Realistic 3D Gaming

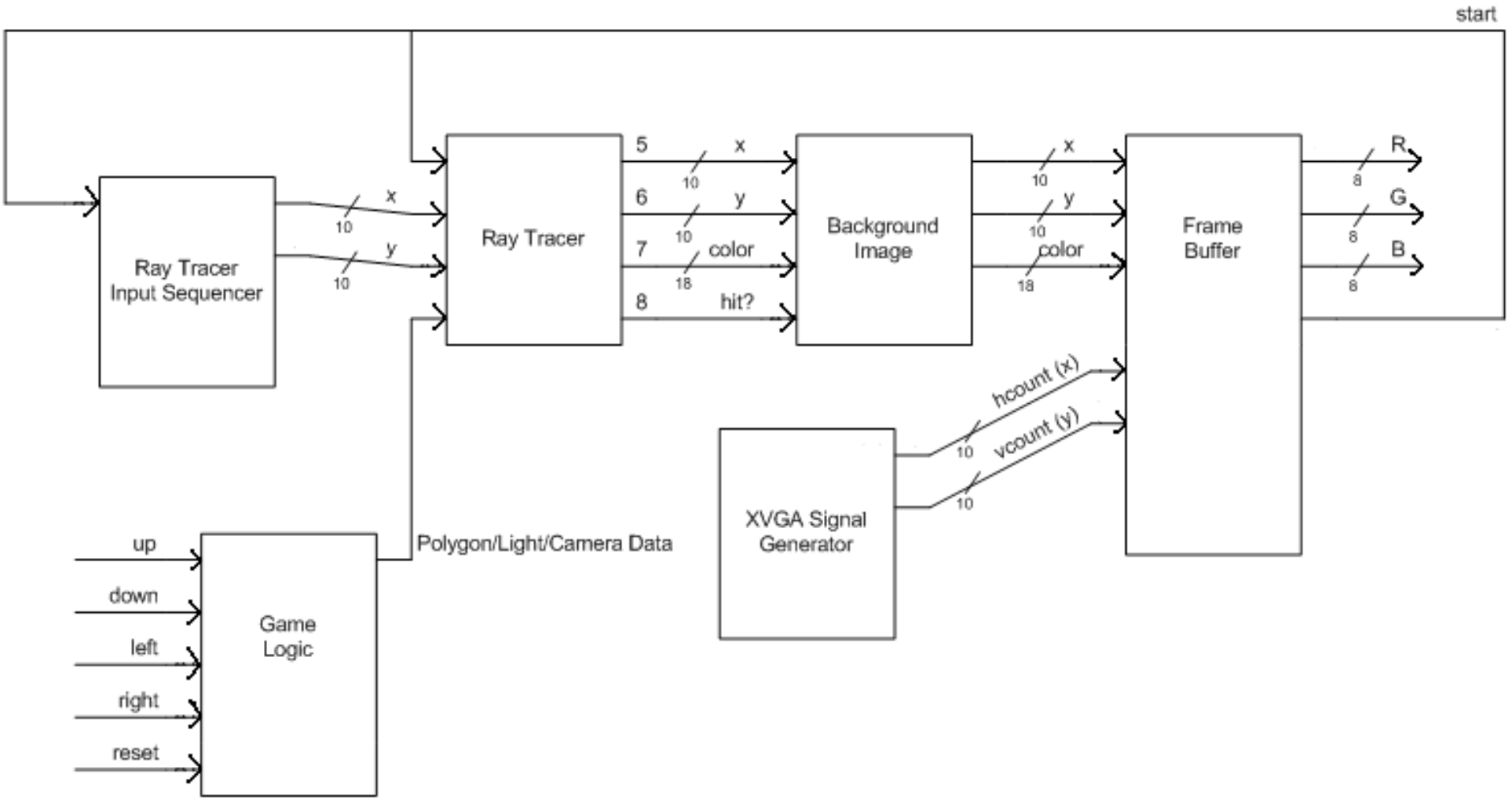
Daniel Whitlow and Ranbel Sun

Nov. 12, 2008
Mentor: Alex Valys

Overview

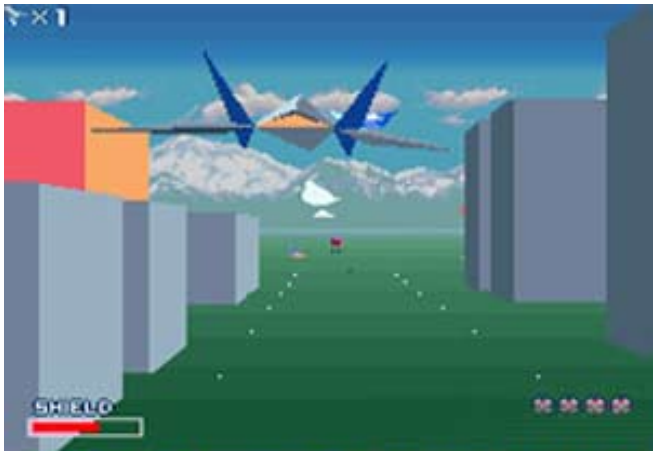
- 3D Ray Tracing renderer
  - Shadows
  - Shading
  - 2 Light sources
  - Camera
  - Primitive shapes
- Game inspired by Nintendo's *Starfox*

Block Diagram



Game Logic

- “Ship” represented by rectangular prism
- Goal: avoid obstacles in path using directional controls
- Scene advances with time along z-axis; objects that get too close vanish from scene
- Game ends when ship collides and is restarted with reset button



Screenshot from SNES *Starfox*
www.racketboy.com

Object representations

- Output from game logic
  - Objects: type (2 bits), location (27 bits), size (27 bits), color (9 bits) = 65 bits / polygon. Up to 32 objects.
  - Lights: location (27 bits), color (3 bits) = 30 bits / light. Up to 2 lights.
  - Camera: location (27 bits), angle (18 bits) = 45 bits.
- Stored in BRAM
  - Will be piped to Ray Tracer serially, or maintained in a buffer similar to the frame buffer
  - ~ 3kbits per buffer

Ray Tracing Algorithm

1. Accept display pixel (P_x, P_y) to be calculated
2. Calculate angle of ray (theta, phi) from (P_x, P_y)
3. Calculate r(t) = (x, y, z) + (cos(theta), sin(theta), sin(phi)/sqrt(2)) [normalized]
4. Perform ray-object intersection tests (runtime increases linearly with number of polygons). 3-4 types of objects: plane, sphere, axis-aligned box, possibly polygon
5. Shadow/reflectivity ray traces (runtime increases linearly with number of lights and levels of reflectivity)

Pipelined Ray Tracing

- Attempt to obtain throughput of 1 pixel/cycle with 65 Mhz clock
- 1,024x768 pixels = 786,432 clock cycles/frame
- 786,432 clock cycles / 65Mhz = 0.012 seconds per frame (ignoring latency)
- Assuming 18x18 multiplier completes in 1 clock cycle, divider in 20, trig LUTs in 5, latency will be approximately 900 clock cycles w/ 32 polygons on screen (1800 extra to do lighting)

Ray Tracing Progress

- Finished:
  - Ray calculation equations
  - Ray-object intersection equations
- To do:
  - Add shading (and possibly reflectivity) into pipeline
  - Determine exact parameters and interfacing for all CoreGen modules used

Background Image

- Stored in Flash ROM
- (1024 x 768) pixels * 18-bit color
  - → Using Byte-wide words, image will use approx. 2500k x 8 bits of memory
- Displayed when no intersection detected by Ray Tracer Unit
- 25 ns (2-cycle) read time

Frame buffer

- 2 Frames buffered in ZBT SRAMs
  - 1 write frame, 1 read frame
  - Each frame uses approx. 400k x 36 of memory
  - Starts Raytracer Unit and Input Sequencer when switching frames
- RGB pixel data output to D/A converter
- 1024 x 768 VGA output

Projected Timeline

- 11/17 – Input sequencer
- 11/24 – Frame buffer, Background image
- 12/1 – Ray Tracing Unit, Game logic