21st Century *Pong*

Greg Byrd, North Carolina State University

Cornell University students built a system that tracks a Ping-Pong ball in real time and keeps score.

In the 1970s, *Pong* premiered as one of the first home video games. It was a 2D version of table tennis, or Ping-Pong, in which an electronic ball bounced between two electronic paddles that players controlled by turning knobs.

Today's video games are more sophisticated, making the bloops and bleeps of simulated Ping-Pong less exciting. So, a Cornell University student design team developed the Table Tennis Tracker system, which applies computing to a live game of Ping-Pong: while humans have fun knocking the ball back and forth, the computer handles the mundane task of keeping score.

Figure 1 shows a high-level view of the system. A video camera is set up to the side of the table, level with the table surface and with the net in the center of the frame. (During initialization, the monitor displays bright purple lines to help the players properly align the camera.) During play, the computing hardware—implemented using an Altera DE2-115 field-programmable gate array (FPGA) board—tracks the ball's position relative to the table and net and determines when a point is scored. The score is displayed on a monitor, overlaid with the live video stream of the game being played. A pair of speakers emits sound effects when a player scores a point or when it's time to switch the serve from one player to the other.

As shown in Figure 2, the system's camera and compute engine are on a cart in the foreground and the scoring monitor is mounted on the wall to the right. A black curtain hangs behind the table to improve ball tracking, which will be discussed later. The speakers are placed under the table, one on each side. The scoring sound plays on one speaker, corresponding to the side that won the point.

See **www.computer.org/computer-multimedia** for multimedia content related to this article.

COMPUTING HARDWARE OVERVIEW

The students used Terasic's DE2-115 Development and Education board as the computing platform. The board features an Altera Cyclone IV E FPGA device, which includes

0018-9162/15/$31.00 © 2015 IEEE

EDITOR GREG BYRD
North Carolina State University; gbyrd@computer.org

114,480 programmable logic elements. On-board memory includes 2-Mbyte static RAM, 128-Mbyte synchronous dynamic RAM, and 8-Mbyte nonvolatile flash.

The board also provides many standard peripheral connections, of which the following are the most important:

- TV decoder (Analog Devices ADV7180), which converts the camera's incoming National TV Standards Committee analog video signal into a digital YCbCr encoding (to be discussed later);
- VGA encoder (Analog Devices ADV7123), which produces the monitor's analog video signal; and
- Audio CODEC (Wolfson WM8731), which produces the speakers' audio signal.

Numerous switches, buttons, and LEDs compose the system's user interface. The system doesn't use a microcontroller or other general-purpose processor. The user interface is implemented directly in FPGA logic.

The FPGA is programmed using Verilog, via the Quartus II design software provided by Altera. Altera also provided an IP block for TV-to-VGA encoding, but the student design team created all the other design components.

PROJECT DETAILS

- Project: Table Tennis Tracker
- School: Cornell University, Ithaca, New York
- Students: Pol Rosello, Taylor Pritchard, Frank Xie
- Faculty mentor: Bruce Land

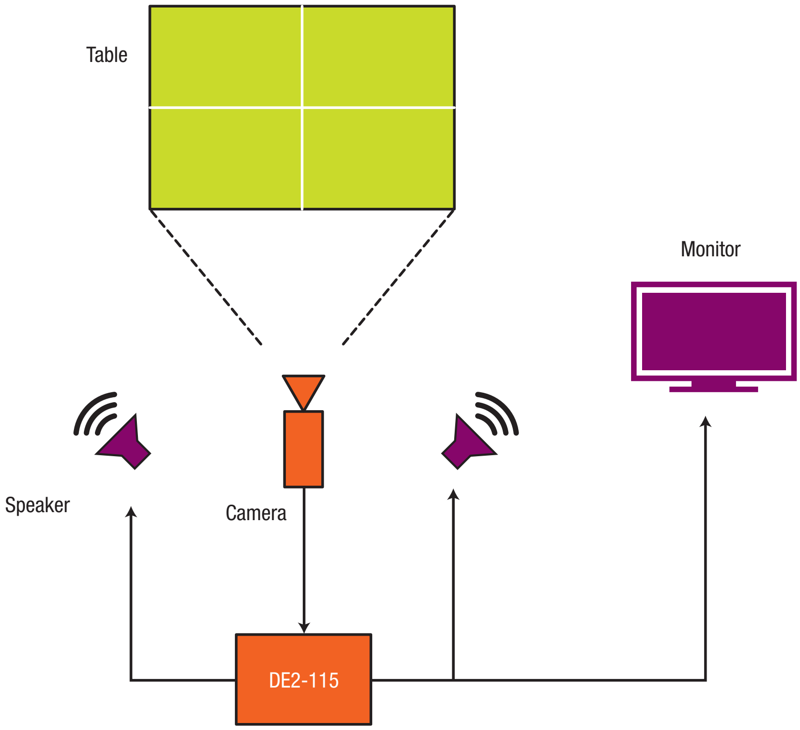

Figure 1. A block diagram of the Table Tennis Tracker system, as seen from above. The computing hardware (DE2-115) analyzes video from the camera, and the VGA monitor displays the score.

BALL TRACKING

In attempting to track the Ping-Pong ball's flight in real time, the team faced three technical challenges. First, how would the system identify the ball among the other objects in the video frame? Second, how would it determine the ball's location and track its motion over time? And finally, how would it distinguish significant events relevant to scoring, such as the ball bouncing on the table, being struck by a paddle, or traveling over the net?

Identifying the ball

To aid identification, the students painted the ball a neon green unlikely to match the color of other objects in the frame. (This is similar to the "green screen" technique used in video special effects.) The TV decoder translates the color of each pixel in the video frame into a 24-bit YCbCr value. Essentially, the 8-bit Y component is a grayscale value indicating the color's brightness, or intensity, while the 8-bit Cb (blue-difference) and Cr (red-difference) components encode

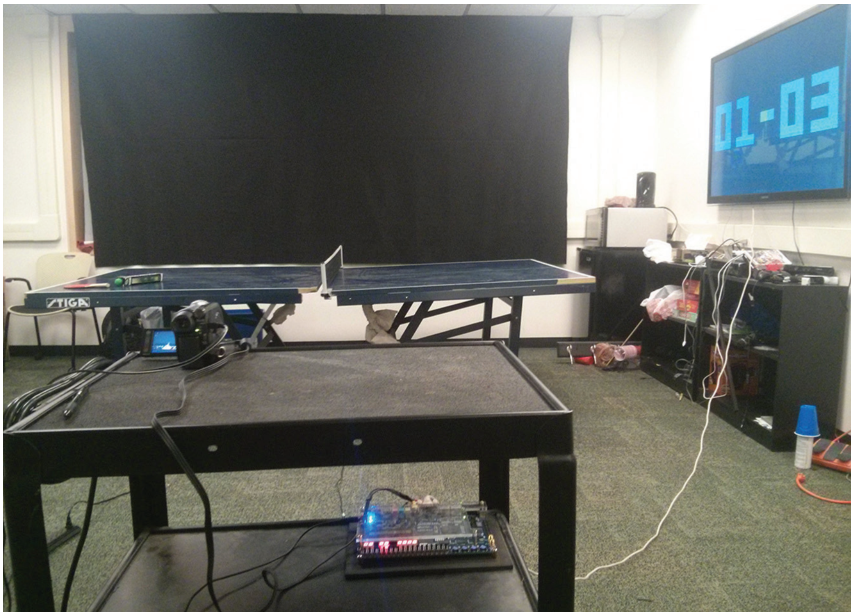

Figure 2. Table Tennis Tracker system components. The camera and compute engine are positioned on a cart, the scoring monitor is mounted on the right wall, and speakers are placed under the game table.

SUBMIT YOUR PROJECT

We want to hear about interesting student-led design projects in computer science and engineering. If you'd like to see your project featured in this column, complete the submission form at www.computer.org/student-showcase.

the color's shade—its red, yellow, and blue components. For this project, the team ignored the Y component to allow flexibility in the lighting. They experimentally determined the Cr and Cb component thresholds to match the green color of the ball.

The students discovered two problems. First, the room's fluorescent lighting had a high green component relative to other illumination forms. This added noise to the signal, because light colors and reflective surfaces could fall into the green-detection threshold. Second, the fluorescent lighting impaired the camera's color accuracy. Although the ball was clearly neon green to the naked eye, it appeared gray or white in the video feed. Adjusting the white balance helped a bit, but the students found that adding a black backdrop—the black curtain in Figure 2—most effectively improved color accuracy.

Locating the ball

The video frame is divided into blocks of 16 × 16 pixels, resulting in 30 rows of 40 blocks each. The block with the most green pixels—the "hotbox"—identifies the location of the ball. The system scans each row to determine the hottest block in that row; if the chosen block is greener than the current hotbox, it becomes the new candidate hotbox.

To improve robustness, the students applied a weighting scheme based on Manhattan distance from the previous hotbox; that is, closer blocks have a higher weight than more distant blocks. This favors the assumption that the ball travels in a smooth, continuous path from frame to frame. Testing upheld this assumption, even for the fastest shots.

Recognizing game events

A stream of ball positions identifies significant game events. A hit is detected when the ball's horizontal direction changes. Similarly, a bounce is detected when the vertical direction changes. The system maintains a running window of six positions: a sequence of three downward transitions followed by three upward transitions indicates that a bounce occurred. Crossing the net is easily detected, because the net's location corresponds to the frame's center vertical line.

SCORING STATE MACHINE

Figure 3 shows the finite state machine that keeps score. The orange arrows show the transitions that occur during scoreless play, and the purple arrows represent scoring events.

The system monitors which player is supposed to serve (put the ball in play). To start, the server must raise the ball to approximately eye level. This allows the system to distinguish the serve from other ball movements that might occur between turns. The speaker emits a confirmation sound to verify that the serve has been detected.

Once the serve is initiated, the ball is expected to bounce once, cross over the net, and then bounce on the other side. If a time-out or bounce occurs before the ball crosses the net, the serve is unsuccessful and the other player scores a point. Likewise, if the ball crosses the net and doesn't bounce—meaning a time-out or hit is detected—the serve fails and the other player scores a point. Once the second bounce occurs, the state machine expects to see a hit. If it doesn't detect a hit, then the server scores a point.

After each hit, the state machine expects to see the ball cross the net, bounce, and then get hit again. A time-out or unexpected event triggers a score for the appropriate player.

The system tracks the number of serves for each player. After five serves, it plays an audio prompt, indicating that it's the other player's turn to serve. The monitor also displays the current server.

The game continues until one player reaches the target score (11 or 21) and is at least two points ahead of the other player. When the system detects a win, it plays a special sound and the monitor displays the winning player's score in green. The system remains in this game-over state until someone pushes the reset button.

SCORING DISPLAY

The current score appears in big block numbers, superimposed over a live video feed of the table. The live feed is simply the incoming video, translated into RGB-encoded pixels and sent to the VGA encoder. Increasing the brightness (Y component) of the YCrCb signal creates the numbers, and adjusting the green or blue levels produces shaded numbers. Figure 4 shows a close-up of the display. The bar in the middle is shaded in green to indicate the server.

After completing the implementation, the student team tested the system for accuracy over five 21-point games. They found that the scoring system was more than 90 percent accurate, correctly recording 167 out of 183 points. The few errors fell into the following categories:

- Lost tracking (4 errors): the ball-tracking or green-detection system misfires.
- Out of frame (3 errors): a player hits the ball to a camera blind spot on the table.
- Serve obscured (4 errors): the system doesn't detect the first bounce of the serve.
- Incomplete logic (5 errors): a point-ending action occurs that the state machine doesn't handle, such as a served ball hitting the net but still going over.

The testers were also the creators of the system, so they were aware of the likely faults. This might have decreased the likelihood of some errors, such as tracking and out-of-frame errors.

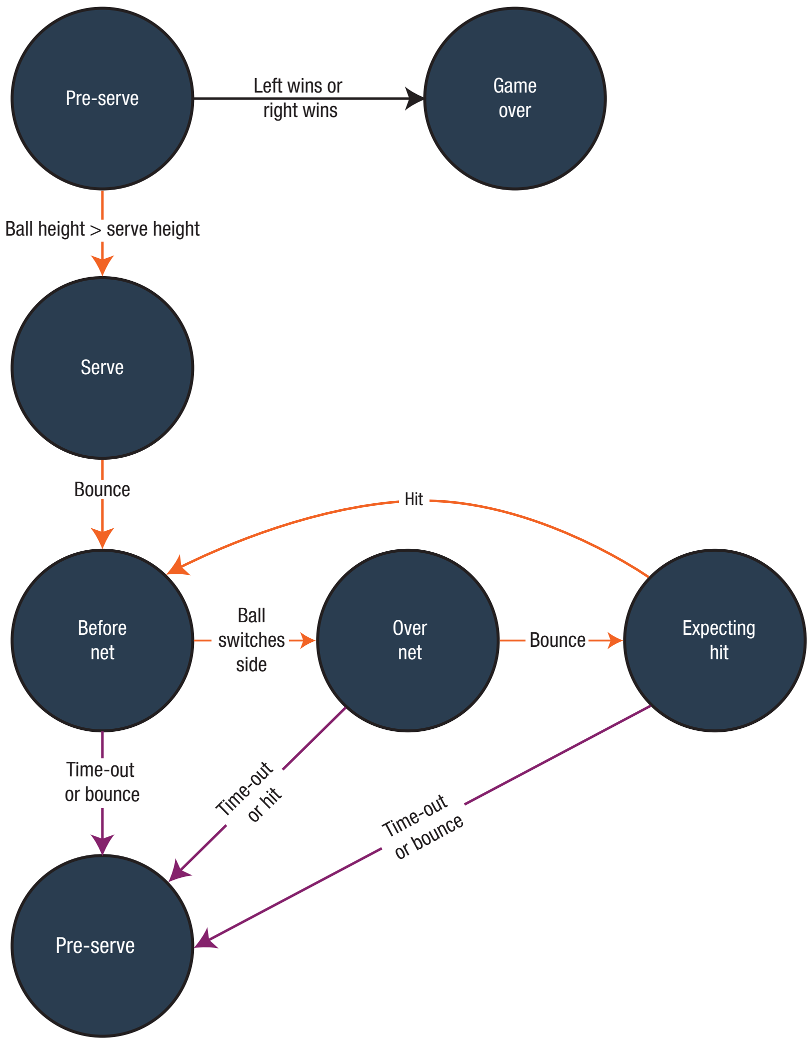

Figure 3. The game scoring state machine. The orange arrows represent transitions during scoreless play, and the purple arrows indicate scoring events.

Several changes could improve the Table Tennis Tracker's performance. A camera with a higher frame rate and better color accuracy would improve ball tracking. A wider shot would detect some out-of-bounds regions that the current setup doesn't. Improved logic could also be used, such as Kalman filtering to predict ball trajectories. This might help decode game logic that's currently difficult to detect, such as a net serve.

Overall, the student team believes that the Table Tennis Tracker system enhances game play, not only by automatically tracking scoring and

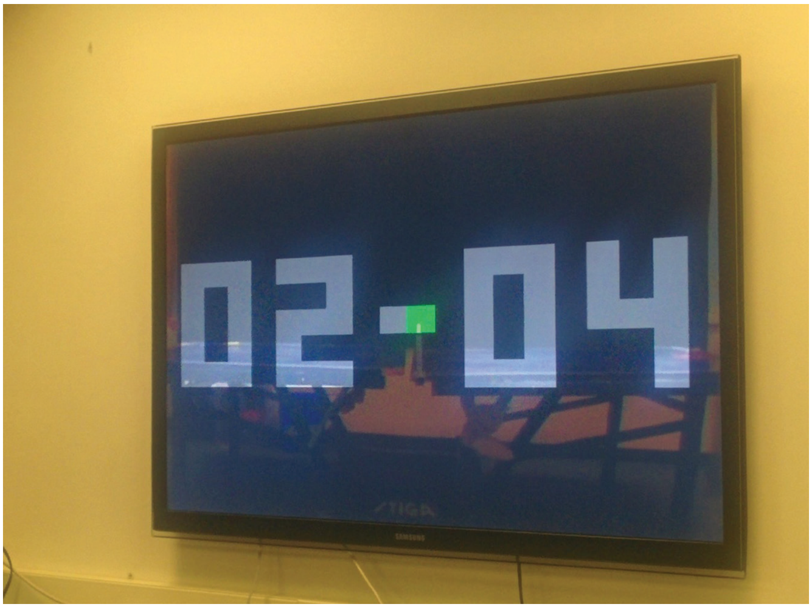

Figure 4. The scoring display. The green part of the middle bar indicates the server. When a player scores a point, their score is shaded in blue. When the game is over, the winner's score is shaded in green.

serving but also by providing audio and visual feedback. Once a game starts, players interact with the system through natural movements such as raising the ball to serve. In this way, the system seamlessly integrates into the game without disrupting it. ■

GREG BYRD is associate head of the Department of Electrical and Computer Engineering at North Carolina State University. Contact him at gbyrd@computer.org.

Selected CS articles and columns are also available for free at **http://ComputingNow.computer.org**.