Weird WarioWare

Instructional Dissonance and Characterization in WarioWare Inc., Mega Microgame$!:

Josh Fishburn

Weird and Well Played

WarioWare, Inc.: Mega Microgame$! is not just weird, it's in-your-face weird. Over the course of the game, the player presses buttons to accomplish the following: pick a nose, protect a quail egg from an earthquake caused by the thunderous hammer of a giant, brush teeth, sniffle to stop a running nose, and bounce a watermelon in the air using a person as a trampoline. In addition, there's a mad scientist who gets diarrhea from drinking lab chemicals, a duo of cab drivers who find out that the player-character is a mermaid/merman, and an afro-wearing disco dancer who designs mobile games.

More strange, and well played by Nintendo, is the instruction manual, which features a sticker book and a set of games within the written instructions that, as I'll argue, end up being diversions crafted by Wario. The instruction manual doesn't don't do much *instructing* at all. Along with all the obviously weird aspects of *WarioWare*, this one stands as unusual, but notable, because it leaves so much of learning how to play up to the player.

WarioWare, Inc.: Mega Microgame$! (or simply *WarioWare*) is a 2003 game for Nintendo's Game Boy Advance console that features brief games (microgames) played in quick succession. The game is also notable for its narrative, its characterization of Wario, the cryptic instructions that the game offers before each microgame, its use of only the directional pad and "A" button to control each microgame, and its consistent, often crude wackiness. *WarioWare* has become a

core series for Nintendo, with sequels on Nintendo's portable and home consoles.

While the original *WarioWare* was experimental by design, it did not necessarily highlight the unique capabilities of the Game Boy Advance. What it did was set a precedent for what I'll call "instructional dissonance", in which Wario's characterization as a greedy narcissist comes to light in the way the game teaches, both through the instruction manual and in the prompts provided to the player before each microgame. Throughout this paper, I will argue that this instructional dissonance is consistently applied across the full game package. To start, I will compare the design of the *WarioWare* instruction manual with the typical approaches that videogame instruction manuals take. We also need to understand the in-game instructions in the context of each microgame that they instruct. But first, to know why Wario, of all of Nintendo's characters, makes sense to lead this strange collection of micro-videogames, we need to know understand his history and emergence as a popular Nintendo character.

The Character of Wario

> *Either way, I'm still a cad! I hate everybody! Yaaarrghh! — Wario (after losing all of his money at the end of WarioWare Inc., Mega Microgame$!)*

By the time that Nintendo released *WarioWare* in 2003, Wario had long been a part of their core cast of characters. He first appeared in 1992's *Super Mario Land 2: 6 Golden Coins* (Nintendo R&D1, 1992) as Mario's antagonist and the final boss, but would get his own games soon after. His early characterization as a greedy narcissist solidified in his first starring game, *Wario Land: Super Mario Land 3* (Nintendo R&D1, 1994), which resulted in a game with a different style and pacing than its prequel. *Wario Land* brought coin and treasure collecting to the fore, making it necessary to advance further in the game and achieve better endings.

While there were changes to the game's rules, Wario's driving narrative force remained his unfettered greed throughout the *Wario Land* series. When he started appearing in the popular *Mario Kart* series, his catchphrases included "Wario is great!", "I lost! To a buncha' losers!", and "I HATE you!" In the *Super Smash Bros.* series, one of his attacks is an explosive fart called the "Wario Waft". As Mario's evil twin, Wario allows Nintendo to explore darker, stranger material (while keeping their kid-friendly sheen intact). While Mario is clean, friendly, and generous, Wario is disgusting, narcissistic, and miserly. But he's not stupid; in fact, he's a game developer!

When we first see Wario in *WarioWare*, he's leaning back in a recliner, picking his nose, and watching television. He sees an advertisement about the latest, greatest videogame and is struck by the thought of mountains of money that he could be making by selling his own videogame. He recruits his friends (although none of the games have explained how Wario has any friends in the first place) to help him create this game, incorporates as WarioWare Inc., and gets to work. In *WarioWare*, you start the game by naming your character, and Wario's game designer friends refer to you by name throughout the game. Each game designer runs into some trouble along the way, and you help them through that trouble by playing through their microgames. Here's where things get confusing.

In-Game Instructions

> *Why does WarioWare make any sense at all? (Gingold, 2005)*

In *Gaming: Essays on Algorithmic Culture*, Alexander Galloway (2006) proposes a model for formal analysis of games, highlighting "four moments of gamic action". This model borrows the concept of diegesis from film theory, with the diegesis of a videogame representing the "game's total world of narrative action." (Galloway, 2006, p.7) In addition to diegetic and nondiegetic elements, the model considers operator (typically referred to as the player of a game) and machine actions, giving four categories of action:

1. Diegetic operator actions (e.g. Moving an avatar)
2. Non-diegetic operator actions (e.g. Pressing buttons, changing game settings)
3. Diegetic machine actions (e.g. Cutscenes)
4. Non-diegetic machine actions (e.g. Powerups, gravity, or glitches)

While playing a videogame, it makes little sense to think of these categories separately, but they provide a useful model of analysis for the instructional dissonance that exists in the pre-microgame instructions in *WarioWare*.

In *WarioWare*, a typical play session involves choosing a genre of microgame (which is represented by one of Wario's game designer friends), then attempting to complete as many of the chosen microgames in succession as possible. A particular genre may include games with the same fictional theme (e.g. Sports or Sci-Fi) but the actions that are required vary widely and are not limited by the chosen genre.

> *Part of the cognitive friction WarioWare creates by changing games so fast, is that you can't map nouns and verbs from one game to the next. One game may contain a snowboarder, and you figured out that by pushing left and right on the directional pad you could guide her through a gate, but the next game has no snow boarder, gate, or even snow. (Gingold, 2005)*

More confusing is that even if the next game had a snowboarder, gate, snow, and an identical color palette, it might still be significantly different. It's not only the rapid context-switching that causes confusion, but also that the game deliberately sows confusion with pre-game instructions. Before each microgame, the player is given an instruction consisting of one or two words. "Catch!", "Pinch!", "Pick!", and "Blast Off!" are just a few examples. These are not

always what they seem. Let's look at three of the most common instructions: "Jump!", "Dodge!", and "Eat!".

Jump!

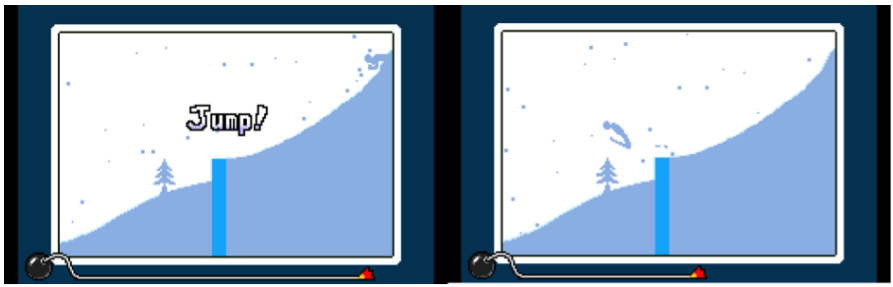

Figure 1. Ski Jump Microgame – Instructions and Action

In the ski-jumping microgame (see Figure 1) there is only one way to succeed, which is a common situation in the *WarioWare* microgames. To win this game, the player must time their press of the A button so that it occurs just as the ski jumper passes over the darker-colored area of the mountain. In this case, it helps to have a previous understanding of how ski-jumping works (i.e. that there is a predetermined launch point for the ski-jumper), but if you fail to make the jump, the game gives you the visual feedback of the jumper tumbling off the launch point. What is not clear from the instructions is that the player cannot force the ski-jumper to jump before the launch point but is not penalized for pressing the button ahead of time. Button-mashing (pressing the A button as fast as possible) is thus a viable strategy for this microgame. Still, the instructions for this game are relatively clear, with the action, "Jump!", mapped directly to the action button (A) on the Game Boy Advance, and also directly to the action in the microgame's fiction (the ski-jumper jumping). In Galloway's terms, the non-diegetic operator action of pressing the A button directly triggers the diegetic player action of a jumping ski-jumper, as long as it is properly timed.

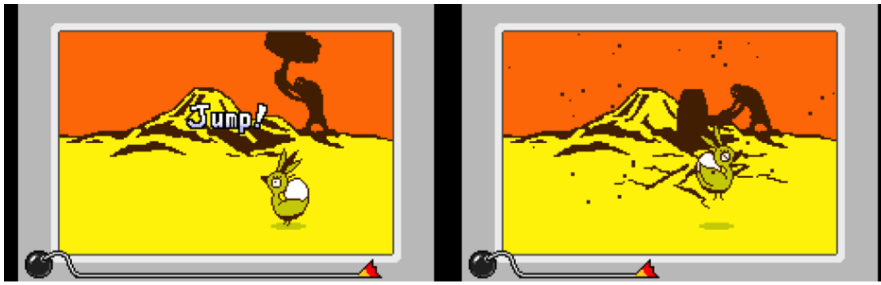

Figure 2. Quail Jump Microgame – Instructions and Action

In the microgame (see Figure 2) the quail must protect the egg it holds on its back by jumping at just the right time to avoid the earthquake caused by the hammering giant in the background. Like the ski-jumping microgame, success here also requires properly timing your jump, and only one jump is required. Unlike the ski-jumping microgame, the player can jump at any time, as many times as the time allows. When jumping, the quail flutters in the air for a couple seconds before returning to the ground. As a result, timing is more important here than it is in the ski-jumping game. Another notable difference is that the goal for the player within this narrative is actually to protect the egg. Jumping is secondary, the means to the end rather than the end itself. If the purpose of the instructions was to tell the player how to succeed, they might be given as "Protect (by jumping) rather than as simply "Jump!" In Galloway's terms, the non-diegetic operator action of pressing the A button directly triggers the diegetic player action of a jumping quail without regard to timing, as long as the quail is on the ground.

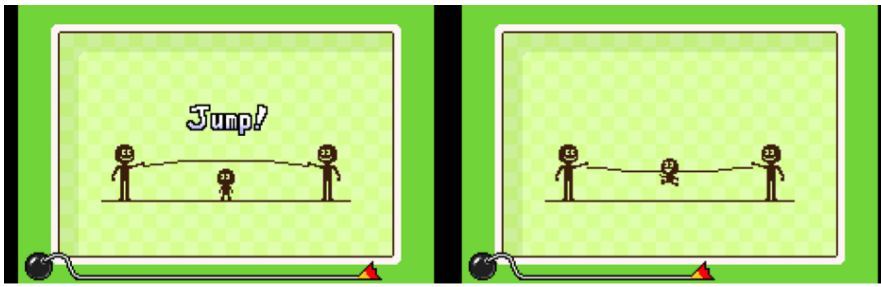

Figure 3. Rope Jump Microgame – Instructions and Action

In the microgame (see Figure 3) the player must again time their jump properly, but unlike in the previous jumping games they must repeat the successful jump until time expires. If they mistime any of their jumps and hit the rope, they lose the game. The previous two "Jump!" games create an expectation that only one jump is necessary to succeed, while here the expectation is changed, and the instruction "Jump!" doesn't help the player understand the new expectation. As in the ski-jumping game, the fiction here is consistent with the instructions – as part of the activity of jumping rope, the fictional character's goal is indeed to jump successfully.

In these three examples of microgames with the "Jump!" instruction, there are differences in game fictions but similarities in how the player interacts with them. The A button controls a jump action, which is always available in one of the microgames (Figure 3) and available with conditions in the other two (Figures 1 and 2). The diegetic operator action of jumping is consistent across all three. In other words, the A button always triggers a jump. Dodging is not so straightforward.

Dodge!

Figure 4. Dodge Microgame – Instructions

Like "Jump!", "Dodge!" is a brief, understandable instruction. We know there will be something coming our way that we need to avoid. Unlike "Jump!", "Dodge" does not easily map onto a specific diegetic operator action (for Wario, in the case of the microgame shown in

Figure 4). There is still only one action button, so perhaps that could map to a dodge action. Because *WarioWare* also uses the directional pad of the Game Boy Advance, Wario could also dodge by moving to a safe location. Although "Dodge!" is a fairly straightforward action (to get out of the way) in the fiction of this microgame, as an operator/player it's not clear what the specific diegetic player action will be. In other words, what the heck does "dodge" mean in the context of this game, and what exactly will Wario do when I press the A button or directional pad?

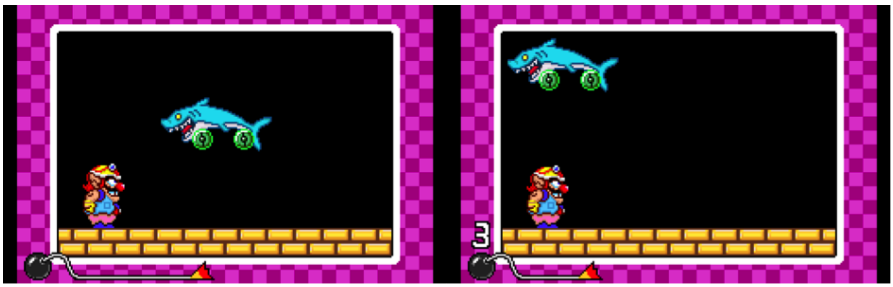

Figure 5. Dodge Microgame – Action

The design of the microgame deliberately exploits this confusion to create its challenge. In the microgame (see Figures 4 and 5) there four possible scenarios, one of which will be picked randomly by the game:

1. The car will come driving at Wario with constant speed. Jumping over the car wins the game.
2. The car will drive, and then jump (see Figure 5). To win, Wario must stand still.
3. The car will drive, then stop briefly, then continue on at the same speed. If Wario jumps as if the car was driving at a constant speed, he will land on the car and lose the game.
4. The car will drive, stop briefly, then turn around. There does not seem to be a possibility to lose this version of the microgame.

The directional pad does nothing in this microgame, so your options are to press the A button, or not. One of those options will be a successful strategy, depending on which scenario is selected by the game. The instruction, “Dodge!”, implies action, but in at least one of the winning scenarios, inaction is required. Instead of always having the meaning, “get out of the way”, “Dodge!” here can also mean, “don’t put yourself in the way.” This is an example of instructional dissonance that, more than the “Jump!” examples, creates a challenge by withholding information about the microgame’s fiction and about its operator actions.

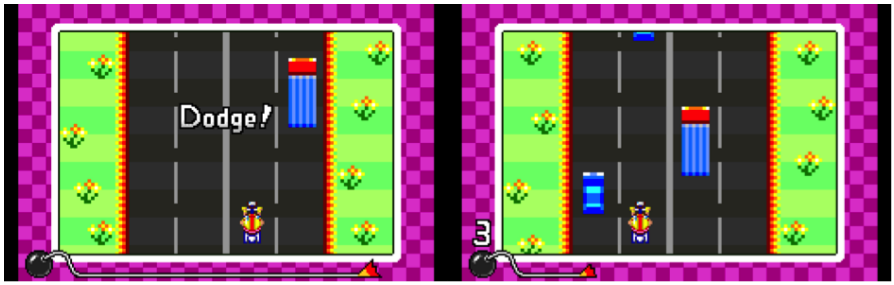

Figure 6. Dodge Microgame – Instructions and Action

A more straightforward example of the “Dodge!” instruction can be seen above (see Figure 6). This microgame puts the player in control of a car barreling down the road at constant speed. Other cars appear, implying strongly that the player should dodge those other cars given that the only buttons that respond are the left and right directional controls. But like the previous example, there are times that the player can win by doing nothing. Unlike the previous example, this seems to have more to do with luck than with deliberately confusing the player. Assuming that the oncoming traffic is generated randomly, the player sometimes gets lucky and doesn’t have any cars coming at them in their lane – all the cars end up to their right and left. Even in this more straightforward example, “Dodge!” sometimes suggests player inaction as a way to win the microgame (“don’t put yourself in the way”). There is a similar microgame that requires action to win, but interestingly does not use “Dodge!” as its instruction.

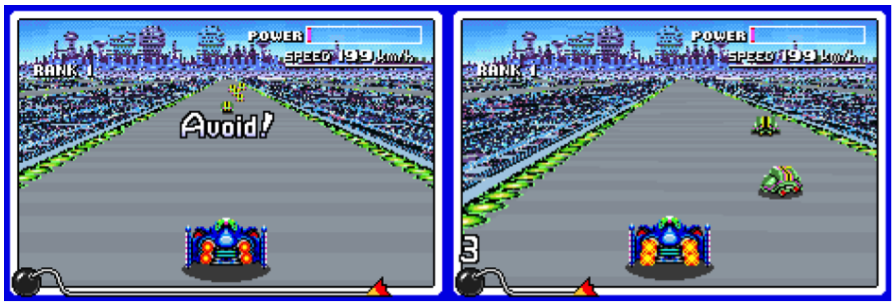

Figure 7. Avoid Microgame – Instructions and Action

"Avoid!" is the instruction given in the microgame pictured above (see Figure 7). This microgame portrays the game *F-Zero* (Nintendo Co., Ltd., 1990), which uses a behind-the-car 3D perspective. In this game, the cars appear larger as they get closer to the player and the screen. Not moving means certain death – randomness will not save you here. Instead, the player must use the left and right directional buttons to move out of the way of several cars. The actions and mapping are much more direct than in the previous "Dodge!" microgames. The nondiegetic operator action of pressing the directional buttons causes diegetic operator action of moving the car left and right, which *avoids* the oncoming cars.

Eat!

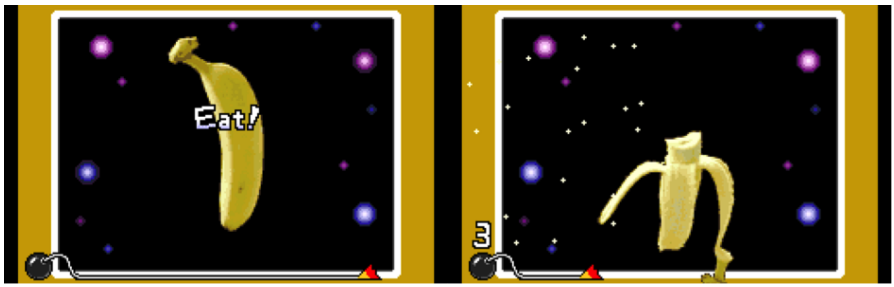

Figure 8. Banana-Eating Microgame – Instructions and Action

Like "Jump!" and "Dodge!" Eat!" is an instruction associated with a distinct action in the game's fiction. One important difference is that

while jumping and dodging suggest movement, eating suggests a variety of possible actions in the process of ingesting food (salivating, biting, chewing, swallowing, etc.) In the example above, the player must consume the banana before the time runs out by rapidly pressing the A button (see Figure 8). The player never sees exactly what is eating the banana; it's implied that something takes a bite each time the player presses the A button (*WarioWare* also features a nearly identical microgame with an apple instead of a banana). In this microgame, button-mashing is the only strategy. As in Wario's personal life, eating is a frequent subject in *WarioWare*, with a few variations on this basic button-mash-to-eat game. Sometimes the verb is "Chomp!" or "Munch!", but the mappings of nondiegetic to diegetic operator action, and those actions to the microgames' fictions, are fairly consistent. Pressing a button triggers a bite of the food item and the player must repeat this until the food item is gone.

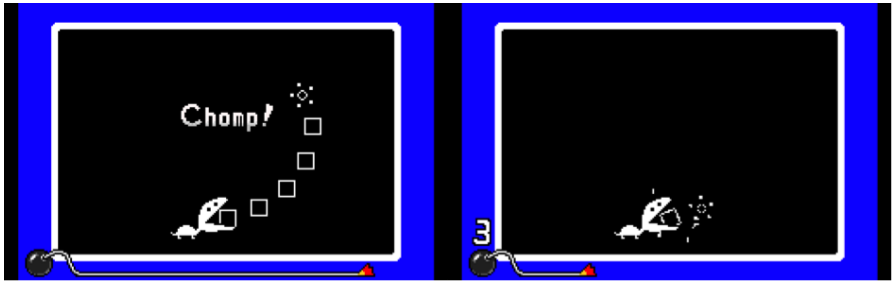

Figure 9. Chomp Microgame – Instructions and Action

The microgame pictured above (see Figure 9) instructs the player to "Chomp!", and one significant difference here is that "chomp" is a direct mapping of instruction to diegetic operator action, with a press of the A button (the nondiegetic operator action) triggering each chomp (If we wanted such a close mapping in the banana-eating microgame, "Bite" would have been an appropriate instruction). Another key difference in the microgame's fiction is that the player can see what is eating the food items, which along with the "chomp" instruction makes it this a clearer relationship. The A button (the nondiegetic operator action) causes the diegetic operator action of the little beast chomping on some food.

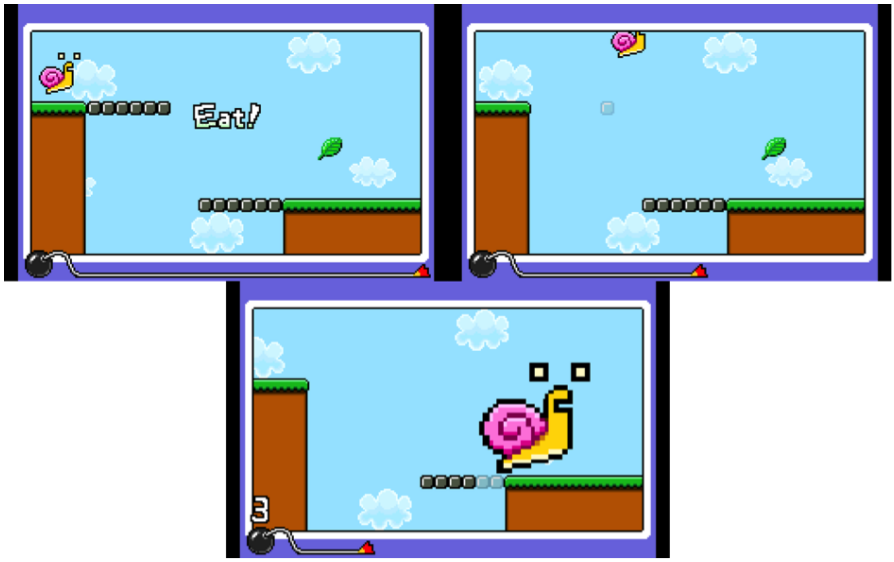

Figure 10. Leaf-Eating Microgame – Instructions and Action

The instructions for the microgame above (see Figure 10) is also "Eat!", but the game could not be more different than the button-mashing eating games described earlier. "Eat" here means to bring the snail to the leaf so that it can have a meal. To align this instruction with the game's fiction, you might instead tell the player to "Reach the Leaf". If the player had just been playing the eating games above, this one might be initially confusing, but these instructions also don't occur in a vacuum. This microgame has visual information that suggests what the player can control (the snail), what is food for the snail (the leaf), but not how to get there. That's where knowledge of other videogames becomes helpful – the platforms look like something out of *Super Mario World*, and so it helps to have that in your visual vocabulary. "Broad knowledge of videogames makes you a better *WarioWare* player." (Gingold, 2005)

As these examples show, there is a wide range in the level of connectivity between the fiction of the microgames, the available nondiegetic operator actions, and the diegetic operator actions. The level of connection, along with the player's knowledge of similar

videogame tropes, determines the difficulty of each microgame for first-time players. The *WarioWare* designers knew this, and it's why we see such experimentation in methods to confuse players. This sowing of confusion through instruction is consistent, even in the printed instruction manual.

The Instruction Manual

> *Dear You,*
>
> *Was Diamond City a blast? Do you love Wario's games? I love my mountain of cash! Tell your friends to buy, buy, buy my games!*
>
> *Wario (Nintendo Co., Ltd., 2003)*

The instruction manual for *WarioWare* is 48 pages long, including front and back covers and sticker sheet insert. Here's a loose breakdown of the type of content found through the manual:

- 6 Pages: Front and back cover, warranty, service, and rating information
- 4 Pages: Removable sticker sheet (see figures 12 and 13)
- 6 Pages: Quick tips – instructions for starting and progressing through the game and menus (see Figure 11)
- 8 Pages: Diversions, silly puzzles, notes pages, and other places to put stickers (See Figures 14 and 15)
- 21 Pages: Introduction to game narrative and character bios

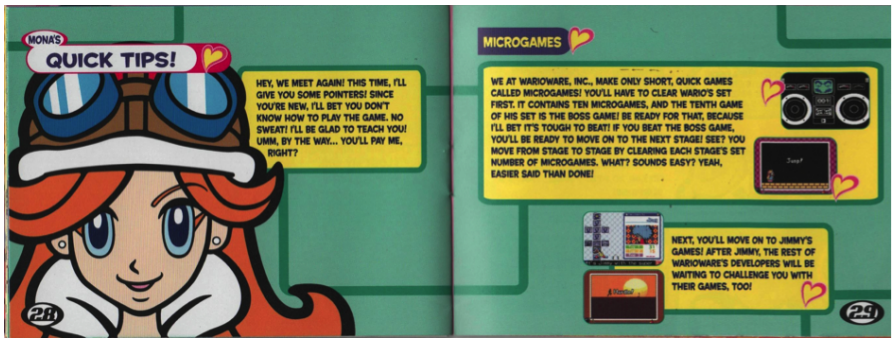

Figure 11. One of the more instructional spreads in the manual

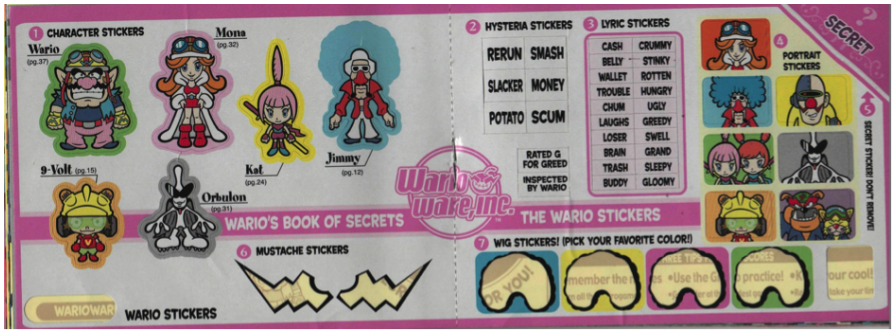

Figure 12. Sticker Sheet with Most Stickers In Place

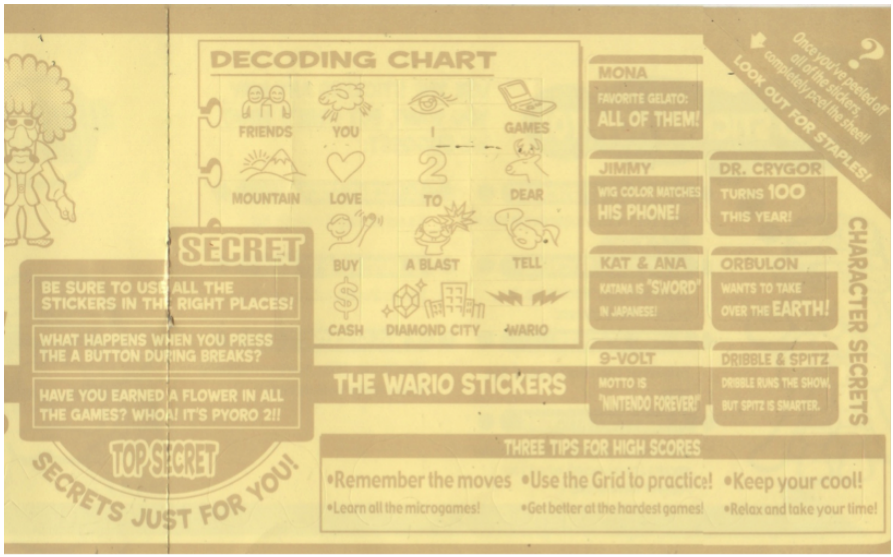

Figure 13. Sticker Sheet with All Stickers Removed

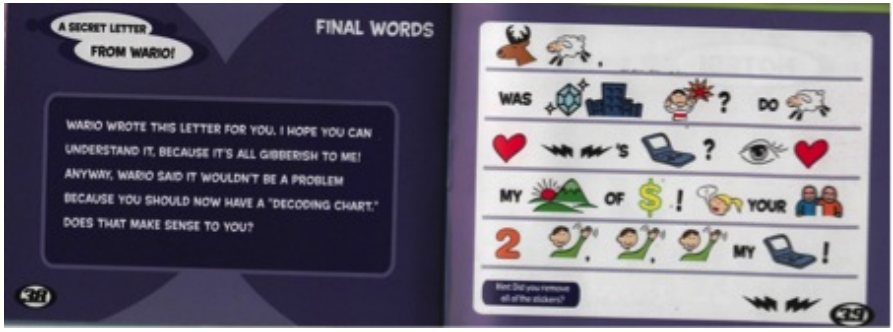

Figure 14. Wario's Song with Stickers to Fill In Words

Figure 15. Wario's Puzzle Page

In *The Role of Instructions in Video Games and Its Impact on Video-Game Translation*, Tomasz Stajszczak identifies three relevant qualities for the instructive features of videogames (2013):

1. Efficiency: "provide as much relevant information as possible within a convenient amount of time."
2. Accuracy: "factual correctness and clarity of instruction."
3. Immersion

While the first two are intentionally thwarted throughout *WarioWare*, immersion provides an interesting lens through which to view the instruction manual of the game.

> *Instructive features may contribute towards the immersion effect by virtually countless means – from making the included text's style fit the game's convention to in-game characters who play the roles of guides to making the instructive stages become a part of the storyline. (Stajszczak, 2013)*

The instructions (see Figure 11) are the closest to actual game instructions that you'll find in the manual, but there's not much of use for progressing through *WarioWare*'s specific microgames. Most of the pages lay out the narrative of the game – Wario and friends design a series of games to generate sales and a boatload of cash.

The puzzles are mostly diversions, but the sticker page appears to be something more significant, with its labels of "Secret" in the top right, which, when peeled away, encourages the reader to remove the entire sticker sheet to reveal what is underneath (see Figure 13). As you can see, what is there is not very useful to complete the microgames, but it does give you information to decode the message (see Figure 14). When decoded, the message revealed is the one quoted at the beginning of this section.

Much like the secret decoder ring in *A Christmas Story* (1983) that infuriates the main character Ralphie with its lame command to "Be Sure to Drink Your Ovaltine", the *WarioWare* manual is successful precisely because it serves as a diversion and reveals no substantial information towards completing the microgame challenges. This misdirection is an intentional and consistent strategy to actually instruct as little as possible and instead to immerse the player in Wario's bizzare narrative and schemes. It reminds the player that this is Wario's world, lest you think for a second that he would actually give away anything resembling a secret about his game. This is consistent with earlier analysis that *WarioWare*'s central initial challenge is to wade through this confusion to establish comfort in each microgame.

Conclusion

While playing *WarioWare*, if you imagine that the instructions are being spoken in Wario's voice, it begins to make sense. He tells you the action, but not how to do it. He tells you the desired outcome without the directions to achieve it. He tells you to do something, but not that it requires you to first do something else. Because the instruction manual is also written from his point of view, it also makes sense that he would be the one barking the instructions to you in the microgames. He is invested in creating an instructional dissonance that makes the games and microgames frustrating, challenging, and fun.

In addition to continuing the tradition of one-button microgames and

the narrative of Wario and friends as game developers, the sequels have frequently served as showcases for new hardware from Nintendo. *WarioWare Twisted!* includes a gyroscopic sensor in the game cartridge (the player is required to rotate their Game Boy Advance system to control aspects of the microgames). *WarioWare Touched!,* for Nintendo's dual-screened, touch-based DS portable system, highlighted both of those aspects of the hardware in its microgames. *WarioWare Smooth Moves* used the gestural Wiimote of the Wii home console to define a set of postures that the player needs to take to complete its minigames. *WarioWare Snapped!* showcases the DSi's front-facing camera. (MobyGames, 2014) Finally, *Game & Wario* showcases the Wii U's gamepad with a collection of minigames (rather than microgames). (Intelligent Systems Co., 2013) Though the diegetic action/formal action of the player or whatever evolves with each new platform in the Wario games, what is consistent throughout the sequels is the instructional dissonance established by the first game.

References

Galloway, A. R. (2006). Gaming: Essays On Algorithmic Culture (Electronic Mediations) (0 ed.). University of Minnesota Press.

Gingold, C. (2005). What WarioWare can teach us about Game Design. GameStudies, 5(1). Retrieved from http://www.gamestudies.org/0501/gingold/

MobyGames. (2014). WarioWare Series. Retrieved from http://www.mobygames.com/game-group/warioware-series

Stajszczak, T. (2013). The Role of Instructions in Video Games and Its Impact on Video-Game Translation. Proceedings from International Conference on Translation and Accessibility in Video Games and Virtual Worlds, Barcelona, Spain.