Legend of Zelda: Ocarina of Time
(LoZ: OoT)
Randomizer Project

Team_Z

Chris Muncey, Demi Obenour, Isaac Stone, Cole Flemmons

Goals

- Produce an executable file that will add a new and exciting level of gameplay by modifying the existing Legend of Zelda: Ocarina of Time ROMs with custom and/or randomized game elements.
- The application would require a .z64 file to modify (not included) and would produce a copy with contents rearranged that will be playable via third party emulator software.
- Our project was intended to be licensed under the GNU General Public License, version 3 or any later edition published by the Free Software Foundation.

Motivation

- Motivation:
  - Since all members of Team_Z are part of the Nintendo 64 generation, we wanted to bring to the table a classic, but refreshing take on the game.
- Why we are the right team:
  - When the forces of evil threaten Hyrule, heroes must rise up and fight to protect it. We are those heroes. We are the right team to accomplish this task because we enjoy Nintendo's iconic Legend of Zelda franchise and wish to engage in a new game experience while capturing memories of growing up with the game.

Approach

- We did not know initially if the project was feasible or not
- After some research, we found that modifying the .z64 ROM was possible
- We took a compartmentalized approach where we each worked on different parts of the project
  - Worked, but project contributions were intermittent and development was somewhat chaotic
  - Issues arose when combining multiple constructed elements

Results/Conclusions

- Most of the project goals were met
  - Randomizer
  - Randomized file plays, is guaranteed ‘beatable’
  - Not able to test as extensively as we had hoped
  - Still trying to work out issues with GUI calling the Randomiser and color changer
- Would use a more structured approach next time
  - time management
  - task assignment
  - project planning

Customer Value

- Intended/target user:
  - Gamers who have completed the game before and would like a new challenge
    - Because the randomizer makes the game unpredictable, users will likely need a decent working knowledge of Ocarina of Time to be able to complete the game
  - Speedrunners, who are people who try to finish the game as fast as possible
    - Randomized game would still be beatable by new players, but would miss plot points and would take a long time
- Customer Value:
  - Replay value, replay value, replay value!
    - New experience every playthrough
    - Modifications allows user to skip over monotonous and time consuming cut-scenes

Developer Value

- Modders/Hackers
  - Large online community of LoZ fans who enjoy modifying LoZ games for enjoyment
  - Someone new to modding can look at our code and see how the game file was modified so they may do the same with other games of improve upon our software
  - Similar game file structure to LoZ: Majora's Mask

- GUI Development
  - Use of Qt libraries and environment
  - Learn about basic construction of how a GUI is constructed

Other Modifications

- Removed various barriers/NPCs blocking regions of the map to allow for more initial randomizing options
- Removed long cut-scenes for quicker gameplay/debugging
- Implemented color-changing option for more customization
- Optional GUI for those inexperienced with the command line

Technology/Resources

- Qt GUI (written by Demi)
- Randomizer written in C++ and will modify .z64 file type (written by Chris)
- Utility of Time - Ocarina of Time level editor/model viewer + editor. version .9. Copyright 2006-2008.
- https://wiki.cloudmodding.com/oot/
- C++ IDEs / C++ compilers, in particular Qt Creator
- GitHub, for code hosting and code review
- Online OoT Interactive Map at https://ootmap.com
- Groupme App (group communication)

GUI

- Small, manageable window with randomizer seed options
- Character color selection
- Drop-down to choose file path
- Calls Randomizer to copy and edit file
- Optional command line interface
- Written using Qt5

GUI Implementation

- Written using Qt5 widgets
- Qt Creator's WYSIWYG design tool used to create it.
- Randomizer runs off the main thread to avoid causing UI to hang.
- Qt also used for portable I/O
  - Abstracts Windows vs. *nix
  - Avoids bugs in Windows C/C++ runtime

Randomizer

- Decompresses .z64 file
- Stores entire binary game file (64Mb) as an array
- Changes indexes (addresses) of elements based on randomized values
- Writes entire edited file to a new .z64 file for user

Technical Info

- Chest struct
- Item struct
- Sort array of items by progression, then available
- Sort chest array by available
- Pick random index for both
- If <= 2 chests, pick a progression item
- Some items unlock more chests

Basic Randomizer Pseudo-Code

Initialize array of all chests
Initialize array of all items

Sort both arrays

While (item list is not empty)
{
    If number of available chests <= 2
        choose an item that unlocks more chests
    else
        Match randomly chosen item to randomly chosen chest

    Change ROM for new item/chest combination
    Remove chosen item and chest from available indices
    If item unlocks any chest(s), add chests to possible indices
}

Randomizer Considerations

To Ensure a Beatable Game:

- Certain items allow access to various regions of the game map (and subsequently, more chests)
- When items that unlock a region have been placed by the randomizer, chests within that region become available for item placement
- All items must be placed
- All chests must be used

Demo to Include:

29 Possible Chests (12 Initial Chests + 17 Unlockable via Items)

29 Possible Items

- 8 Progression Items
- 21 Other Items

Color Changer

- Additional program that changes certain characters' in-game colors
- Finds the RGB or RGBa values in the game's memory
- Rewrites those values to reflect the colors that the player chose

Hardening

- Original code did was not robust
  - Missing bounds checks
  - Memory leaks
  - Lots of error-prone pointer arithmetic
- Code was hardened
  - Missing bounds checks added
  - Compiler hardening
    - -D_FORTIFY_SOURCE
    - -fsanitize=undefined -fsanitize-trap-on-error (on Linux)
  - Assertions
  - Goal: should be secure even on malicious ROMs

Changes We Would Make

- Add more structs (Like Locations)
- Add a “requirement” for every location
- Every location has a list of other locations that it connects to, and a list of chests
- Open chests when chest and location requirements are met
- Implement NPC item randomisation and item progression

External Sources

- Utility of Time - Ocarina of Time level editor/model viewer + editor. version .9. Copyright 2006-2008.
- Game file addressing and structure: https://wiki.cloudmodding.com/oot/
- C++ IDEs / C++ compilers, in particular Qt Creator
- GitHub, for code hosting and code review
- Online OoT Interactive Map: https://ootmap.com
- Groupme App (group communication)
- Google Docs (proposal and reports)
- Slideshow Wallpaper: http://videogamegallery.blogspot.com/2012/07/the-legend-of-zelda-triforce-wallpaper_22.html