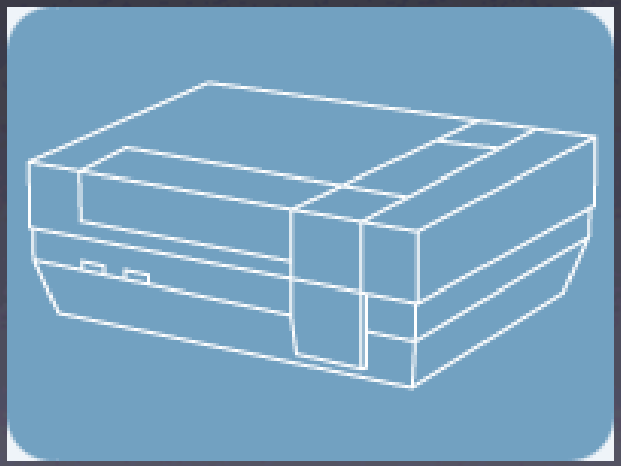
98-026
Nintendo

Bob Rost

January 14, 2004



Today

- Email from the real world. What do other people care about in NES development?
- nesmus Announcements
- Drawing tips
- Some programming magic
- Practical sprite drawing

nesmus

- I double checked the slides. The G was just a really sharp F, so the code is correct. I don't make mistakes.
- The slide shows that f2 means an F of twice the current length. That's actually not supported yet, and you have to change the current length. I made a mistake.
- It's on the Resources page, in case you haven't found it.

Email From the Outside

- December 21: "Asnake" informs me that nbasic for windows actually works if it's recompiled with cygwin
- January 7: Philipp Lenssen from Germany plays Sack of Flour on his Nokia phone. I think he wants me to buy a Nokia. He wants to buy simple game creation software, but he's not a programmer.
- January 12: Jeff Pimper requests my mailing address so he can send me a dollar for the SOF development fund through US Postal.

Email From the Outside

- January 13: Mark Farnsworth wants to use nbasic to make a new game for his Computrainer, a NES-based bicycle simulator
- January 13: "Some Dude" asks if anyone had attempted to get SOF on cartridge.

Email From the Outside

- January 25: Cory Arcangel has been working on an animation engine in nbasic. He wonders if he can play two different nesmus songs in the same program.
- January 26: Cory figured it out. And he thinks the class is "totally cool!!!!!!!"
- February 8: Hugo Desmeules likes nbasic but wants documentation on sound. I pointed him to the last lecture.

Drawing Tips

- Ask an artist to teach you to actually draw
- Picking a palette
- Drawing from reference, scaling
- Drawing from scratch

Drawing Tip

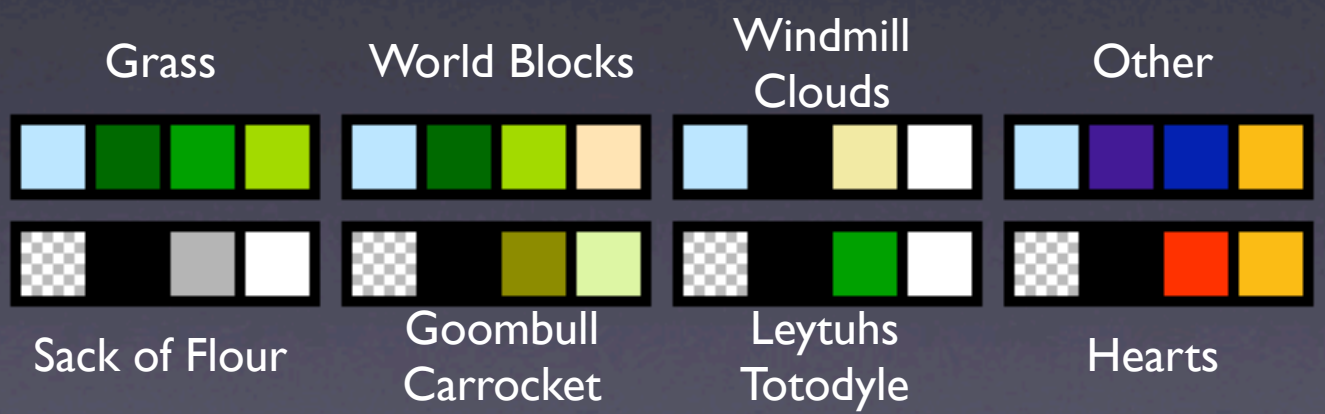
- Sprites are often 16 or 32 pixels wide.
- Your monitor is at least 1024 pixels wide.
- Zoom in.

Picking a Palette

- Choose the 3 most important colors for your character
- Often, entry 1 for outline, entries 2 and 3 for color
- Try to be consistent. Dark to light?
- Good general scheme:
  one sub palette for the hero
  two for enemies
  one for powerups and items

Example Palette

- Foreground has outline color for sprites
- Consistent gradient direction allows smooth palette changes



Drawing from Reference

- Start with a large image
- Scale it down to sprite size
- Use as a guide to draw on top
- Keep original around for reference

Original Image (640x480)

Scaled Image (32x16)

Drawn Image (32x16)

Drawing From Scratch

- It helps if you're actually an artist
- Kind of necessary for original art
- Maybe draw big and scale down?
- Pick a palette before you draw
- Iconic simplicity is key

iconic and simple

NESPaint

- By Dennis Cosgrove and Steve Audia
- Demo

Frame Timing

- vwait
- NMI
- Sprite 0 Hit

vwait

- Loop until the television begins/ends the vertical retrace (port $2002)
- Safe to write to PPU during vwait
- Set scroll values and PPU address to 0 by end of vblank
- Example code in provided demos

Non Maskable Interrupt

- Enabled by bit 7 of $2000
- Triggered at start of vblank, program automagically jumps to a specified label
- Use `resume` keyword at end of NMI

Sprite 0 Hit

- Flag cleared at end of vblank
- Flag set when sprite 0 draws a pixel over a background pixel (forground not transparent, background not primary color)
- Bit 6 of $2002
- Split screen scrolling. Wait for clear and set

Good nbasic Coding

- Where to declare arrays
- Code readability
- Goto and Gosub
- Documentation, new versions

Code examples in this section are not real. Don't fret over them.

nbasic Arrays

```
main:
    gosub init
    blah blah
    blah blah
    gosub thing2
    goto main
init:
    array absolute $8001 array1 2
    if this_code = dumb set myvar 0
    return
thing2:
    set blah + 1 blah
    array array2 1
    inc myvar
    return
```

```
array absolute $8001 array1 2
array absolute $8000 array2 1

main:
    gosub init
    blah blah
    blah blah
    gosub thing2
    goto main
init:
    if this_code = dumb set myvar 0
    return
thing2:
    set blah + 1 blah
    inc myvar
    return
```

Quick! Find the array declarations!

nbasic Arrays

```
main:
    gosub init
    blah blah
    blah blah
    gosub thing2
    goto main
init:
    array absolute $8001 array1 2
    if this_code = dumb set myvar 0
    return
thing2:
    set blah + 1 blah
    array array2 1
    inc myvar
    return
```

BAD: Hard to Find

```
array absolute $8001 array1 2
array absolute $8000 array2 1

main:
    gosub init
    blah blah
    blah blah
    gosub thing2
    goto main
init:
    if this_code = dumb set myvar 0
    return
thing2:
    set blah + 1 blah
    inc myvar
    return
```

GOOD: Easy to Find

Readable Code

Programmers who don't know this:
shame on you!

- Be smart, not "clever"
- Tabs are your friend
- Good code is easier to debug

Readable Code

```
label:
blah blah
if thing <> 0 then
blah
   blah
blah
   endif if 1 = 2
goto label
```

BAD: Hard to Find Logical Blocks

```
label:
   blah blah
   if thing <> 0 then
      blah
      blah
      blah
      endif
   if 1 = 2 goto label
```

I line my endifs up like this. Easier to follow the logic.

GOOD: Much Easier to Read

Goto and Gosub

- Avoid spaghetti code!
- Use goto for loops
- Use goto for downward jumps within a single "function"
- Use gosub/return everywhere else

nbasic Other

- New language documentation is in the works. Check the Resources page.
- New compiler version supposedly fixes a math bug. OSX and source for now.
- nbasic v3.0 getting there...

Multiple Sprites

- Facing left and right
- Being smart about sprite memory
- Clearing extraneous sprites

Facing Left and Right

- We arrange the hero in the pattern table facing one way (right) and draw the sprites in order.
- To face him left, we flip the sprites, but then he's split in two if he's made of multiple sprites.
- Need to swap the horizontal positions.

table layout

facing right, as laid out in table

individual sprites flipped

sprites flipped, positions swapped

Facing Left and Right

Assuming coordinate (x,y) is upper left of the character

```
draw normal (facing right):
   sprite (y, tile 0, normal, x)
   sprite (y+8, tile 1, normal, x)
   sprite (y, tile 2, normal, x+8)
   sprite (y+8, tile 3, normal, x+8)

draw flipped (facing left):
   sprite (y, tile 0, normal, x+8)
   sprite (y+8, tile 1, normal, x+8)
   sprite (y, tile 2, normal, x)
   sprite (y+8, tile 3, normal, x)
```

Flipping the character correctly requires adding 8 (or not) to the right sprites.

Smart Sprite Memory

- Fill sprite memory during the frame draw, DMA during the vblank.
- We should be able to support a changing number of sprites on screen
- But drawn sprites stay onscreen until you manually clear them
- Keep track of what you've drawn in the frame, then clear the rest

Main Draw Function

```
array absolute $4014 SPRITE_DMA
array absolute $200 spritemem

draw_stuff:
  set next_sprite 0
  gosub draw_hero
  gosub draw_enemies
  gosub clear_end_sprites
  gosub vwait
  set SPRITE_DMA 2
  return
```

next_sprite tells us the next unused piece of sprite memory.

clear_end_sprites will remove unused sprite garbage that may be left.

Drawing the Guys

```
draw_hero:
  if hero_facing = 0 then
    gosub draw_hero_left
    return
    endif
  gosub draw_hero_right
  return

draw_enemies:
  set which_enemy 0
  draw_enemies_1:
    gosub draw_enemy
    if which_enemy <> num_enemies
      goto draw_enemies
  return
```

It's easier to have a separate function for each facing direction.

Abstract the concept of drawing a single enemy. Make use of the which_enemy variable.

Drawing A Guy

```
draw_hero_left:
  set [spritemem next_sprite] hero_y
  inc next_sprite
  //repeat for all sprite information
  return

draw_enemy:
  set [spritemem next_sprite]
      [enemy_y which_enemy]
  inc next_sprite
  //repeat for all sprite information
  return
```

use next_sprite to control which sprite memory you fill next

Clearing Sprites

- The NES always draws all 64 sprites, no matter what.
- Once you set a sprite to draw something, it will always draw that until you tell it otherwise.
- You can't delete them, but you can make them unseen.

Clearing Sprites

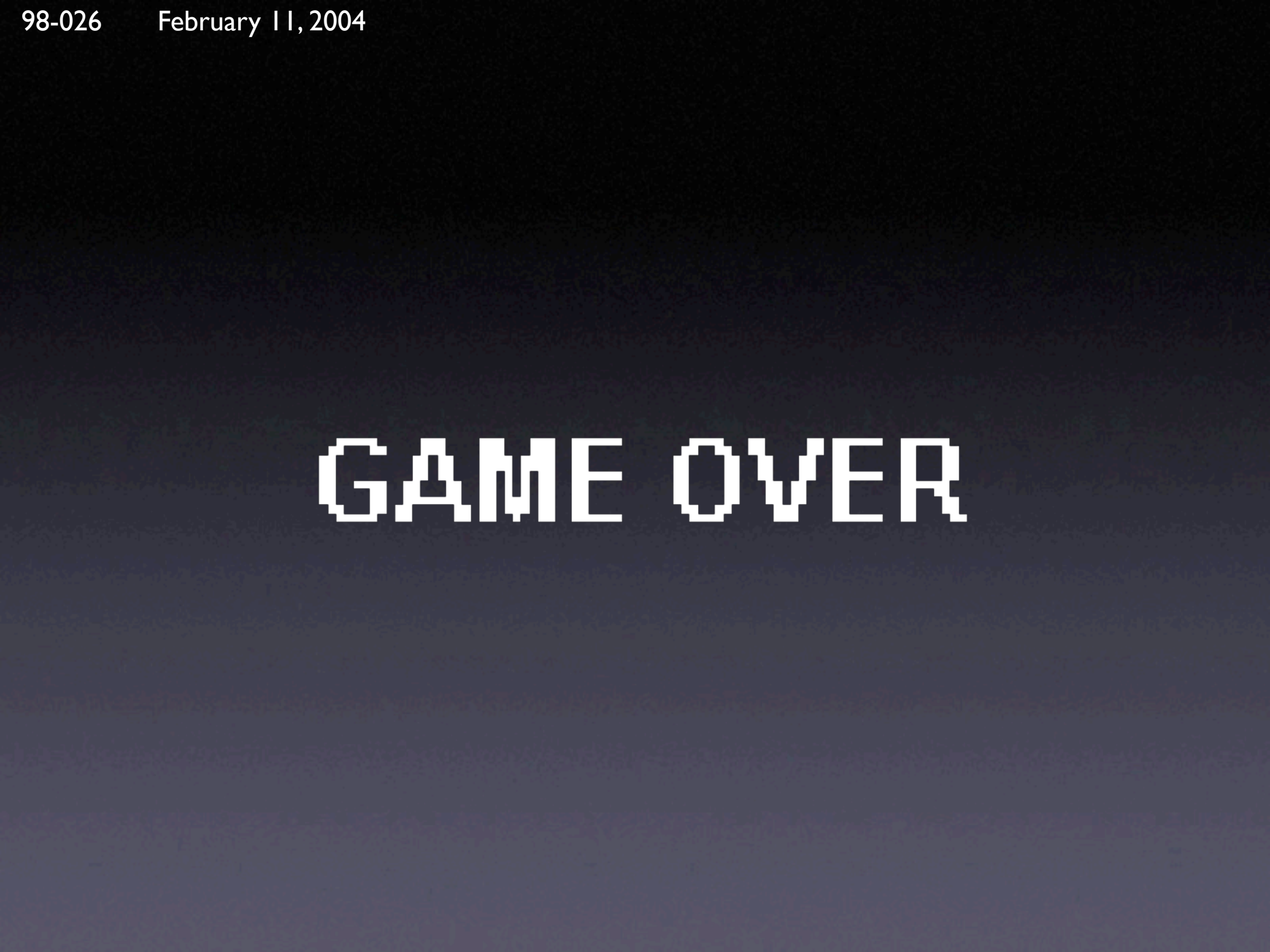
- Make sprites invisible!
- Set the tile to a blank tile, often 0 or 255
- Problem: 8-sprite scanline limit
- Problem: The tile may change with bank swapping

Clearing Sprites

- Shove them off screen!
- The NES addresses 256 vertical pixels, but only 240 are visible
- Set the y coordinate of unused sprites past the bottom of the screen

Mappers and ROMs

- how about next week?


GAME OVER