Harley Vispo, Kevin Zheng and Thomas Frederick

CS 477

Speedrunning in Video Games: Routing, Knapsack and Optimization of Mario (TAS)

Introduction

Speedrunning is a particularly interesting subject. Obviously, almost all games are meant to be completed. Speedrunning however, proposes a simple question with a rabbit hole of answers: How fast can that game be completed? To try and answer that question, people run through a video game as fast as possible, using whatever methods possible to improve the time, with the most common one being exploiting glitches within the game to ignore once thought to be necessary segments of the game. At times, this can even require performing acts that are outside of the realm of human capabilities. Speedrunning has become an incredibly popular activity to both perform and watch as of late, thanks to the large number of video streaming websites that have popped up nowadays. Some people even make a living off of streaming themselves perform the same actions over and over for hours on end, just to cut seconds off of the record at a time.

In an attempt to explain the complexities of speedrunning, we will be describing the often-used methods of speedrunning in terms of the knapsack problem, breaking each section of the game *Super Mario Bros.* into sections of time, and how by performing proper glitches and routing, most of the game can be skipped. To help explain this, we created a TAS, or tool assisted speedrun. A TAS is a “perfect” speedrun, taking every possible shortcut, human-capable or otherwise, to achieve the fastest known time possible.

Glitches And Routing

Glitches and routing are crucial elements to almost any given speedrun. Glitches are a flaw in the game's hardware or coding of the game itself. By exploiting particular glitches in a game, one can perform unintended actions that can potentially move a player further into the game than a player not performing the glitch. For *Super Mario Bros.* the glitches in question are fast acceleration, wrong warps, flagpole skips, wall jumping and walking through walls. Most of these glitches require either frame-perfect timing or performing actions impossible to humans.

Fast acceleration is an exploit to make Mario move faster from a standing position by inputting both left and right into the controller in one frame. However, it is physically impossible to perform this glitch, as the standard controller cannot accept more than one directional input at a time, and thus requires either an artificial input or a modified controller, which is ineligible for most speedruns. Walking through walls is a glitch with many purposes and the source of the rest of the glitches explained. By moving at an optimal speed and performing a frame-perfect jump, Mario can be pushed into walls, allowing him to cross into areas that would otherwise be more time-consuming. Wrong warps are a glitch caused by Mario attempting to cross a loading zone before the proper flags check where he should be going, sending Mario to an unintended location. Wall jumping lets Mario jump off walls due to an error where if he is close enough in proximity, walls count as a floor for a single frame. Flagpole skipping is performed by moving Mario into the baseblock of the end flagpole, skipping the flagpole falling animation.

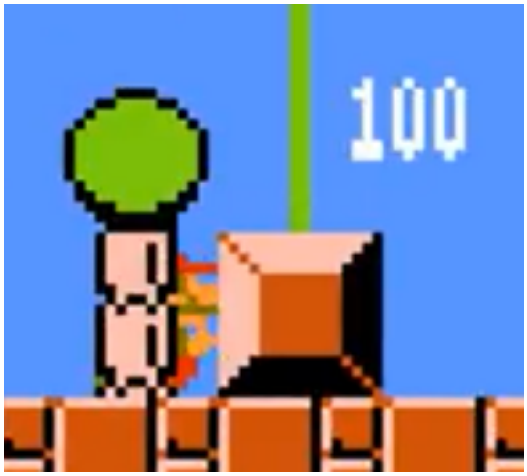

Figure 1. An example of the flagpole glitch. Mario has connected with the bottom of the flagpole through the block, allowing the player to skip the animation for running to the castle.

Routing on the other hand is fairly simple. *Super Mario Bros.* has eight Worlds with four levels per world. However, there are certain locations in the game that allow one to progress much further in the game. Within the first level of the first World, there exists a pipe that Mario can enter that takes him to a bonus area whose exit lets him skip most of the level. Within the second level of the first World, there exists a "Warp Zone" in which Mario can use to move to World 2, 3, or 4. Similarly, in the second level of the fourth world, another "Warp Zone" exists which can take Mario to Worlds 6, 7 or 8. By using both "Warp Zones," one can complete the game by only playing through levels 1-1, 1-2, 4-1, 4-2, 8-1, 8-2, 8-3, and 8-4, skipping 5 Worlds and 24 levels.

In a typical speedrun of the game the major events of the game are such: Start the game, Complete 1-1 with the pipe to skip most of the level and touch the flagpole to end the stage. Complete 1-2 by accessing the "Warp Zone" to skip to level 4-1. Complete 4-1 and touch the flagpole to end the stage. Complete 4-2 by accessing the "Warp Zone" to skip to level 8-1. Complete levels 8-1 through 8-3 by touching the flagpole to end the stage. Complete 8-4 by touching the axe and finish the game.

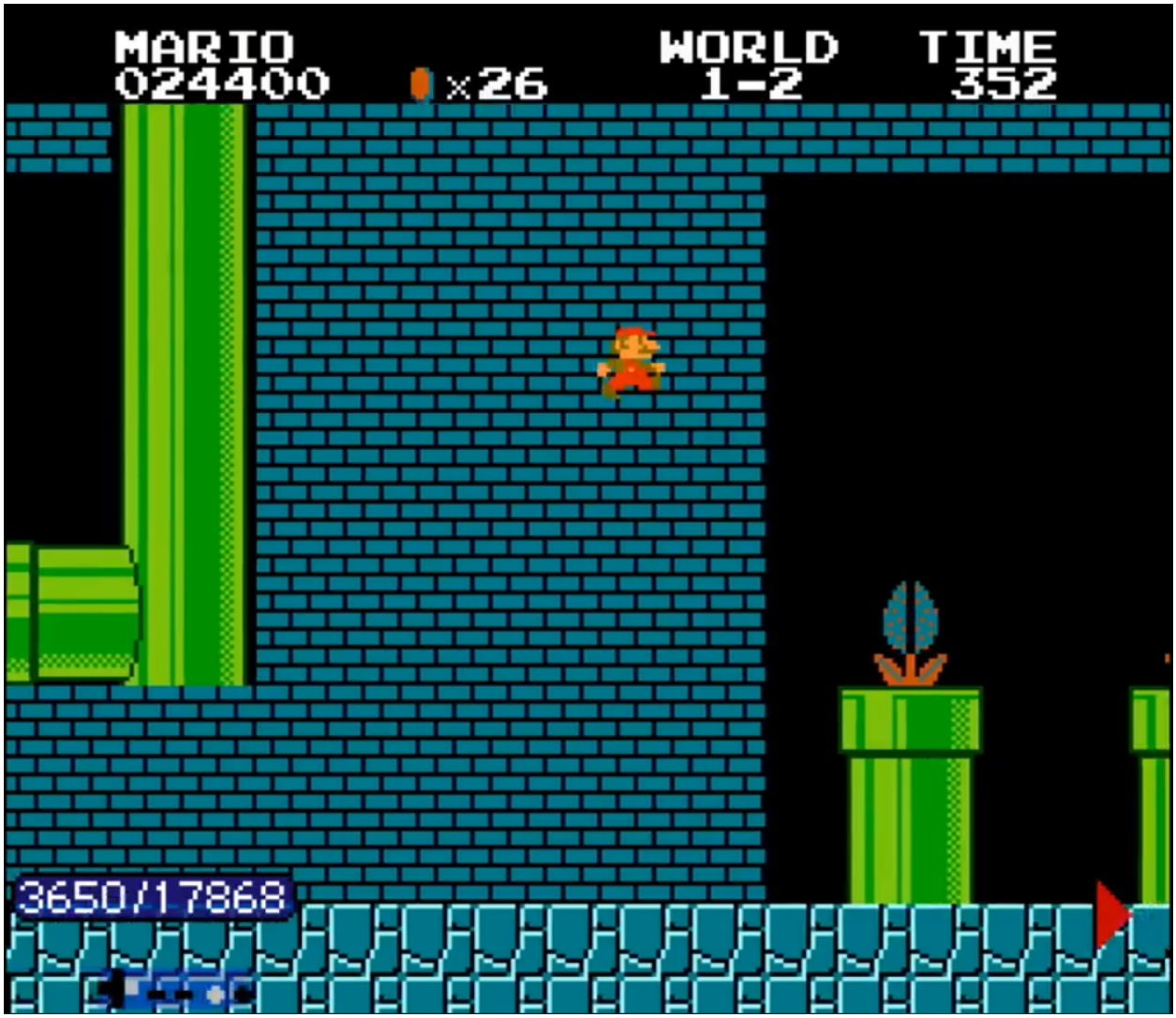

Figure 2. Mario deploys the wall glitch and is actively walking through walls to enter the warp zone area. Depicting both a glitch and an example of routing and its application.

The Knapsack Problem

The knapsack problem is a scenario in combinatorial optimization that describes an issue where one has a number of items, each having their own weight and value, and must decide how much of each item can be collected so that the total weight is less than or equal to a given limit, while maximizing the total value. The name of it is derived from the example scenario of someone being given a knapsack that can only hold a maximum weight, and asked to fill the bag with as much value as possible.

Applying it to speedrunning a game, one could say that the entire speedrun is a modified knapsack problem, treating sections of the game as events that are performed in a specific amount of time, and removing unnecessary sections to minimize the total amount of playtime. Removal, of course, being done via skips and glitches.

Each level is performed under an in-game timer of 400 units, of which translate to 0.4 seconds in real time, placing each stage's time limit to about 160 seconds. By seeing how long each glitch takes its respective "Event" compared to an non-glitched attempt of the same event we can see how much time is being saved.

Assuming each level can be completed

- Completing 1-1 via the bonus pipe took only 12 seconds, while completing it without the pipe takes about 24 seconds, saving one 12 seconds of time from that skip.
- The flagpole's descent takes 1 seconds to complete, so by ignoring the flagpole, this saves 1 seconds of time per level with a flagpole, and with 5 flagpole levels, this saves 5 seconds of time.
- Completing 1-2 via the Warp Zone skips over levels 1-3 and 1-4, along with the entirety of Worlds 2 and 3. Assuming each level takes about 24 seconds of time to complete, this saves one about 240 seconds.
- 4-1 is completed normally, with the only notable exception being the flagpole glitch, which was previously mentioned.
- Completing 4-2 via the Warp Zone skips over levels 4-3 and 4-4, along with Worlds 5, 6, and 7. Once again, assuming each level takes about 24 seconds to complete, this saves 336 seconds.

- The rest of the game is completed normally.

Although these are the major events to be played, the glitches mentioned improve the performance of each event. The Fast Acceleration glitch mentioned saves about a second each time Mario moves from a starting zone, such as exiting a pipe or starting a stage. Walking through walls saves Mario about 4-5 seconds per use, ignoring sections of the level that would otherwise require unnecessary time to complete.

Methods/Attempts

In an attempt to try and show these glitches off and how efficient of a speedrun we could make, we attempted to create a TAS. We created a TAS rather than attempt to run the game ourselves in order to access the non-human capable exploits, along with the issues of none of us having enough experience with *Super Mario Bros.* to attempt a proper speedrun. We used the emulator BizHawk, as it was designed for speedrunning in mind, and had the tools to help record and edit a TAS as necessary.

Using a TAS run of 4:54 as a guide, we attempted to create our own speedrun via going through the game frame by frame, and changing the inputs that were performed at specific times.

Problems/Issues

We ran into several problems as we created our speedrun. The most notable issues occurring were due to the emulator. Originally, we attempted to try and work with the same game that the paper this one is based off of used, *Castlevania*, but in our attempt of the TAS, inputs were not being read and simply failed. As a result, we were forced to edit our topic to some degree and switched games to *Super Mario Bros.*

Another issue we ran into was with the emulator itself. The emulator, for whatever reason, was prone to stuttering and frame drops, which made rewatching the runs frame by frame

difficult at times. It worked for the most part, but inexperience with the emulator and its settings were the most likely cause of why the issue was not entirely solved.

End Result

Creating a tool assisted speedrun is not easy even when creating one based off of another TAS run. The TAS run we created was not as efficient as the TAS run it was based off of. Due to inexperience with TAS runs and our limited knowledge of performing speedruns, creating the most optimal tool assisted speedrun was not in the realm of possibility. Creating a TAS run though did teach the importance of knapsack to get the most out of a speedrun. Being able to maximize the speed of a speedrun through the minimal steps and levels throughout the game shows how important it is.

Conclusion

Speedrunning is the act of completing a game as quickly as possible using as many possible means to do so. Although a seemingly simple task, the methods used to improve times often require more than just human inputs. These can range from creating the most efficient route to the end or to exploiting glitches in the game's programming to perform unintended but time-saving actions. Glitches range from walking through walls to a small time save at the end of most levels. Creating a tool assisted speedruns had its own handful of problems, however it can skillfully depict the differences in optimization with all of these other factors in play. Since all of the TAS inputs are done manually, it can be tedious and time consuming to create a TAS run, however it is still highly beneficial since it can heavily disregard human error while simultaneously saving time with the usage of an emulator, rather than having a human attempt multiple runs, multiple times.

Bibliography

Lafond, M. (n.d.). (rep.). *The complexity of speedrunning video games*.

Saradoc. (n.d.). *Useful Glitches*. speedrun.com. https://www.speedrun.com/smb1/guide/09v5d.

YouTube. (2017). *[Tas] Super Mario Bros. Glitchless in 5:02.55*. *YouTube*. https://www.youtube.com/watch?v=Zn0unJPFtqI&t=132s.

YouTube. (2017). *[Tas] Nes Super Mario Bros. in 4:54.03 by tubby*. *YouTube*. https://www.youtube.com/watch?v=sNBWxmItfpU&t=1s.