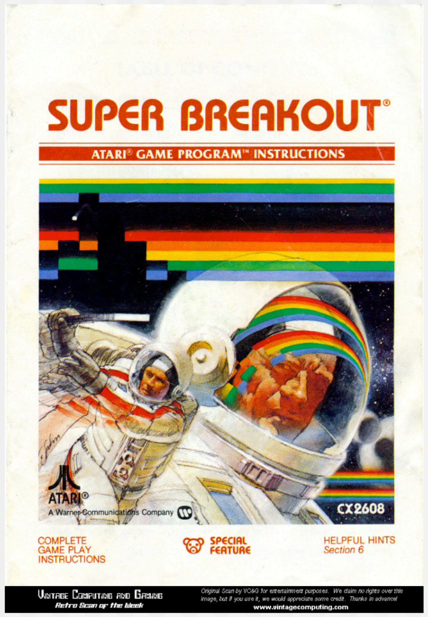
ARKAN∞ID

Breaking out of a finite space

Nels E. Beckman
CMU ISR

Breakout

- On May 13, 1976 Atari Inc. shocked the world:
  - Released the game *Breakout*
  - Features
    - Single Player Pong
    - Totally sweet
    - Destroying Bricks


SUPER BREAKOUT®
ATARI® GAME PROGRAM™ INSTRUCTIONS
ATARI®
A Warner Communications Company
CX2608
COMPLETE GAME PLAY INSTRUCTIONS
SPECIAL FEATURE
HELPFUL HINTS Section 6

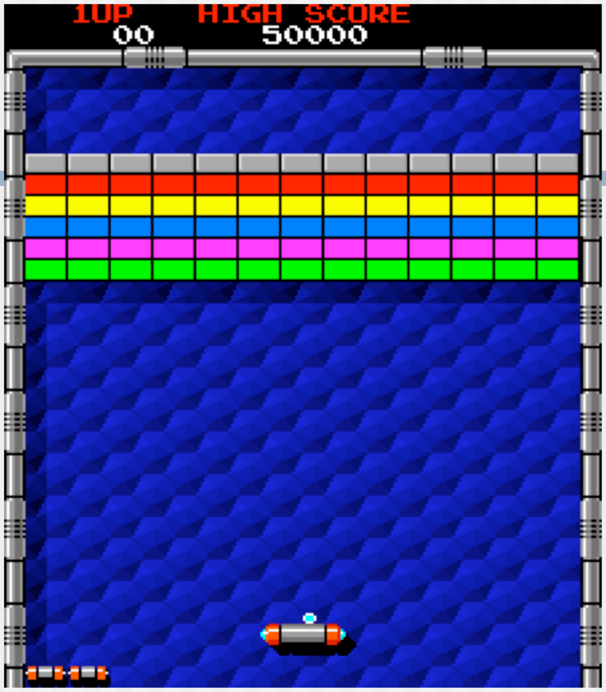
http://a8.nelsbeckman.com

http://a8.nelsbeckman.com

ARKANOID

ARKANOID
TAITO
DISCOVERY SOFTWARE INTERNATIONAL
©Copyright 1987 Discovery Software Intl., Inc.
©Copyright 1986,87 Taito America Corp.

- 1986 Taito Corp. takes it one step further
  - Releases Arkanoid. Truly an epic mega-game.
  - Features
    - Gives us the back-story
      - Our "paddle" is actually the spaceship "Vaus"
    - Power-ups



THE ERA AND TIME OF THIS STORY IS UNKNOWN. AFTER THE MOTHERSHIP "ARKANOID" WAS DESTROYED, A SPACECRAFT "VAUS" SCRAMBLED AWAY FROM IT. BUT ONLY TO BE TRAPPED IN SPACE WARPED BY SOMEONE........

What followed next…

- Never-ending stream of clones
  - Available on many platforms
- Suffered from two basic flaws:
  - Fun
  - Finite
  - Both copied from the original Arkanoid

Why finity sucks

- Physicists have shown (maybe) that space is infinite.
  - Why then are the number of bricks in space bounded?
  - Why are there “levels?”
- ARKAN∞ID
  - Corrects these longstanding issues

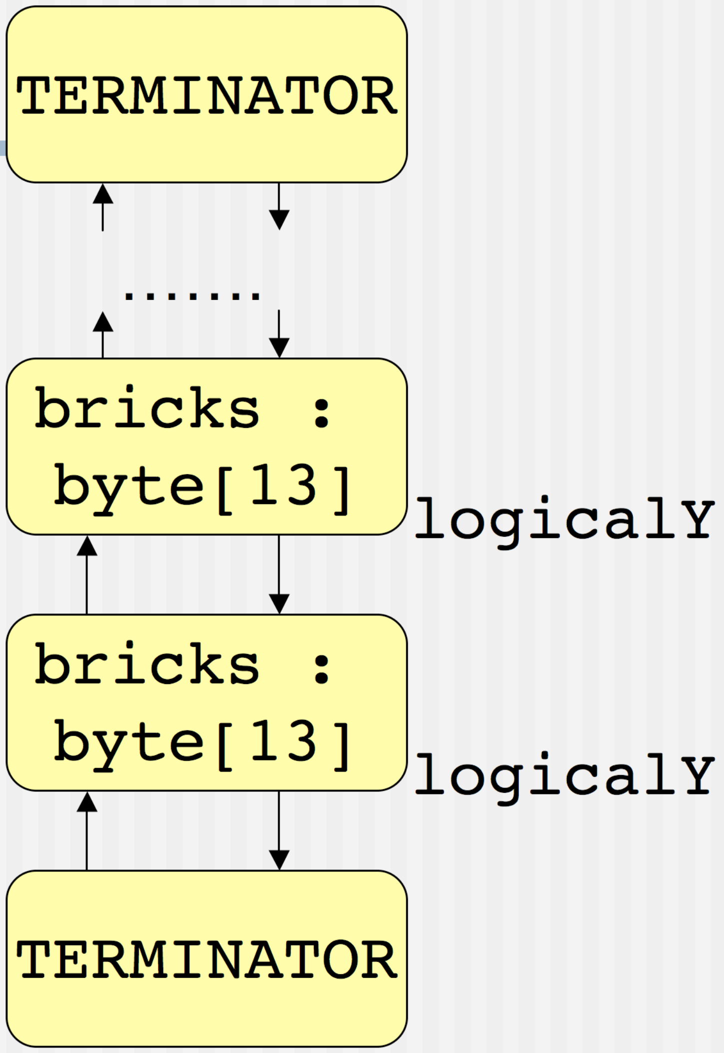
ARKAN∞ID

- The infinite brick-breaking game
  - Broken bricks always reveal more bricks
  - Blackberry application
    - Allows you to play continuously
      - E.g., At work, on subway, in space
  - CPU and Memory efficient
- Download now
  - http://a8.nelsbeckman.com

Demo

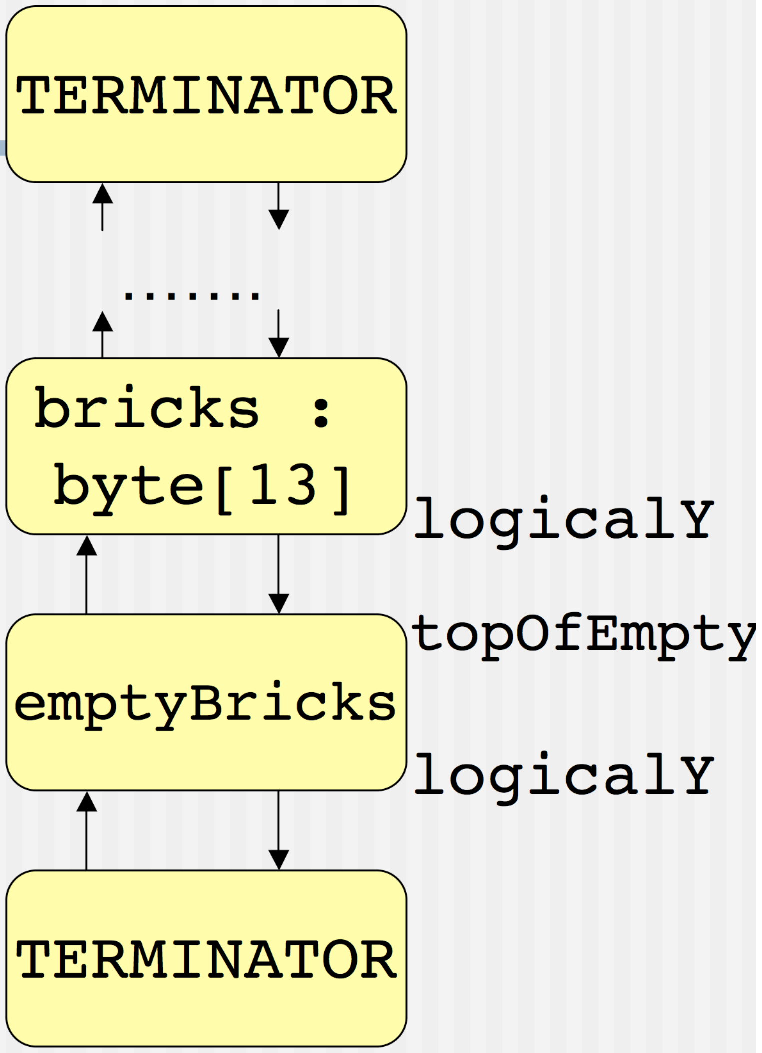
Implementation

- BrickBoard represents all bricks
  - Linked list of byte arrays
    - One node per screen of bricks
    - Generated on-demand
    - One byte per row of bricks (8 bricks each)



Implementation

- Garbage Collection
  - We don't want our heap to overflow storing empty bricks!
  - When a brick array at the bottom is empty, we collect it.
  - EmptyBrick symbolizes all empty space up to the first non-empty screen-full.
- Analysis
  - Pathologically heap usage could grow, but not in practice



http://a8.nelsbeckman.com

Analysis

- Is ARKAN∞ID really infinite?
  - Heap usage is O(n) w/ some high constant
  - Limiting factor:
    - Representation of Y axis
    - Eventually will overflow
    - When does it overflow?

Analysis

- Is ARKAN∞ID really infinite?
  - Heap usage is O(n) w/ some high constant
  - Limiting factor:
    - Representation of Y axis
    - Eventually will overflow
    - When does it overflow?

2^{32} bits ÷ (5 pixels ÷ 50ms) = 497 days

Analysis

- Is ARKAN∞ID really infinite?
  - Heap usage is O(n) w/ some high constant
  - Limiting factor:
    - Representation of Y axis
    - Eventually will overflow
    - When does it overflow?

2^{32} bits ÷ (5 pixels ÷ 50ms) = 497 days

LAME

http://a8.nelsbeckman.com

Analysis

- Is ARKAN∞ID really infinite?
  - Heap usage is O(n) w/ some high constant
  - Limiting factor:
    - Representation of Y axis
    - Eventually will overflow
    - When does it overflow?

2^{32} bits ÷ (5 pixels ÷ 50ms) = 497 days

LAME

2^{64} bits ÷ (5 pixels ÷ 50ms) = 58 billion years

Analysis

- Is ARKAN∞ID really infinite?
  - Heap usage is O(n) w/ some high constant
  - Limiting factor:
    - Representation of Y axis
    - Eventually will overflow
    - When does it overflow?

2^{32} bits ÷ (5 pixels ÷ 50ms) = 497 days

LAME 2

2^{64} bits ÷ (5 pixels ÷ 50ms) = 58 billion years

Conclusion

- ARKAN∞ID
  - The infinite brick-breaking game
- Future Work
  - Use BigInteger instead of 64 bit long
    - Y axis only limited by heap size
  - Worst-case analysis
    - Theoretical bound on heap usage?