USING GLITCHES FOR… SPEEDRUNNING?

Ocarina of Time beaten in ~~3 hours~~ ~~7 minutes~~ 4 minutes!

By: Rebecca Reid

The Legend of Zelda: Ocarina of Time (OoT)

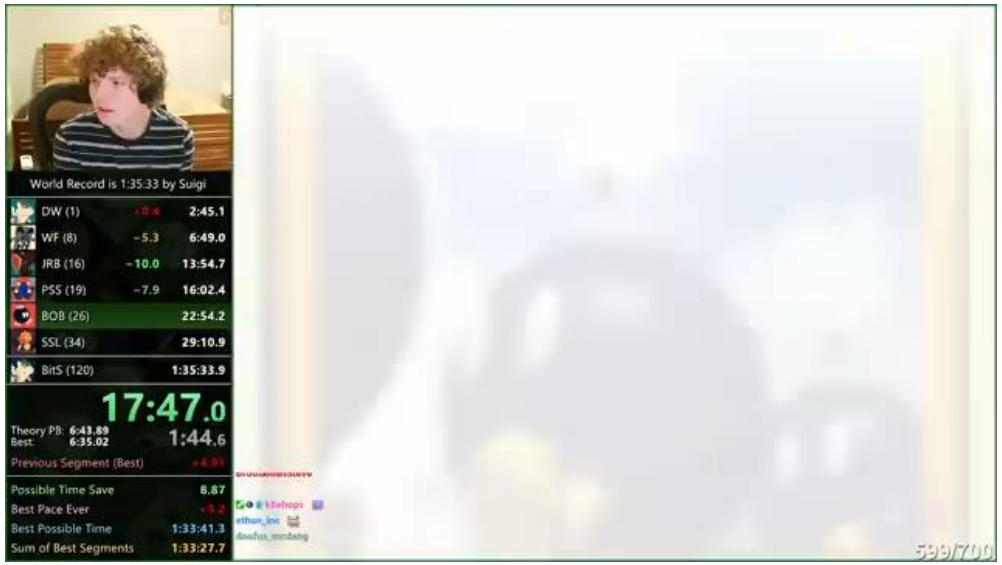
- The Legend of Zelda: Ocarina of Time(OoT) is a Nintendo 64 game that came out in 1998.
- Similar to Super Mario 64, it was the Zelda franchise's first foray into 3D graphics.

Introduction to OoT Speedrunning

- OoT was intended to be beaten casually in 25-35 hours, with plenty of time for experiencing the story, finding collectables and fighting enemies.
- However, to some, this was taken as a challenge: How fast could one beat OoT – or in other words – *speedrun* OoT?

What is speedrunning?

- **Speedrunning** is the act of playing a video game, or section of a video game, with the goal of completing it as fast as possible. Speedrunning often involves following planned routes, which may incorporate sequence breaking and exploit glitches that allow sections to be skipped or completed more quickly than intended.

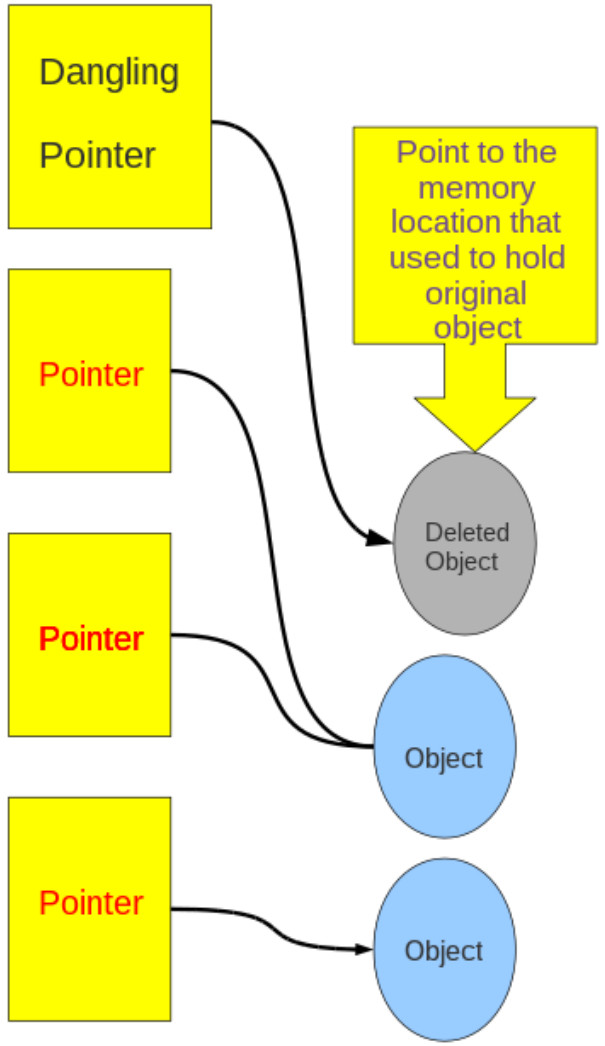


OoT Speedrunning: A history

- The oldest record of speedrunning OoT was from 2003, with a time of 6h:45m. At the time, many speedrunning communities discouraged the use of glitches/exploits. Eventually, glitches caught on as speedrunners found glitches that could shave hours off their times.
- The first intentional usage of a glitch in OoT was an invincibility-frame glitch that allowed a speedrunner to set a 5h:04m run in 2005.
- As more glitches were discovered, the time was brought down to 2 hours, then 1 hour, then 30 minutes, then even 18 minutes in 2017, with very little of the game actually played. The record hovered around ~18 minutes from 2013-2020. That is, until **stale reference manipulation** was discovered.

Stale Reference Manipulation

- Stale reference manipulation(SRM) is a glitch discovered in January 2020 that allowed an OoT player to execute arbitrary code.
- Through a series of convoluted, specific actions, you could run ANY code you wanted. This included warping anywhere, enabling debug mode, and more. The discovery pushed the game's current world record time to 3 minutes and 47 seconds!
- This glitch works using a form of use-after-free, where a reference to an unloaded object(a stale reference, if you will) continues to update even if other data is loaded in that object's place.



SRM: How does it work?

- In OoT, interactable things like enemies, items, and the player character Link, are stored in a data structure called an actor. These actors have information like X,Y,Z, position, angle, speed, ID, and other variables. They are loaded and stored on the heap.

- When the player picks up an object, like a pot or a rock, Link's actor points its "held actor" reference to the held object. This lets the game update the position of the object in case the player wants to throw the object.

SRM: How does it work? - 2

- In normal gameplay, when Link throws the rock, Link's reference dereferences, the rock plays its throwing animation, breaking animation, then unloads. If Link tries to walk through a loading zone, the rock tries to unload, but a "don't unload" flag set on the object when Link picks it up, preventing unloading and letting Link continue to carry it.
- However, there is a set of actions discovered that allow Link to continue "holding" an actor even after it unloads, which updates the position in the heap the object used to occupy.
- A camera glitch lets Link pick up an object without setting the "don't unload" flag, so when he walks through a loading zone, the object unloads, without Link's reference dereferencing.

- Here, Link uses the camera glitch to lock the camera in place, which lets him avoid the "don't unload" flag. When he crosses the loading zone, he finds himself holding nothing??

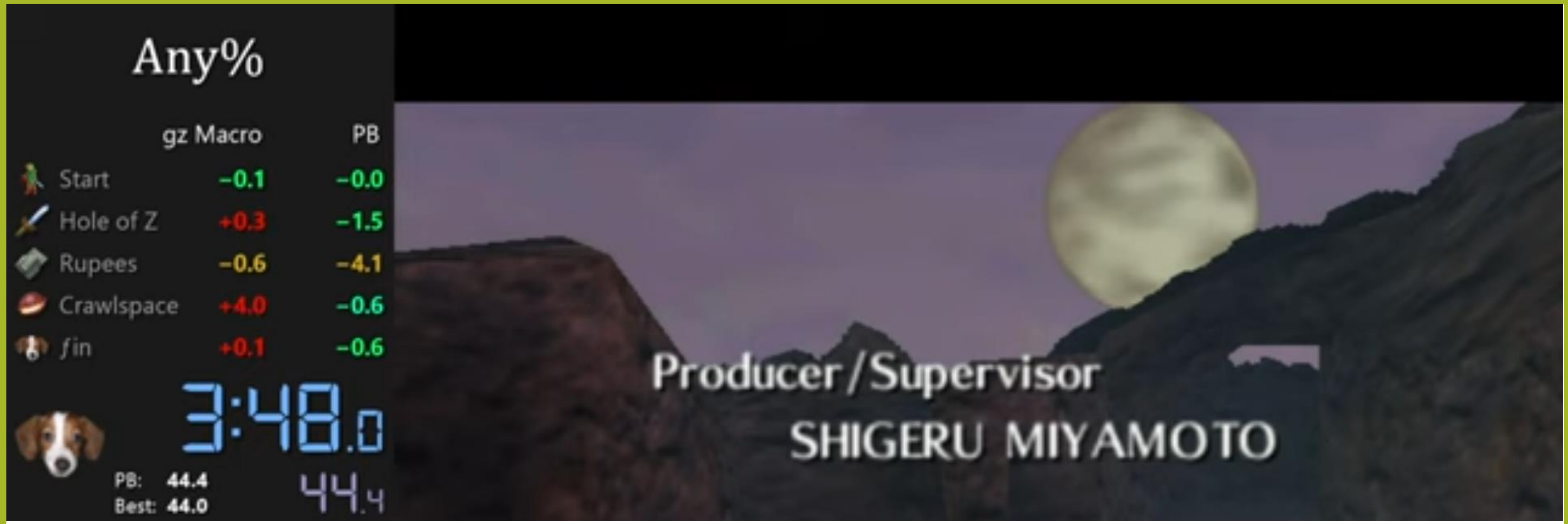
- Since objects load/unload from the heap all the time, when Link's reference tries to update the reference's coordinates, it could be writing to literally anything, and usually causes a crash.

Arbitrary Code Execution

- Speedrunners can take advantage of the stale reference by setting up the heap by moving around and acting in a specific way. This allows Link's stale reference to write to the same location every time.
- In the actual speedrun, the stale reference is set to code that belongs to a hidden fairy. This code runs when fairy unloads, which can be triggered by the player changing maps.
- The speedrunner will obtain a stale reference to this fairy code, overwrite code-like values into the fairy code, then trigger a map change, which runs the code-like values as code.

The Payload

- Normally when fairies are unloaded, they run a function. This function, which removes a lighting effect, modifies memory in two places.
- Runners use SRM to modify this function based on the first and second halves of the player's filename.
- the first half of the filename sets all the grass polygons to a specific map exit within Kokiri Forest, and the second half to set the current cutscene value to 7, When the runner moves maps, it unloads the fairy and activates the effects of the SRM, which wrong warps the runner to the credits.


Any%
gz Macro
PB
Start
-0.1
-0.0
Hole of Z
+0.3
-1.5
Rupees
-0.6
-4.1
Crawlspace
+4.0
-0.6
fin
+0.1
-0.6
3:48.0
PB: 44.4
Best: 44.0
44.4
Producer/Supervisor
SHIGERU MIYAMOTO

AND CREDITS!

Bonus: Could this have been prevented?

- I personally think any method used to prevent this would have either made the game file too large to fit on the game cartridge, or was unknown to programmers at the time.
- For example, a protection could be Non-executable stack. However, the game made heavy use of code on the stack, and changing this could have increased development time by a lot. Most of Nintendo's previous games also used executable stacks, so it was likely unknown or not a viable option for them.
- In addition, this was one of the first 3D games ever. The architecture had to be made from scratch, with little/no material to re-use or learn from.
- These days, modern games make heavy use of updates over the internet, which was not an option for physical cartridges.

Bonus: The fairy code

```
void LightContext_RemoveLight(GlobalContext*
  globalCtx, LightContext* lightCtx, LightNode*
  node) {
 if (node != NULL) {
     if (node->prev != NULL) {
         node->prev->next = node->next;
     } else {
         lightCtx->listHead = node->next;
     }

     if (node->next != NULL) {
         node->next->prev = node->prev;
     }

     Lights_FreeNode(node);
 }
}

typedef struct LightNode {
  /*0x0*/  LightInfo* Info;
  /*0x4*/  struct LightNode* prev;
  /*0x8*/  struct LightNode* next;
} LightNode;
```

At offset 0x264 in En_Elf, there exists a pointer to a LightNode (pointing to 8011xxxx). When En_Elf is unloaded, the function LightContext_RemoveLight is run on the node:

This means when you remove a LightNode using LightContext_RemoveLight, you essentially perform the following operations:

Get the address found at node+0x4, add 0x8 to it, and write the value at node+0x8 to the address

Get the address found at node+0x8, add 0x4 to it, and write the value at node+0x4 to the address

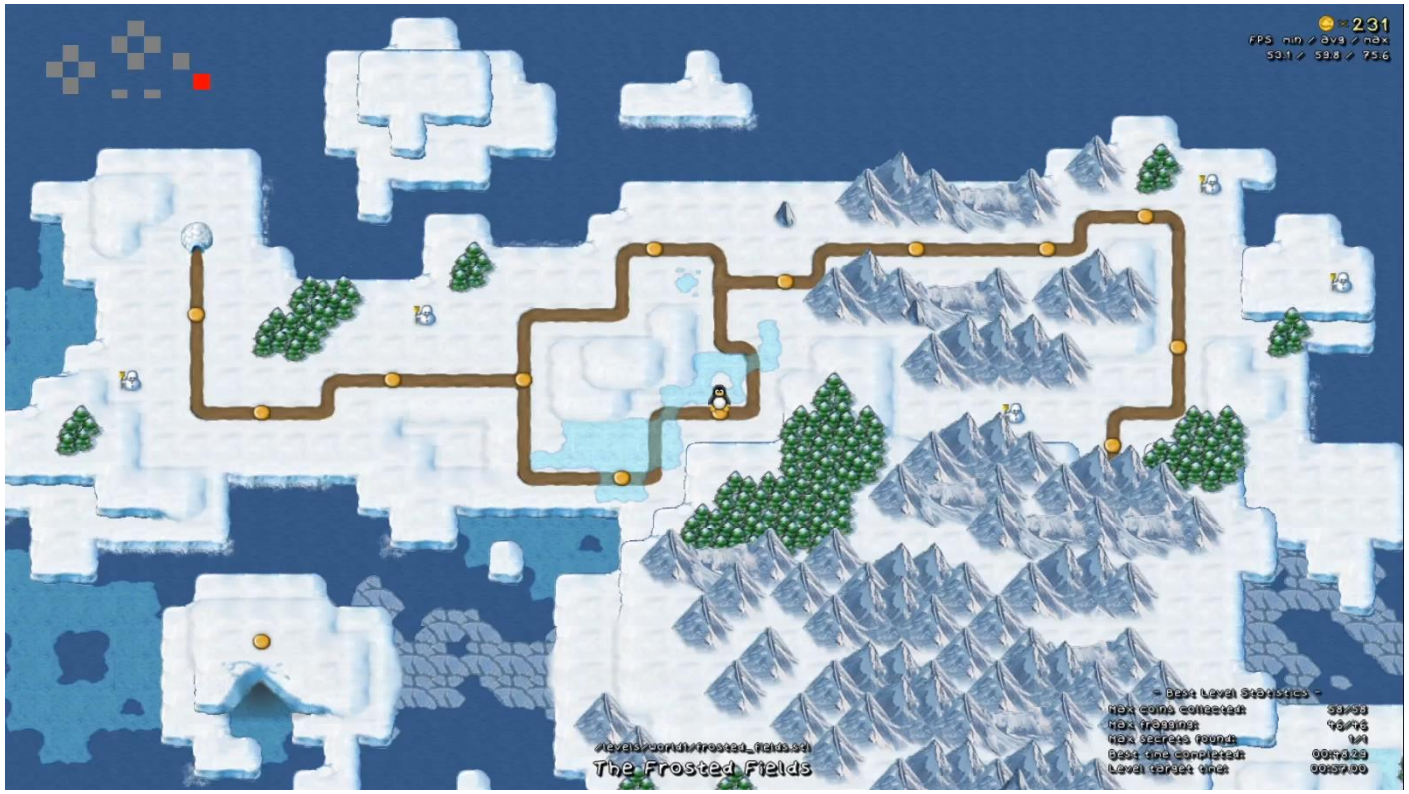
If we SRM this pointer to point to filename, we can perform an arbitrary RAM write to anywhere in memory.

Bonus: Not just for speedruns

- It's interesting that this discovery was only made after 22 years, but there is a growing number of old(and new) software and games that have exploits that allow arbitrary code execution. There's games that allow a user to jailbreak their consoles. Super Mario World had an arbitrary code execution glitch that programmers used to make it play snake and pong. Ocarina of Time's glitch also has been used to add new features to the game, like adding in Arwings from Star Fox.

Bonus: The World Record

Bonus: SuperTux



- ***SuperTux*** is a free and open-source 2D side scrolling platform video game[7] inspired by Nintendo's *Super Mario Bros.* series. Players control Tux, the mascot of the Linux kernel, through a variety of levels and worlds to rescue his girlfriend, Penny, from the evil Nolok.[8] Originally developed for Linux operating systems, the game has since been ported to Windows, Mac OSX, Steam, and other platforms.