Vertex
(the game, not to be confused with Virtex)

Don Goldin
Mark Sullivan
Overview

- A game where you are in a ship and must survive as long as you can
- Use 2D wireframe graphics

Screenshot from Geometry Wars: Retro Evolved (PC)
High Level Modules

- Three top level modules
  - Input Module
  - Game Module
  - Graphics Module
Input Module

- First iteration will simply be labkit buttons
- Once other modules are complete, interface with two joysticks
- Joysticks use SPI

http://www.digilentinc.com/
Input Module

**SPI Communication Module**
Interfaces with Joystick and passes on relevant information

**CORDIC Arctan Module**
Returns quadrant-correct angle

Signals:
- 1/MISO
- 1/MOSI
- 2/!SS
- 1/CLK
- 10/x1
- 10/y1
- 1/start
- 10/x2
- 10/y2
- 8/angle
Game Module

- Entities

0001_0000000001_0000000001_00000001


- “Public” and “Private” information
- Information stored in two 256x32 BRAMs
- Public Table, Private Table
- FSM: Spawning, Moving, Colliding, Done
Game Module

**Entity Movement Module**
- When ready signal is asserted, processes input
- Decides based on ID what behavior should be
- Outputs new coordinates and orientation, and asserts movement_done signal

**Random Number Generator**
- Uses linear congruential generator
- Produces 32 bit signal every clock cycle

**Collision Module**
- When ready signal is asserted, processes input
- Checks the ship and bullets against each enemy with circular collision detection
- Outputs colliding enemy index or zero, and asserts collision_done signal

**Game Module**
- Manages unit information
- Handles unit creation and deletion
- Hub for communication of game modules
- Communicates with graphics modules

**Input**
- 10/x
- 10/y
- 8/angle
- 1/start

**Output**
- 32/random
- 32/public_entry
- 32/private_entry
- 1/movement_done
- 1/movement_ready
- 32/public_entry
- 32/private_entry
- 32/player_data

**To Graphics**
- 8/index
- 32/entry

**From Graphics**
- 32/entry
- 8/index
- 8/angle
- 10/x
- 10/y
- 1/start
Graphics Module

- Double buffers using ZBT memories
- Lookup tables for entity shapes: 16:16:32
- Translates and rotates line segments
- Bresenham's line drawing algorithm
- Pipelines line segments
Graphics Module

Shape Module
- Every other vsync pulse, looks up the line segments of each entity and applies translation and rotation. Sends endpoints to the Bresenham module along with the color.

Bresenham Module
- When the shape and buffer modules are ready, draws the current line segment to the offscreen buffer using Bresenham’s line algorithm.

Buffers
- Each 1/60 second, draws screen buffer to svga out.
- Every other 1/60 second, swaps buffers, then erases and takes writes to offscreen.

Sin Module
- 8/angle
- 8/sin

Cos Module
- 8/angle
- 8/cos

svga_video
- 1/vsync (game_clock)
## Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Mark</th>
<th>Don</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/14</td>
<td>Movement</td>
<td>Double Buffers</td>
</tr>
<tr>
<td>11/21</td>
<td>Spawning</td>
<td>Bresenham</td>
</tr>
<tr>
<td>11/28</td>
<td>Collisions</td>
<td>Shape Module</td>
</tr>
<tr>
<td>12/5</td>
<td>Input</td>
<td>Integrate</td>
</tr>
<tr>
<td>12/10</td>
<td>Debug</td>
<td>Debug</td>
</tr>
</tbody>
</table>