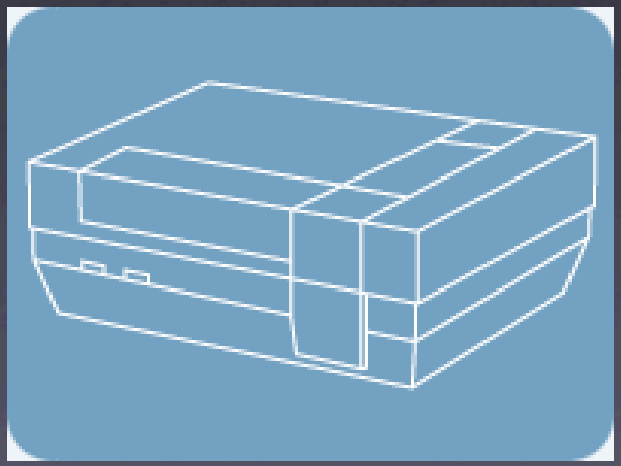
98-026
Nintendo

Bob Rost
January 28, 2004



Today's Schedule

- Postmortem: Sack of Flour, Heart of Gold
- Emulator image scaling overview
- The game loop
- Joystick input
- "A" Demo ROM

Postmortem

- Sack of Flour, Heart of Gold

Image Scaling

- NES has a fixed resolution of 256x240
- Your monitor is bigger
- Playing games in a tiny window is no fun

Image Scaling

Nearest Neighbor

Bilinear Filtering

Horizontal 50%

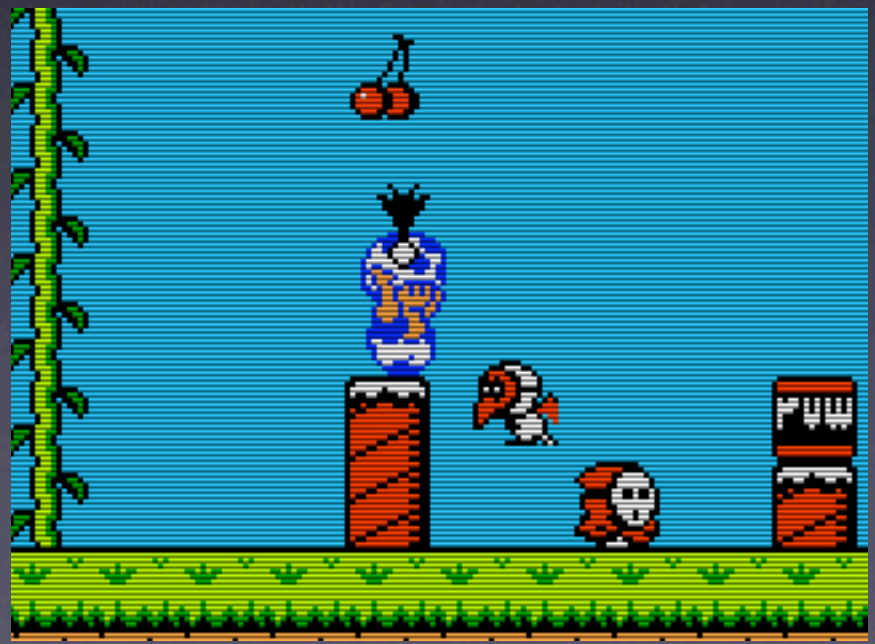


Image Scaling

Simple Smoothing

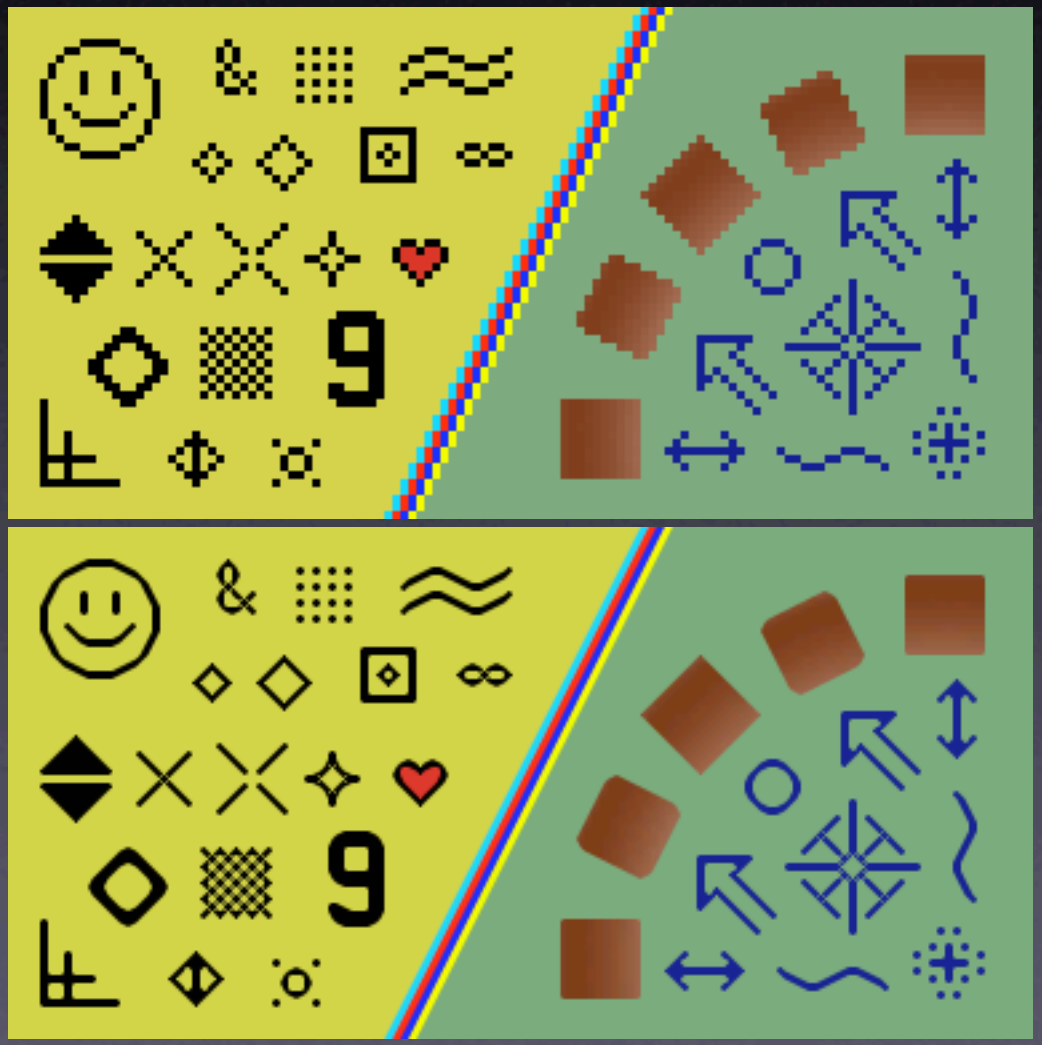
2x Sal

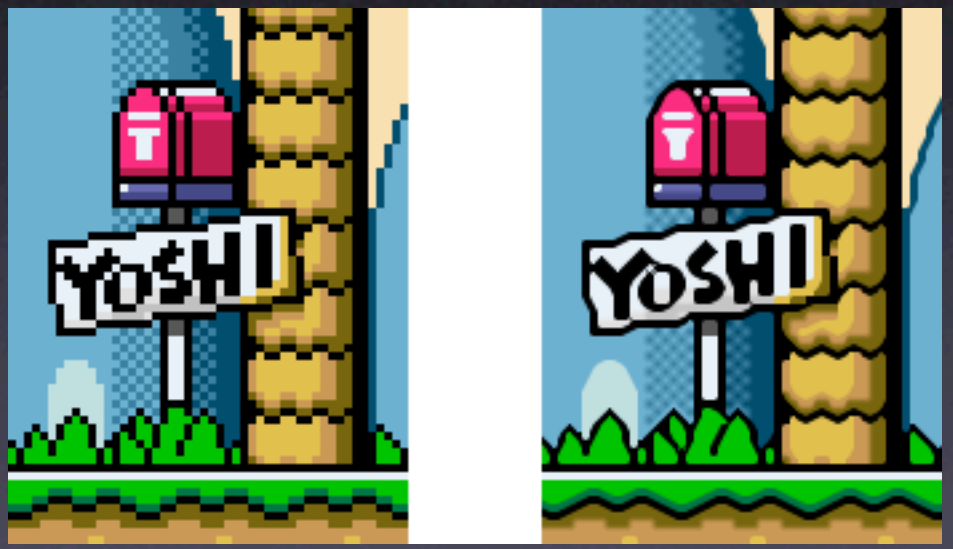
Super Eagle

Super 2x Sal

Image Scaling

hq3x





hq3x image scaling examples
(compared to nearest neighbor)

How a Game Works

- Initialization
  - Set up graphics
  - Set initial values
- The Game Loop
  - Draw the screen
  - Get player input
  - Calculate the next frame

Example nbasic Game

```
start:
     gosub init_screen
     gosub init_vars
mainloop:
     gosub vwait //wait for next frame
     gosub draw_stuff //draw the new frame
     gosub handle_joystick //get input
     gosub calculate_things //game logic
     gosub update_sound //make some noise
     goto mainloop
```

NES Registers

- Special memory addresses that allow interfacing with parts of the system

Common Registers

$2000, $2001	PPU Initialization
$2003, $2004	Sprite Control
$2005	Scrolling
$2006	PPU Memory Address
$2007	PPU Memory Access
$4014	Sprite DMA
$4000-$400F	Music Control
$4015	Sound Setup
$4016, $4017	Joysticks

Joystick Input

- Joystick 1 is register $4016
- Joystick 2 is register $4017
- Strobe by writing 1 then 0 to port
- Read one byte for each button
  (A, B, Select, Start, Up, Down, Left, Right)
- Bit 0 tells button status

Example Joystick Code

- The demo roms I create usually have a file "common.bas", including common graphics and input routines. The "joystick1" function called below is one of these routines.

```
handle_joystick:
     gosub joystick1 //strobe and read joystick 1
     if joy1a = 1 gosub jump
     if joy1b = 1 gosub shoot
     if joy1right = 1 gosub move_right
     if joy1left = 1 gosub move_left
     if joy1start = 1 gosub pause
     return
```

“A” Demo Rom

- Drawing a sprite
- Handling user input

Accessing PPU Memory

- PPU has its own memory space, which you can access through registers
- Write high and low byte of target PPU address to $2006
- Read and write $2007. Internal pointer auto-increments

Drawing Sprites

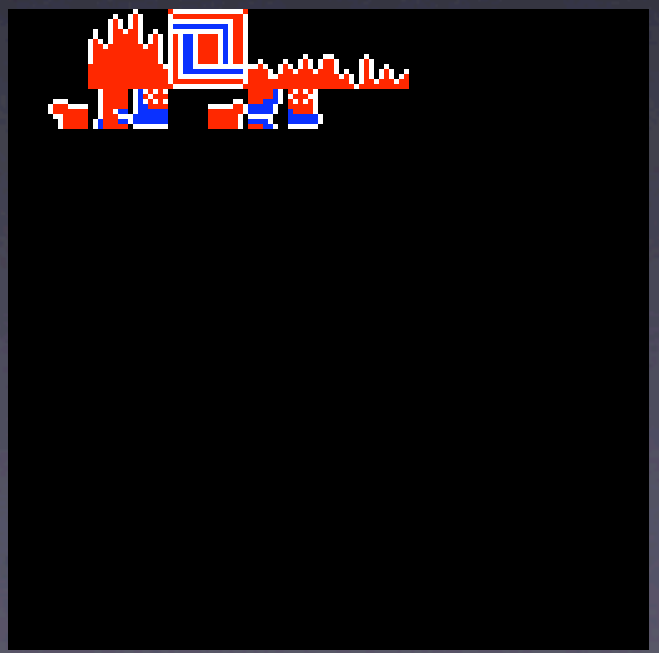
- Each sprite has 4 attribute bytes
  - Y coordinate
  - Tile number
  - Attribute byte
  - X coordinate
- Set the sprite memory, through either PPU memory access or DMA

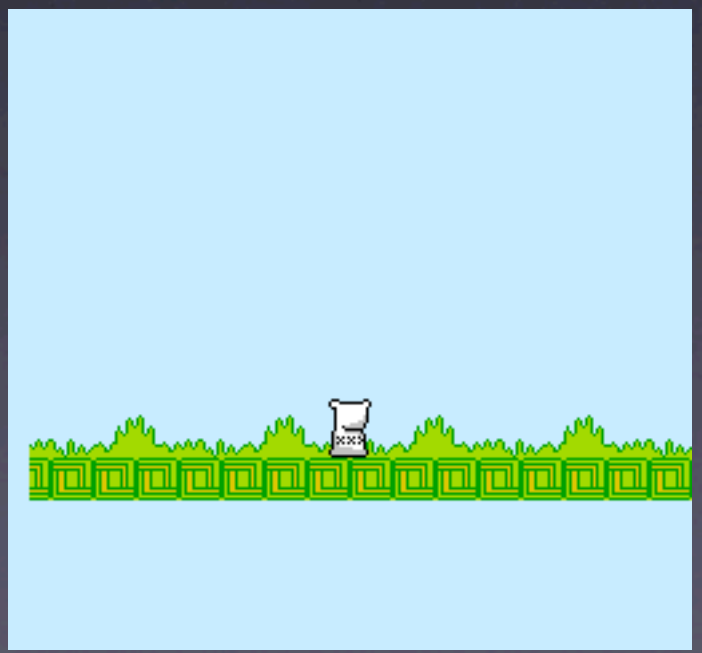
Sprite Attribute Byte

bits 0-1	Sub palette
bits 2-4	*unused*
bit 5	behind background
bit 6	flip horizontal
bit 7	flip vertical

Background Sprite Demo Rom

- Drawing a multi-sprite character
- Going behind the background
- Simple gravity physics





Drawing Sprites

Standard (Slow) Method

- Works for small numbers of sprites
- Write the starting sprite memory byte to $2003 (Sprite Number * 4)
- Start writing sprite memory to $2004
  The internal pointer auto-increments

Drawing Sprites

Awesome (DMA) Method

- Necessary for handling lots of sprites
- Keep your own copy of sprite memory, update during calculation part of game loop
- Write (address / $100) to $4014
- Note that your memory must be aligned by $100 for this to work

Setting the Palette

- Background palette has PPU address $3F00
- Foreground palette has PPU address $3F10
- Write the global palette indices to PPU memory

```
set $2006 $3f
set $2006 $00
set $2007 [palette 0]
          ...
set $2007 [palette 31]
```

example nbasic code to set foreground and background palettes

Scrolling

- Wait for vertical retrace
- Write horizontal scroll value to $2005
- Write vertical scroll value to $2005

Deconstructulator

- Dynamic demonstration of sprite memory during game play
- http://acg.media.mit.edu/people/fry/deconstructulator/

Assignment 2

- If you're going to drop the course, do it before you screw over other people
- Groups of 1-4 people
- I encourage you to work with people you don't normally work with
- Take advantage of each other's talents
- Due around mid semester break (before? after?)
- Your project must be approved by me

Assignment 2

pick one

- Make a useful development tool
  Must be pre-approved
- Make a complete game
  Works well
  Quality artwork (in-game, box art, manual)
  Sound and Music
  Decent story if appropriate
  Might result in fame and riches
- Make a "partial" game
  Some restrictions may apply
  Not for use with some sets

Assignment 2

- Form groups now? Later?

98-026 January 28, 2004

GAME OVER