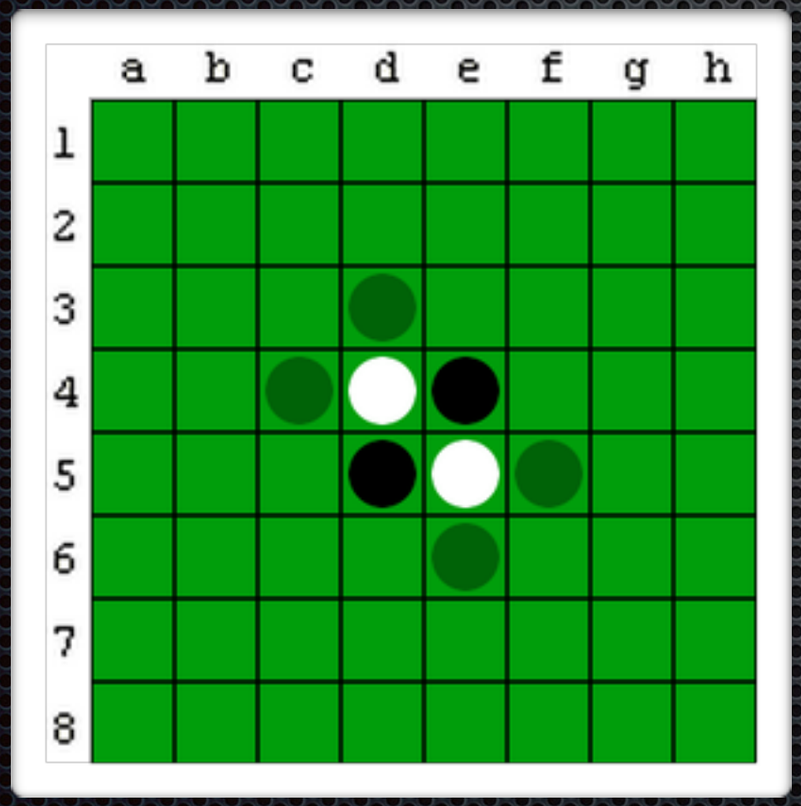
Reversi

Eric Beales & James Frank

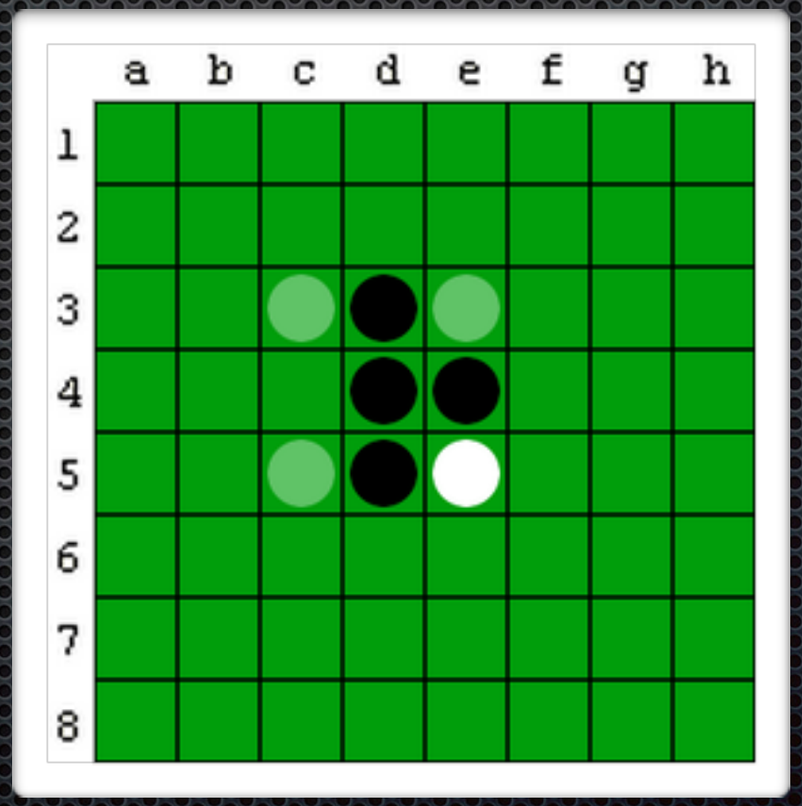
Reversi Review

- Two players
- Most pieces wins
- Each play must capture pieces
- Game over when nobody can play

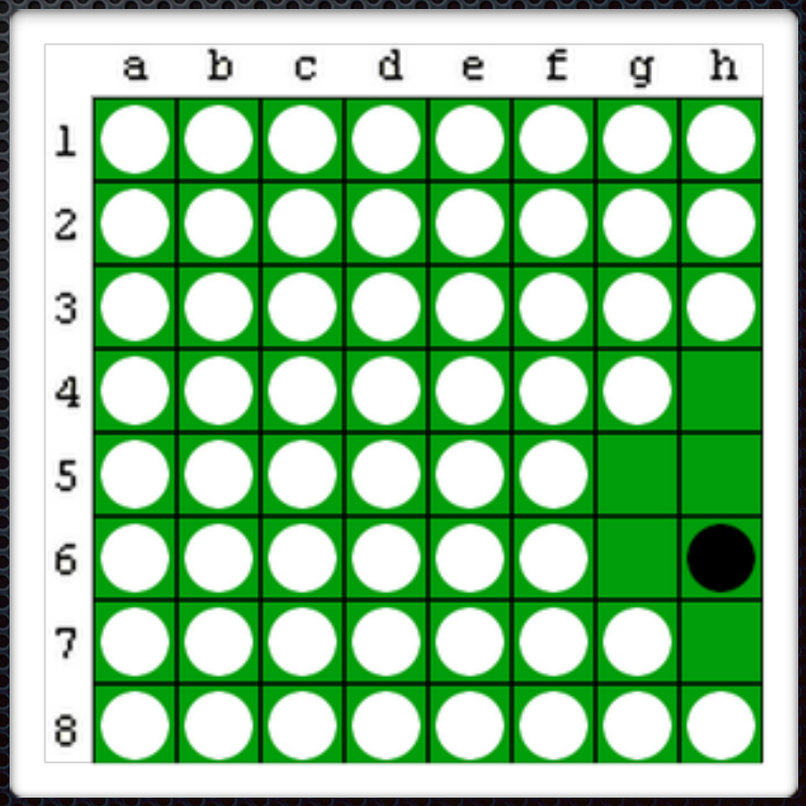
Starting Position



Next Play



Game Over

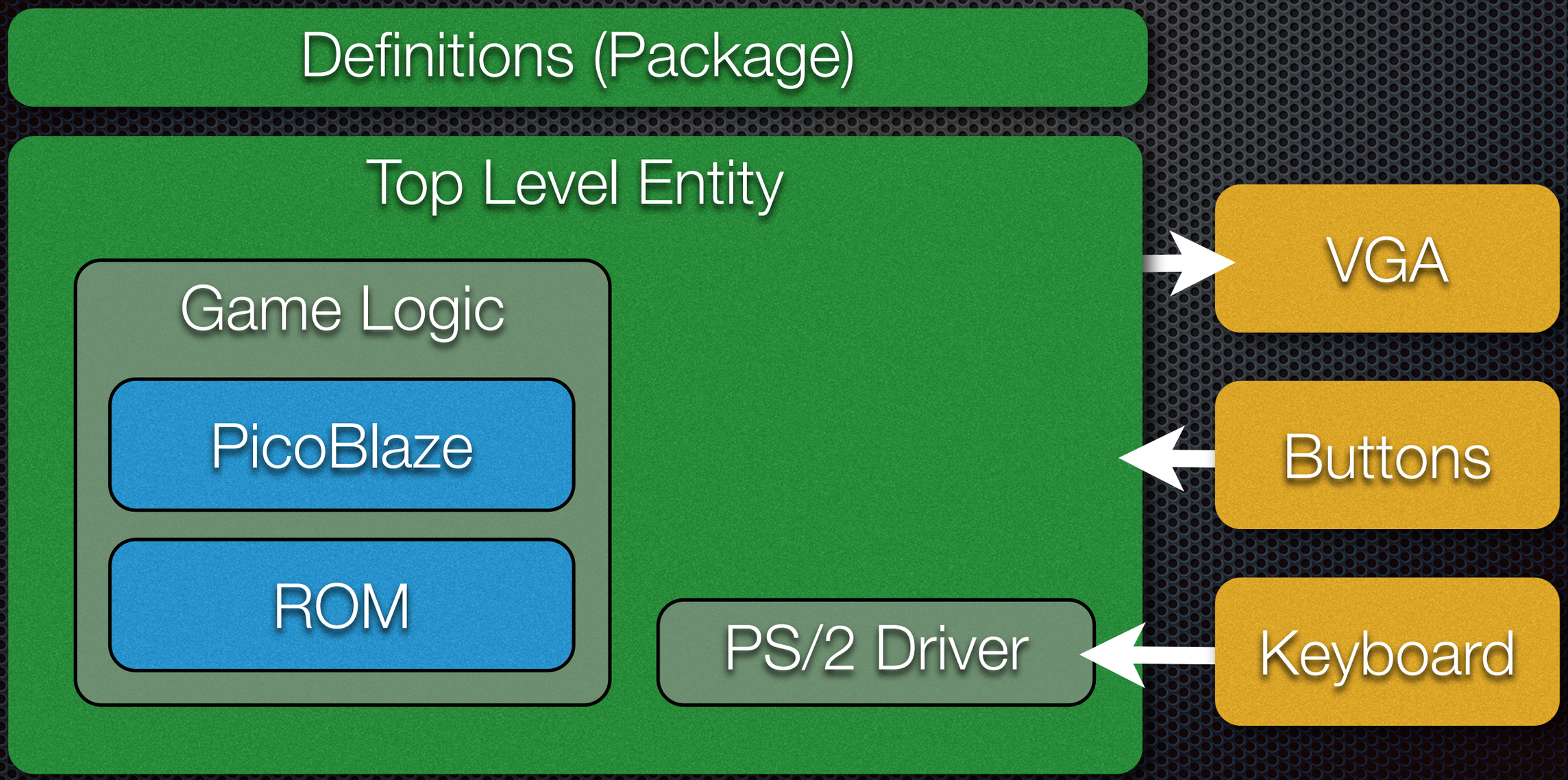


Note: Images blatantly stolen from Wikipedia

Project Goals

- Output to VGA display
- External input device (mouse or keyboard)
- Illegal move detection
- Piece capture detection
- Win condition detection

Architecture



Game Logic

- All game logic done in PicoBlaze Assembly
- Accesses game board via 'Input' and 'Output'
- Initializes board then waits until 'Play' interrupts
- Interrupts will play a piece and flip anything necessary, then re-creates the entire board's next-state array.
- Originally debugged in C then translated to assembly

Game Board

- Array of 64 bytes
- Each spot on the board gets one byte
- Bottom 4 bits are the current state
- Top 4 bits are the state if selected by the current position
- Each 4 bit state represents a color or picture
- Current position represented by 6-bit number

Board Indexing:

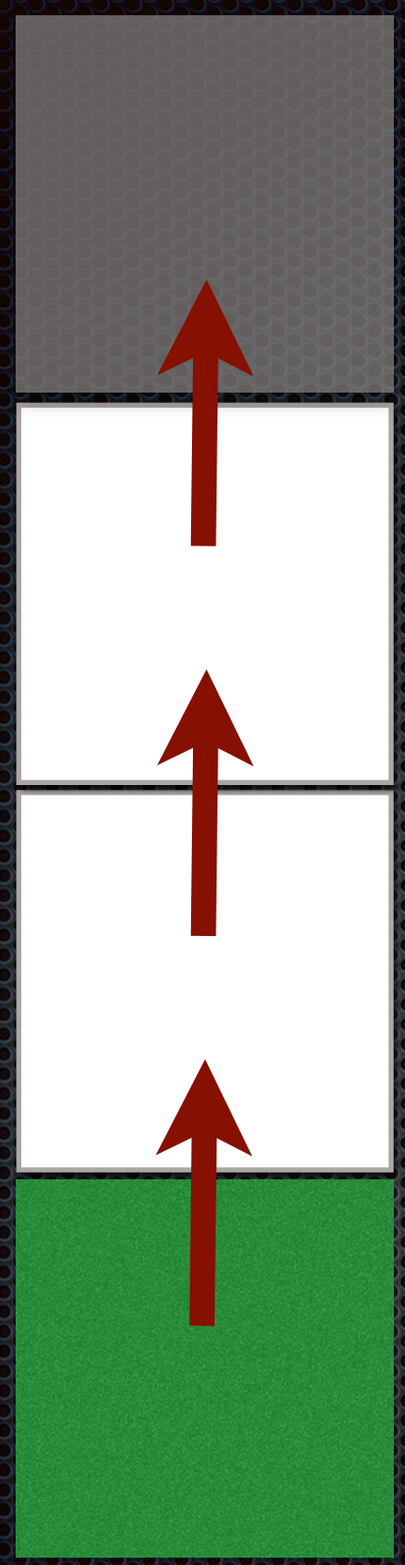
0	1	..	7
8	..		
..			55
56	..	62	63

Spot Representation:

0	0	0	0	0	0	1	1

Next State 0 (Red / Unplayable)

Current State 3 (Black Piece)

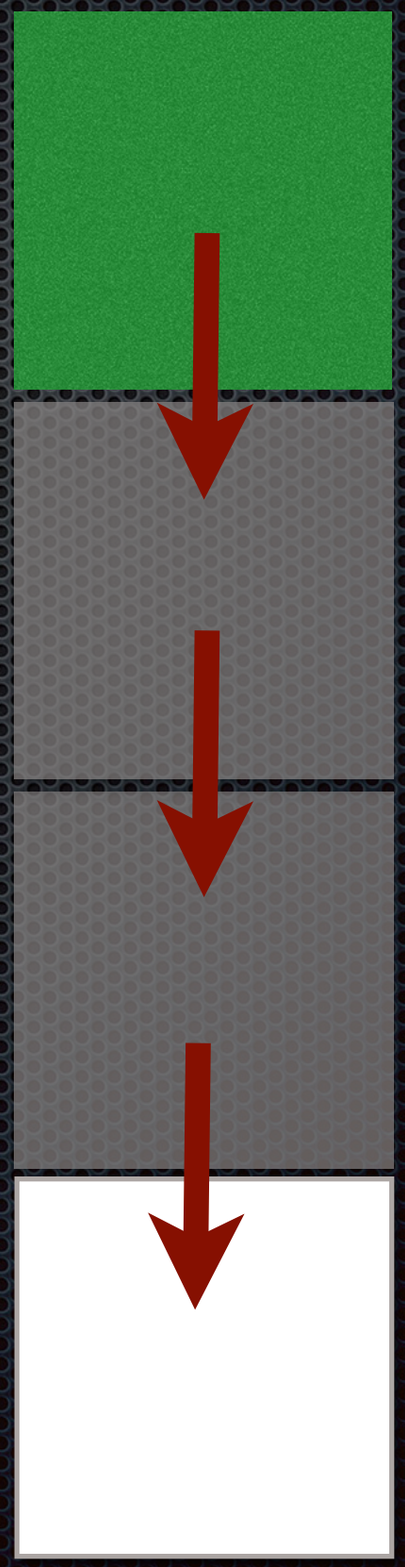


isPlayable Logic

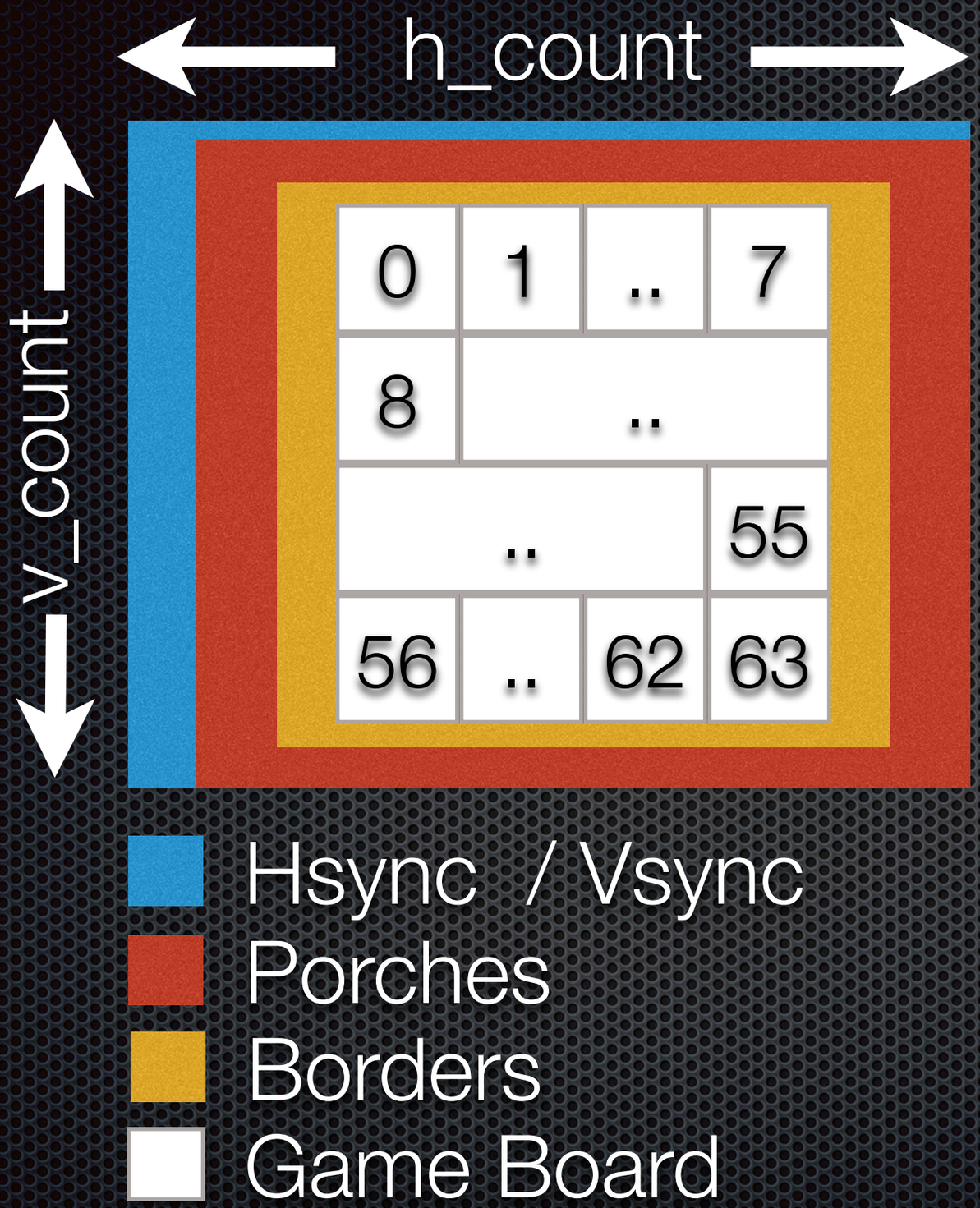
- Must start at a 'board' space.
- Walk each direction
  - Clear a flag before walking
  - If we see the opposite color, set the flag and keep walking
  - If we see the current color, return the flag
  - If anything else is seen or we fall off the edge of the board, return 0

Playing Logic

- If the spot is not playable just return
- Set the spot to be the current color
- Walk each direction
  - Check isPlayable on that direction
  - Set the spot to the current color
  - If the spot equals the current color, return



VGA



- 640x480 @ 60 Hz
- Changes about every 25 MHz (using every other clock cycle)
- Sets everything to black when not in the game board
- Sets Hsync and Vsync when below their 96 & 2 respectively
- Game board:
  - Horizontal: 145 - 784
  - Vertical: 35 - 515
- Total
  - Horizontal: 0 - 799
  - Vertical: 0 - 524

Keyboard

- Standard PS/2 packets
- Keyboard encoder handles debouncing
- Driver receives scan codes:
  - "Make" code (right arrow: 0xE0, 0x4A)
  - "Break" code (right arrow: 0xE0, 0xF0, 0x4A)

Testing and Simulation

- Initial board setup
- Piece captures
- Illegal moves
- Whole game simulation
- Win conditions

Timing and Utilization

- Move processing: approx. 600 PicoBlaze cycles (12 μs)
- Delay: 15.794ns (maximum frequency: 63.317MHz)
- Occupied slices: 819 (17%)
- Adders/subtractors/xors: 6
- Comparators: 31
- Registers: 629
- Counters: 3

Work Breakdown

- Eric:
  - VGA output
  - Game board design
  - Primary game logic
- James:
  - PS/2 Keyboard Input
  - Architecture and integration
  - Test bench
  - Current player marker

Results

- Fully playable (multiplayer only)
- Navigation using arrow keys
- Play using enter key
- Restart game using escape key

Challenges

- Assembly code hard to write and test
- Win condition logic was difficult
- PicoBlaze ISim simulation

Potential Improvements

- Bitmap graphics
- Computer opponent

Fork me on GitHub

Code and binaries available at:
https://github.com/jamesfrank/Reversi

Questions?

Demo