Secrets of Animal Crossing

Adventures in ROM Hacking

James Chambers
Introduction

• Animal Crossing
  • Released in 2002
• “Life simulator” game
  • Interact with villagers
  • Customize house, collect items
  • Events and holidays happen in real time, whether or not you’re playing
Tank

Tell you what! How's about I hook you up with this rad blue bureau as a little housewarming gift!
Introduction

• Does N64 emulation (graphics, some system properties) for main game

• NES emulator for NES games you can acquire in-world
Introduction

• Connects to Game Boy Advance
  • Can transmit NES games to run on Gameboy Advance
  • E-Reader peripheral for collectible cards that grant items, etc. (like an early version of Amiibo)
  • Unlocks game features
Introduction

- GameCube contains customized PowerPC processor
- Extended instruction set
Introduction

• Halloween was approaching and it’d be fun to make a spooky mod
  • Lack of tutorials on doing comprehensive ROM hacking
  • I could make a tutorial in the style that I learned RE/cracking from

• Goals for mod:
  • Create a new holiday based event

• Targets:
  • Dialogue system
  • Event system
  • Quest system
Looking Inside

File formats, symbols, IDA scripts
Looking Inside

• Open up the disc image:
  • boot.dol
  • foresta.rel.szs
  • forest_1st.arc
  • forest_2nd.arc
  • famicom.arc
  • statica.map
  • foresta.map

• Lots of proprietary formats:
  • File format analysis will be important
  • Some documentation on common GameCube/Wii formats already exists
Looking Inside

• .ARC files: archives
  • Contain most of the interesting data files

• Some tools for opening ARC files but not creating them (except maybe sketchy EXEs)

• Found a Python extractor and added archive creation to it
  • Noticed why the other tools didn’t support it
  • It’d be nice to have some generic tools for defining/analyzing binary formats
    • Kaitai and the other thing for game archives
Looking Inside

- Binary files
  - boot.dol
  - foresta.rel
- Importing to IDA
  - Custom PPC instructions
    - “Paired singles are a unique part of the Gekko/Broadway processors used in the Gamecube and Wii. They provide fast vector math by keeping two single-precision floating point numbers in a single floating point register, and doing math across registers.”
  - PPC Altivec IDA plugin: https://github.com/nihilus/PPCAltivec
- Custom REL/DOL loaders
  - https://github.com/heinermann/idawii-loaders
- Kaitai definitions for debugging loaders
Looking Inside

• The symbol map files
  • GameCube build script options:
    # -map – create a .MAP file that shows final memory layout of all sections
• Make simple IDA script for populating database with names
Reversing the Dialogue System
Reversing the Dialogue System

• Initial analysis
• Find the files that contain the message strings
  • *data.bin and *data_table.bin files
<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000470</td>
<td>7F 00 00 09 00 00 15 49 20 63 61 6E 27 74 20 6C</td>
</tr>
<tr>
<td>00000480</td>
<td>65 74 20 79 6F 75 20 67 6F 20 77 69 74 68 6F 75</td>
</tr>
<tr>
<td>00000490</td>
<td>74 CD 67 69 6F 6D 67 20 79 6F 75 20 61 20 6C</td>
</tr>
<tr>
<td>000004A0</td>
<td>69 74 74 6C 65 20 73 6F 6D 65 74 68 69 6E 67 2E</td>
</tr>
<tr>
<td>000004B0</td>
<td>7F 03 06 CD 48 65 72 65 2C 20 74 68 69 73 20 6F</td>
</tr>
<tr>
<td>000004C0</td>
<td>73 20 6D 79 20 66 61 76 6F 72 69 74 65 20 6B 69</td>
</tr>
<tr>
<td>000004D0</td>
<td>6E 64 CD 6F 66 20 73 74 61 74 69 6F 6E 65 72 79</td>
</tr>
<tr>
<td>000004E0</td>
<td>2C 7F 03 08 20 7F 1C 21 CD 7F 01 7F 09 00 00 15</td>
</tr>
<tr>
<td>000004F0</td>
<td>7F 1A 2C 7F 03 08 20 49 20 77 61 6E 74 20 79 6F</td>
</tr>
<tr>
<td>00000500</td>
<td>75 CD 74 6F 20 68 61 76 65 20 74 68 69 73 20 73</td>
</tr>
<tr>
<td>00000510</td>
<td>74 61 74 69 6F 6E 65 72 79 2E 20 49 20 6A 75 73</td>
</tr>
<tr>
<td>00000520</td>
<td>74 CD 77 61 6E 74 20 74 6F 20 73 68 69 6E 77 20</td>
</tr>
<tr>
<td>00000530</td>
<td>74 65 20 67 79 CD 67 72 61 74 69 74 75 64 65 2C</td>
</tr>
<tr>
<td>00000540</td>
<td>7F 03 08 00 7F 1C 21 CD 7F 01 7F 09 00 00 FF 53</td>
</tr>
<tr>
<td>00000550</td>
<td>6E 66 2C 7F 03 0A 20 49 20 68 61 76 65 20 61 20</td>
</tr>
<tr>
<td>00000560</td>
<td>6F 75 20 6D 79 CD 67 72 61 74 69 74 75 64 65 2C</td>
</tr>
<tr>
<td>00000570</td>
<td>6F 75 20 6D 79 CD 67 72 61 74 69 74 75 64 65 2C</td>
</tr>
<tr>
<td>00000580</td>
<td>6E 65 20 73 74 61 74 69 6F 6E 65 72 79 2C 7F 03</td>
</tr>
<tr>
<td>00000590</td>
<td>08 CD 7F 1C 2E CD 7F 01 7F 09 00 00 FF 4C 65 74</td>
</tr>
<tr>
<td>000005A0</td>
<td>20 6D 65 20 74 68 61 6E 6B 20 79 6F 75 21 7F 03</td>
</tr>
<tr>
<td>000005B0</td>
<td>0C 20 4D 61 79 62 65 7F 03 06 CD 73 6E 6F 6D 75</td>
</tr>
<tr>
<td>000005C0</td>
<td>20 6D 65 20 73 74 61 74 69 6F 6E 65 72 79 2C 7F</td>
</tr>
<tr>
<td>000005D0</td>
<td>77 69 6C 6C 20 64 6F 2C 7F 03 08 CD 7F 1C 21 CD</td>
</tr>
<tr>
<td>000005E0</td>
<td>7F 01 7F 09 00 00 15 54 68 65 20 64 65 6C 69 63</td>
</tr>
<tr>
<td>000005F0</td>
<td>61 74 65 20 70 61 74 74 65 72 6E 20 6F 66 20 74</td>
</tr>
<tr>
<td>00000600</td>
<td>68 69 73 CD 73 74 61 74 69 6F 6E 65 72 79 7F 03</td>
</tr>
</tbody>
</table>

...I can't let you go without giving you a little something. Here, this is my favorite kind of stationery, ...!...!

I want you to have this stationery. I just want to show you my gratitude, ...!...!

I o, ... I have a little reward for you! It's so nice stationery...!...!

Maybe I'm some ate pattern of this stationery...
<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>0A</th>
<th>0B</th>
<th>0C</th>
<th>0D</th>
<th>0E</th>
<th>0F</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>48</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>B3</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>B5</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>B7</td>
</tr>
<tr>
<td>00000010</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>B9</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>BB</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>BD</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>DC</td>
</tr>
<tr>
<td>00000020</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>DE</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>E0</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>E2</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>E4</td>
</tr>
<tr>
<td>00000030</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>E6</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>E8</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>EA</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>EC</td>
</tr>
<tr>
<td>00000040</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>EE</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>F0</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>F2</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>F4</td>
</tr>
<tr>
<td>00000050</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>F6</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>FA</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>FC</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>04</td>
</tr>
<tr>
<td>00000060</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>FE</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>02</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>04</td>
<td></td>
</tr>
<tr>
<td>00000070</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>06</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>08</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>0A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>0C</td>
</tr>
<tr>
<td>00000080</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>0E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>10</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>12</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>14</td>
</tr>
<tr>
<td>00000090</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>16</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>18</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>1A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>1C</td>
</tr>
<tr>
<td>000000A0</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>1E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>20</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>22</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>24</td>
</tr>
<tr>
<td>000000B0</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>26</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>28</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>2A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>2C</td>
</tr>
<tr>
<td>000000C0</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>2E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>30</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>32</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>34</td>
</tr>
<tr>
<td>000000D0</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>36</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>38</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>3A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>3C</td>
</tr>
<tr>
<td>000000E0</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>3E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>40</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>42</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>44</td>
</tr>
<tr>
<td>000000F0</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>46</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>48</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>4A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>4C</td>
</tr>
<tr>
<td>00000100</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>50</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>52</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>54</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>56</td>
</tr>
<tr>
<td>00000110</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>56</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>58</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>5A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>5C</td>
</tr>
<tr>
<td>00000120</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>5E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>60</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>62</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>64</td>
</tr>
<tr>
<td>00000130</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>66</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>68</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>6A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>6C</td>
</tr>
<tr>
<td>00000140</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>6E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>70</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>72</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>74</td>
</tr>
<tr>
<td>00000150</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>76</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>78</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>7A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>7C</td>
</tr>
<tr>
<td>00000160</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>7E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>80</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>82</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>84</td>
</tr>
<tr>
<td>00000170</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>86</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>88</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>8A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>8C</td>
</tr>
<tr>
<td>00000180</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>8E</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>90</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>92</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>94</td>
</tr>
<tr>
<td>00000190</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>96</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>98</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>9A</td>
<td>00</td>
<td>00</td>
<td>03</td>
<td>9C</td>
</tr>
</tbody>
</table>
This is my favorite outfit, but you can have it. It'll look good on you, too!

What should I give you in return? Oh, how about this? I just bought this outfit the other day.

Think these clothes will work for you? The fabric is like a massage for your skin. I hope they're comfortable. You can have these clothes. I'm sure you can tell from them that I'm a pretty darned fashionable guy.

Since I certainly don't want to owe you one, you can have this outfit. It's brand new.

Thanks for your help. Consider this outfit your reward, alright? I picked it out myself.

These clothes are your reward for a job well done. I'm sure you'll like them.

I wasn't sure what I should give you for your trouble.
Animation plays
Speech pauses

What?!? What do you want to do first, li'l dude???
Oh! So you're back, huh, lil' dude?
I know it's kind of clunky to be carrying around, but please accept this wallpaper!

This time, I think I'll thank you with some nice wallpaper!

Ooh! I know! I'll give you wallpaper!

Eureka! I've totally got it! You can have this wallpaper!
Reversing the Dialogue System

- Iteratively add codes
  - Focus on what non-printable bytes are left
- Basic editor
  - Doesn’t have all the codes defined
  - Doesn’t support writing special codes back
  - Good for analysis
- Use IDA to figure out the rest of the special codes

```plaintext
[ANIM:NPC0:DEFAULT]I know it's kind of clunky to be carrying around,[PAUSE:0x06] but please accept this wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]

#0x010b @ 0x000001b6c (0x4d bytes):
[ANIM:NPC0:HAPPY_BROWS]This time,[PAUSE:0x06] I think I'll thank you with some nice wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]

#0x010c @ 0x000001bb9 (0x3a bytes):
[ANIM:NPC0:DEFAULT]Ooh! [PAUSE:0x07]I know![PAUSE:0x08] I'll give you wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]

#0x010d @ 0x000001bf3 (0x51 bytes):
[ANIM:NPC0:HAPPY_BROWS]Eureka! [PAUSE:0x05]I've totally got it! [PAUSE:0x0c] You can have[PAUSE:0x04] this wallpaper,[PAUSE:0x08] [PHRASE]!
[CONTINUE]
```
Reversing the Dialogue System

- “ControlCursol” functions for each code handle reading the special bytes and doing something with them
.globl mMsg_Main_Cursol_CursolSetTime_ControlCursol
.globl mMsg_Main_Cursol_CursolSetTime_ControlCursol:

.set var_18, -0x18
.set var_14, -0x14
.set var_10, -0x10
.set arg_4, 4

stwu r1, -0x20(r1)  # Store Word with Update
mfsptr r0, LR       # Move from sprg,
stdw r0, 0x20+arg_4(r1)  # Store Word
stdq f31, 0x20+var_10(r1)  # Store Floating-Point Double-Precision
psq_st %fr31, 0x18(r1), 1, 0# Paired Single Quantized Store
stw r31, 0x20+var_14(r1)  # Store Word
stw r30, 0x20+var_18(r1)  # Store Word
mr r31, r4  # Move Register
mr r30, r3  # Move Register
lwz r4, 0(r4)  # Load Word and Zero
bl mMsg_Get_CursolSetTimeCode  # Branch
fmr f31, f1  # Floating-Point Move Register
lwz r4, 0(r31)  # Load Word and Zero
mr r3, r30  # Move Register
bl mMsg_Set_SizeCode  # Branch
lwz r0, 0(r31)  # Load Word and Zero
add r0, r0, r3  # Add
stw r0, 0(r31)  # Store Word
lwz r0, 0(r31)  # Load Word and Zero
stw r0, 0x40(r30)  # Store Word
lwz r0, 0x48(r30)  # Load Word and Zero
cmpwi r0, 0  # Compare Word Immediate
beq loc_803C30A4  # Branch if equal
.globl mMsg_Get_CursolSetTimeCode  # weak
mMsg_Get_CursolSetTimeCode:

.set arg_4, 4

stwu r1, -0x10(r1)  # Store Word with Update
mfspr r0, LR  # Move from sprg,
stw r0, 0x10+arg_4(r1)  # Store Word
lwz r3, msg_window_t.msg_data_ptr(r3)  # Load Word and Zero
addi r3, r3, 0x20  # message text pointer
bl  mMsg_Get_CursolSetTimeCode_forData  # Branch
lwz r0, 0x10+arg_4(r1)  # Load Word and Zero
mtspr LR, r0  # Move to sprg,
addi r1, r1, 0x10  # Add Immediate
blr  # Branch unconditionally
# End of function mMsg_Get_CursolSetTimeCode
Double check it begins with 0x7F 0x03

Extract pause amount from text buffer

Convert time integer to float
Return from extracting pause interval

Report size of the code

Save interval to msg_window_t.

do stuff with extracted data for code
Reversing the Dialogue System

- Still don’t know where these cursor controller functions are used
  - Cross-references are a dead end
  - Search for the function addresses as raw bytes...
    - They’re held in a function table
    - Data type of the bytes wasn’t defined, so the references didn’t show up
- Referenced by
  mMsg_Main_Cursol_Proc_ControlCursol
    - Performs the table lookup by code
    - Referenced in turn by
      mMsg_Main_Cursol_ControlCursol

cursol_proc_table: .long mMsg_Main_Cursol_Last_ControlCursol# 0
  # DATA XREF: mMsg_Main_Cursol_Proc_ControlCursol
  # mMsg_Main_Cursol_Proc_ControlCursol
  .long mMsg_Main_Cursol_Continue_ControlCursol# 1
  .long mMsg_Main_Cursol_Clear_ControlCursol# 2
  .long mMsg_Main_Cursol_CursorSetTime_ControlCursol# 3
  .long mMsg_Main_Cursol_Button_ControlCursol# 4
  .long mMsg_Main_Cursol_CursorColor_ControlCursol# 5
  .long mMsg_Main_Cursol_AbleCancel_ControlCursol# 6
  .long mMsg_Main_Cursol_UnableCancel_ControlCursol# 7
  .long mMsg_Main_Cursol_SetDemoOrderPlayer_ControlCursol# 8
  .long mMsg_Main_Cursol_SetDemoOrderNPC_ControlCursol# 9
  .long mMsg_Main_Cursol_SetDemoOrderNPC_set_ControlCursol# 0x0A
  .long mMsg_Main_Cursol_SetDemoOrderNPC_set2_ControlCursol# 0x0B
  .long mMsg_Main_Cursol_SetDemoOrderQuest_ControlCursol# 0x0C
  .long mMsg_Main_Cursol_SetSelectWindow_ControlCursol# 0x0D
  .long mMsg_Main_Cursol_SetNextMessageF_ControlCursol# 0x0E
  .long mMsg_Main_Cursol_SetNextMessage0_ControlCursol# 0x0F
  .long mMsg_Main_Cursol_SetNextMessage1_ControlCursol# 0x10
  .long mMsg_Main_Cursol_SetNextMessage2_ControlCursol# 0x11
  .long mMsg_Main_Cursol_SetNextMessage3_ControlCursol# 0x12
mMsg_Main_Cursol_ControlCursol

- Timing and cancel request handling
- Check data at cursor; Print or handle proc code
- Timing and talk animation/sound handling
Reversing the Dialogue System

- GUI editor for string tables
- Translates special codes to serialized text format, e.g. `{{PAUSE:0x03}}`
- Handles special character set
- Note: Adding more entries to a table requires generating a patch
  - Highest entry ID is compiled in, and used for bounds check
1542  Hey. ((PAUSE:0x08))((PLAYER_NAME))(((CLEAR))) Haven't seen you around these days. ((PAUSE:0x08))So I fig... you'd moved, ((PAUSE:0x08))((TAIL)) (((CONTINUE)))

1543  ((SET_DEMO_ORDER_NPC:0x000000))((CLEAR))((SET_DEMO_ORDER_NPC:0x000000))(((PLAYER_NAME)))(((CLEAR)))(((SET_DEMO.ORDER_NPC:0x000FF))) My instincts are the greatest in the entire universe. ((PAUSE:0x08))Yeah, ((PAUSE:0x08))((TAIL)) (((CONTINUE)))

1544  Hey. ((PAUSE:0x08))What's up with... this evening? ((BUTTON)) ((CLEAR))((SET_DEMO.ORDER_NPC:0x000000))(((PLAYER_NAME))) I talked to you, huh. ((PAUSE:0x08))((TAIL))(((CONTINUE)))

1545  How-diddily-oo! ((PAUSE:0x14)) So you're sort of a loner. ((PAUSE:0x08))aren't you, ((PAUSE:0x08))((TAIL))(((CLEAR))) You know, if you don't s... to me every now and again, ((SET_DEMO.ORDER_NPC:0x000000))((TAIL))((PAUSE:0x08)) No lid! ((CONTINUE))

1546  ((SET_DEMO.ORDER_NPC:0x000000))((CLEAR))((SET_DEMO.ORDER_NPC:0x000000))(((PLAYER_NAME))) How I haven't seen you lately. ((PAUSE:0x08))and here you are wandering in the dark. ((PAUSE:0x08))((TAIL)) (((CONTINUE))))

Text:

{{SET_DEMO.ORDER_NPC:0x000003}}Hello, fellow night owl!{{PAUSE:0x0c}}
If it isn't {{PLAYER_NAME}}}{{SET_DEMO.ORDER_NPC:0x000015}}{{BUTTON}}
{{CLEAR}}{{SET_DEMO.ORDER_NPC:0x000000}} was just thinking about how I haven't seen you lately,{{PAUSE:0x08}}
and here you are wandering in the dark, {{PAUSE:0x08}}{{TAIL}}
{{CONTINUE}}}
Unlocking Developer Features
Finding debug features

- Noticed a bunch of functions and variables with “debug” in the name
- Debug features would be useful for testing out mods
- What does `new_Debug_mode` do?
Finding debug features

• Called by **entry** (right after the Nintendo logo splashscreen)

• Allocates a **0x1C94** byte structure and saves the pointer to it

• Value at offset **0xD4** is set to zero right away

• What happens if it’s set to 1?
Animal Crossing™

Press START!

© 2001, 2002 Nintendo
Finding debug features

• Looking for more code that references the debug mode structure...

• There are a bunch of references to “zuru mode” in the context of debug display behavior
  • No idea what it is or what “zuru” means (zulu?)

• zurumode_flag looks important
Finding debug features

• Looked up functions with zurumode in the name:
  • zurumode_init
  • zurumode_callback
  • zurumode_update
  • zurumode_cleanup
zurumode_init

- **Sets** zurumode_flag to 0
- **Checks** some bits in a thing called osAppNMIBuffer
- **Stores** pointer to zurumode_callback in padmgr structure
- **Calls** zurumode_update
zurumode_update

- Checks some bits in `osAppNMIBuffer`.
- Conditionally update the value of `zurumode_flag` based on the bits.
- Prints a format string to OS debug console.
- Characters are not ASCII, so I tried Japanese encodings. It's Shift-JIS:
  - “zurumode_flag が %d から %d に変更されました”
  - “zurumode_flag has been changed from %d to %d”
- Doesn’t mean much yet, but knowing the encoding helped with other debug strings and untranslated game text.
zurumode_callback

• Calls zurumode_check_keycheck first
• Checks a bunch of bits in osAppNMIBuffer
• Prints value of zuru mode flag
• Calls zurumode_update

...what’s zurumode_check_keycheck?
• Didn’t know what zuru mode was or how crucial it was to debugging

• Tried getting translations of “zuru” or slight changes in spelling through Google Translate, got “shake”
  • Thought this might refer to original Wii remote, which was actually made for Gamecube, or some other special kind of developer input device

• Didn’t know if “key check” referred to cryptographic key, controller buttons, or keyboard keys

• Noticed missing/inconsistent symbols

• Held off and looked for path of least resistance
Finding debug features

• Problem with the symbol loader
  • First script parsed out address/name and added it
  • Section addresses in *foresta* map all start from 0
    • Resulted in symbols clobbering each other

• New scripts set up values for each section with:
  • Name at correct address
  • Function or data definition
  • Segment named after the source object (e.g. *m_player_lib.o*)

• The new *bss segment for m_debug_mode.o* had some variables like:
  • *quest_draw_status* and *event_status*.
  • Cross-references from these data entries to a huge piece of code that checks
    *debug_print_flg* (located in the same *bss segment*)
Finding debug features

• Approach in reverse
  • Go from debug display behaviors back up to debug mode activation

• Did some simple NOPping to bypass checks and get displays to activate

• Found `debug_print_flg` and some status variables related to it
  • Set breakpoint where `debug_print_flg` is checked. Never hits.
  • Why? `zurumode_flag` gets checked first.

• `zurumode_flag` keeps showing up throughout debug code and simple patches get the various displays to activate
  • No avoiding it any longer, I have to figure out what zuru mode is
zurumode_init

- Returning to zurumode_init, it initializes a few things:
  - 0xC(padmgr_class) is set to the address of zurumode_callback
  - 0x4(zuruKeyCheck) is set to the last bit of the 32-bit value at 0x3C(osAppNMIBuffer)
- Only runs once on game start
- Patching it to set 0x4(zuruKeyCheck) to 1 causes this text to appear on the title screen
  - But none of the other displays show up during play
zurumode_update

- Checks the last bit of \texttt{0x3C(osAppNMIBuffer)} and updates \texttt{zurumode\_flag} based on its value:
  - If it's zero, the flag is set to zero.
  - If not, it extracts bit 28 from the NMI buffer value and adds 1 to it:
    - The result will always be 1 or 2. The flag is set to this value.
    - When the result is 2 a bunch of interesting stuff shows up.

- Checks whether the flag has changed:
  - If so, it calls some functions from \texttt{boot.dol}:

```c
manager = JC_JUTDbPrint_getManager()
if (flag == 0) {
    JC_JUTAssertion_changeDevice(2)
    JC_JUTDbPrint_setVisible(manager, 0)
} else if (BIT(nmiBuf+0x3c, 25) || BIT(nmiBuf+0x3c, 31)) {
    JC_JUTAssertion_changeDevice(3)
    JC_JUTDbPrint_setVisible(manager, 1)
}
```

zurumode_callback

- Runs each time the gamepad state updates
- Calls the crazy `zerumode_check_keycheck` function
- Checks and sets some bits in `0x3c(osAppNMIBuffer)`
- Calls `zurumode_update`

The last bit of the NMI buffer value is set if:

- bit 26 is set, or...
- bit 25 is set and controller 2 is plugged in, or...
- `0x4(zuruKeyCheck)` is non-zero

Otherwise, the bit is set to zero (disabling zuru mode)
Zuru Mode Activation

0x3C(osAppNMIBuffer)
Bit 26

0x3C(osAppNMIBuffer)
Bit 25

0x4(zuruKeyCheck)
osAppNMIBuffer

• What is osAppNMIBuffer?
• Found it in N64 SDK docs.
  • “osAppNMIBuffer is a 64-byte buffer that is cleared on a cold reset. If the system reboots because of a NMI, this buffer is unchanged.”
• NMI refers to soft reset (via non-maskable interrupt).
• Where do the bits get set?
osAppNMIBuffer

- Bits 25, 26, 28, and 31 of 0x3c(osAppNMIBuffer) control zuru mode
  - 25 and 26 control whether it’s enabled
  - 28 controls the flag level (1 or 2)
- A series of checks in the main function of boot.dol set bits in osAppNMIBuffer
  - Large, somewhat complex function
  - Look for OR instructions with 0x1, 0x8, 0x20, 0x40
Bit 26

- First up: there's an ori r0, r0, 0x20 instruction
  - Applied to the buffer value at 0x3c
  - Sets bit 26, which always results in zuru mode being enabled.
- To reach this block, the eighth byte of the disk ID must be 0x99
  - Try a simple patch for it in emulator…
SDK VERSION: 12Dec2001 Patch4

Nintendo

<DISK ID>
GAME NAME: GAFE
COMPANY: 01
VERSION: 0x99(153)

COPYDATE: 02/08/01 00:16:48
Zuru Mode Activation

0x3C(osAppNMIBuffer)
Bit 26

Game disk ID is 0x99
Instant unlock

0x3C(osAppNMIBuffer)
Bit 25

?

0x4(zuruKeyCheck)

?
Bit 25 and 28

- Bits 25 and 28 get set if the disk ID is greater than 0x90
- Bit 28 controls zuru mode level (1 or 2)
- Bit 25 was associated with that controller connection check...
One of the conditions for enabling zuru mode was:

- Bit 25 is set
- A controller is connected to port 2

If the game disk ID is between 0x90 and 0x98, zuru mode can be enabled by plugging in a second controller, and...

- The second controller controls all of the debug displays!
Zuru Mode Activation

0x3C(osAppNMIBuffer)
Bit 26
Game disk ID is 0x99
Instant unlock

0x3C(osAppNMIBuffer)
Bit 25
Game disk ID between 0x90 and 0x98
Unlocked when a controller is in port 2

0x4(zuruKeyCheck)

?
• The last mystery is zuruKeyCheck
• It gets updated by zurocheck_key_check
  • Cross-reference didn’t show up before because of the way the address is calculated
• What we want at the end is for register 5 to hold 0xB
  • This will toggle the value of 0x4(zuruKeyCheck), enabling or disabling zuro mode
• r5 is stored in 0x0(zuruKeyCheck)
  • Loaded at the beginning
  • Updated at the end
• Follow the blocks up to the beginning and find the constraints:
  • 8040ED74: r5 must be 0xB
  • Sets r5 to 0xB
  • 8040ED60: r0 must be 0x1000
  • 8040EBE8: r5 must be 0xA
  • 8040EBE4: r5 must be less than 0x5B
  • 8040EBA4: r5 must be greater than 0x7
  • 8040EB94: r6 must be 0x1
  • 8040EB5C: r0 must not be 0x0
zeruchek_key_check

• The blocks right before the end will update r5 to some number or reset it to zero based on a comparison.
• It’s a state machine:
  • r5 stores state index and is advanced on correct conditions, or reset to zero.
  • The condition is a comparison to the value of r0.
The values of r0 looks like bit flags...
  - Where do they come from?

Function called every frame via callback function passed to gamepad manager class.

Holding down various buttons on the second controller changes the value
  - Affects 16-bit value at offset 0x2
  - So it is checking for certain button combinations on a controller

The first thing key check does is load the state.

Second thing is load the previous and current button press flags
  - (new XOR old) AND new leaves only the changed button press flags
  - The input to this function is new button presses – this is r0
zerucheck_key_check

• Look up button values in N64 SDK
• It's a cheat combo!
  1. Hold L + R triggers and press Z
  2. D-UP
  3. C-DOWN
  4. C-UP
  5. D-DOWN
  6. D-LEFT
  7. C-LEFT
  8. C-RIGHT
  9. D-RIGHT
 10. A + B
 11. START

<table>
<thead>
<tr>
<th>Button</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_BUTTON</td>
<td>0x8000</td>
</tr>
<tr>
<td>B_BUTTON</td>
<td>0x4000</td>
</tr>
<tr>
<td>L_TRIG</td>
<td>0x0020</td>
</tr>
<tr>
<td>R_TRIG</td>
<td>0x0010</td>
</tr>
<tr>
<td>Z_TRIG</td>
<td>0x2000</td>
</tr>
<tr>
<td>START_BUTTON</td>
<td>0x1000</td>
</tr>
<tr>
<td>U_JPAD</td>
<td>0x0800</td>
</tr>
<tr>
<td>L_JPAD</td>
<td>0x0200</td>
</tr>
<tr>
<td>R_JPAD</td>
<td>0x0100</td>
</tr>
<tr>
<td>D_JPAD</td>
<td>0x0400</td>
</tr>
<tr>
<td>U_CBUTTONS</td>
<td>0x0008</td>
</tr>
<tr>
<td>L_CBUTTONS</td>
<td>0x0002</td>
</tr>
<tr>
<td>R_CBUTTONS</td>
<td>0x0001</td>
</tr>
<tr>
<td>D_CBUTTONS</td>
<td>0x0004</td>
</tr>
</tbody>
</table>
Zuru Mode Activation

0x3C (osAppNMIBuffer)
Bit 26
Game disk ID is 0x99
Instant unlock

0x3C (osAppNMIBuffer)
Bit 25
Game disk ID between 0x90 and 0x98
Unlocked when a controller is in port 2

0x4 (zuruKeyCheck)
Enter 11-step button combo on port 2 controller
Toggle unlock with button combo
Special menus

- Famicom menu
- Map select
- Player select
- Scene selection
QFC ver.011012   (C)2001 Nintendo

[3/21]   R: back   B: demo

:GAME

:GAME/01

->GAME/01/01 Nes Clu Clu 3.bin.szs

GAME/01/02 USA Balloon.nes.szs

GAME/01/03 Nes Donkey Kong 3.bin.szs

GAME/01/04 USA Jr. Math.nes.szs

GAME/01/05 Pinball 1.nes.szs

GAME/01/06 Nes Tennis 3.bin.szs
Bonus
Translations, localization, development history,
Haniwa / Gyroids
Kamakura

• Googling it returns a city
• Look up related message ID in message table:
  • “So what do you think? Isn't this a great igloo, {{TAIL}}?”
• Originally based on snow hut festival in Japan
  • Igloos are the localized version
Death / Funeral