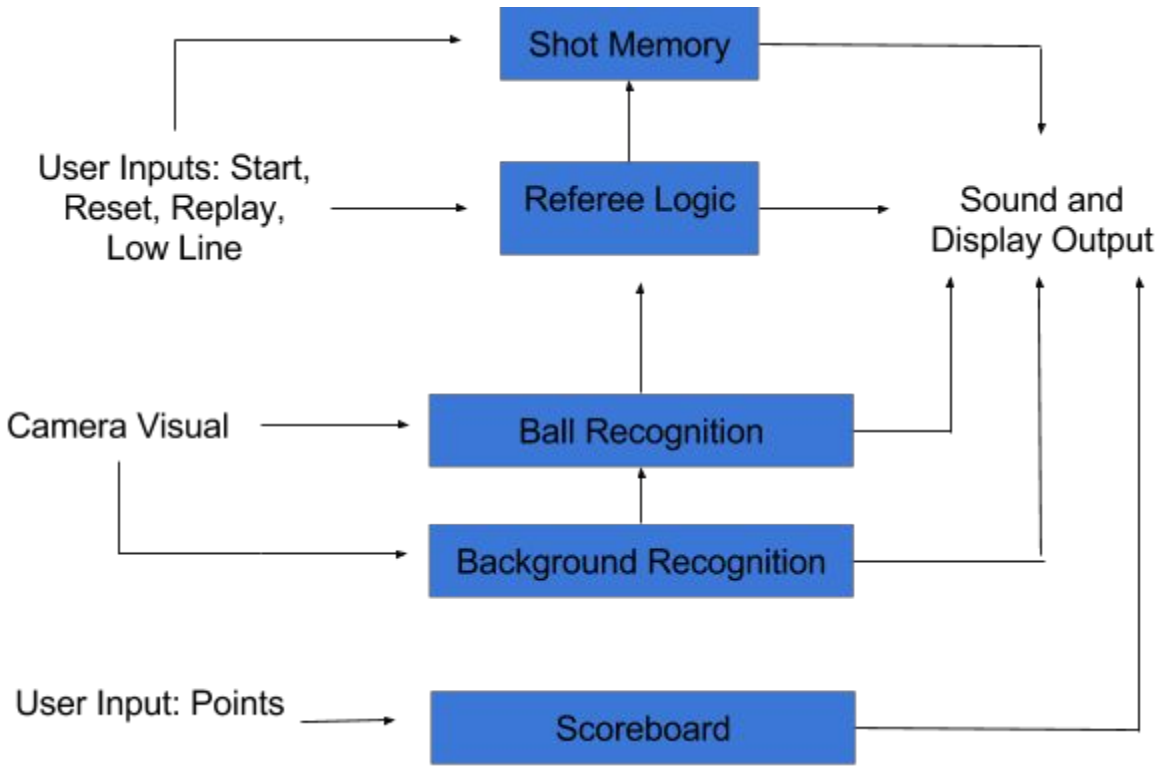

Figure 1: This is a system block diagram. Modules are displayed in Blue. Note: Referee Logic is our Finite State Machine

3. 1. Shot Memory Block

This is the cornerstone of our project's ability to replay any throw. As explained in the Design section, once the player pushes the "Start" button to begin a throw, the system will switch into the recording state, and begin storing video input (in the form of hue bit numbers) as Addresses, overriding the oldest video input once memory is filled. Recording only stops when our FSM switches from a recording to an idle state, signifying the end of a shot. Once a player wants to replay a throw, the Referee will call on this memory to be placed upon the display.

3. 2. Referee Logic Block

This module is the arbiter of the project. It will take in the position of the ball in space, the background of the ball (the arena), plus the user inputs of Start, Reset and Line Positions in order to determine a result. This system has five states: start, idle, recording, replaying, and game over. Note that predetermined sounds will be played based on the states of the machine, as outlined in the design section. The states transition as such:

- The start state is where the system powers on and resets. The system sets all variables to their default values, plays a tune signifying the start of the game, and then transitions to the idle state.
- In the idle state, the system is waiting for any of a few user inputs. The threshold lines and the score can be adjusted using their designated button inputs in this state. The system will transition to the replay state if the replay switch is flipped and will transition the recording state if the button is pressed signifying the start of a shot.
- In the replay state, the system will loop through the recorded video clip of the previous shot. The speed of this replay can be modified with push buttons, and the system will return to the idle state by flipping the replay switch back to its original position.
- When the recording state is entered, the system will begin overwriting the memory holding the previous shot by utilizing our Memory Shot module. The system will loop through the address space of the allocated memory, overwriting each address with the current camera input until the end of the shot has been determined. The shot is over either when the ball leaves the field of view after the shot begins, or 2 seconds after the ball comes in contact with the table line. Once the shot is over, the system returns to the idle state. The memory will be large enough to hold approximately 5 seconds of video, so the replay will contain the last 5 seconds of the shot. We believe this replay window will be large enough to convey all desired information to the viewer.
- Once one side reaches 7 points (win by 2), the system will enter the game over state from the idle state. In this state a new screen will be displayed showing the final score and commemorating the winning side. Music will be played in the background to add to the

fanfare of the victory ceremony. From here, the only transition is for the user to reset for the next game.

3. 3. Ball Recognition Block

In this module, the different hue values being input into the system by the camera will be processed in order to determine where the tennis ball is located. We have chosen to utilize this object due to its bright color and modest size (allowing for consistent motion-tracking). Once the system has located the ball, this module will keep track of it by storing its position as an x-y value from the camera input, continuously updating it based on where it sees the bright color of the ball again. This information will be placed on the display.

3. 4. Background Recognition Block

This module keeps track of the spatial position of the threshold lines in order to give the necessary data to our FSM so that a decision about the validity of a shot can be made. Hence, this module takes in user inputs so that the lines can be moved. The threshold lines are as follows:

- Low Line: This is the line the thrown ball must cross in order to be a valid throw
- Table Top: This line signifies where the top of the table is in space. This will be utilized to determine when recording of a throw will end.
- Table Lines: These lines will be set to the edges of the table to provide the FPGA with boundary information on the table. This boundary information will be used to determine where a throw left the table, for our possible stretch goals.

3. 5. Scoreboard Block

This module keeps track of points being input by the players. As explained in the design section, we chose to have the Snappa referee only as an unbiased guide, but the ultimate decision lies on the players themselves. This module gives information that will be displayed, plus is utilized to tell our Referee Logic when to transition into the game over state, once a team wins.

3. TIMELINE AND TESTING

Our basic plan of design can be summarized by Figure 3. In order to allow for division of labor in the replay, scoreboard, and referee logic modules, it is vital to get the video input and tracking modules working as soon as possible.

Week	Implementation	Testing	Writing
11/1 - 11/7	Ball Tracking and Background Tracking	Debugging of video tracking modules (edge cases, different colors)	Project Proposal Draft Revision, Prepare for Project Presentation, Block Diagram Meeting
11/8 - 11/14	Referee Logic Module	None	Project Proposal Due, Project Presentations, Revise Block Diagram
11/15 - 11/21	Referee Logic Module, Scoreboard Block	Debugging of Referee Logic (state switching)	Project Checkoff Checklist Meeting
11/22 - 11/28	Memory Replay, Basic Sound Effects	Memory Replay test benches	None
11/29 - 12/05	Stretch Goals (motion sensors)	Debugging	Project Status Update with Mentor
11/6 - 11/11	None	Debugging for Check-off	Check-off due, Project Report due, Video of Projects

Figure 3: Outline of Module creation, debugging, and milestone due on a weekly basis

The bulk of our work will be spent with the Referee module, and interfacing video input with logic in order to obtain conclusions. Memory playback is done once we can effectively get the referee to come to correct conclusions based on the environment, with stretch goals being fitted on the last week as time allows.

Testing of modules will be done mostly through physical tests in order to understand what video input is given to the system, and how the tracking modules can understand this information in order to keep track of the ball and threshold lines. Testing for the referee module is much more multifaceted, as it will have a good bit of benchmark testing to see that state switching occurs properly, plus actual real-live testing to make sure interfacing is done correctly. Testing for the Scoreboard block would be essentially all through test benches, as the scoreboard does not rely on visual input, just user inputs.

4. RESOURCES

In order to successfully implement our idea of an ideal Snappa Referee, we would need a video camera for our visual inputs in the tracking modules, a tennis ball of a bright color (such as orange or even lime-green), and possibly a white tarp for the background to facilitate object tracking. These all seem to be provided by the 6.111 lab mentors.

For the stretch goal of better detection of the ball going off a side edge versus a back edge of the table, we will need to build corner mounted proximity sensors. These sensors will consist of IR emitters paired with IR receiver proximity sensors, which could be ordered by Professor Gim using the small 100 dollar budget given to us. When a ball passes over one of these sensors, a signal will be asserted to inform the FPGA of where the ball left the table.

5. CONCLUSION

The Snappa Referee is an exciting pursuit, as it truly takes a very interactive and fun game, and aims to patch up the zones where contentious decision-making takes place. With the option of replays, with different speeds, plus user-input threshold lines, an actual referee, and even gaming soundtracks, the Snappa Referee brings about a gaming environment. Perhaps most important is that while it does provide useful decision-making to the players, it is only an advisor in this sense. In keeping with the spirit of Snappa, the teams can always make the final judgement about a shot's validity, and then proceed to input the score themselves.