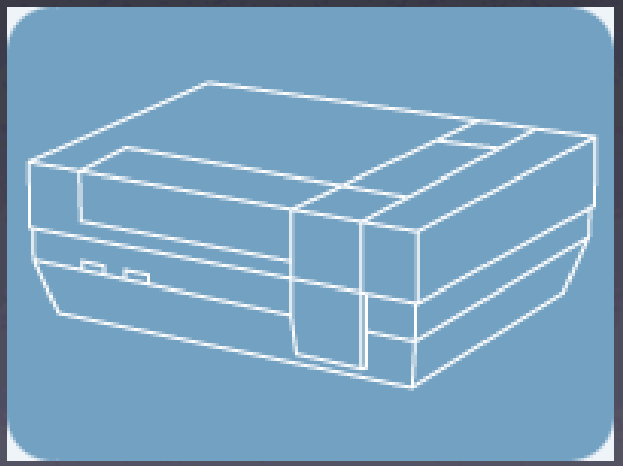
98-026
Nintendo

Bob Rost

February 25, 2004



Today

- Demos
- Making levels (broad overview)
- Level graphics compression
- SOF level data
- Some Sound (just a little)
- Palette cycling
- Sprite 0 Usage
- Types of two-player gameplay
- Something else

Demos

screenshots from Ben Rotskoff's Full Contact Bowling

INTRODUCTION
Ben Rotskoff's Full Contact Bowling is a game for the 8-bit Nintendo Entertainment System (NES.) The game will be playable using various NES emulators (Jnes, Rocknes, etc.)

Ben Rotskoff's Full Contact Bowling will include two modes: Bowling Mode and Fighting Mode.

During the course of play, the game will switch between these modes automatically based on the player's actions.

Throughout this document, the player will be referenced as a male purely for readability and consistency. The player will also be considered singular, though in two-player games both players will participate equally.

START SCREEN
When the game first begins, the player will see a Start Screen that includes:
The name of the game
A list of possible choices
Start 1-player game
Start 2-player game
Game Options
The names of the people who created the game

The player will also hear the background theme music, which should begin (and continue to loop) while the Start Screen is showing.

Once the player chooses to start a 1- or 2-player game, the player will enter his name, and then start in Bowling Mode.

GAME OPTIONS SCREEN
Currently there are no game options. This section will be revised as necessary during the development of the game. This options screen will eventually be similar to the options screen from Tetris.

NAME ENTRY SCREEN

When the player first begins the game, he will be asked to enter his name. The name will have a limit of 8 (?) digits, including letters, numbers, and spaces. On the Score Screen, the player will see his initials (the first letter, the first letter following the first space (if any), and the first letter following the second space (if any).) When the player is bowling, he will see his full name.

The interface for input will be a cursor that moves around the various letters, numbers, and spaces, as well as a "delete" key and a "done" key.

If this name entry is too elaborate for the available time, the player will be asked to enter his initials instead, up to three letters.

BOWLING MODE
In Bowling Mode, the top 85% (?) of the screen will be devoted to the Bowling Panel. The bottom 15% (?) of the screen will be devoted to the Score Panel, which player's current score.

In the Score Panel, the player's initials will be shown, along with 4 frames of the player's game. The score visuals should match a traditional bowling scorecard. Each frame is shown as a large square with a smaller square in the upper right; the first ball is in the upper left, the second ball is in the upper right, and the current total score is on the bottom.

The Score Panel also includes a vertical bar labeled "Rage" which is the character's current level of frustration. Rage starts at zero.

In a two-player game, at the beginning of each frame, there will be a separate Score Screen that shows both of the players' current scores. This screen will stay visible for 3 seconds or until either player clicks a button, at which point the first player will bowl.

The Bowling Panel will be a back view of the player in a bowling alley, similar to this picture:

First, the player will have the opportunity to move his character left and right, and closer or farther to the release line. Once the player hits a button to lock-in his character's location, there will be three interface elements that appear in order:

Making Levels

- Look for tools on nesdev, or roll your own
- Not everything on nesdev is appropriate. Many of the tools are DOS only.
- Open tUME looks promising...

Another Graphics Tool

- Hisham is extending the sprite tools from the webpage
- Provide an image from Photoshop
- The tool will automatically create the palette, pattern table, name table, and attribute table for you
- Check the webpage soon...

Drawing the Level

- You may be tempted to have code such as:

```
set $2007 1
set $2007 5
set $2007 1
...
set $2007 4
```

- This works, but is generally a bad idea

Drawing the Level

- It is more convenient, smaller and robust to have a level data format
- Use a load function to read that data

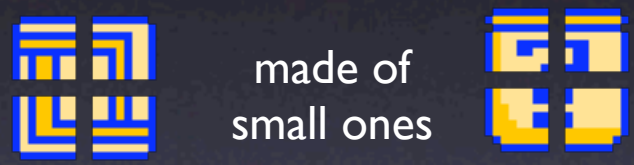
```
load_level:
        //do stuff here
        return
level_data:
        asm
        .incbin “level.dat”
        endasm
```

Graphics Compression

- A full screen (name table and attribute table) takes 1k of data. That adds up quickly, and takes away space for music or other data
- If you can take away redundant information, you'll have room for more

4-Square Compression

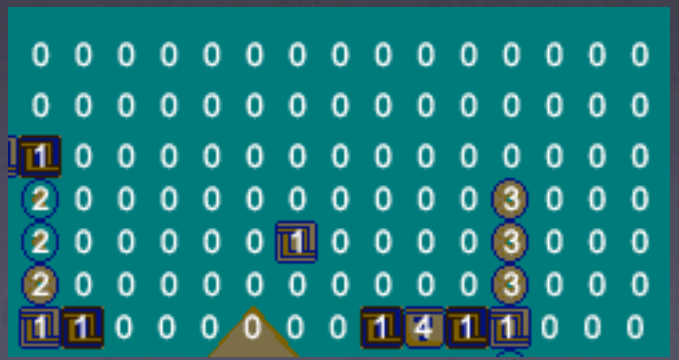
- We often draw 4 blocks together as a single large block
- It's often silly to draw them any other way
- We can use just 1/4 of the data by specifying one block on screen, and assuming the other three
- Store all such blocks in the pattern table with the same relative positions (all 4 tiles in a row, or bunched as they appear on screen)



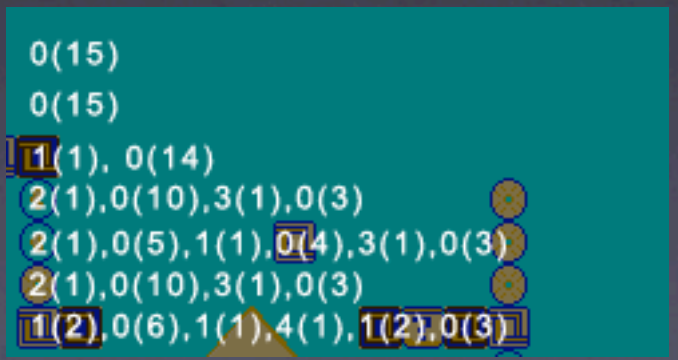
silly arrangements

RLE Compression

- Backgrounds often have the same tile many times in a row.
- Run Length Encoding stores the tile number, and how much to repeat it
- We can often save lots of data this way
- The example below is 15x7 large tiles



Raw Data, 105 bytes



RLE Data, 48 bytes

Other Compression

- ZIP, LZW, ARJ - Would work well, but they're probably too complex for practical use in a NES game
- Scene description, game understands concepts of door, plant, staircase, vine
- Anybody know of other practical methods off-hand?

SOF Level Data

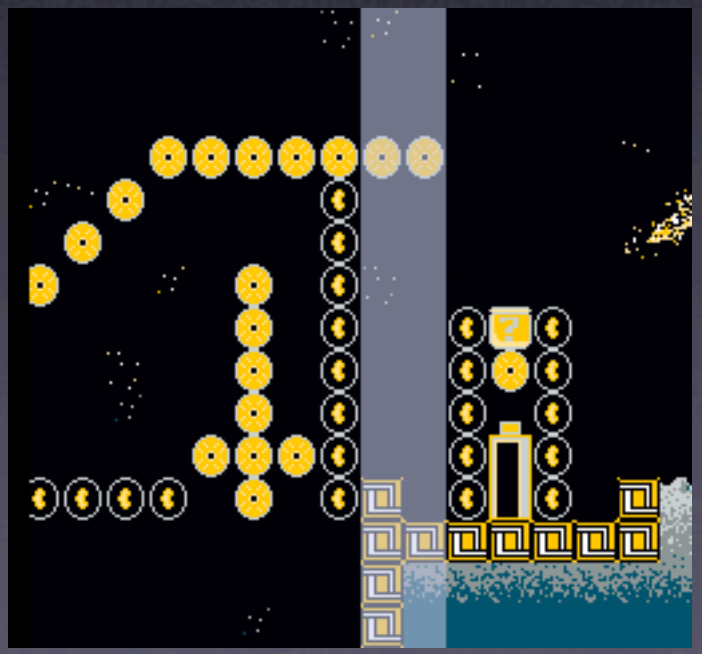
- Column order (4 tiles, 1 attribute byte wide)
- 4-square compression
- 30 blocks, 4 attribute bytes per major column
- 4-state Collision map
- Locations of doors and question blocks are stored elsewhere

SOF Level Data

- Column order, store major columns

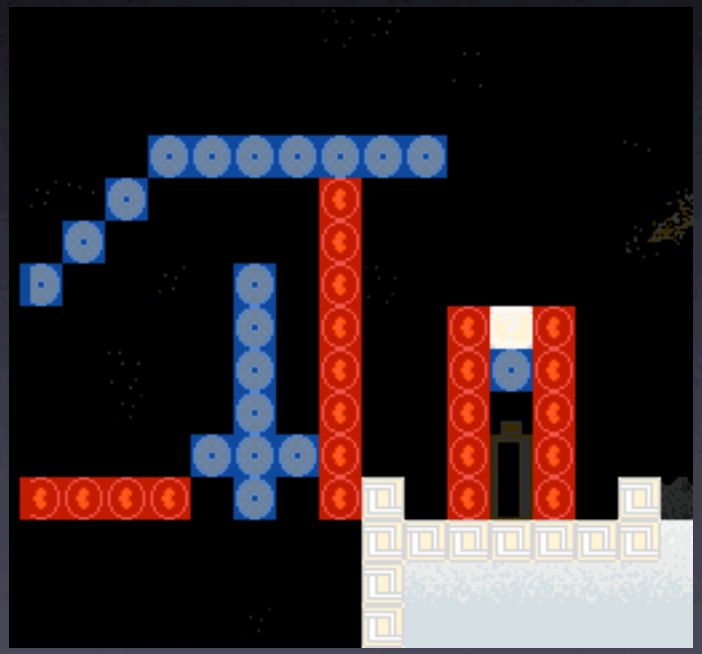
in-game

major column highlighted



SOF Collision Data

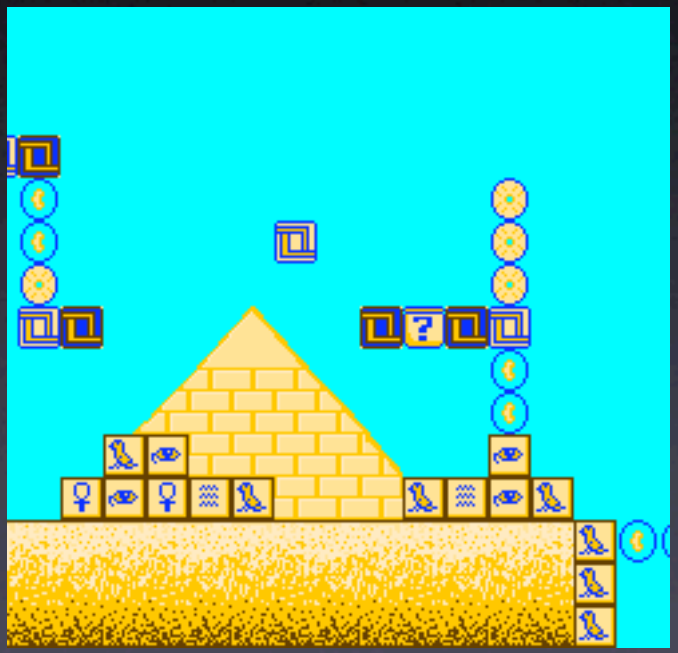
- 4 states (none, solid, sun, moon)

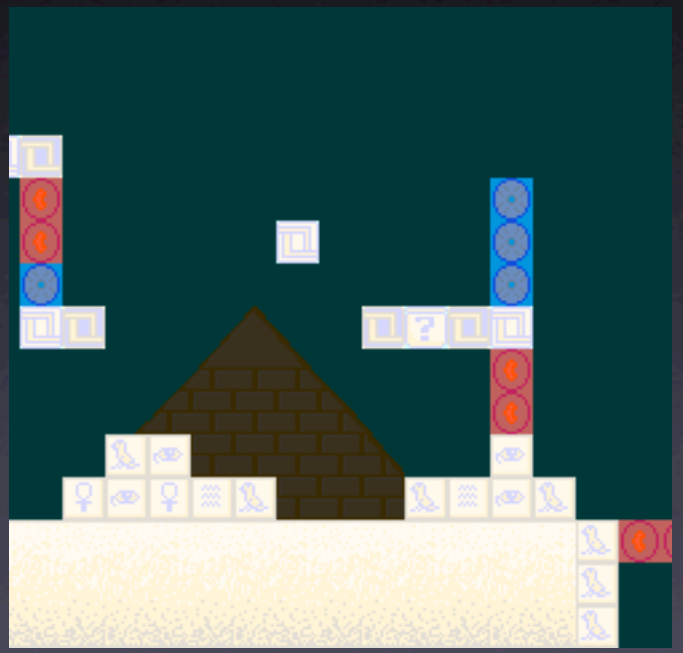


Space Example

SOF Collision Data

- 4 states (none, solid, sun, moon)





Egypt Example

SOF Special Data

- Each level has a limited number of question blocks, doors, and “magical drops”
- Every time SOF hits his head, the game checks if he is below a question block
- Every time the player presses up, the game checks if SOF is in front of a door

Sound Effects

- Check the webpage for NES Sound Test
- This ROM lets you set the bits any sound channel and immediately hear what it sounds like.
- Play around with it to create your sound effects. Write down the values when you make a sound you like.

Sound Effects

some samples to start you off

```
init_sound:
      set $4015 0 //turn off all channels
      set $4015 %00011111 //turn them back on
      return
```

```
playsound_jump:
   set $4000 %10011000
   set $4001 %10001100
   set $4002 %01001101
   set $4003 %10010101
   return
```

```
playsound_squish:
   set $4000 %10001000
   set $4001 %01001000
   set $4002 %00100101
   set $4003 %01001011
   return
```

```
playsound_thump:
   set $4000 %10011111
   set $4001 %10000100
   set $4002 %11010011
   set $4003 %11111100
   return
```

Programming Palettes

- Loader function
- data array

```
load_palette:
   set $2006 $3f
   set $2006 0
   set x 0
   load_palette_1:
      set $2007 [palette x]
      inc x
      if x < 32 branchto load_palette_1
   return


palette:
   data 0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3
   data 0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3
```

Multiple Palettes

- You may want the palette to change when something important happens
- Easy approach: multiple load functions, multiple palette data arrays

```
load_overworld_palette:
   //do some stuff
   return

load_evil_palette:
   //do some stuff
   return
```

```
overworld_palette:
   //some data

evil_palette:
   //some data
```

Palette Cycling

- Sometimes you want certain colors to change over time (gold coins in Mario, player gets hurt and flashes)
- Every frame, or every few frames, change part of the palette

Palette Cycling

- When called each frame, this code will make one palette color cycle through shades of gold every 8 frames
- Powers of 2 are easy cycle times to work with

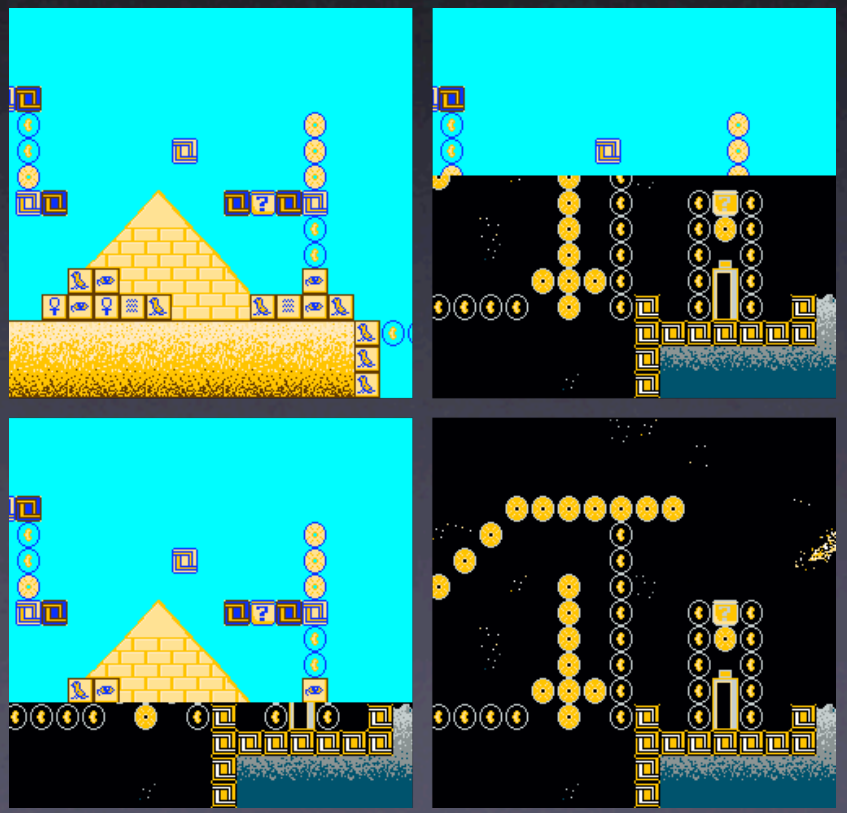
```
cycle_coin_color:
   set $2006 $3f
   set $2006 19 //foreground color 3
   set which_color & current_frame %111
   set $2007 [coin_colors which_color]
   inc current_frame
   return

coin_colors:
   data $18,$26,$27,$28,$28,$27,$26,$18
```

Scene Change Palette

- A scene change (new level, going through a door) often needs to redraw the whole background
- Rewriting an entire name table cannot be done in a single vblank
- You need to either take several frames or turn off the PPU during that time
- If you take several frames to draw, blanking the palette can avoid unsightly graphics.

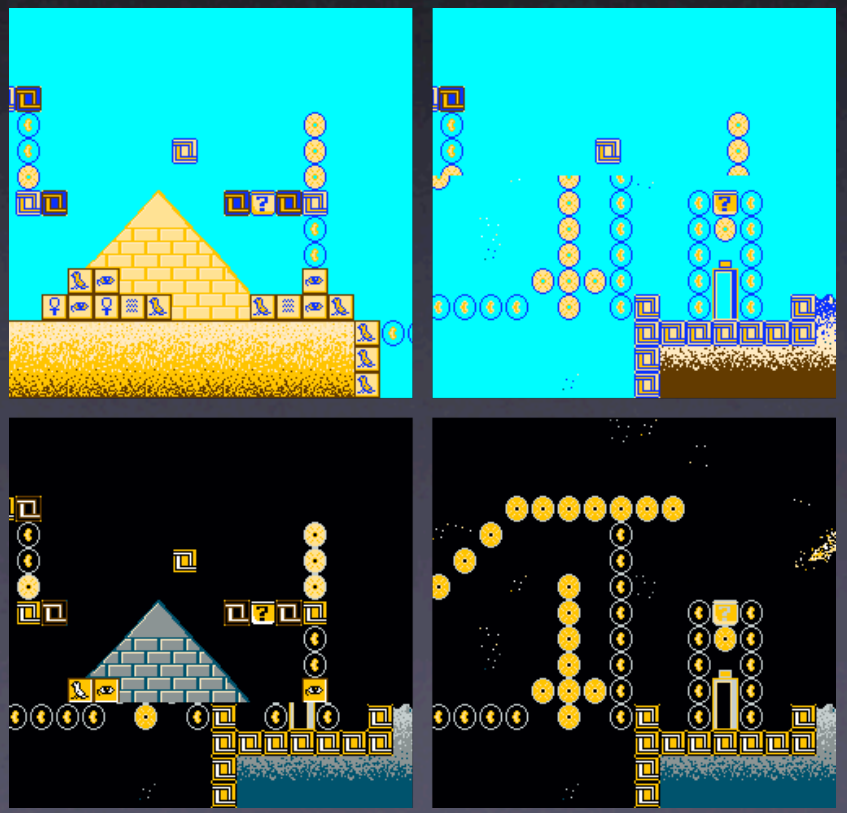
Multi-Frame Change

- Best case bad situation, scenes share a palette, or cause no color conflicts



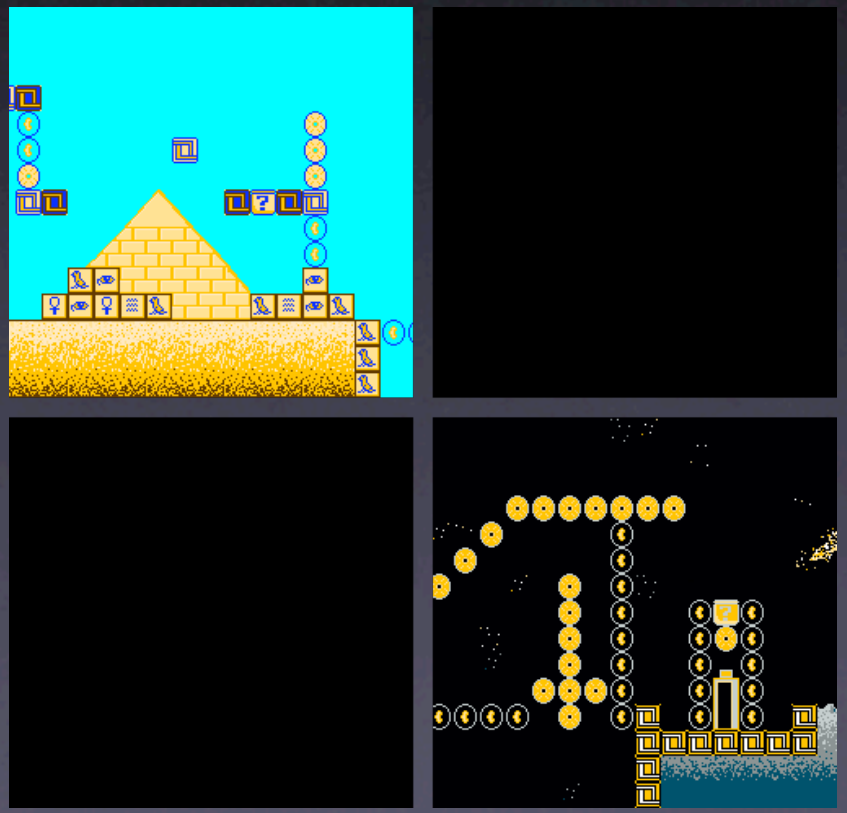
Multi-Frame Change

- Usual case bad situation, scenes have different palettes and screw each other up



Multi-Frame Change

- Blank the palette during the transition
- Ninja Gaiden actually fades the palette



Sprite 0 Hit Tricks

- Group Sooperdooper has been playing around with split screen scrolling
- Ninja Gaiden uses it for parallax scrolling
- Many games use it to scroll one part of the screen and have a static status bar at the top or bottom
- You *may* be able to detect multiple times and locations in a frame, if you want to be really cool

Sprite 0 Usage

- Bit 6 of $2002 tells sprite 0 hit status
- Three loops for proper usage:
  Wait for end of vblank
  Wait for bit 6 to clear
  Wait for bit 6 to hit
- There's some code on the webpage for you

If We Have Time...

- Two-Player Games

Two-Player Game Types

- One at a time
- Simultaneous
  - Competitive
  - Cooperative
  - Ambiguous

One at a Time

- Player one goes until dying or completing a level
- Player two's turn
- Repeat until someone is really dead
- Really easy to design

Simultaneous 2-Player

- Much more interactive and exciting than one at a time
- If you are making a two-player game, consider the various kinds of common simultaneous gameplay

Competitive Games

- The usual sports titles. (football, ice hockey, soccer, dodgeball, volleyball, tennis)
- Fighting games (Trojan)
- Puzzle games (Dr Mario, Puyo Puyo, Puzzle Bobble / Bust a Move)
- Other (World Class Track Meet, other Power Pad games)

Cooperative Games

- Platformers (Chip n Dale, Contra, Mickey Mousecapade)
- Overhead (Gauntlet, Ikari Warriors, various vertical scrollers)
- Puzzle-ish (Pipe Dream, Bubble Bobble, Thunder & Lightning)
- Players have different functions (Gyromite, Gotcha!, Duck Hunt)

Ambiguous

- Games don't necessarily have to be clear whether the two players are helping or fighting each other
- Gyromite and Duck Hunt both let secondary player make it difficult for the other
- Rampage! Destroy the city, but destroy each other too.

Pop Quiz (sort of)

Not graded

- Use a small piece/scrap of paper
- Answer the questions from the next slide
- You can leave after you hand me your answers on the scrap of paper

"Quiz" Questions

1. What is your name?

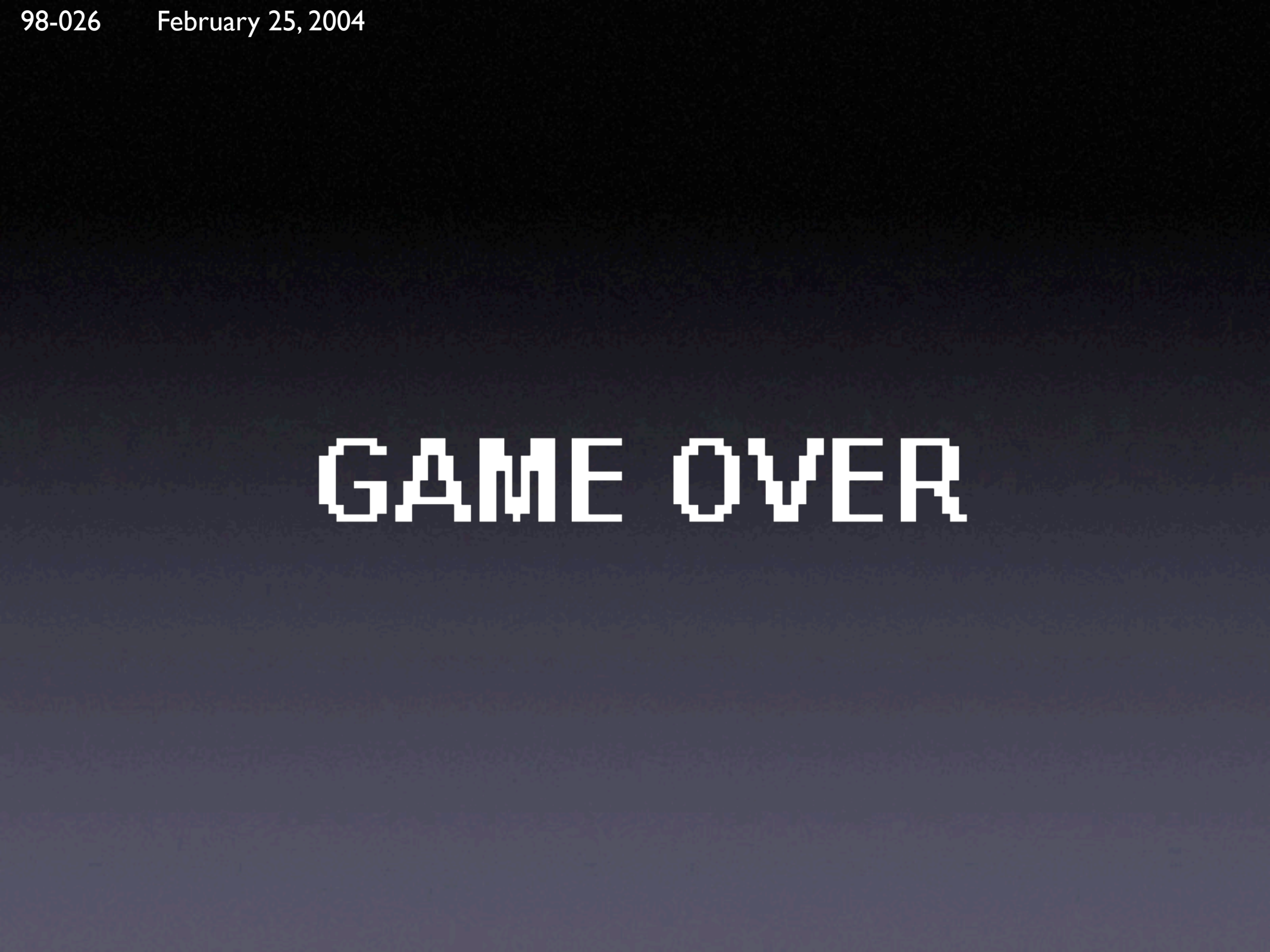
(2-6) How well do you understand...

2. ...the artistic restrictions of making NES graphics?
3. ...the way graphics work on the NES?
4. ...using nesmus?
5. ...how to write a game? *(in general)*
6. ...programming in nbasic?

7. How is your project going?
8. What (if anything) do you need me to explain in more detail?
9. What would you like to me to cover in a future lecture?

How Well:

1 = not at all
2 = okay, if I look at notes
3 = pretty well, maybe looking at notes
4 = I could explain most of it to someone
5 = I could explain it coherently while drunk


GAME OVER