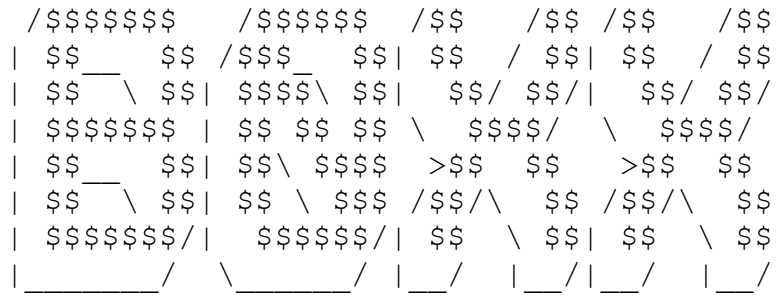


Designed by Aziz “Hax$” Al-Yami

Produced & Engineered by Kyle “simple” McDowell

Written by Aziz “Hax$” Al-Yami

Last Updated: 05/01/2018

****

Table of Contents

(use CTRL + F)

****

****

[1] Introduction

****

Within the competitive Super Smash Bros. Melee scene, the ergonomic shortcomings of the Nintendo Gamecube controller have taken their toll on many players. Four years ago, I infamously joined this club. On May 4th, 2014 I incurred an injury while playing Melee that led to a series of surgeries on my left wrist. While this was ultimately resolved, it shed light on a condition that had even greater implications. My first MRI results revealed that my left thumb was showing signs of arthritis at the age of 19. This inevitably led me to learn that the Gamecube controller was no longer an option.

In December 2016, an opportunity arose. Hit Box, a company known for their moderately popular stickless controllers, had been looking to dip their toes in the world of Melee with their latest product: the Smash Box. Advertised as precise and ergonomic, the Smash Box was widely regarded as revolutionary. Confident that I could provide valuable insight, I decided to reach out.

Fatefully, Hit Box shipped me a prototype Smash Box for playtesting purposes. The moment it arrived in the mail, I started evaluating its performance. But just as quickly as I began, I realized that finding flaws in the current model was the least of my concerns. The real question was *how* one would even go about rationalizing the Smash Box's design. While it wasn't hard for me to point out improvements that could be made to the Smash Box's button layout, nearly everything software-related was beyond my comprehension at the time. Like most Melee players, I had been competing for over 10 years without ever giving the game engine much thought. Suddenly, I was staring at a bunch of coordinates I couldn't recognize, and an instruction manual I could barely understand. Hit Box had opened the gateway to the future, and there was no going back.

As I slowly familiarized myself with the Smash Box, it became clear just what I had gotten myself into. Hit Box's invention had seemingly endless potential, and while I wasn't sure where to start, I knew that I was determined to perfect it. As the weeks

went by, these aspirations of mine struggled to align with a company that had deadlines to meet. In order to rebuild the Smash Box from the ground up, I needed complete creative control, which meant cutting ties with Hit Box on January 6th, 2017. While this decision wasn't easy, I knew that it would be justified by the contribution I'd eventually make to the game. In the most interesting journey back to Melee I could've asked for, I sought out to give Hit Box's concept the iteration it deserved. I called this the B0XX.

The B0XX would end up taking me over a year to complete. As I continued to make breakthroughs in my research and gain skill with the controller, my approach constantly shifted along the way. Eventually, I settled on the goal of solving the hand health epidemic without isolating the Gamecube controller playerbase. This seems to resonate with most people in need of the B0XX. Among us, there is little-to-no interest in competing on an unfair playing field; thus, the model I'll be presenting is intended to mimic the performance of a Gamecube controller as closely as possible. And while a perfect replica may not exist, I believe I've achieved something evenly-matched.

For those who are curious, the advantages of a digital controller generally stem from two things. The first is travel time (not the same as actuation time, which is discussed in Section 11.1.2). Since physical recoil doesn't exist with directional keys, certain motions (i.e. pivot U-tilt) are made reasonable by these controllers. The second is precision. Since any coordinate can be pinpointed with certainty, there is effectively no difference between performing an Ice Climbers desync (a single X or Y-value on the entire grid) and a tap jump.

Wherever possible, I targeted and removed what stood out as unrealistic to reliably perform on the Gamecube controller. Despite these efforts, it must be understood that the B0XX is not obsolete to the Gamecube controller, nor should that be what is expected of it. There are some inherent advantages that cannot be taken away. In Chapter 10, I will make it clear what these are.

Similarly, the B0XX has several inherent disadvantages. These also tend to fall under two categories. The first is a lack of

intuition. Several techniques, such as angling Firefox, undoubtedly have a steeper learning curve with this controller. The most meaningful disadvantages, though, have to do with the B0XX being objectively incapable of performing certain techniques. Since the majority of the coordinates on the plane aren't present on this controller (in addition to the rules I've enforced further reducing what it's capable of), its potential is capped in several areas.

That being said, it would be foolish to think that taking the time to learn a more complex controller with limited options ought to be rewarded with the advantages the B0XX grants if it isn't restricted. Perhaps the most obvious example that regulation is needed in some capacity is the ability to pinpoint the 16.8° (shallowest) wavedash, a feature I consider game-breaking. Some of these advantages aren't so crystal clear until someone has a significant amount of mastery under their belt, which is why it took me so long to finalize my rationale.

Throughout my decisions, I attempted to design the B0XX as objectively as possible. Not a single coordinate on the controller is arbitrary, though that isn't to say that the B0XX is perfect. The potential to revise this controller, whether for the sake of fluidity or game balance, is absolutely still there. Nonetheless, I am confident I'll be providing a starting point that is more than sufficient.

Lastly, we must be aware of the social good that comes alongside the solution to our biggest controller-related problem remaining. In 2017, we were able to find ways to fix inconsistent performance among Gamecube controllers (which, ironically, likely stemmed from the digital controller revolution), but health concerns still remain. Third-party controllers may be a foreign concept to the Melee community, but they are something we have overlooked for too long. In most games and sports, several different models of equipment are permitted, and understandably so; personalization is critical to the enjoyment of any activity. The Melee community, however, is faced with something much more than just that. For the longevity of our players, it is important that we work together to legalize some iteration of the B0XX.

****

[2] Universal Rules

****

Parameters that any tournament legal controller should abide by.

[2.1] Hardware

[2.1.1] Quantity of Buttons

All controllers should consist of a 1:1 remapping* of the Gamecube controller. For clarification, the actuators I am referring to are the A, B, digital L, digital R, X, Y and Z buttons, the analog stick/C-stick, and the D-pad. A controller with a digital analog stick and/or C-stick should contain one of each cardinal direction per stick. The C-stick may also transform into the D-pad through the use of a toggle (so long as this cannot conflict with the actuation of C-stick inputs).

A controller may also contain buttons that guarantee certain analog L/R-values. This is a logistical requirement for controllers that do not have springs, as they would otherwise be unable to generate these. While analog L/R buttons inherently come with a precision advantage, they are inferior to the full analog range of a spring.

For logistical reasons, the Gamecube controller must resort to a spring for analog L/R-values. It should also be noted that the Gamecube controller can guarantee certain analog L/R-values through the use of either a stopper or calibration exploits (see Section 11.2.1).

***Modifier buttons** are only permitted on controllers with at least one digital stick, and are only allowed to influence digital directional inputs. Modifier buttons may also influence L/R's analog value. There is no harm in having an unlimited amount of modifier buttons (so long as they abide by the rules listed in Section 4.2), although it is best to keep this number to a minimum (the B0XX has two).

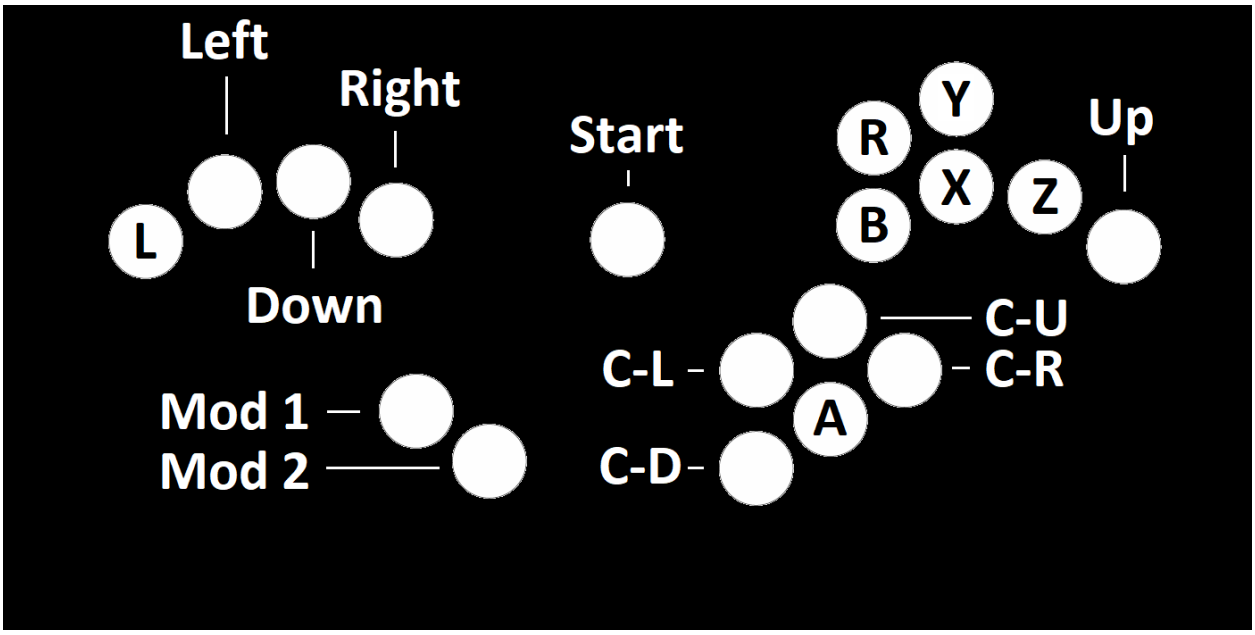
[2.1.2] Button Locations

Barring a few exceptions (see Section 2.2.2), **controller manufacturers should be allowed to arrange their actuators however they feel is logical.** In general, it would be backwards for us to demand anything short of a well-designed controller.

For the most part, the Gamecube controller's layout is fine for performing in-game techniques frame perfectly. The main exception is the C-stick, which is in a much less ideal location. As a result, the best way to grip the Gamecube controller is rather unorthodox.

"Javi claw," a style of grip that has flown under the radar.

Employing some variation of right hand claw is the only way to access the Gamecube controller's C-stick at all times. While this grip isn't without its downsides, there are ways to mitigate nearly all of them. Despite this, right hand claw remains unpopular, likely due to its awkward nature. That is to say that most players opt not to wield the Gamecube controller optimally due to the discomfort that comes with doing so.



Unlike the Gamecube controller, any modern controller will have its C-stick divided into four easily accessible buttons.

Sometimes, upholding game purity means compromising quality of life to a degree that isn't worth it. In the same vein that the Melee community has come to accept that certain aspects of the game (dash back/shield drop) unquestionably warrant redesign, the C-stick is an artifact that is unfavorable for competitive play. I believe that we should not be opposed to, but *supportive* of a controller that rectifies this.

As to offer Gamecube controller users the same luxury, **the B0XX is compatible with the Wii Nunchuk (once plugged in, it replaces the analog stick and L button).** This will give everyone the option to play with the B0XX's superior right hand layout. The Nunchuk utilizes the same stickbox as certain runs of the Gamecube controller, making for an easy transition.

As far as the rest of the B0XX's layout goes, the elephant in the room is surely the Up button's location. This was chosen by process of elimination. It is a must that the left pinky is assigned shield (L) and the left thumb operates the modifier buttons. If the controller is then given a WASD arrangement, Up forces the user to shift their left wrist just to access it. This makes the right pinky the most efficient location for Up.

Finally, I should clarify L/R/Z's recommended roles. As I mentioned, L is intended to be used for shield, though it should also be used to tech. L-cancels should be performed with Z, and downwards airdodges (wavedashes) should be performed with R (you'll have to shift your right wrist for these, since R and Y aren't on the home row). Upwards airdodges are a special case that will be covered in Section 8.3.4.

[2.2] Software

[2.2.1] Override

Within the context of Melee, **opposite cardinal directions must be allowed to override each other (Left -> Right = Right. Pressing Left + Right on the same frame generates a neutral input).** This mimics an analog stick most closely, as it shouldn't be possible to receive an unwanted neutral input by failing to release the first cardinal before pressing the second (which would happen if opposite cardinals negated each other).

When a cardinal is overriden by the actuation of the opposite cardinal, it should not reactivate once the opposite cardinal is released. All actuations must be performed manually.

[2.2.2] Macros & Button Binds

Macros (inputs occuring on future frames. Can also pertain to different outcomes being generated if certain inputs take place within a specific # of frames of each other) are illegal.

Action-action button binds* (more than 1 of A/B/C/L/R/X/Y/Z bound to a single button) are illegal.

Action-direction button binds* (A/B/C/L/R/X/Y/Z bound to an impact on the analog X/Y-coordinates) must be evaluated on a case-by-case basis. Traditional action-direction button binds (i.e. Up bound to A to produce U-tilt) are illegal; however, the B0XX uses some of its action buttons in a unique way to pinpoint a few niche coordinates (see Section 4.2 and Chapter 8).

*While these rules ban button binds on a software level, it remains possible to achieve the same effect through hardware. Up and A, for example, could be arranged in a yin-yang formation in order to guarantee simultaneous actuation. Even though 1:1 remapping encompasses this exploit to a degree (since it is illogical to devote your only Up button to your only A button, etc.), it is still worth stating that **constructing buttons in a manner that guarantees simultaneous actuation is illegal**.

[2.2.3] Order-Dependency

The B0XX contains two instances of buttons generating different outcomes depending on whether or not another button is already being held (see Sections 7.3 and 8.3.4). These are needed to distinguish the player's intention in situations where compatibility with two or more techniques would otherwise conflict (i.e. the L button can be used to shield *or* airdodge).

Order-dependency is innocuous so long as:

- All inputs are in line with what is normally legal for an actuator to perform.

- **"After" inputs are inclusive of simultaneous presses.** For example, if Up and L are pressed on the same frame, the controller must read this as "Up *after* L" *and* "L *after* Up." This preserves a manual system of inputs, whereas recognizing the inputs as "simultaneous" would be a macro.

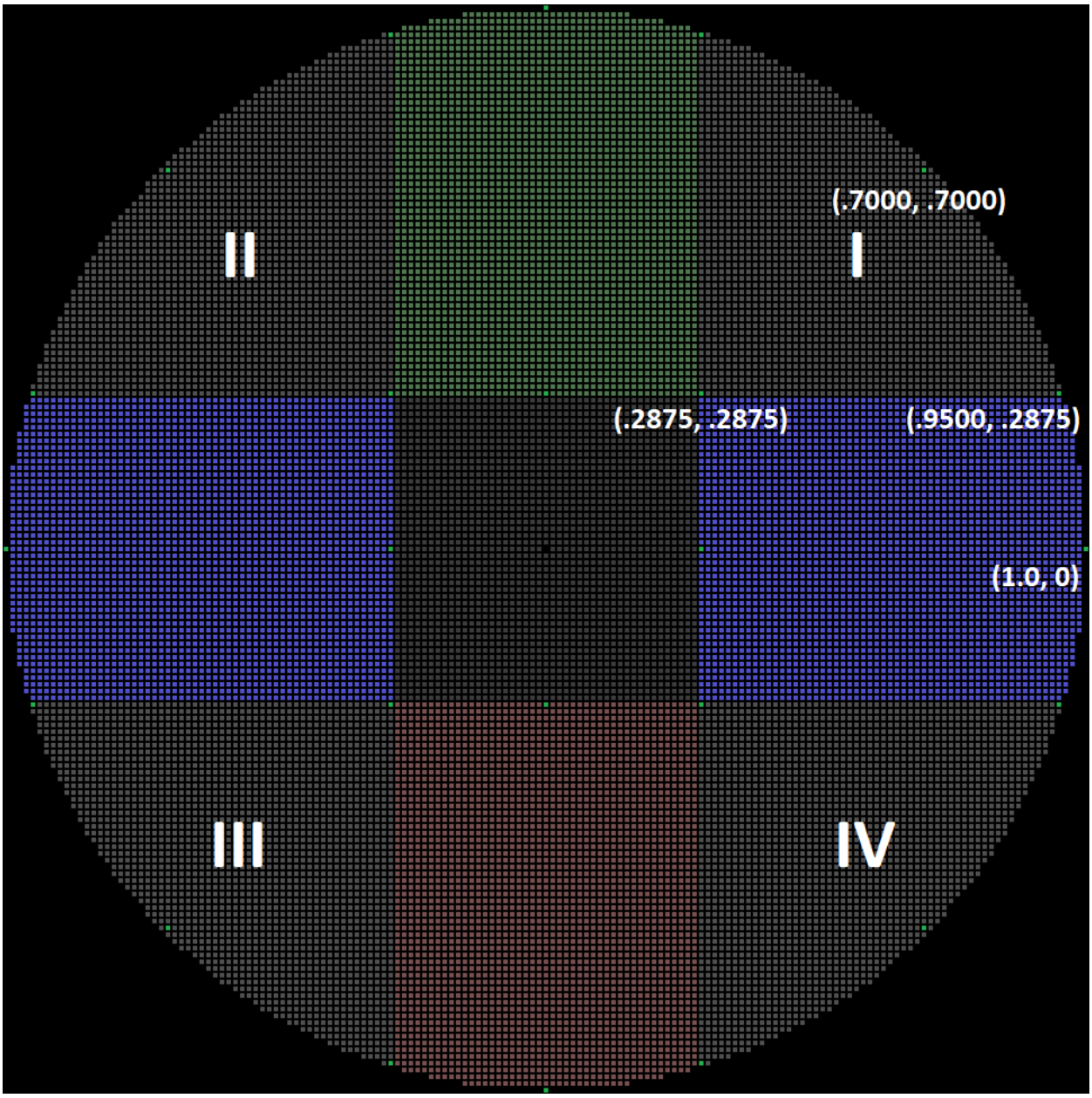
****

[3] Gamecube Controller Overview

****

A thorough understanding of the analog stick's coordinate plane will play a crucial role throughout this document. Excluding Section 3.2.3, all of these concepts apply to the C-stick as well.

[3.1] Basics



The above diagram contains every coordinate in the game, and clearly outlines the 9 sections of the grid (deadzone/4 cardinals/4 quadrants). The 4 quadrants begin in northeast, and ascend counter-clockwise.

The X and Y axes operate in increments of .0125. Among these coordinates, there are some notable ones:

-.2875 is the minimum value that activates an axis (i.e. X .2875 activates east/X .2875 Y .2875 activates northeast).

-1.0 is the highest magnitude X/Y-value, and can only be paired with 0 on the other axis. Due to its microscopic range, **X or Y +/-1.0 is notorious for being difficult to pinpoint on the Gamecube controller (see Chapter 12).**

-X +/-.7000 Y +/-.7000 (45°) are the intended diagonal corners.

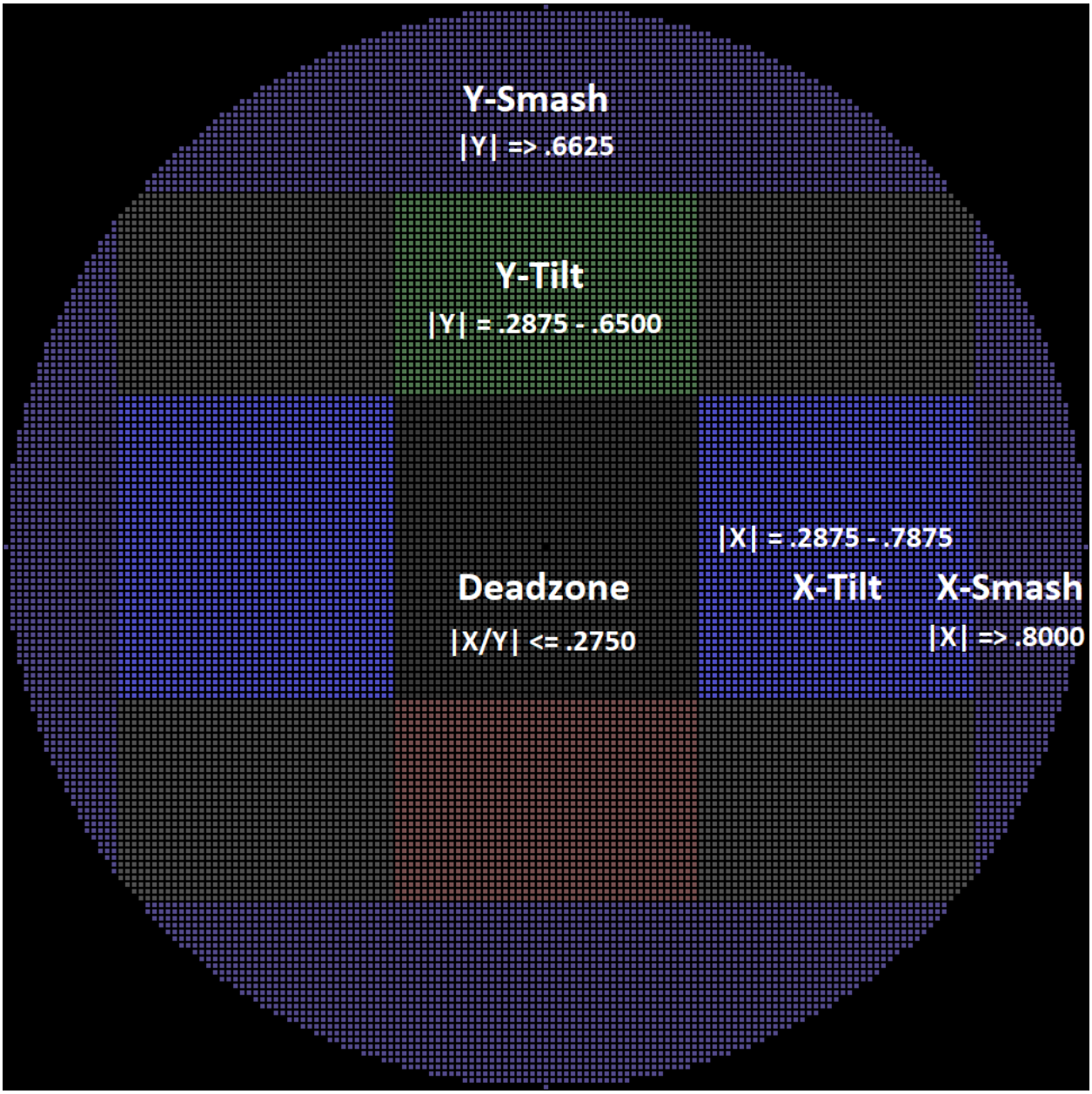
-X +/-.9500 Y +/-.2875 (16.8°) and X +/-.2875 Y +/-.9500 (73.2°) are the shallowest* and steepest** angles respectively.

**Shallow* implies that the vector hugs the X-axis.

***Steep* implies that the vector hugs the Y-axis.

[3.2] Zones

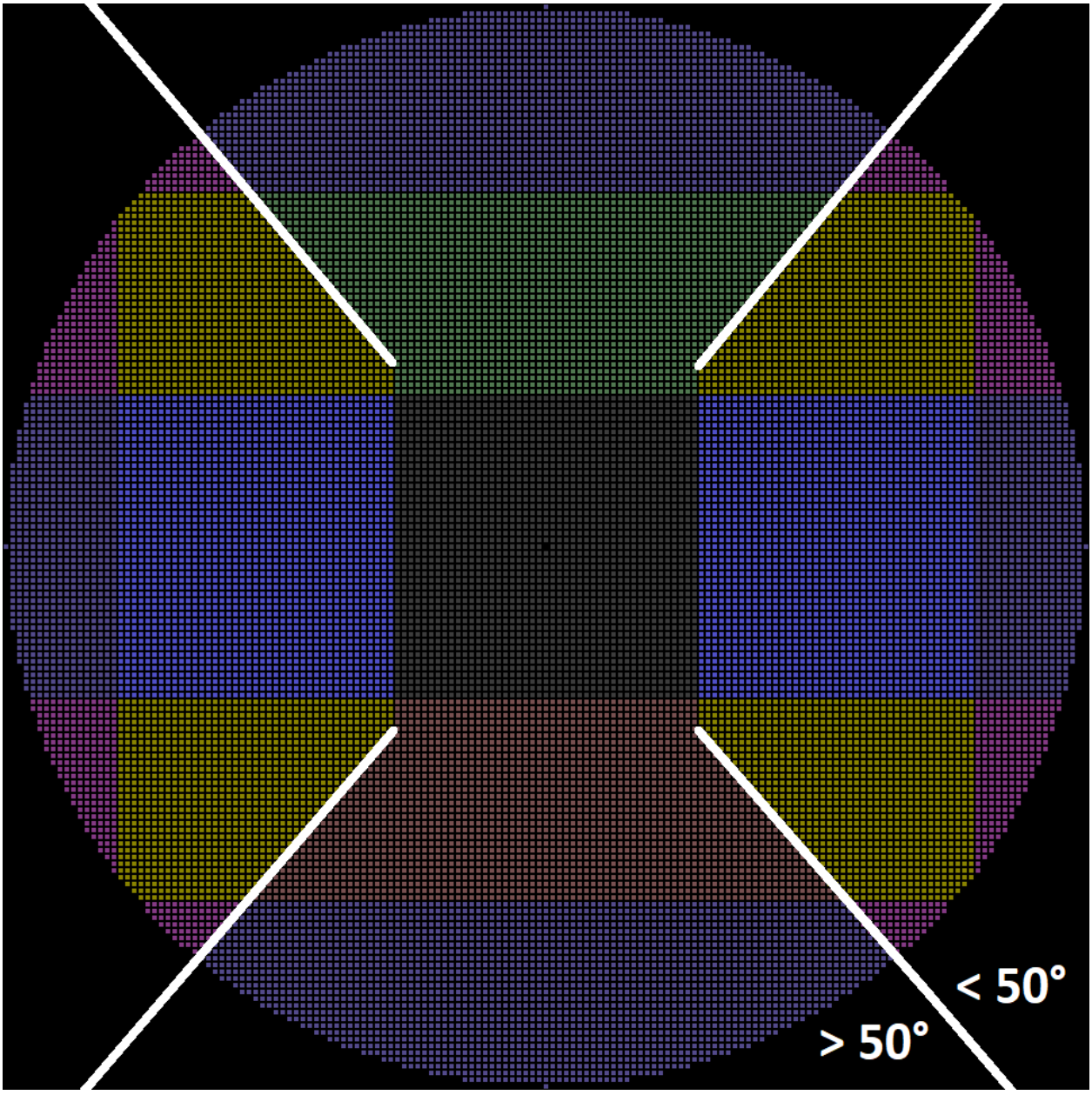
[3.2.1] X-Tilt/X-Smash & Y-Tilt/Y-Smash



On either axis, there are thresholds that serve similar purposes. The X-axis has had these documented extensively due to the infamous dash back dilemma, the Y-axis not so much. Along the X-axis, .2875 through .7875 is X-tilt, while .8000 through 1.0 is X-smash. Along the Y-axis, .2875 through .6500 is Y-tilt, while .6625 through 1.0 is Y-smash. X .8000 and Y .6625 aren't

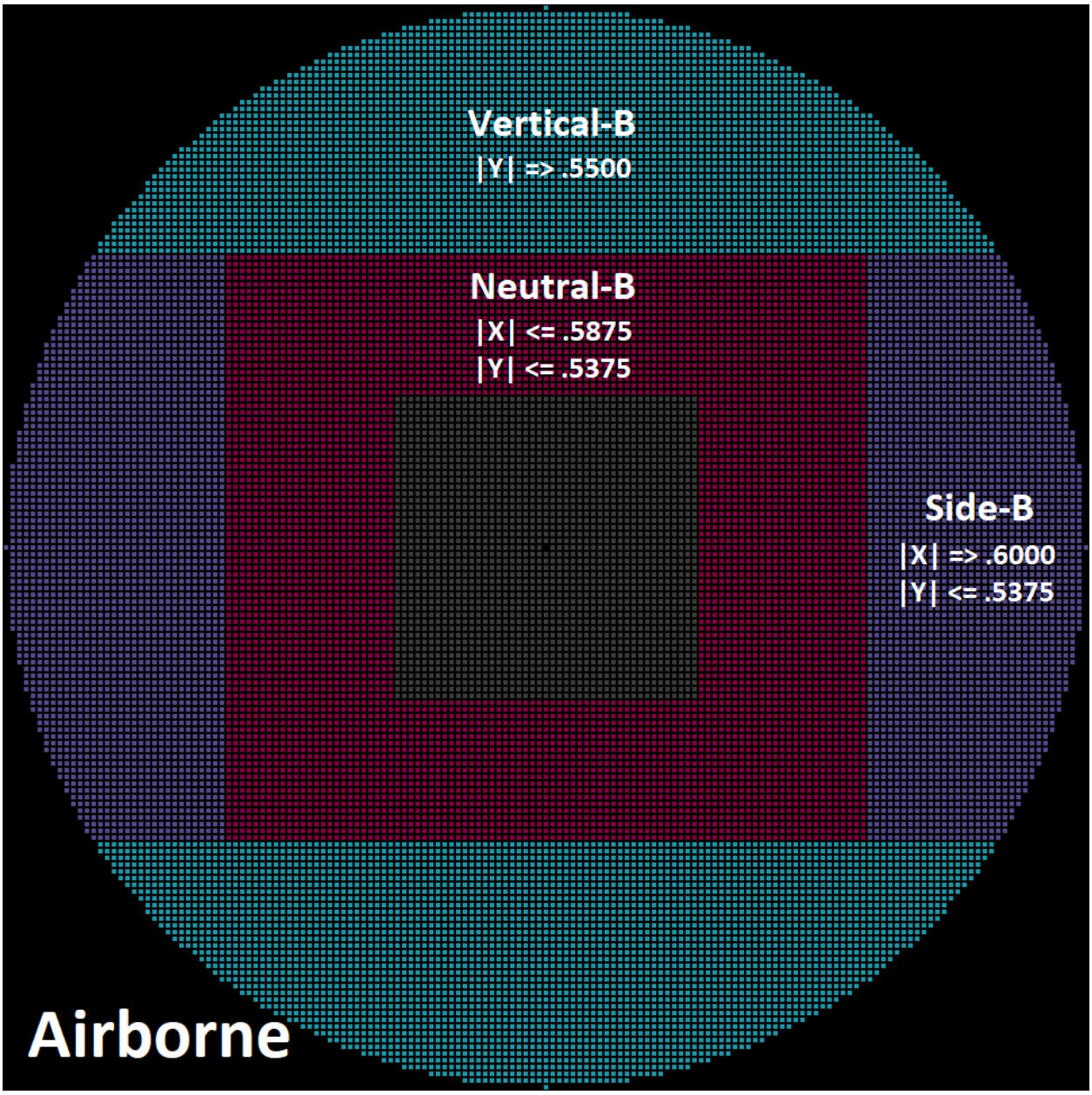
just the cutoffs for tilts/smashes, but several other things as well, such as dash/pivot on the X-axis, and tap jump/shield drop/ fastfall on the Y-axis.

[3.2.2] 50° Line

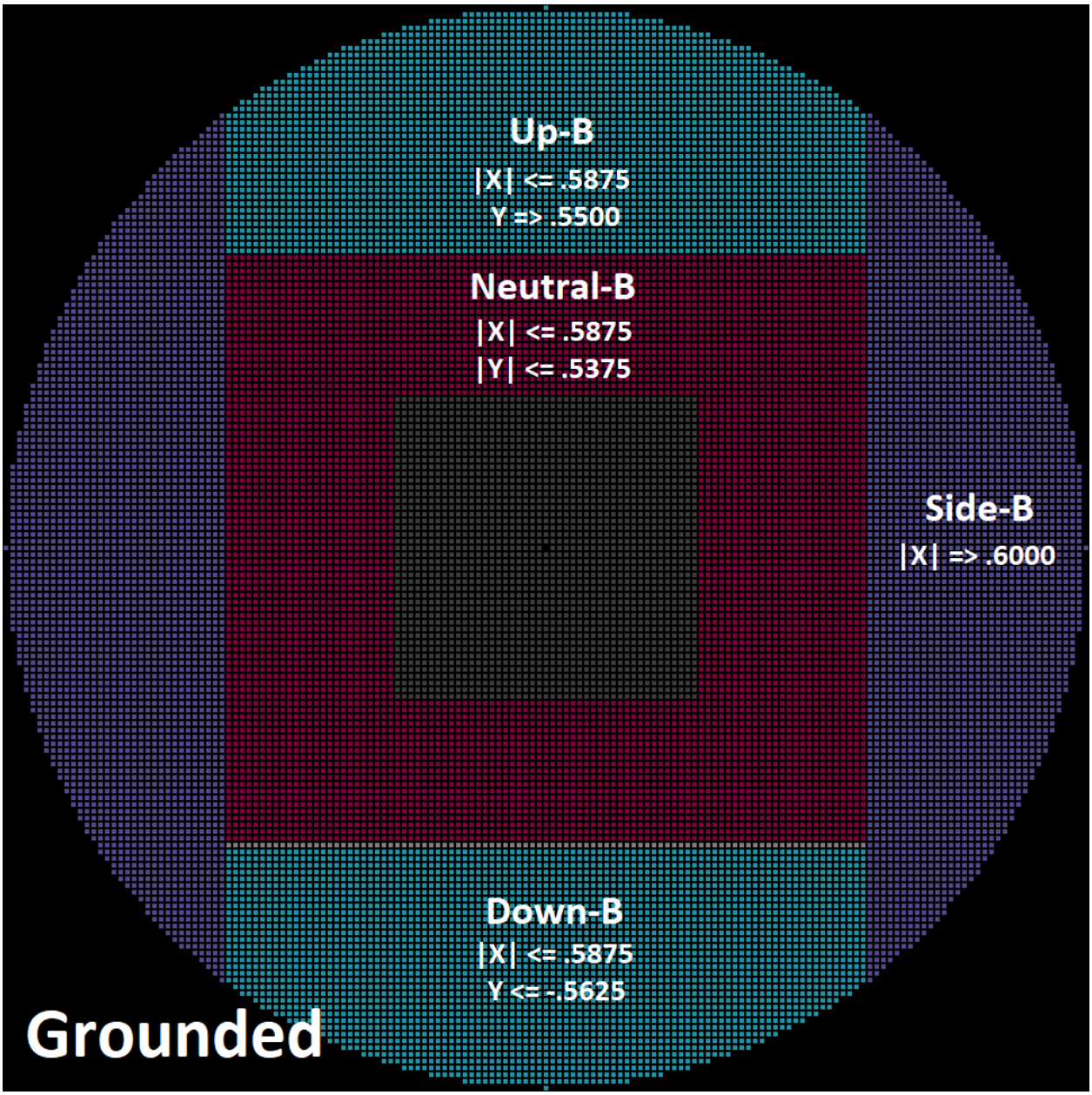


There is a 50° line that separates several techniques in the quadrants. Examples include angled F-tilts/vertical tilts, ledge get-up/ledgefall, and horizontal/vertical aerials upon pressing the A button.

[3.2.3] Special Moves



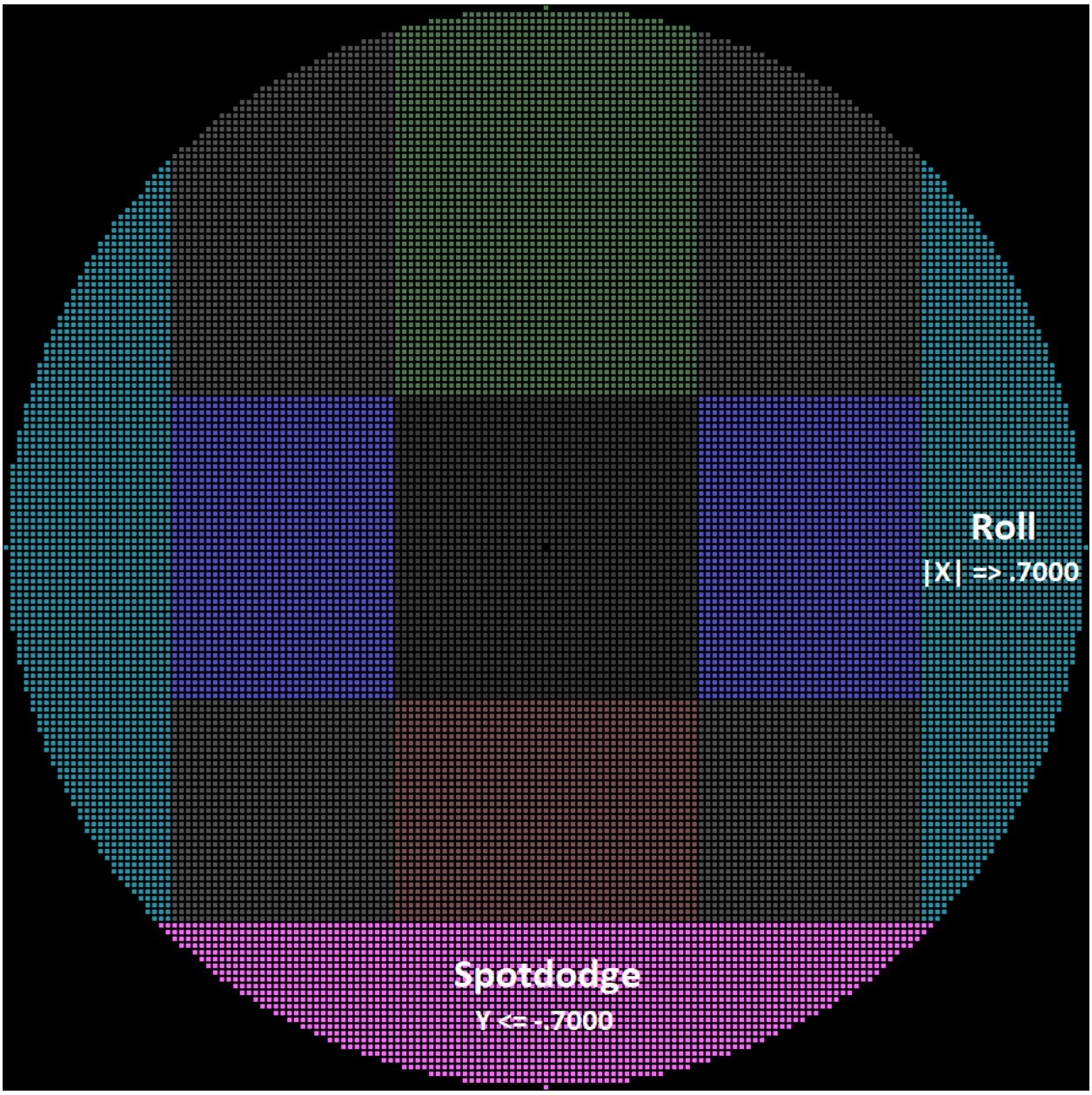
Neutral-B is unconditionally |X| <= .5875 with |Y| <= .5375. When your character is airborne, vertical-B is |Y| => .5500, and side-B is |X| => .6000 with |Y| <= .5375.



When your character is grounded, the zones for vertical-B and side-B are *usually* swapped as shown. This is true for most, but not all grounded states. Crouch, for example, uses the same zones as airborne.

|X| <= .5875 with Y -.5500 will not produce a B move when these zones are used. This was likely a programming oversight.

[3.2.4] Roll & Spotdodge



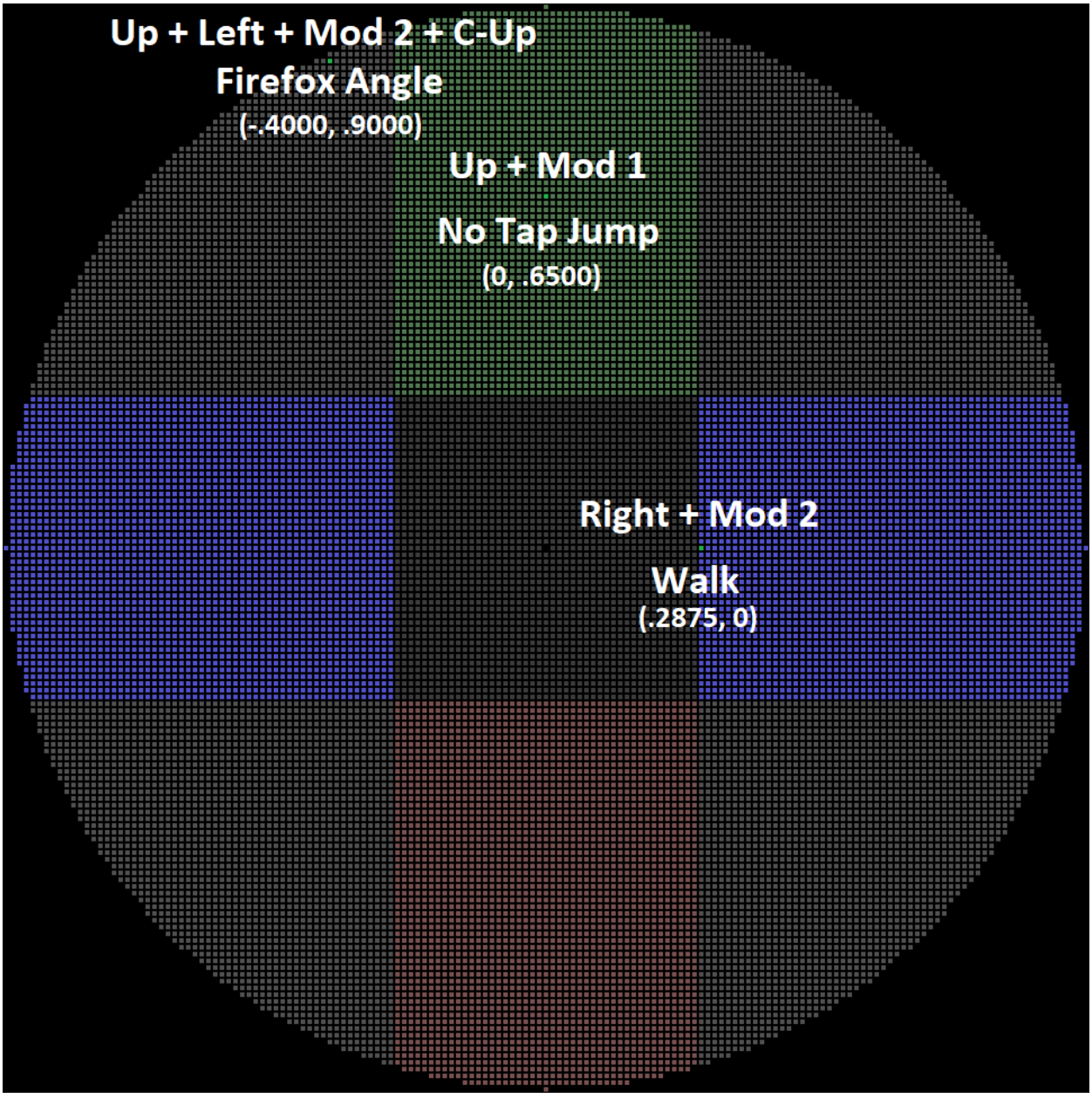
|X| => .7000 causes roll and Y <= -.7000 causes spotdodge. Y <= -.7000 also causes crouch.

****

[4] Digital Controller Overview

****

[4.1] Modifiers



Through the use of modifiers, digital inputs are able to adequately mimic an analog stick. Without them, the 4 arrow keys can only pinpoint the highest magnitude cardinals (X or Y +/-1.0) and 45° angles (X +/-.7000 Y +/-.7000). **Modifiers can be thought**

of as shift keys: when held alongside the arrow keys, they can be used to select any coordinate within the designated section of the grid.

The provided diagram shows examples of techniques that require the use of modifiers. Slight presses of the stick and angles that aren't 45° can't be performed with the 4 arrow keys alone.

[4.2] Restrictions

All modifiers on the B0XX abide by three overarching rules.

Modifiers cannot change the section of the grid you're in (deadzone/4 cardinals/4 quadrants). This job is reserved for the 4 arrow keys.

Modifiers cannot pinpoint banned coordinates (see Section 5.2). These are the areas of the game where digital inputs have too much of a precision advantage over an analog stick otherwise.

Action buttons (A/B/C/L/R/X/Y/Z) can possess modifier properties and influence the analog X/Y-coordinates. In doing so, they must abide by several rules that ensure they can't cause disingenuous behavior (see Chapter 8). For clarification, this feature is *not* meant to permit action inputs bound to directional inputs in the traditional way (i.e. U-tilt button). It is simply meant to accommodate niche coordinates that Modifiers 1 and 2 don't have room for. **For example, the Firefox angle pictured in Section 4.1 is comprised of Up + Left + Modifier 2 + C-Up.** While C-Up remains actuated throughout this interaction, its action input bears no significance: C-Up merely serves as the physical button needed to identify these coordinates.

The third rule (also known as **non-dedicated modifiers**) can be considered the hallmark of the B0XX. Without it, the controller's minimalistic button layout would be unattainable.

****

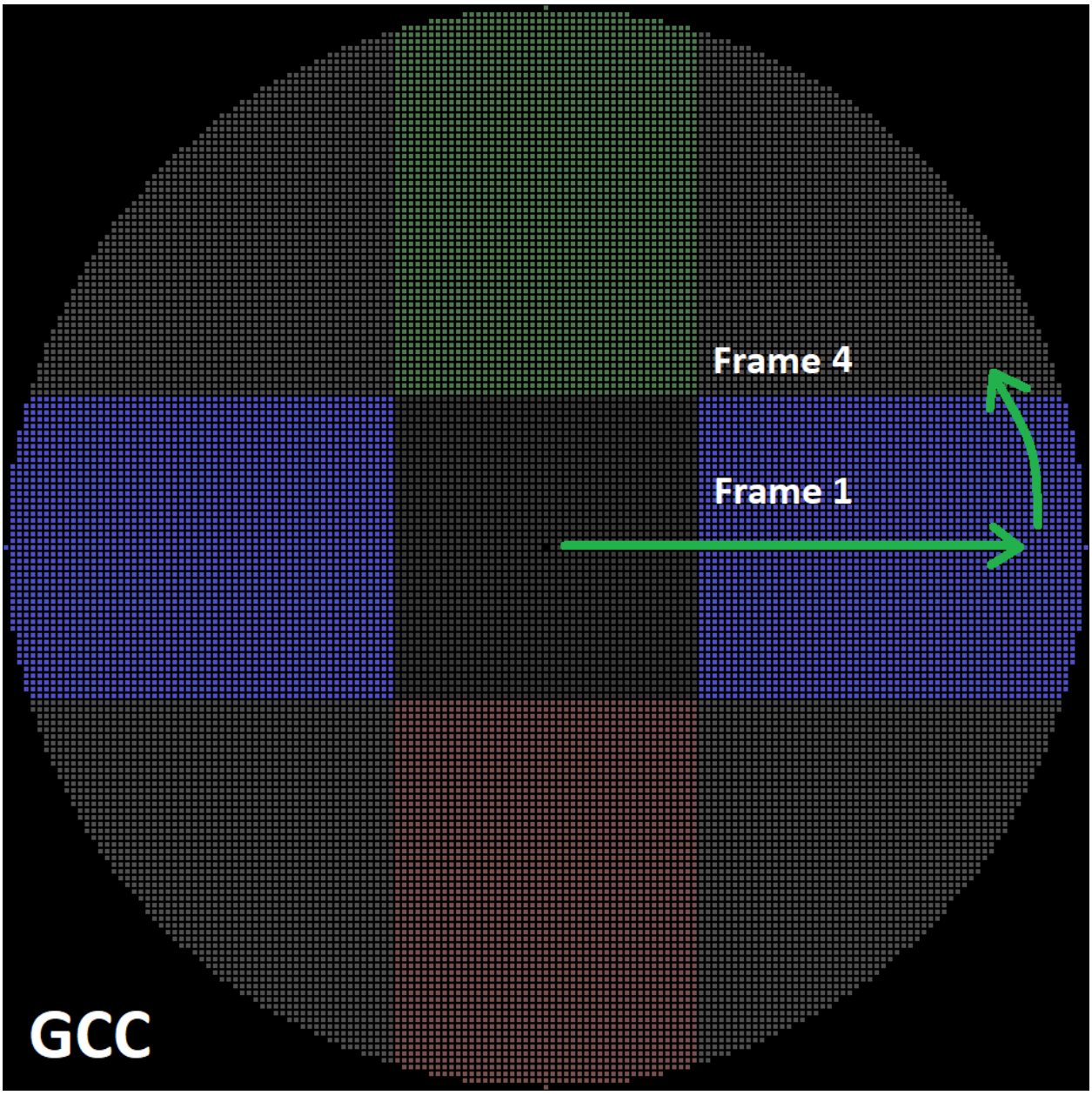
[5] Nerfs

****

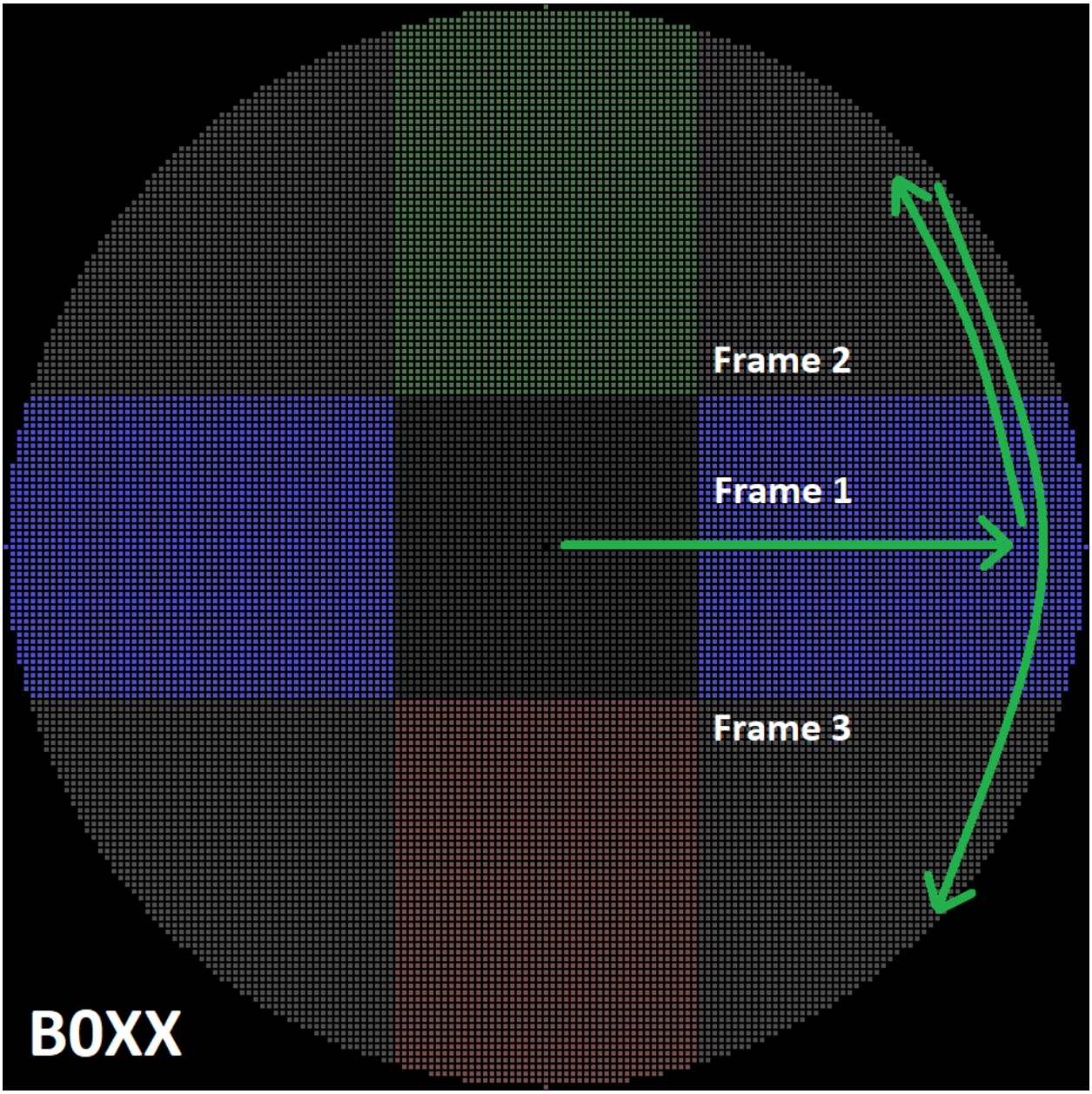
[5.1] Travel Time

Banning button sequences that involve physically impossible analog stick motions.

[5.1.1] Smash DI



On the Gamecube controller, the go-to method for SDI is quarter-circle SDI. This is when you start in a cardinal, then go into an adjacent diagonal to SDI twice by no later than frame 4.



Due to the lack of physical recoil on the B0XX, an additional quarter-circle SDI motion is feasible. This makes for a total of 3 SDI inputs, which can also occur as rapidly as a frame apart from each other (although not spacing them out increases the chance that they overlap). Compared to the Gamecube controller, this is excessive.

To balance things out, the second quarter-circle SDI motion must be removed. But before this can happen, we must establish the hitlag windows that are consistent with competitive play. Usually, these last for <= 9 frames, since almost all of the viable moves in the game hit for <= 20% (18%/19%/20% = 9 frames of hitlag). From there, it is relevant that you cannot SDI on the first frame of hitlag. This leaves us with SDI windows of <= 8 frames.

Frame 1: Cannot SDI
Frame 2: East (SDI)
Frame 3: Northeast (SDI)
Frame 4: Southeast (Banned)
Frame 5: Southeast (Banned)
Frame 6: Southeast (Banned)
Frame 7: Southeast (Banned)
Frame 8: Southeast (Banned)
Frame 9: Southeast (Banned)

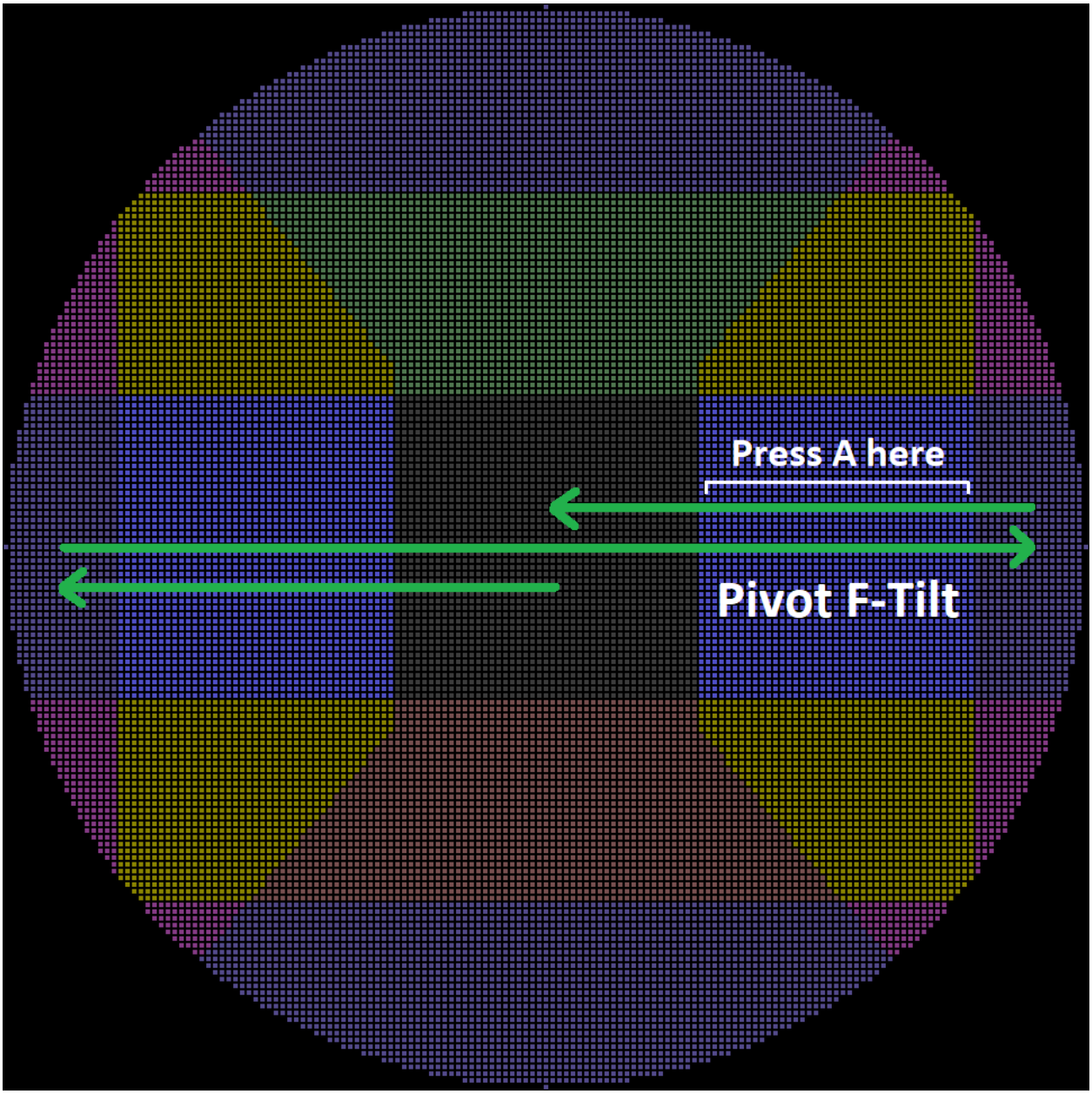
In accordance with the Gamecube controller, the B0XX has been programmed to limit the player to only one quarter-circle SDI motion (2 SDI inputs) per 9-frame hitlag window. **When a cardinal is followed by a diagonal on a later frame, the correct diagonal to go to next is banned for 6 frames.** This input, if attempted, is not "pushed" to frame 10 (as that would be a macro); it is killed entirely.

Following this nerf, the only remaining concern involves a technique known as double-tapping. This is when the middle and index fingers (in that order) are used to actuate a single button twice in succession. In theory, this can be used to bypass the current SDI nerf. By double-tapping a cardinal, then going to an adjacent diagonal, 3 SDI inputs within 9 frames of hitlag remains possible.

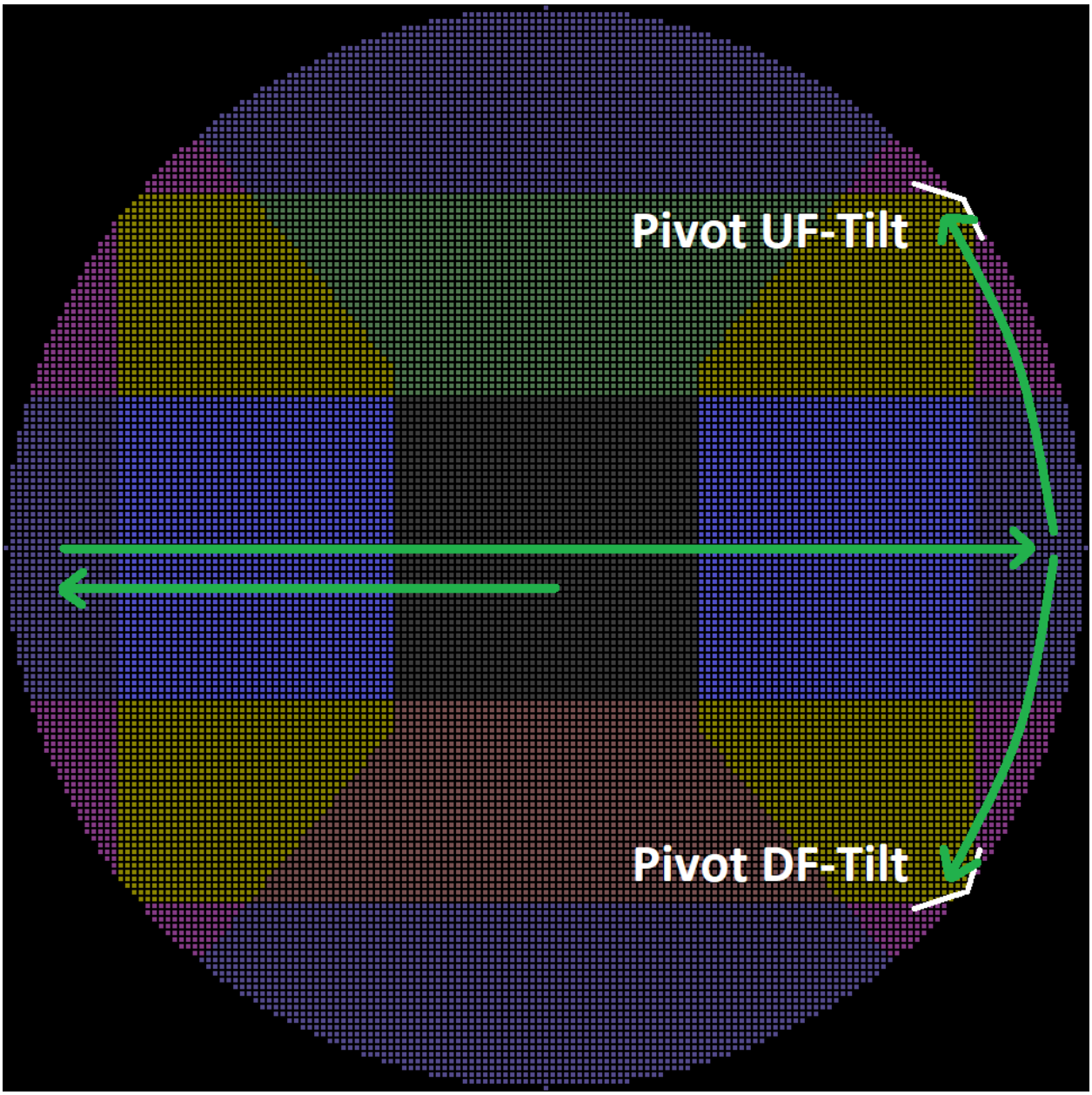
For the time being, I chose not to address double-tapping due to my inability to recreate it in practice (double-tapping has a steep learning curve). If double-tapping proves to be exploitable, I will remove it from the B0XX.

[5.1.2] Pivot Tilts

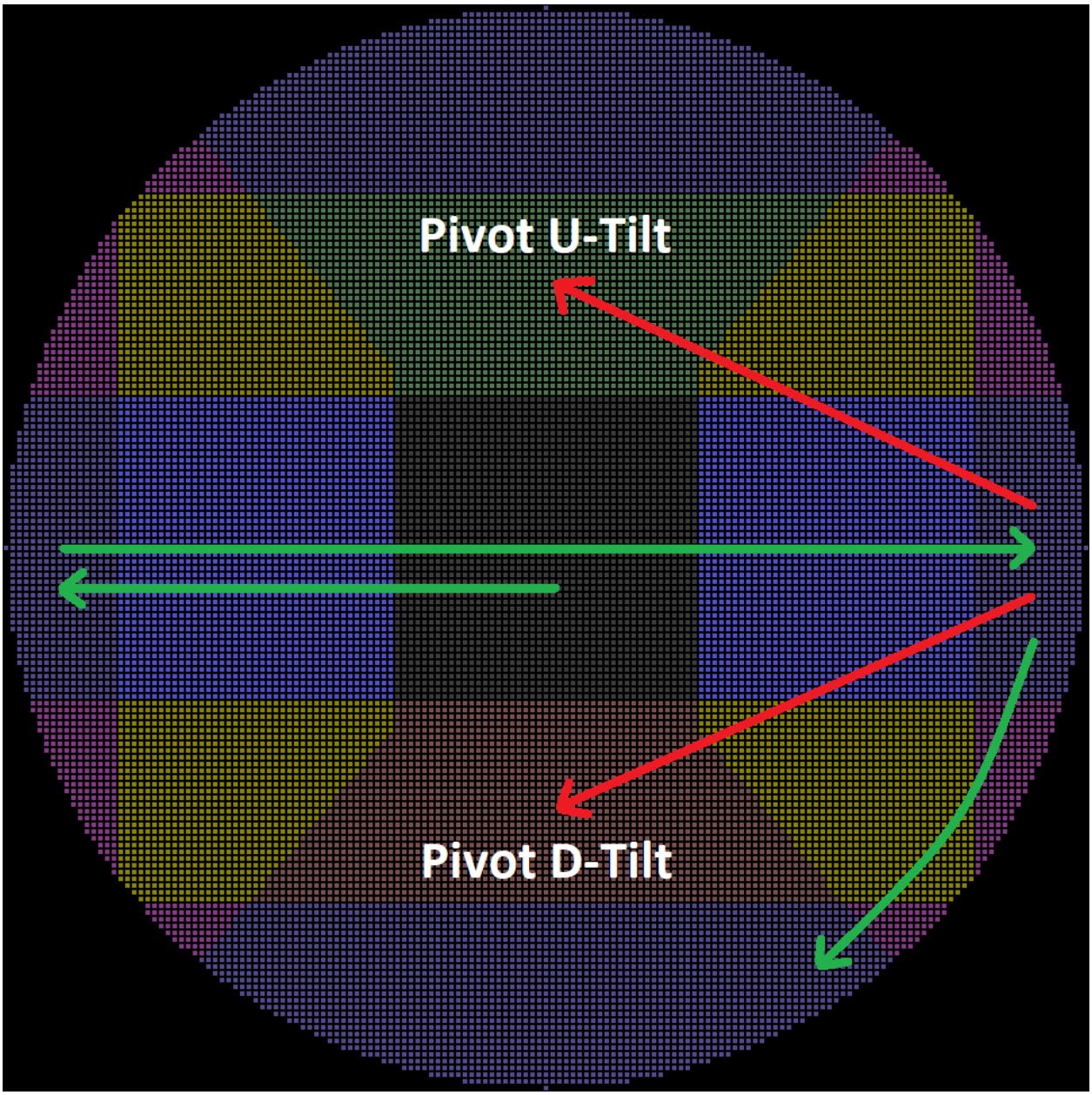
Due to the physical recoil caused by a pivot motion, it is nearly impossible to pinpoint the ideal zones for certain tilts shortly afterwards on the Gamecube controller. Determining which of these the B0XX can have will require case-by-case evaluation.



Pivot F-tilt isn't a culprit. It can reliably be done by flicking the stick and pressing the A button as the stick makes its return across X-tilt.



Likewise, pivot UF/DF-tilt can be made very reliable. This is due to the fact that their ideal zones come into contact with the case of the Gamecube controller, making it possible to notch for them. By centering your corners on Y .6125 through .6500, you'll be able to perform these by quarter-circle pivoting.



The vertical pivot tilts are where the Gamecube controller runs into problems. It is nearly impossible to reach the ideal zones for these immediately after the pivot; however, there is still a good method for pivot D-tilt. By quarter-circling downwards into Y-smash + > 50° territory and waiting 4 frames (this timer begins upon entering Y-tilt), you can perform a D-tilt afterwards. Pivot U-tilt cannot be performed in this manner due to tap jump.

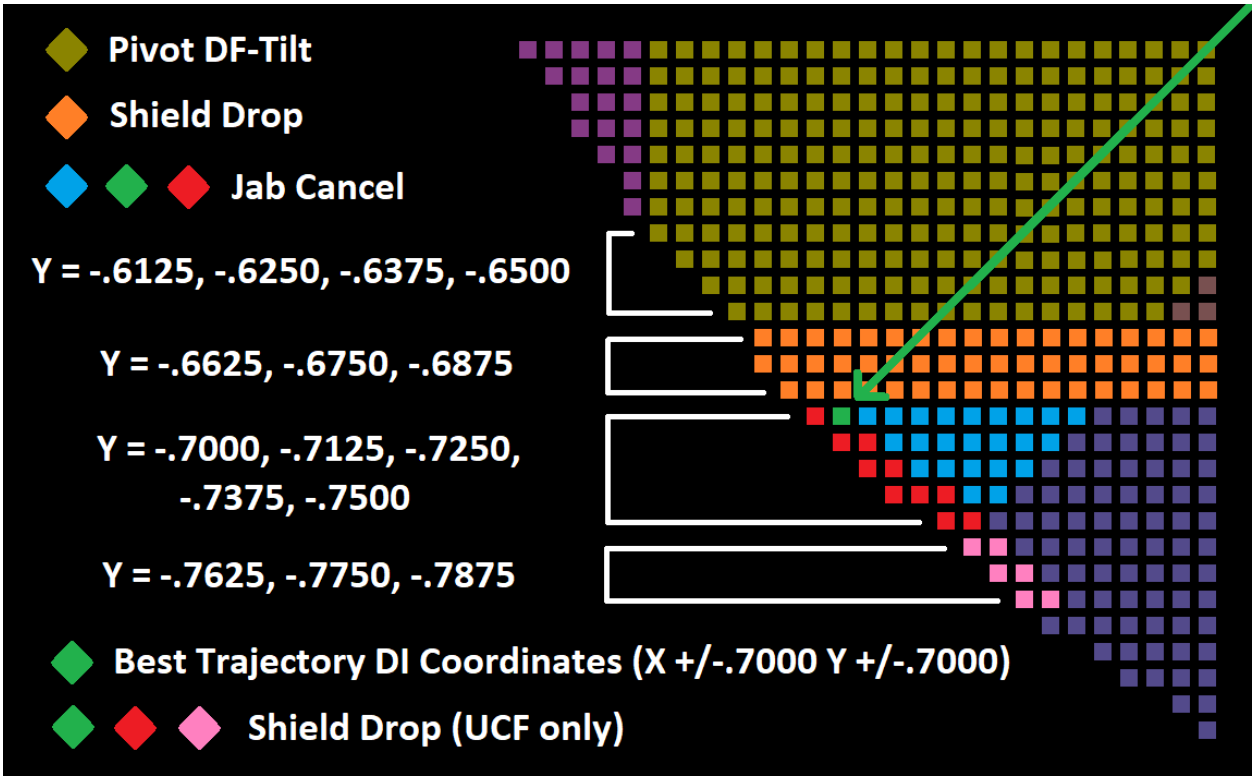
On the B0XX, the vertical pivot tilts have been brought in line with the Gamecube controller. **When any of the coordinates capable**

of performing a dash (X => .8000) are actuated for specifically 1 frame (this is how a pivot is performed), the A button won't work within the ideal zones for these tilts for a set period of time.

Pivot U-tilt: 15 frames (removed)
Pivot D-tilt: 4 frames

[5.1.3] Notch Integrity

In Section 5.1.2, I brought up some rather uncommon notches that assist with pivot UF/DF-tilt. **While these notches do exist, they are ultimately bad to have.** This is because it is impossible to construct them without losing out on better notches.



There are several points of interest in proximity of the NE/NW/SW/SE grooves.

In all 4 quadrants, the pivot UF/DF-tilt notch sites happen to be near X +/-.7000 Y +/-.7000. These are significant for being the best* trajectory DI coordinates in the game against moves that cause trajectory 361 knockback (nearly half of the moves in the game do this). Therefore, any notch must be weighed against how

far it takes you away from X +/-.7000 Y +/-.7000; in other words, how much DI it costs you. Pivot UF/DF-tilt notches both take you at least 4 degrees away from these coordinates. This alone isn't a worthwhile trade-off.

*Technically, X +/-.7125 Y +/-.7000 and X +/-.7000 Y +/-.7125 are the best trajectory DI coordinates. These aren't listed because they have special traits that make them difficult to pinpoint.

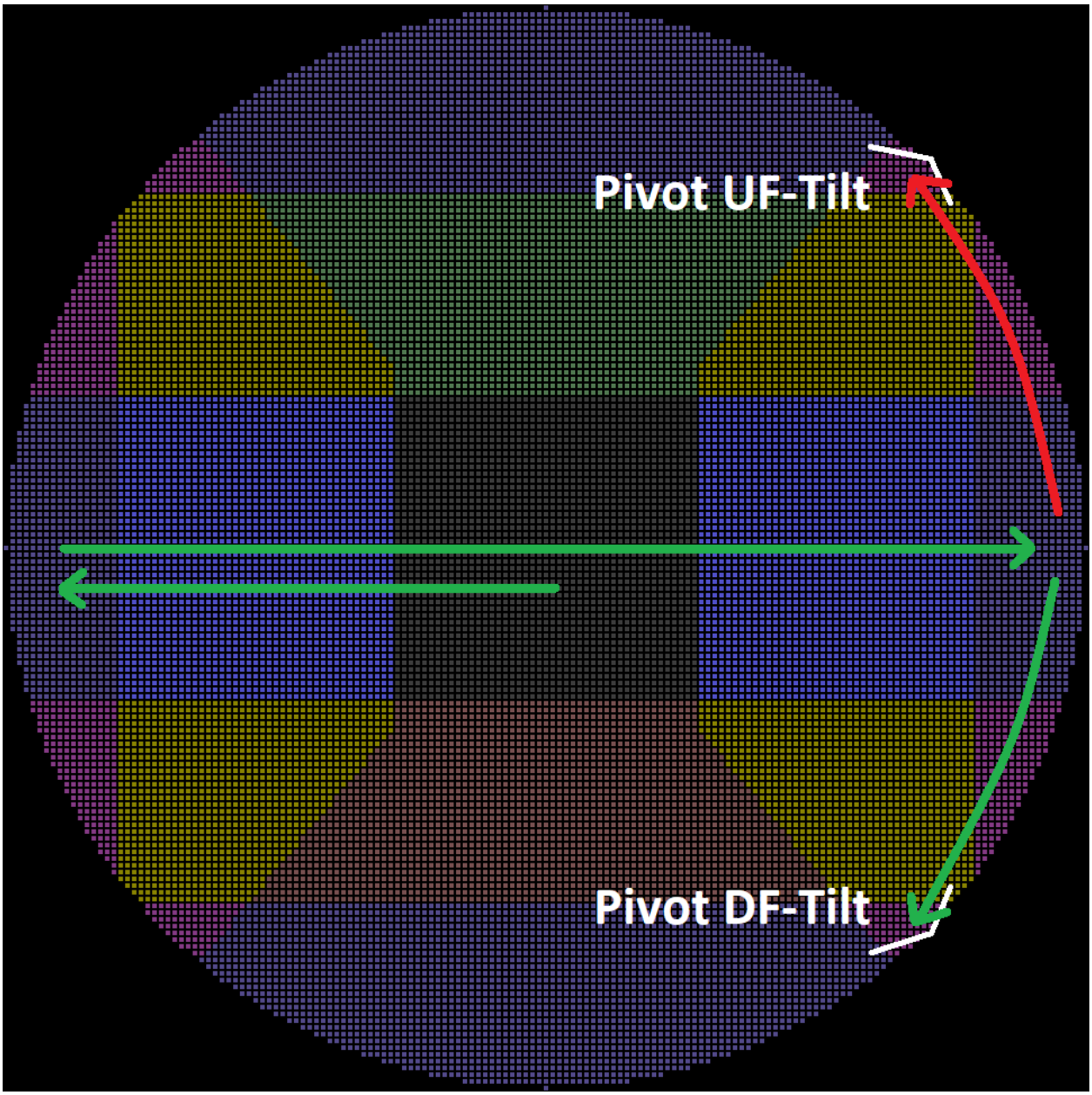
In quadrants 3 and 4, there are two more techniques that benefit from notches. The first is jab cancel, which spans from Y -.7000 to -.7500. This is when you cancel a jab's IASA frames with a crouch in the opposite direction, then press A to jab again. The second is shield drop, which spans from Y -.6625 to -.6875 on vanilla and Y -.6625 to -.7875 on Universal Controller Fix (a mod that has become the tournament standard). Pivot DF-tilt notches (Y -.6125 through -.6500) cannot coincide with the notches for either of these techniques (regardless of game version).

Based on this information, the best coordinates to center your corners on (if playing on UCF) are:

Quadrants 1, 2, 3 and 4: X +/-.7000 Y +/-.7000

These coordinates give you the best trajectory DI in quadrants 1 and 2, and the best trajectory DI, jab cancel, and UCF shield drop in quadrants 3 and 4.

Having your corners on these values means not having pivot UF/DF-tilt notches, which I believe the B0XX should remain true to. At this point, both of these pivot tilts have to be re-evaluated based on how well a Gamecube controller with X +/-.7000 Y +/-.7000 corners can perform them.



In the same fashion as pivot U/D-tilt, there is still a good method for pivot DF-tilt, but not UF-tilt. Whereas X +/-.7000 Y -.7000 is valid for a DF-tilt after 4 frames, X +/-.7000 Y .7000 will prompt tap jump. Pressing the A button within the ideal zones for these tilts (after pivoting) has been restricted accordingly.

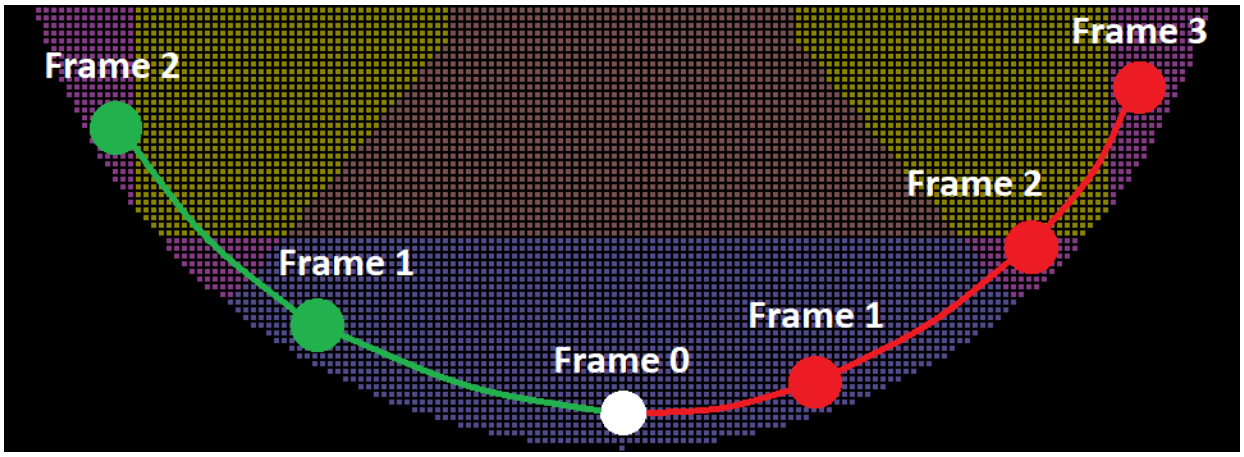
Pivot UF-tilt: 15 frames (removed)
Pivot DF-tilt: 4 frames

[5.1.4] Dash Back Out of Crouch

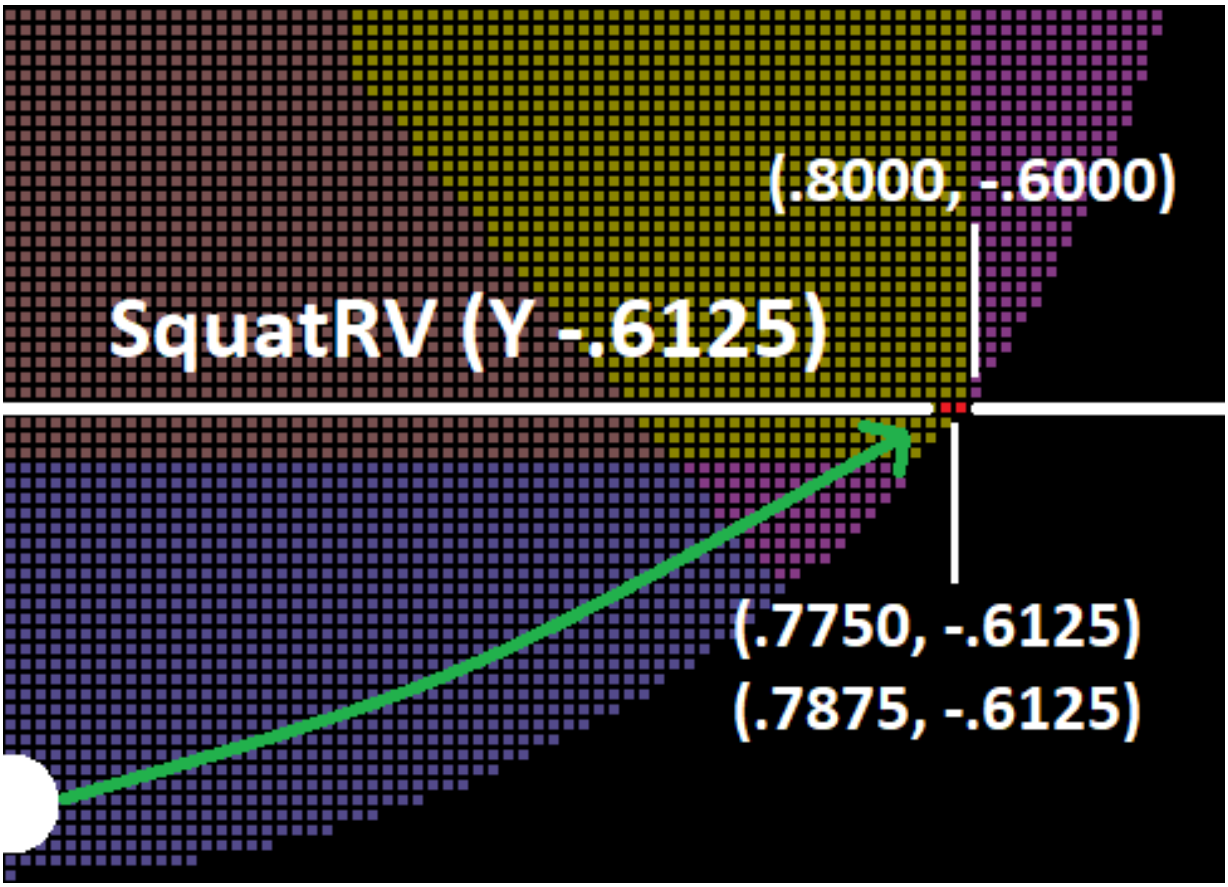
Disclaimer: UCF does not affect dash back out of crouch (dash back and dash back out of crouch are two separate techniques).

In today's metagame, an increasingly popular choice when techchasing is crouching in front of a knocked down opponent. This removes the need to react to their get-up attack, since crouch cancel renders get-up attack invalid. From there, dash out of crouch can be used to follow your opponent's wake-up roll. While dash out of crouch isn't without execution requirements of its own, many players find the most success techchasing by rinsing and repeating this strategy.

Unfortunately, this strategy has a glaring weakness: **it is humanly impossible to successfully dash *back* out of crouch 100% of the time.** While dashing forwards and backwards out of crouch share certain criteria, there is a key difference between them that attaches a failure rate to the latter.



The execution test incurred by dashing in either direction out of crouch is **being able to traverse from crouch to dash (X-smash) within 2 frames (if this motion takes 3 or more frames to complete, your character will walk).** Through the use of one method or another, this criteria isn't hard to satisfy. Some players swear by rolling the analog stick along the bottom of the rim in order to minimize its travel route, while others prefer to return the stick to its centerpoint before pressing it horizontally. With practice, either of these methods should ensure a 100% success rate on dash *forward* out of crouch.

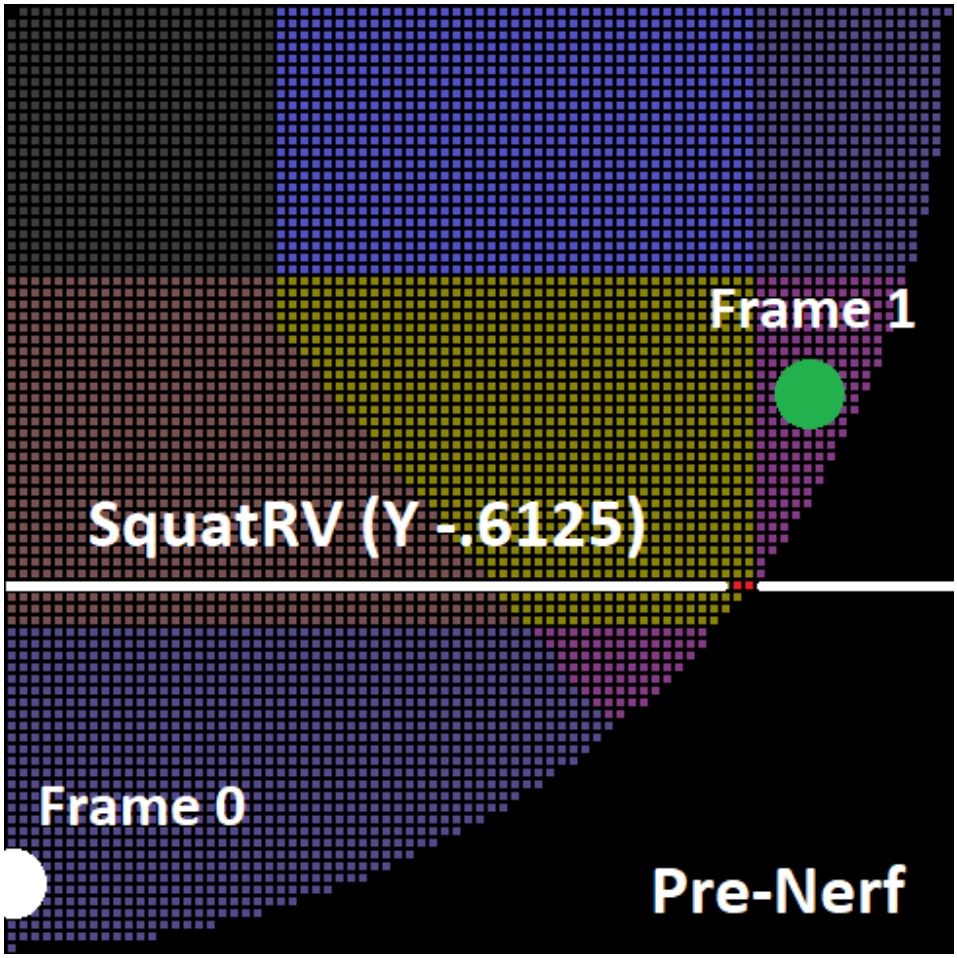


Along the way to X-smash, X +/-.7750 Y -.6125 and X +/-.7875 Y -.6125 (squatRV) cannot be avoided 100% of the time.

When it comes to dash back out of crouch specifically, any method that does not involve hugging the bottom of the rim is made unviable by squatRV. Located at Y -.6125, squatRV will cause your character to stand up from crouch. SquatRV can be disregarded when dashing forwards out of crouch, as it can be cancelled into

dash forward; however, squatRV cannot be cancelled into dash back. This wouldn't have been a problem if not for the fact that **squatRV cannot be avoided 100% of the time.**

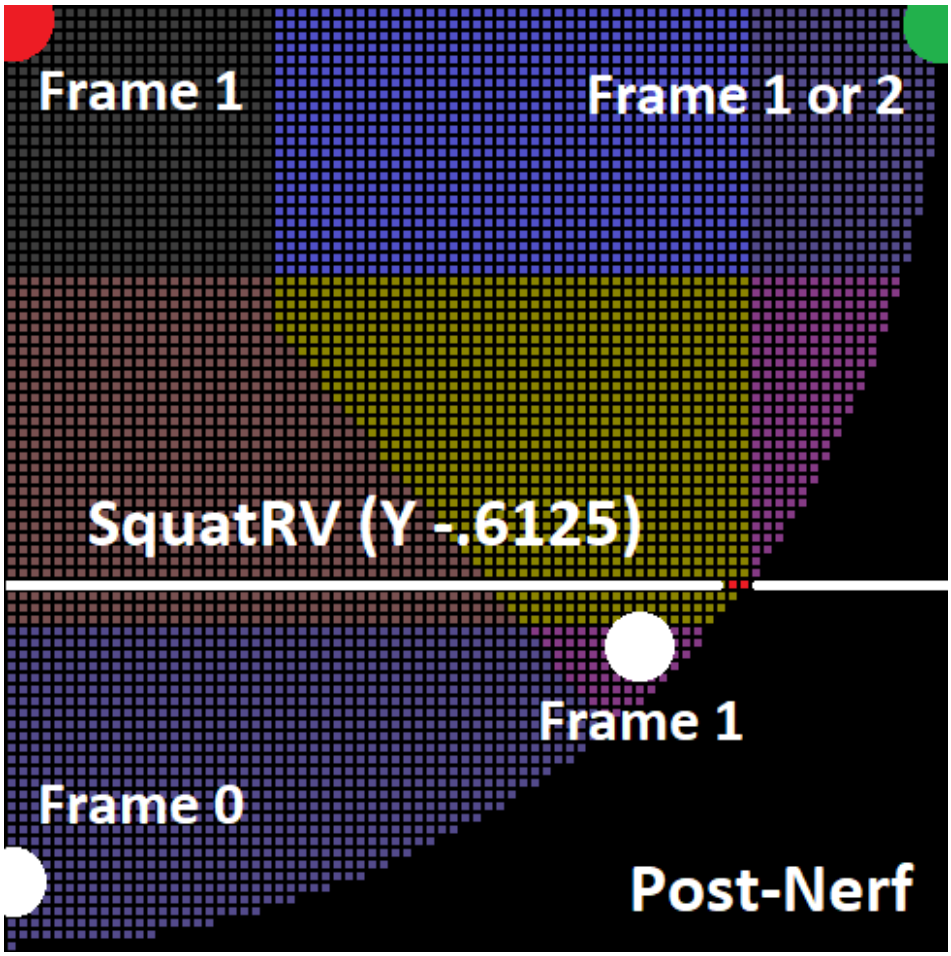
Even if the most direct route from crouch to dash is taken, two dreaded coordinates must be traversed in the process: X +/-.7750 Y -.6125 and X +/-.7875 Y -.6125. If you are polled in these coordinates, dash back out of crouch will fail. Since this is impossible to account for, there is always an element of luck when attempting dash back out of crouch with an analog stick.



By crouching, then dashing in a quadrant, it is impossible to fail dash back out of crouch with digital inputs.

Because digital inputs don't have a travel route, they are able to skip from crouch to dash without traversing squatRV. While removing this advantage may seem hopeless, it is actually just a

matter of banning the button sequence shown in the diagram. This button sequence in particular is concerning because it consists of Down -> Down-Forward, which means **the player never has to release Down in order to perform dash back out of crouch.** For as long as this is true, dash back out of crouch cannot fail; however, if Down must be released, then digital inputs are forced to incur risk. **In order to create the need to release Down, it must be illegal for the B0XX to travel from crouch territory to diagonal dash territory.** While the B0XX's programming doesn't outright ban this button sequence, it is made impossible through means that will be revealed in Sections 8.2.1 and 8.3.3. For now, operate under the assumption that the B0XX must fall back on a secondary dash back out of crouch method.



Once Down has to be released, the combination of squatRV and the 2-frame dash window creates an execution test for dash back out of crouch.

With dashing in the quadrants out of the picture, the B0XX is forced to perform dash out of crouch in a cardinal direction. There are only two valid button sequences for this:

Sequence A (Success)

Frame 0: Crouch
Frame 1: Dash (X +/-1.0 Y 0)

Sequence B (Success)

Frame 0: Crouch
Frame 1: X +/-.7000 Y -.7000
Frame 2: Dash (X +/-1.0 Y 0)

Having to release down also makes for the possibility of a third button sequence. This will cause dash back out of crouch to fail:

Sequence C (Failure)

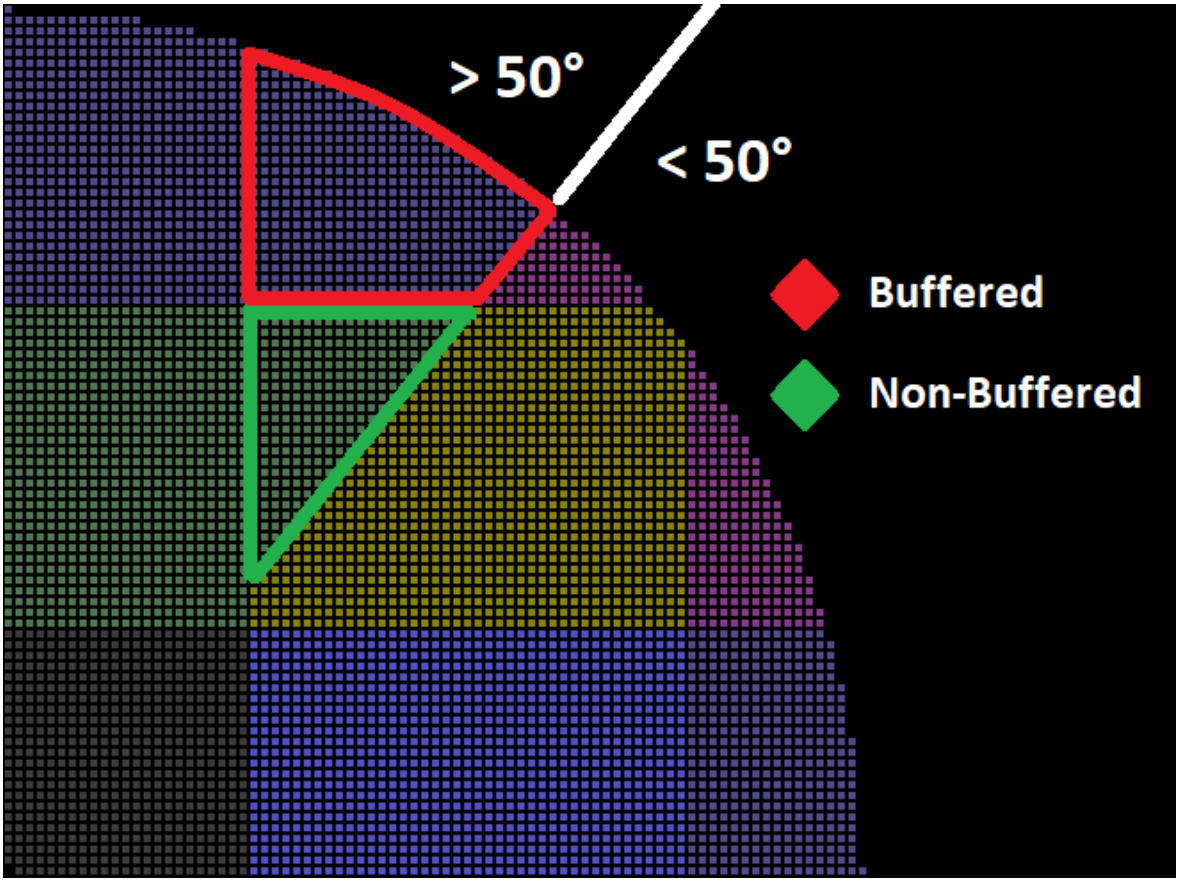
Frame 0: Crouch
Frame 1: SquatRV (X 0 Y 0)

Sequence A is a one-frame link, while Sequence B is a pseudo-two-frame link (Down and Forward can overlap for one frame, but Down must be released on the next frame for dash to occur). Meanwhile, Sequence C is an accidental series of inputs that can result from having to perform a skillful motion. All in all, removing the B0XX's ability to crouch, then dash in a quadrant forces the player to think twice about performing dash back out of crouch.

[5.2] Precision

Removing the ability to pinpoint certain coordinates with 100% accuracy.

[5.2.1] Y-Tilt + > 50°



Within the quadrants, > 50° territory can be used to perform turnaround vertical tilts. In doing so, it is relevant whether you are pointing in Y-tilt or Y-smash. Whereas the former can be used in any situation, the latter can only be used in buffered situations (i.e. L-cancel lag). This is because Y-smash is associated with techniques that conflict with your ability to perform tilts, such as tap jump and smash attacks.

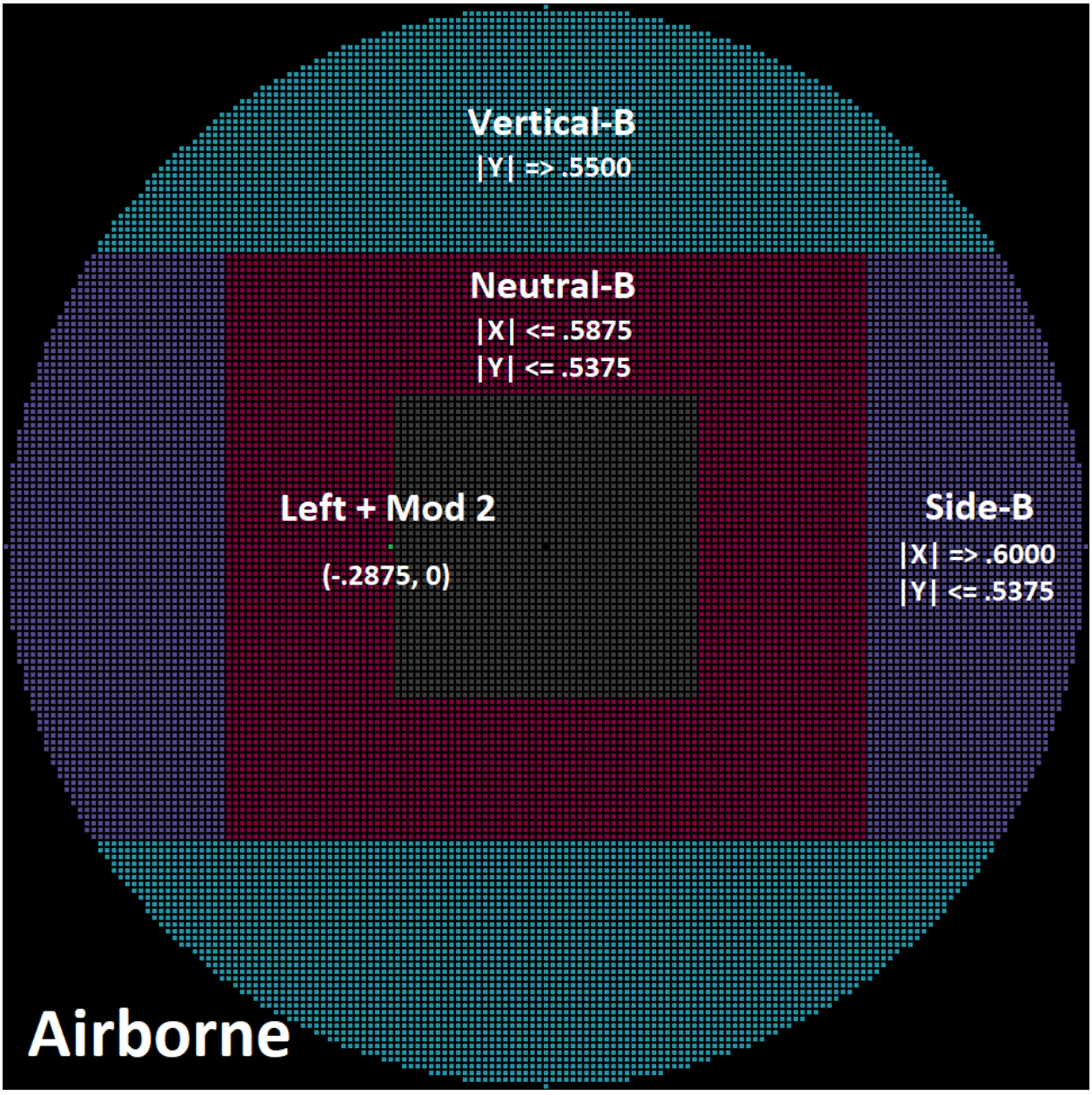
On the Gamecube controller, the elusive Y-tilt + > 50° is extremely difficult to pinpoint. As a result, people usually perform non-buffered turnaround vertical tilts by turning around, *then* pointing vertically (in the north or south corridor), which

can only be equal to or slower than pointing directly at Y-tilt + > 50°. To recreate this inconvenience, **the B0XX cannot pinpoint Y-tilt + > 50°.**

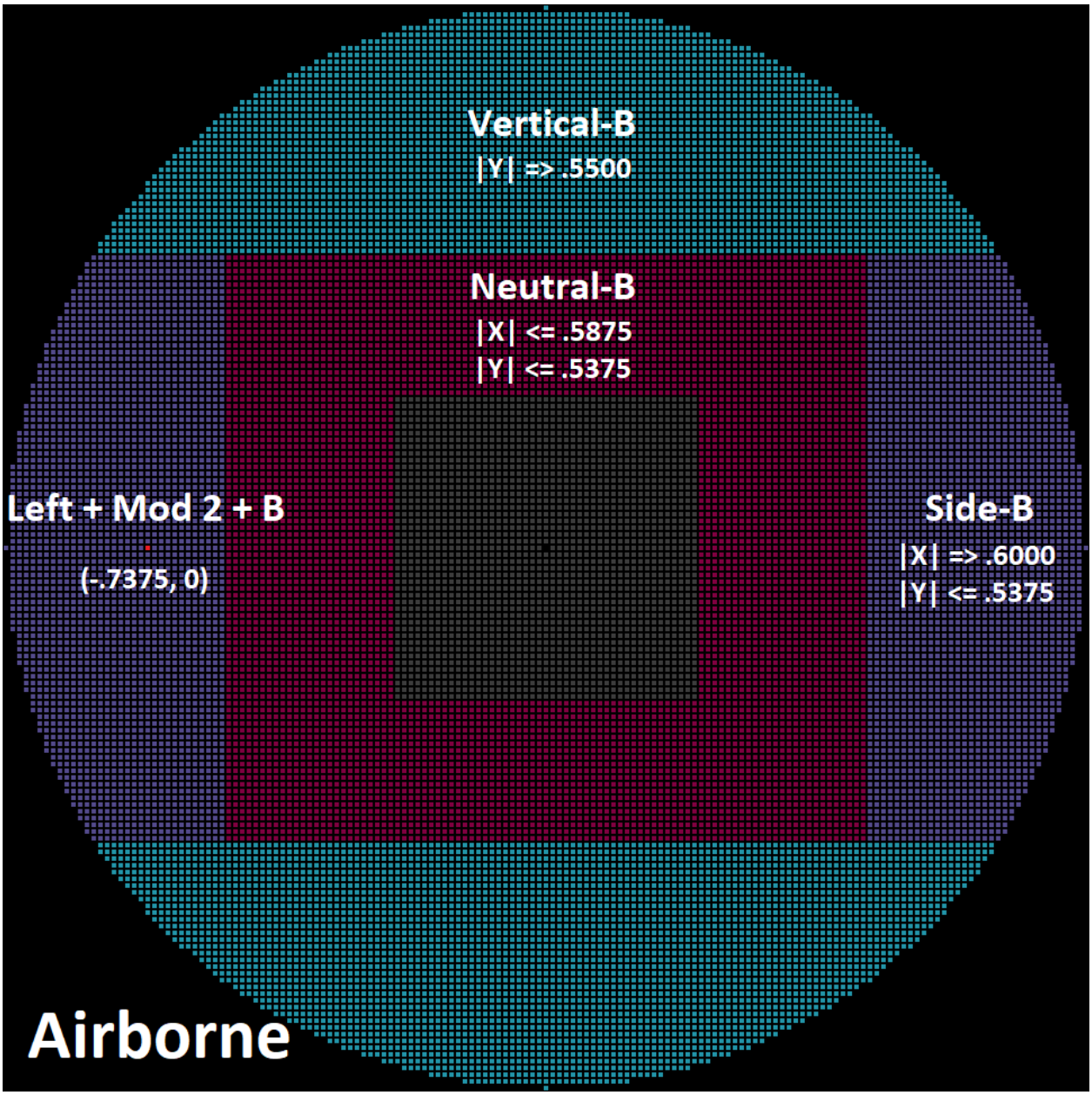
Additionally, **the B0XX cannot pinpoint Y -.6625, -.6750, or -.6875.** Even though these coordinates are not in Y-tilt territory, they can be used to perform a buffered turnaround D-tilt (Y <= -.7000 causes crouch, which prevents this).

[5.2.2] Neutral-B Integrity

One of the Gamecube controller's intrinsic risks is overshooting into side-B territory when attempting to neutral-B in the other direction.



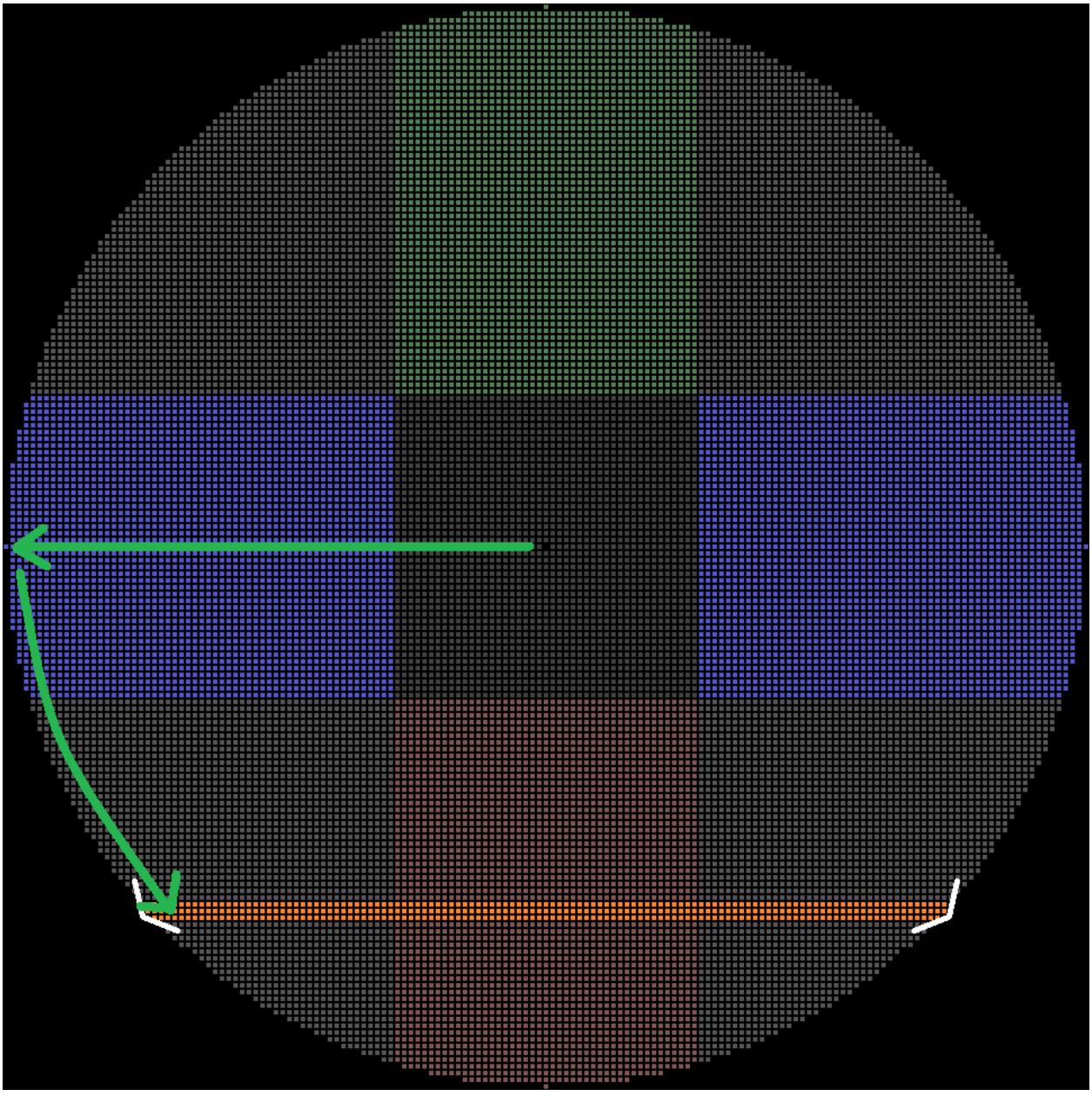
Digital inputs can circumvent this risk by situating the analog stick in X .2875 through .5875 (the zone that ensures a neutral-B in the chosen direction).



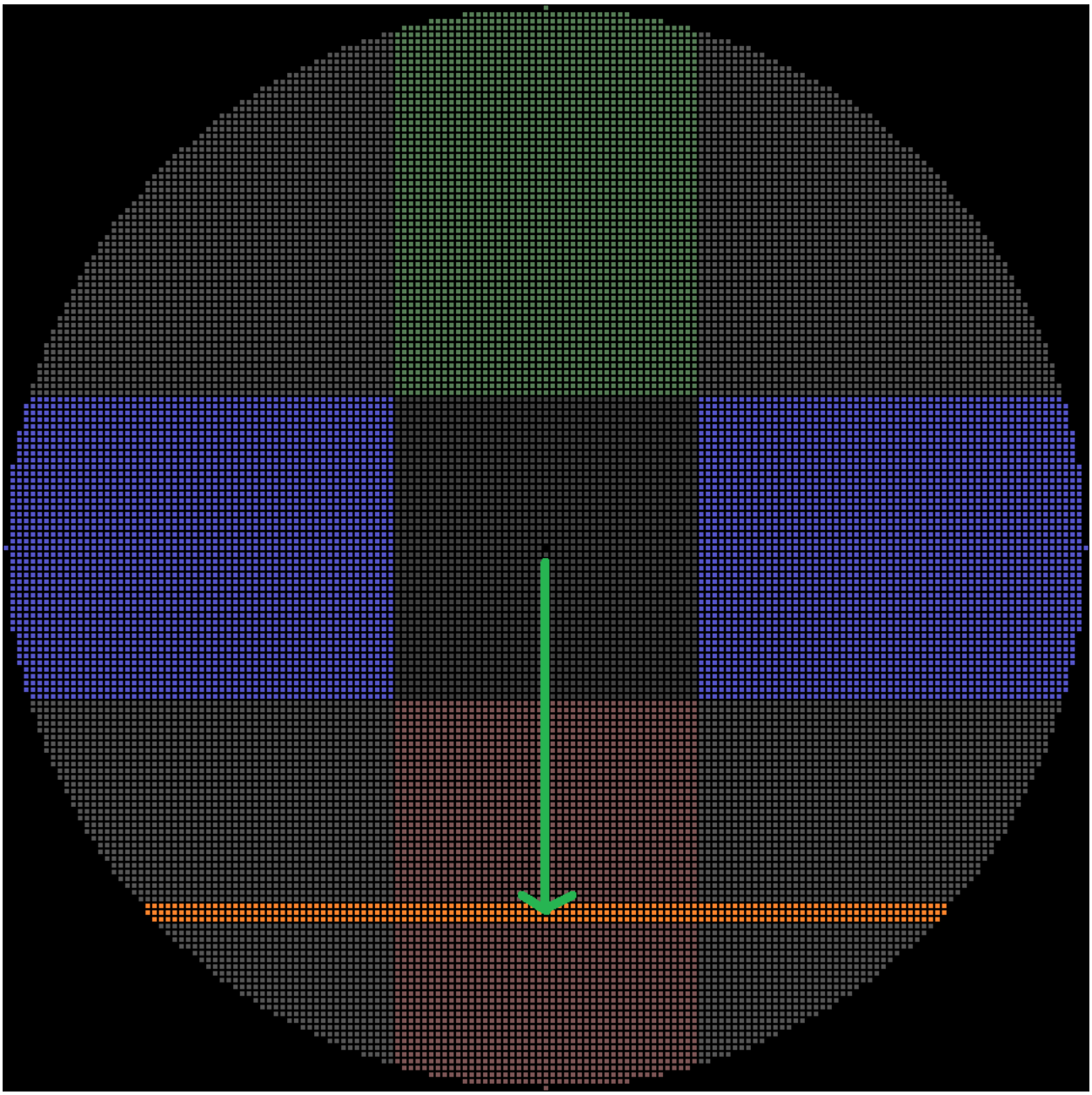
Since this would otherwise feel disingenuous, **attempting to neutral-B with the analog stick in X .2875 through .5875 will result in your X-value being pushed into side-B territory.** X .7375, the coordinate this pushes you to, is a non-arbitrary one that the B0XX already contains elsewhere. This means that the only way to neutral-B is to completely release the horizontal(s).

[5.2.3] Shield Drop Down

With the Gamecube controller, there are two drastically different ways to shield drop.



The first shield drop method involves shutting off roll, then going to the SW/SE corner of the stick (which has been notched). This has become well-known for its ease and reliability in recent years. Colloquially, it is called the “Axe method.”



The second method involves pointing directly down at shield drop's Y-values. This lets you shield drop as early as frame 2 (if you haven't shielded yet). In theory, there is no reason not to shield drop this way every time. In reality, however, this method is far too difficult to perform consistently.

The key to removing shield drop down from the B0XX lies in the order of priority within the game. If chosen on the same frame, the game will prioritize:

Spotdodge (Y -.7000) > Roll (X +/-.7000) > Shield Drop (Y -.6625)

Therefore, **if the only shield drop Y-values on the B0XX are paired with roll X-values, shield dropping without shutting off roll is impossible*.** The B0XX uses the coordinates X +/-.7250 Y -.6875, which abide by this rule.

*This nerf removes the B0XX's ability to shield drop down on frames 2, 3 and 4. There remains a method that involves tilting shield downwards (in Y-tilt) for 4 frames, which then turns the entire spotdodge range (Y <= -.7000) into shield drop on specifically frames 5 and 6. This does not need to be targeted, as it can reliably be performed with the Gamecube controller.

[5.2.4] Lightshield

Analog L/R 43, the largest lightshield in the game.

In Melee, analog L/R-values span from 0 to 140, with 43 being the lightest press that generates a shield. Since L/R 0 through 42 do not generate anything, it can be difficult to press the L/R trigger precisely enough to pinpoint L/R-values equal to or close to 43.

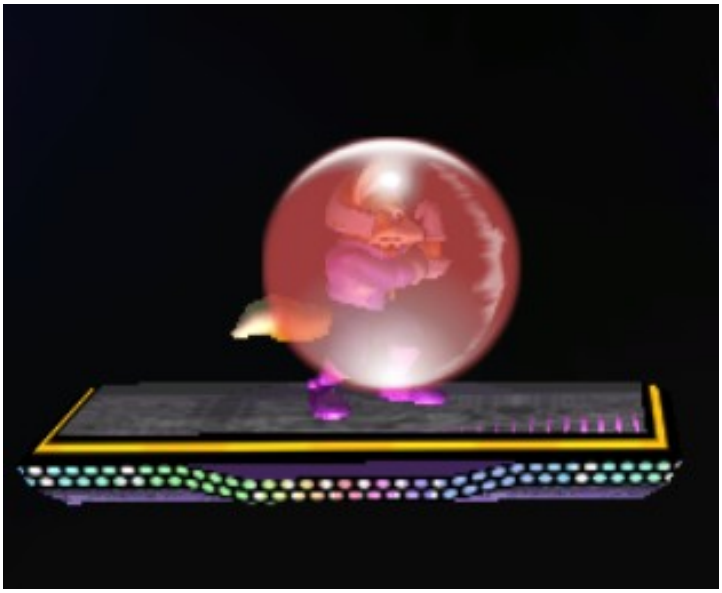
There are two ways to modify a Gamecube controller to bypass this difficulty entirely, however, neither of them are without their

flaws. Most commonly, a physical stopper is inserted in the L/R trigger. This lets the player press down forcefully and be halted at the analog L/R-value of choice. While stoppers serve their intended purpose, they come with the downside of preventing the trigger from being digital pressed. This is why, in my opinion, the better method is to manipulate the trigger's calibration upon plugin. In doing so, L/R 44 through 140 can be disabled on a software level. This allows L/R 43 to be pinpointed by resting the slider atop its digital press.

The second method allows you to pinpoint L/R 43 with one trigger and have access to the entire analog L/R range with the other. This would have been flawless if not for the existence of **analog-digital transition**, a game mechanic that can cause your shield to fail to protect you from physical attacks. Analog-digital transition occurs when you are polled in L/R 43 through 140, followed by digital L/R on the next frame. This results in an analog shield protecting your character on frame 1, **no shield protecting your character from physical attacks on frames 2 and 3**, then the expected digital shield on frame 4 (despite a shield being visibly displayed the entire time). Similar to vanilla Melee's dash back dilemma, this polling sequence is impossible to account for.

Luckily, a Gamecube controller can (and should) be made immune to analog-digital transition. By removing the spring from an L/R trigger (or manipulating its calibration), the entirety of L/R 43 through 140 can be disabled. This guarantees digital L/R presses with the modified trigger, **but conflicts with being able to devote a trigger to analog L/R 43**.

Based on this information, the ideal Gamecube controller contains one springless trigger, and one trigger that has not been manipulated in any way (so that it can select from the entire analog L/R range). Since this Gamecube controller does not contain a way to pinpoint L/R 43, it is appropriate to keep the B0XX a small distance away from L/R 43 as well. The only non-arbitrary L/R-value to nerf to is 49 (the size of the shield generated by the Z button); therefore, **the B0XX cannot pinpoint L/R 43 through 48**.



Analog L/R 140. Visibly identical to a digital shield, but functionally different.

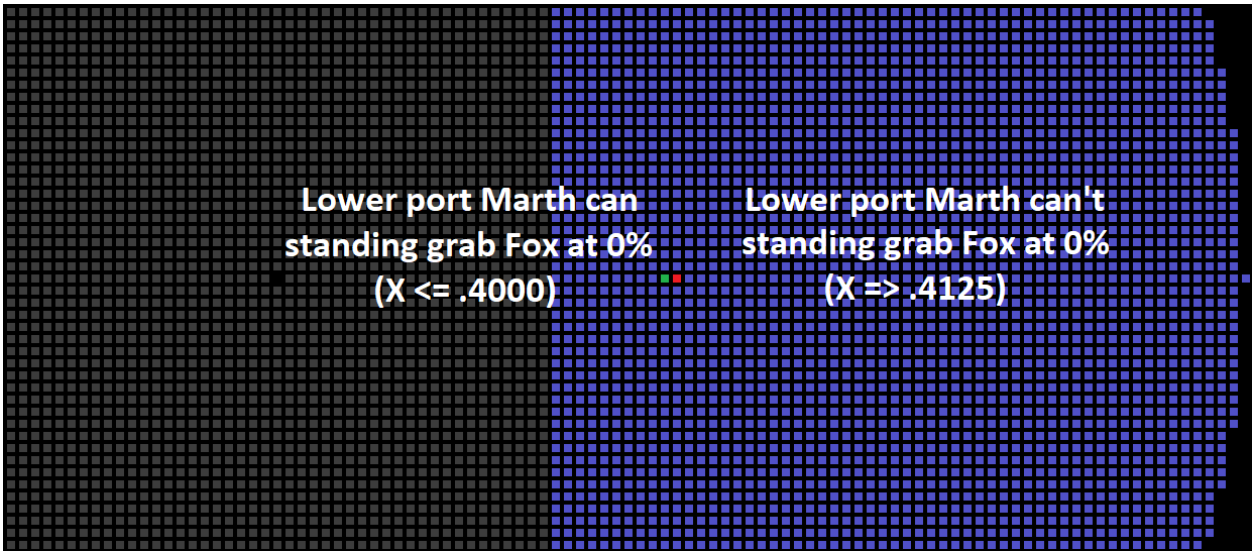
Additionally, there is a lightshield-related technique very few players are aware of. By situating the L/R trigger's slider atop its digital press, L/R 140 will be generated. **The significance of L/R 140 is that it incurs the same amount of shield stun as a digital shield, but cannot powershield.**

Contrary to popular belief, **powershielding physical attacks can actually be disadvantageous.** Because a successful powershield pushes your character back, it can push them out of range for certain out of shield options. In theory, L/R 140 would have been useful in these situations. In reality, however, a Gamecube controller cannot make use of this technique. Since L/R 140 can only come *after* L/R 43 through 139, it is humanly impossible to guarantee that you aren't polled while traversing these L/R-values. In the event that you are, an L/R 43 through 139 shield will be generated for *two* frames, followed by analog L/R 140 on the third (even if you are polled in L/R 43 through 139 on frame 1, followed by L/R 140 on frame 2). This defeats the purpose of attempting this technique on a Gamecube controller.

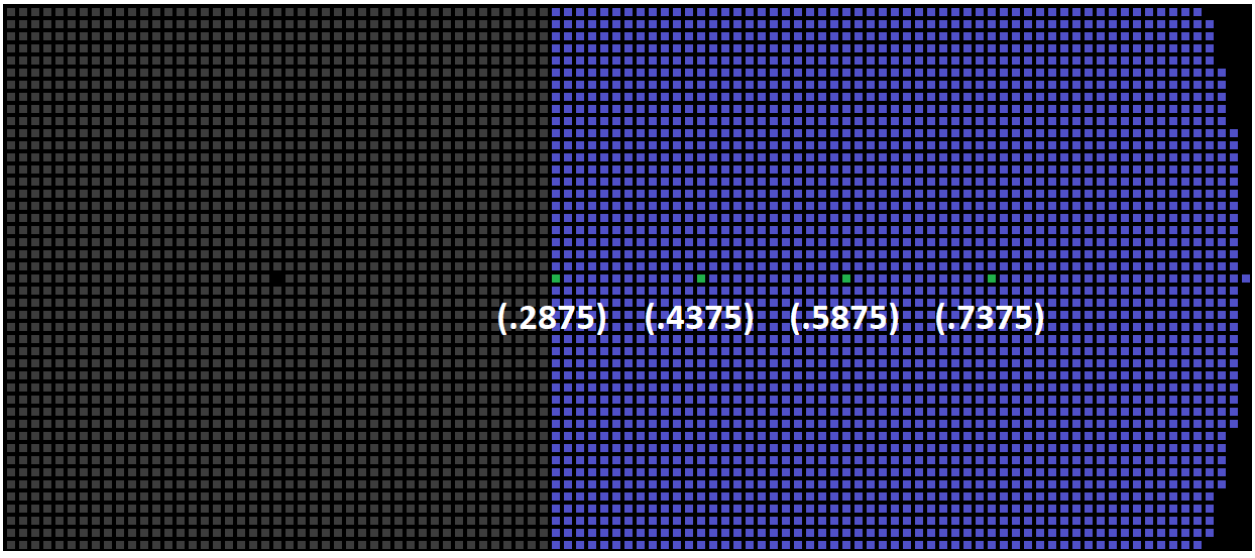
As to avoiding raising any issues with lightshielding, **the only analog L/R-value the B0XX is capable of pinpointing is 49.**

[5.2.5] Ambiguous DI

When you are thrown vertically, the best option is often to DI ambiguously. This is when you make it difficult for your opponent to discern which side of them you're on. In these situations, there are always "perfect coordinates;" ones that make it as hard as possible for your opponent to tell.



In this situation, for example, X .4000/.4125 is the perfect mix-up. In theory, if the B0XX had access to every X-value, it would be able to pinpoint both of these coordinates; therefore, the line has to be drawn somewhere.



The clear choice is **4 coordinates along the X-axis.** Since less than 4 (2) is grossly underpowered, while more than 4 (8) digs into user-friendliness, this is the correct amount.

The significance of X +/-.7375, the final coordinate in this range, will be explained in Section 7.1.

[5.2.6] Ice Climbers Desyncs

The B0XX does not contain any of the coordinates that cause Popo and/or Nana to perform isolated actions. These are:

X +/-.8000 (Popo X-Smash/Nana X-Tilt)
Y +/-.6625 (Popo Y-Smash/Nana Y-Tilt)
X +/-.7000 Y Not Along Rim (Popo Roll)*
X Not Along Rim Y -.7000 (Popo Spotdodge/Nana Shield Drop)*
X Along Rim Y -.8000 (Popo Spotdodge/Nana Shield Drop) (UCF Only)**
X .6250 (Popo Run/Nana Runbrake)***
X .7500 (Popo Teeter Break/Nana Teeter)***
Y .5625 (Popo Jump out of Dash/Run/Runbrake/Turnrun)
|X| <= .5875 Y -.5500 (Nana Neutral-B) (Grounded Only)
X Y or C X Y in Proximity of 50° Line (2 Different Aerials)
C X +/-.8000 (Popo F-Smash)
C Y +/-.6625 (Popo U/D-Smash)

*Based on the logic presented in Section 5.1.3, it is acceptable to pinpoint X +/-.7000 and/or Y -.7000 (the sources of these desyncs) along the rim.

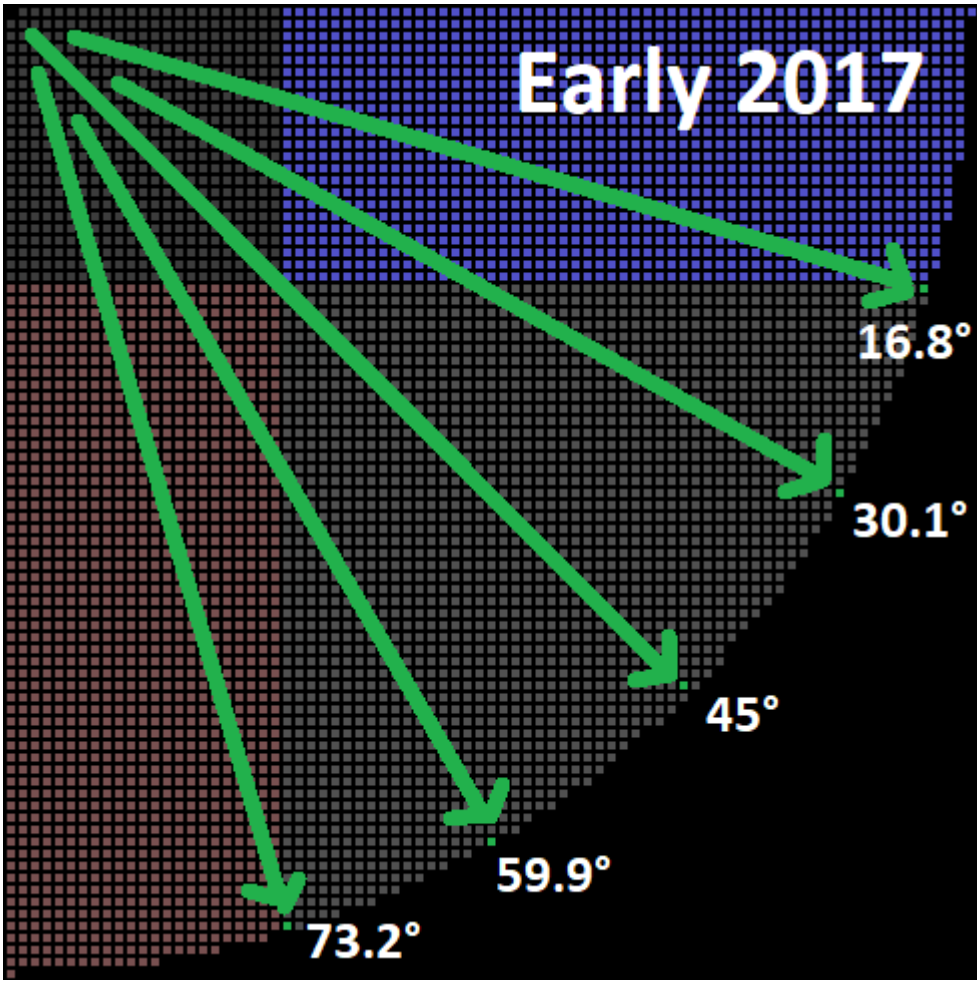
**This desync is a byproduct of UCF's modified shield drop range.

***These desyncs only work facing east. This is due to the Gamecube controller's X-axis spanning from -128 to 127 before conversion to Melee values occurs, resulting in imbalances in absolute values.

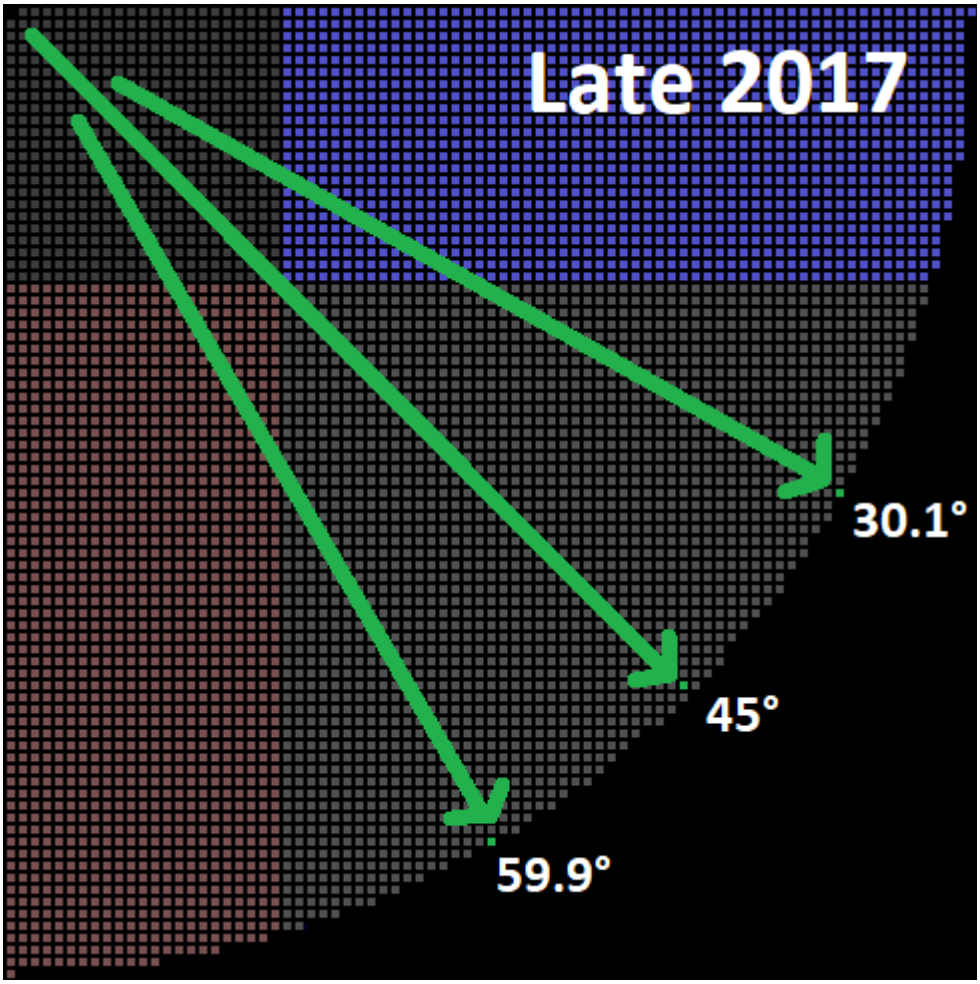
[5.2.7] Wavedash

Disclaimer: The B0XX is based on a stock Gamecube controller, not one that contains wavedash notches.

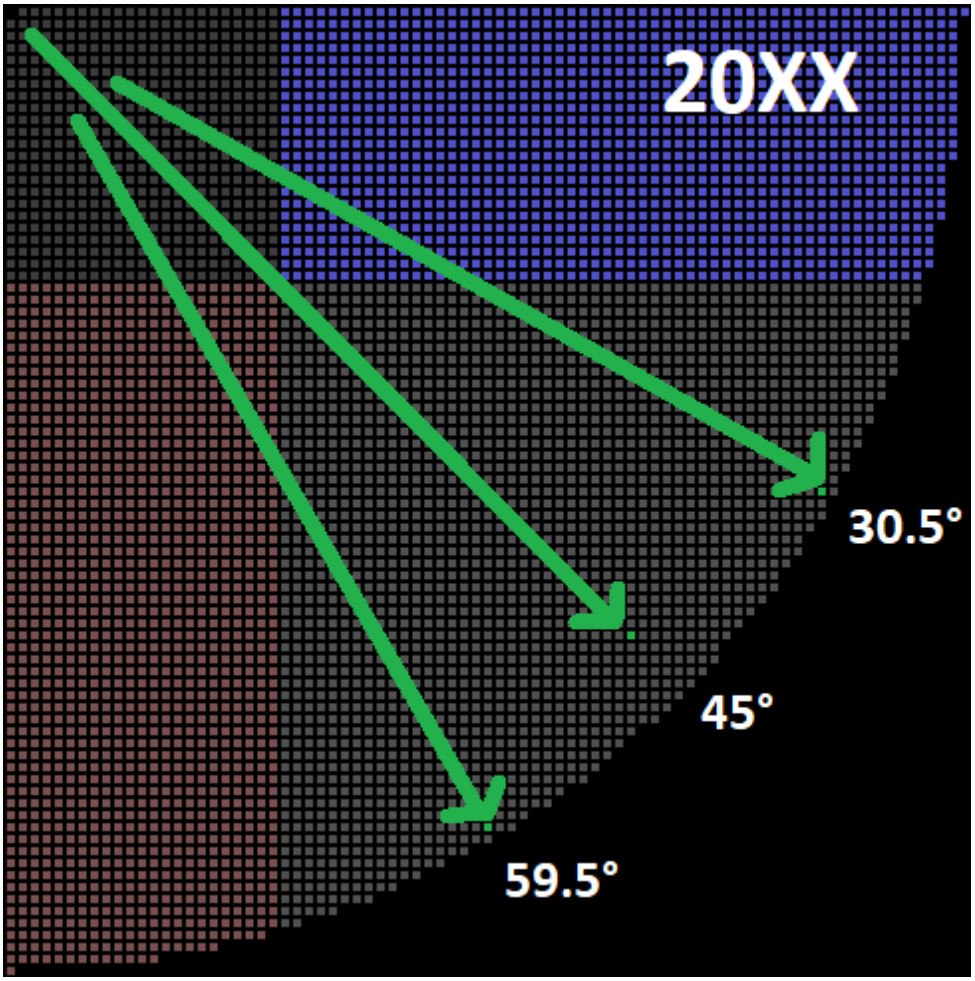
If unrestricted, the B0XX's most blatantly overpowered feature is its ability to pinpoint the shallowest wavedash angle in the game. It doesn't take much experience with this controller to realize that this needs to be nerfed in some capacity.



Initially, the B0XX had five wavedash angles. Just like the 16.8° airdodge, the 73.2° wavedash was very unrealistic (albeit not nearly as competitively advantageous). The lower and upper limits clearly needed to be kept healthy distances away from the perfect angles.



Following this realization, there wasn't a need for more than three wavedash angles. Choosing wavedash distances is extremely unintuitive past the point of "short"/"medium"/"long," anyway, so this worked out well. These wavedashes were clearly around the right lengths, but I eventually wanted to settle on angles that weren't just benchmark numbers. 30.1° and 59.9° had been chosen because they were closest to 30° and 60°; the final product needed a better explanation than that.



This would prove to be the most complex nerf of all, but I was eventually able to find **non-arbitrary wavedash angles** within the game. Coincidentally, these were only a coordinate off of the ones I started with. They also mirrored each other, making for the most elegant outcome I could've hoped for. The coordinates for these angles are:

X +/-.8500 Y +/-.5000 (30.5°)*
X +/-.5000 Y +/-.8500 (59.5°)*

To learn about what makes these coordinates significant, refer to Chapter 6.

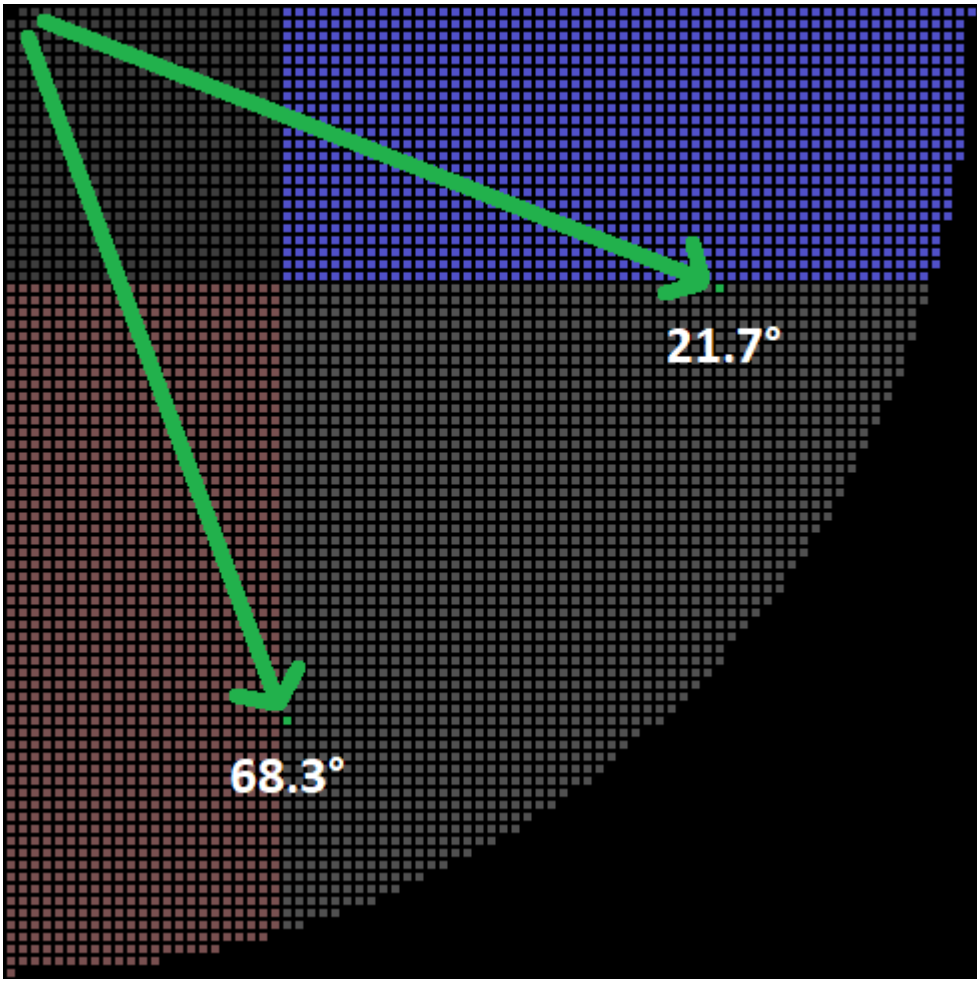
I would also end up finding a reason to use X +/-.6500 Y +/-.6500 as the coordinates for the 45° wavedash (instead of X +/-.7000 Y +/-.7000). This will be revealed in Section 8.3.2.

*These airdodge coordinates are only used in quadrants 3 and 4. Airdodges that take place in quadrants 1 and 2 are adjusted slightly (see Section 5.2.9).

[5.2.8] Firefox

Disclaimer: The B0XX is based on a stock Gamecube controller, not one that has Firefox notches.

As with airdodging, the Gamecube controller incurs risk in approaching 16.8° and 73.2° when angling certain up-B's (such as Firefox). The key difference between these two techniques is that an airdodge takes off on frame 1, whereas up-B's tend to take off after a generous period of time. Fox's up-B, for example, gives the player 42 frames to aim. This makes it significantly easier to perform shallow/steep up-B angles than shallow/steep airdodge angles.



Furthermore, an up-B nerf is inherently extreme, since your opponent *knows* that you cannot recover at certain angles. For these reasons, this nerf is kept on the lighter end. The coordinates for the B0XX's shallowest/steepest up-B angles are:

X +/-.7375 Y +/-.2875 (21.7°)
X +/-.2875 Y +/-.7375 (68.3°)

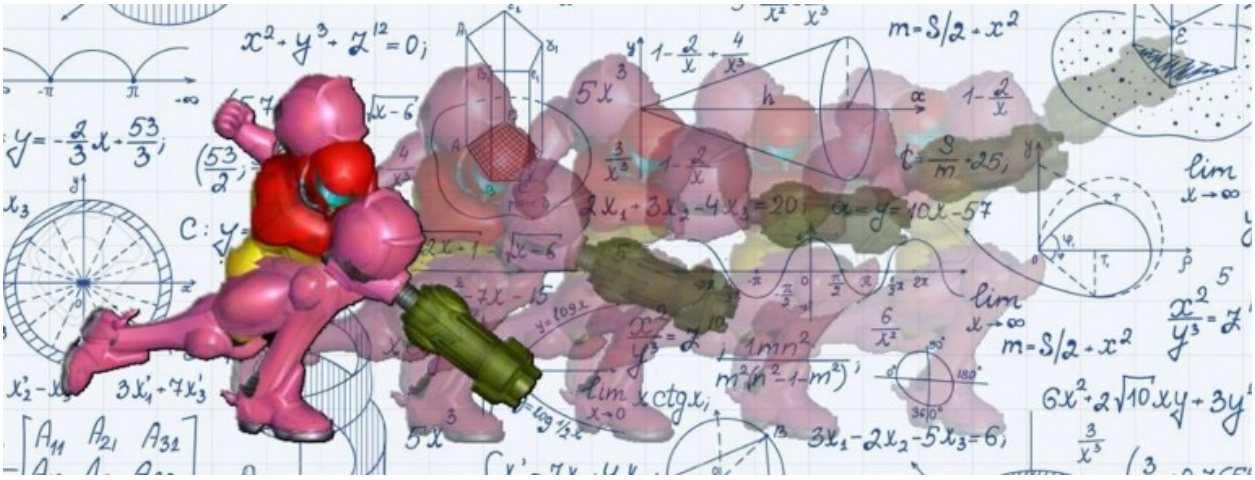
These coordinates aren't necessarily significant in-game, however, **they are coordinates that the B0XX already contains.** Since these coordinates produce angles that are appropriate for an up-B nerf (4.5° from perfect), it is efficient to endorse them as the shallowest/steepest up-B angles.

The significance of these coordinates will be revealed in Chapter 7.

[5.2.9] Other

There are a few cases of techniques requiring incredible precision that don't fall under any of the aforementioned categories. All of the coordinates listed in this section are illegal for the B0XX to have.

Middle UF/DF-Tilt & Middle UF/DF-Smash



One of the most heated debates during Melee's design phase was how the deadzone ought to be shaped and sized. There are remnants

on the game disc suggesting that circular deadzones, as well as smaller deadzones were considered at some point. When these ultimately didn't make the cut, a technique that was shafted in the process was the ability to F-tilt/F-smash at two more angles.

These less well-known F-tilt/F-smash angles still made it to the final version, seemingly as an Easter egg. In each quadrant, a *single* set of coordinates will let you perform them. These coordinates are X +/-.9500 Y +/-.2875* (and C X +/-.9500 Y +/-.2875), the ones that hug the X-deadzone corridor hardest.

Pikachu & Pichu's Double Upwards Up-B

The equation for Pikachu and Pichu's up-B pulses is ridden with loopholes. While there's no need to go into detail, the takeaway is that there are exploitative coordinates that allow you to up-B directly upwards twice. These are:

X +/-.5000 Y 0
X 0 Y -.5000
X +/-.4000 Y +/-.3000
X +/-.3000 Y -.4000

Peach's Ledgedash

Widely perceived as one of the least threatening characters from the ledge, Peach happens to have an unconventional "ledgedash" that can grant her up to *nine* frames of actionable intangibility. Unfortunately, this technique is nearly impossible to perform. First, Peach must manipulate her ECB (environmental collision box) by up-B'ing to the ledge. Then, she must wait out the 8 unactionable frames that occur upon grabbing the ledge and fall on either frame 9 or 10. This fall can last for either 1 or 2

frames, however, if Peach falls for 2 frames, she must not fastfall on frame 2. Then, Peach must jump on frame 2 or 3 (to manipulate her ECB once again) and airdodge *upwards* on frame 3 or 4. This airdodge must take place at one of the following coordinates along the rim:

Frame 3 Airdodge:

X +/-.5375 Y .8250 (56.9°)
X +/-.5375 Y .8375 (57.3°)
X +/-.5250 Y .8375 (57.9°)
X +/-.5250 Y .8500 (58.3°)
X +/-.5125 Y .8500 (58.9°)
X +/-.5000 Y .8500 (59.5°)
X +/-.5000 Y .8625 (59.9°)

Frame 4 Airdodge:

X +/-.5125 Y .8500 (58.9°)
X +/-.5000 Y .8500 (59.5°)
X +/-.5000 Y .8625 (59.9°)
X +/-.4875 Y .8625 (60.5°)
X +/-.4750 Y .8625 (61.2°)

If all of these instructions are followed, Peach will collide with the stage on either frame 19 or 20 of this sequence. She will then gain actionability on either frame 29 or 30, which will last 9 or 8 frames respectively.

Of the three nerfed areas of the game within Section 5.2.9, Peach's ledgedash plays the biggest role in the B0XX's design rationale. Since X +/-.5000 Y .8500 (59.5°) (the airdodge coordinates that would have been used in quadrants 1 and 2) are now banned, the B0XX is forced to choose between X +/-.5500 Y .8250 (56.3°) and X +/-.4750 Y .8750 (61.5°) in order to keep out of Peach's ledgedash's range. Given that 56.3° is an unreasonably weak airdodge angle, **I opted for 61.5° (and its counterpart, 28.5°) within quadrants 1 and 2.**

[5.3] Summary

Smash DI
Only 2 smash DI inputs can be performed during a hitlag window.

Pivot Tilts
Pivot U-tilt is removed / Pivot D-tilt (fast method) is removed.

Notch Integrity
Pivot UF-tilt is removed / Pivot DF-tilt (fast method) is removed.

Dash Back Out of Crouch
The risk of a failed dash back out of crouch is incurred.

Y-Tilt + > 50°
Y-tilt + > 50° cannot be pinpointed.

Neutral-B Integrity
The risk of an accidental side-B is incurred.

Shield Drop Down
Shield drop down (fast method) is removed.

Lightshield
Only analog L/R 49 can be pinpointed.

Ambiguous DI
There are no ambiguous DI mixups.

Ice Climbers Desyncs
There are no Popo/Nana desync coordinates.

Wavedash
The shallowest/steepest airdodge angles are each kept 13.6° from the respective perfect angles in quadrants 3 and 4. 30.5° and 59.5° are non-arbitrary airdodge angles that will be discussed in Chapter 6.

Firefox

The shallowest/steepest Firefox angles are each kept 4.5° from the respective perfect angles. 21.3° and 68.7° are non-arbitrary Firefox angles that will be discussed in Chapter 7.

Other

Various obscure coordinates are illegal. Most notably, the shallowest/steepest airdodge angles are each kept 11.7° from the respective perfect angles in quadrants 1 and 2 in order to ban Peach’s ledgedash.

****

[6] Wavedash Mechanics

****

From the beginning, 45° was the only airdodge angle set in stone. With the shallowest/steepest airdodges angles in the game (as well as anything near them) out of the question, I had two left to find. These needed to be in the vicinities of 30° and 60°.

Given how subjective the act of choosing these airdodge angles might seem, yet how directly they correlate with the B0XX's strength, I felt the need to study this area of the game thoroughly. I kept my hopes up that the most important coordinates on the entire controller would have some sort of justification behind them. If not, the B0XX would forever remain arbitrarily designed.

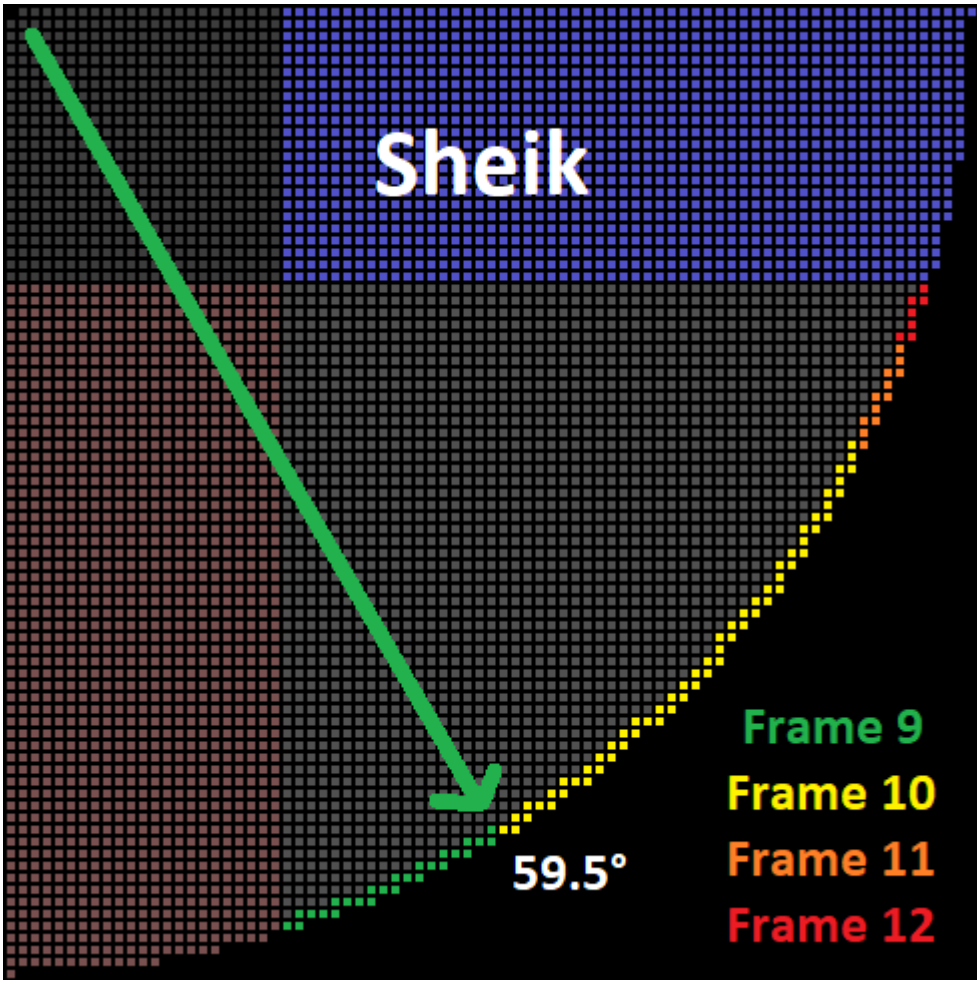
[6.1] The Ledge

My first instinct was to investigate the high tier characters' (Fox/Falco/Sheik/Ice Climbers/Pikachu/Luigi) ledgedashes. This isn't as obvious with an analog stick, but digital inputs make it easy to tell that airdodge angles have an impact on ledgedash frame data.

As it turned out, I had a very shallow understanding of the ledge all along. It is a mechanic with many subtleties I wasn't formerly aware of.

[6.1.1] Intangibility Thresholds

I first learned that, when ledgedashing, **steeper airdodges cause you to make contact with the floor faster. This means that intangibility is inversely correlated with distance.** Despite this being overlooked by the scene at large, it is often better to prioritize the former.



The frame Sheik will land on when she falls on frame 1, jumps on frame 2, then airdodges at various angles on frame 9.

Sheik is unique in that her most intangible ledgedash requires an *extremely* steep airdodge: => 59.5°. This allows her to maximize her actionable frames by landing on frame 9 (if Sheik airdodges at an angle shallower than 59.5°, she cannot land until frame 10 or later).

X +/-.5000 Y -.8500 (59.5°) are the last valid coordinates along the rim for Sheik's frame 9 ledgedash. I figured Sheik ought to be able to land on frame 9 with the B0XX, so I went with these coordinates. Even though this is a bit of a TAS airdodge angle for Sheik, distance isn't important in this particular case. The purpose of her frame 9 ledgedash is intangibility, and nothing else.

Intangibility thresholds exist for all characters who can ledgedash, and are worth being aware of. Finding the B0XX's final airdodge angle, however, would require yet another discovery.

[6.1.2] Ledge Elevation

A little-known fact about the stages in Melee is that ledgedashing is not uniform across them. This is because **the elevation of the stage relative to the ledge varies.** For whatever reason, the game developers made this decision. They also went on to reuse certain ledge positionings.

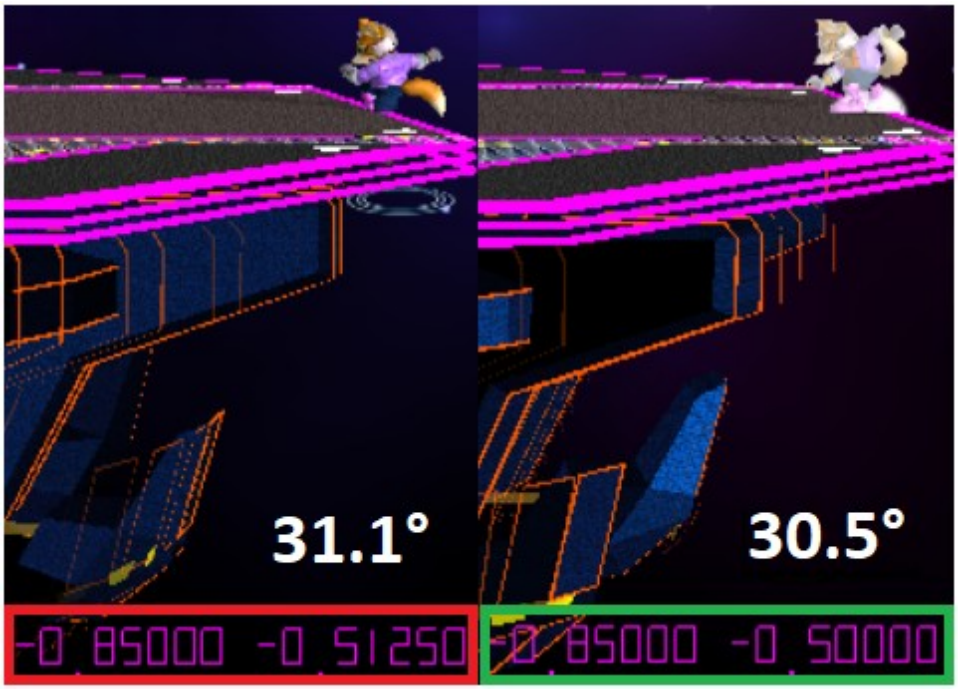
Easier Stages (Ledge is Higher):

Harder Stages (Ledge is Lower):

Conveniently, half of the legal stages share an easier difficulty (BF, DL64, and FoD have a higher ledge), while the other half share a harder difficulty (FD, YS, and PS have a lower ledge). By difficulty, I quite literally mean that certain ledgedash motions will succeed on the easier stages and SD on the harder ones.

[6.1.3] Airdodge Angle

As far as the criteria that determine whether a ledgedash succeeds or not goes, there are two components. **The first is the airdodge angle, which (usually) must be shallow enough to rise above the stage.** With this in mind, Fox's frame data led me to the final set of airdodge coordinates.



Fox's frame 6 ledgedash consists of falling (without fastfalling) for 2 frames, jumping on frame 3, then airdodging on frame 6. This ledgedash requires a shallower airdodge angle than usual to succeed. On the easier stages, Fox needs a <= 34.7° airdodge to make it onstage. On the harder stages, he needs <= 30.5°.

Due to certain intricacies of the B0XX's ledgedash methods, it is mandatory that Fox is able to ledgedash on frame 6 on the harder stages with this controller (it is overly difficult to ledgedash with Fox otherwise). These intricacies will be explained in full in Section 11.1.1, although you'll probably figure out what they are by the end of this chapter. For the time being, this means Fox needs access to **X +/-.8500 Y -.5000 (30.5°),** the first angle along the rim that enables this ledgedash.

[6.1.4] Jump Trajectory

The other component that the success of a ledgedash hinges on is the horizontal trajectory of the midair jump. This is calculated on the jump frame (as early as frame 2) of a ledgedash sequence.

To demonstrate this, I will analyze Fox's frame 5 ledgedash. X .7000 aerial drift/jump trajectory, and a 45° airdodge (X +/-.7000 Y -.7000) will be used.

Frame 0:

Fox hangs from the ledge. Along Battlefield's X-axis, he is located at 70.32.

Frame 1:

Fox falls from the ledge. If he falls diagonally, he'll advance a small amount in that direction. In this case, he falls straight down, which doesn't result in any horizontal movement.

Frames 2, 4 & 5 (No Jump Trajectory):

On frame 2, Fox jumps, but he has failed to do so with horizontal trajectory. He remains at 70.32.

On frames 3 and 4, Fox drifts at X -.7000, but this has a negligible impact on his positioning. He only makes it to 70.13.

Then, on frame 5, Fox airdodges at 45°, but this results in a SD.

Frames 2, 4 & 5 (Jump Trajectory):

This time, Fox jumps at X -.7000 trajectory on frame 2. He has already made it to 69.71.

By frame 4, he has made it to 69.31.

Then, on frame 5, the 45° airdodge makes it onstage.

In the first example, Fox sealed his fate on frame 2 when he jumped without any trajectory. At that point, the 45° airdodge couldn't have succeeded. In the second example, Fox jumped with trajectory. This brought him closer to the stage, granting him more leniency on his airdodge angle. All characters' ledgedashes benefit from jump trajectory in this manner.

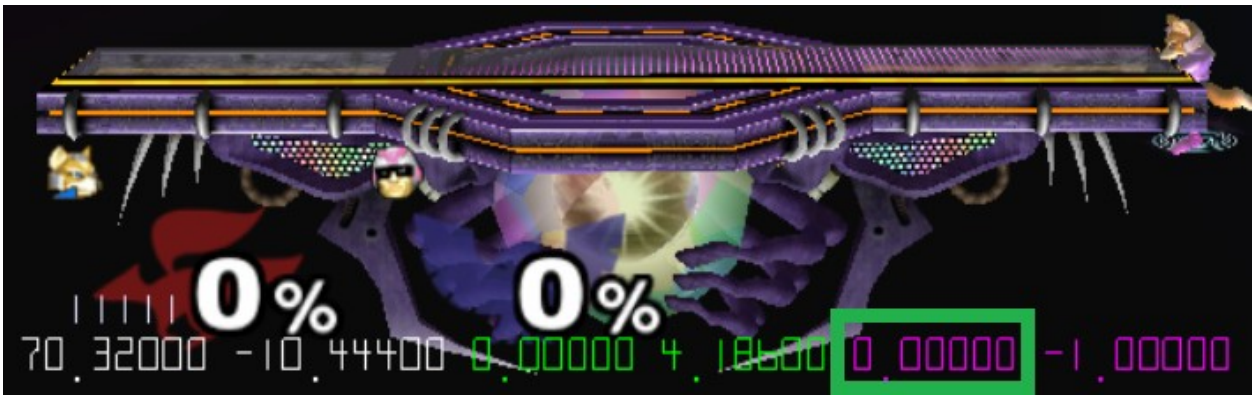
[6.2] Skill System

When designing the B0XX, I had to take all of the subtleties of ledgedashing into consideration. This wasn't just a matter of choosing airdodge angles, but preserving the Gamecube controller's intrinsic challenges. So far, I have yet to show parallels for some of these. For example, the stage a Gamecube controller user is playing on should influence the airdodge angle they attempt, as certain angles won't work on every stage. Similarly, the all-important role of jump trajectory on the B0XX has neither been demonstrated, nor explored in full. Rest assured, these subtleties are all intact.

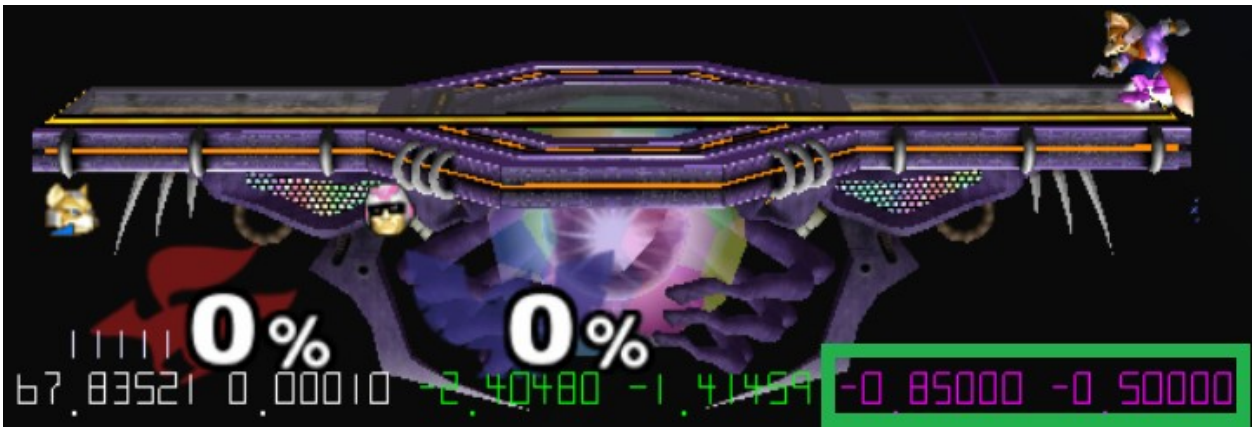
[6.2.1] Difficulty

On the B0XX, the relationship between stage difficulty (ledge elevation) and jump trajectory is preserved across the cast. To demonstrate this, I'll once again analyze Fox's frame 5 ledgedash.

Frames 2 & 5 (Easier Stage / No Jump Trajectory)

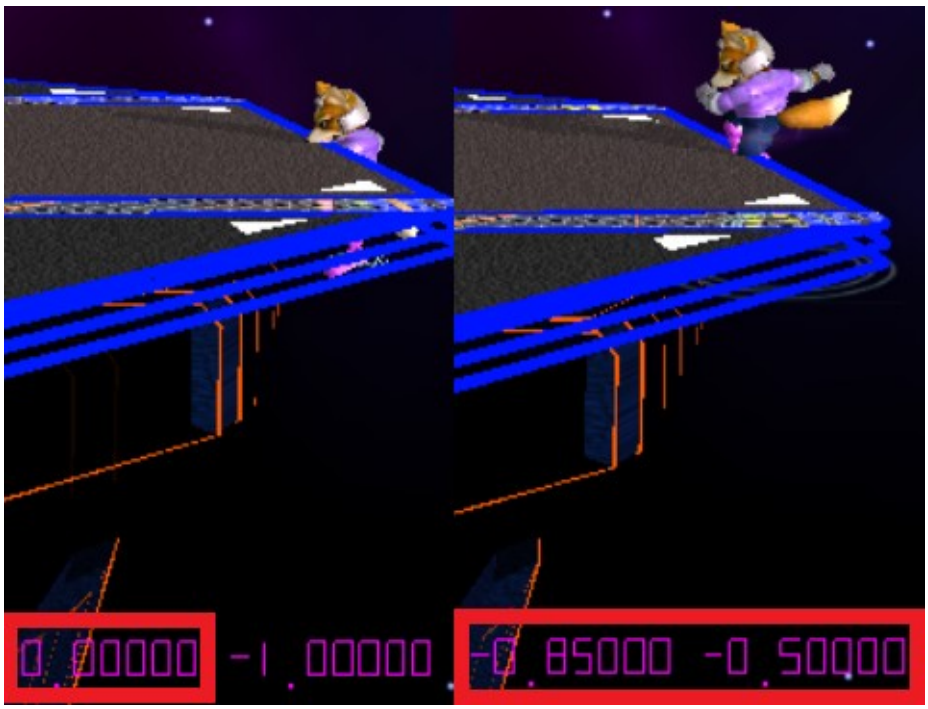


On the B0XX, it is fine for Fox to jump without any trajectory (on frame 2) on the easier stages.

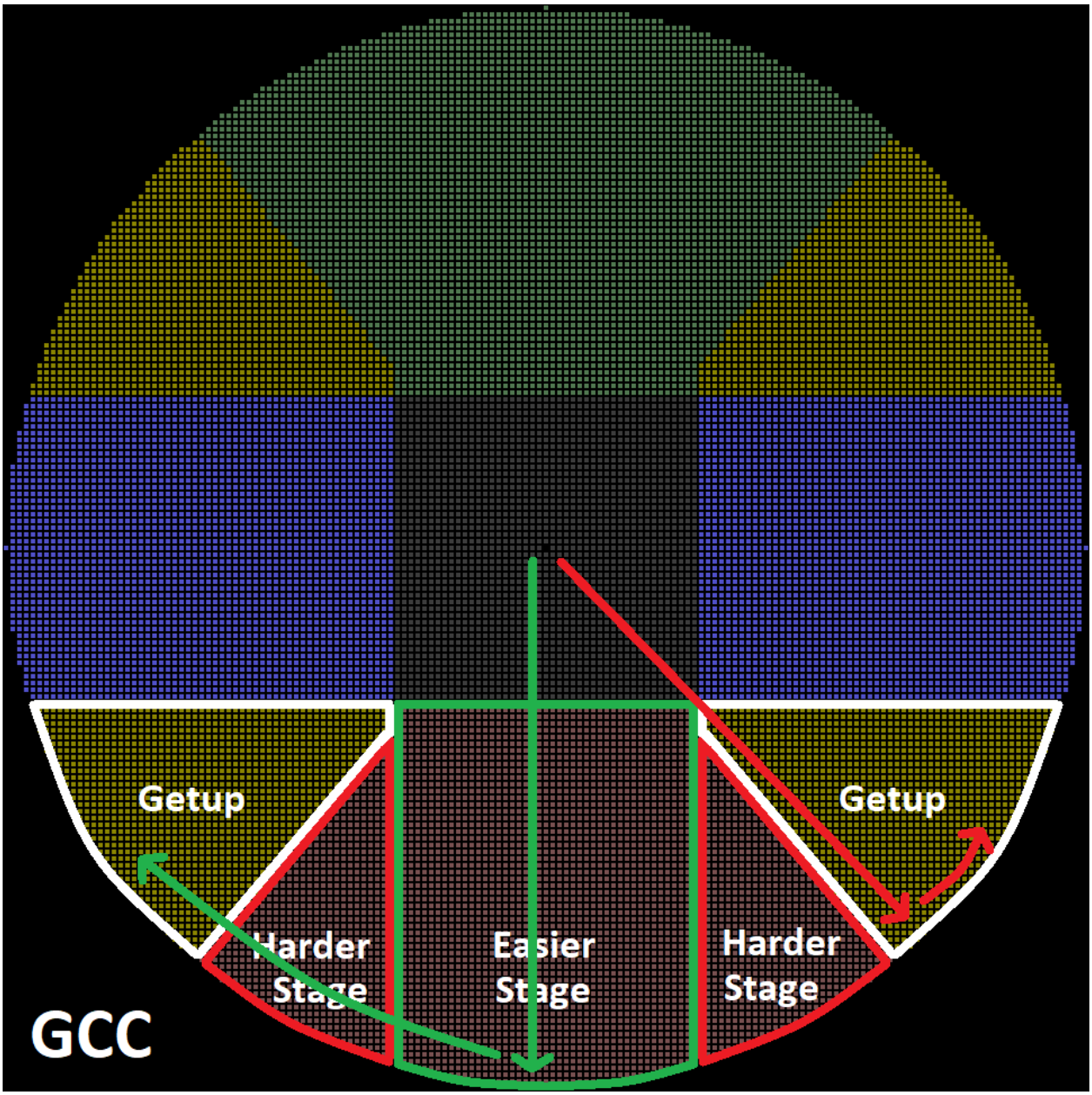


He'll need a single frame of horizontal drift (can be done as late as frame 4), but he'll still be able to make it onstage with the 30.5° airdodge.

Frames 2 & 5 (Harder Stage / No Jump Trajectory)



On the harder stages, this isn't the case. If Fox fails to jump with trajectory, no matter how well he drifts from that point on, he will die if he attempts a ledgedash on frame 5. For anyone curious, a <= 28.8° airdodge would have been needed for Fox to survive from this position.



Harder stages increase the risk of an unintentional ledge get-up.

In summary: a Gamecube controller user can afford to steer clear of ledge get-up (< 50° territory) on the easier stages because jump trajectory isn't as necessary. On the harder stages, however, they should look to jump with trajectory in order to reduce the shallowness needed on their airdodge. But in doing so, the risk of an unintentional ledge get-up becomes very real.

On the B0XX, this challenge is preserved. The player must avoid pressing Forward on the fall frame of a ledgedash*, otherwise, they too will receive a ledge get-up. **To perform your character's most intangible ledgedash with jump trajectory, you will have to one-frame link the Forward press after falling** (either frame 1 Down, frame 2 Down-Forward or frame 1 Back, frame 2 Forward).

*This one-frame link is only needed for < 50° ledgedashes. When performing > 50° ledgedashes, you can ledgefall diagonally (see Section 8.1.3).

[6.2.2] Distance

Not all ledgedashes that occur on the same frame are created equal. Regardless of stage, there is always incentive to perform a more skillful motion and jump with trajectory.

Fox's Frame 5 Ledgedash (No Jump Trajectory):

If Fox jumps without trajectory, then airdodges at 30.5°, he only makes it to 44.46 along Battlefield's X-axis.

Fox's Frame 5 Ledgedash (Jump Trajectory):

If Fox jumps with X 1.0 trajectory, he makes it to 43.25. This is the best ledgedash the B0XX is capable of.

This extra 1.2 units along the X-axis equates to airdodging at a ~2° shallower angle, which, believe it or not, is meaningful. If you take a close look at Fox's feet, you'll be able to tell that jumping with trajectory caused him to travel further. This is a prime example of why the B0XX's wavedash mechanics had to be fine-tuned so carefully. Seemingly small details like these can make a difference.

Due to the variable that is jump trajectory, ledgedashing on the B0XX is by no means static. Skill is always rewarded with longer ledgedashes, and in some cases, more intangible ones.

[6.3] Traction Anomaly

There is one final aspect of Melee's physics engine that is worth mentioning.

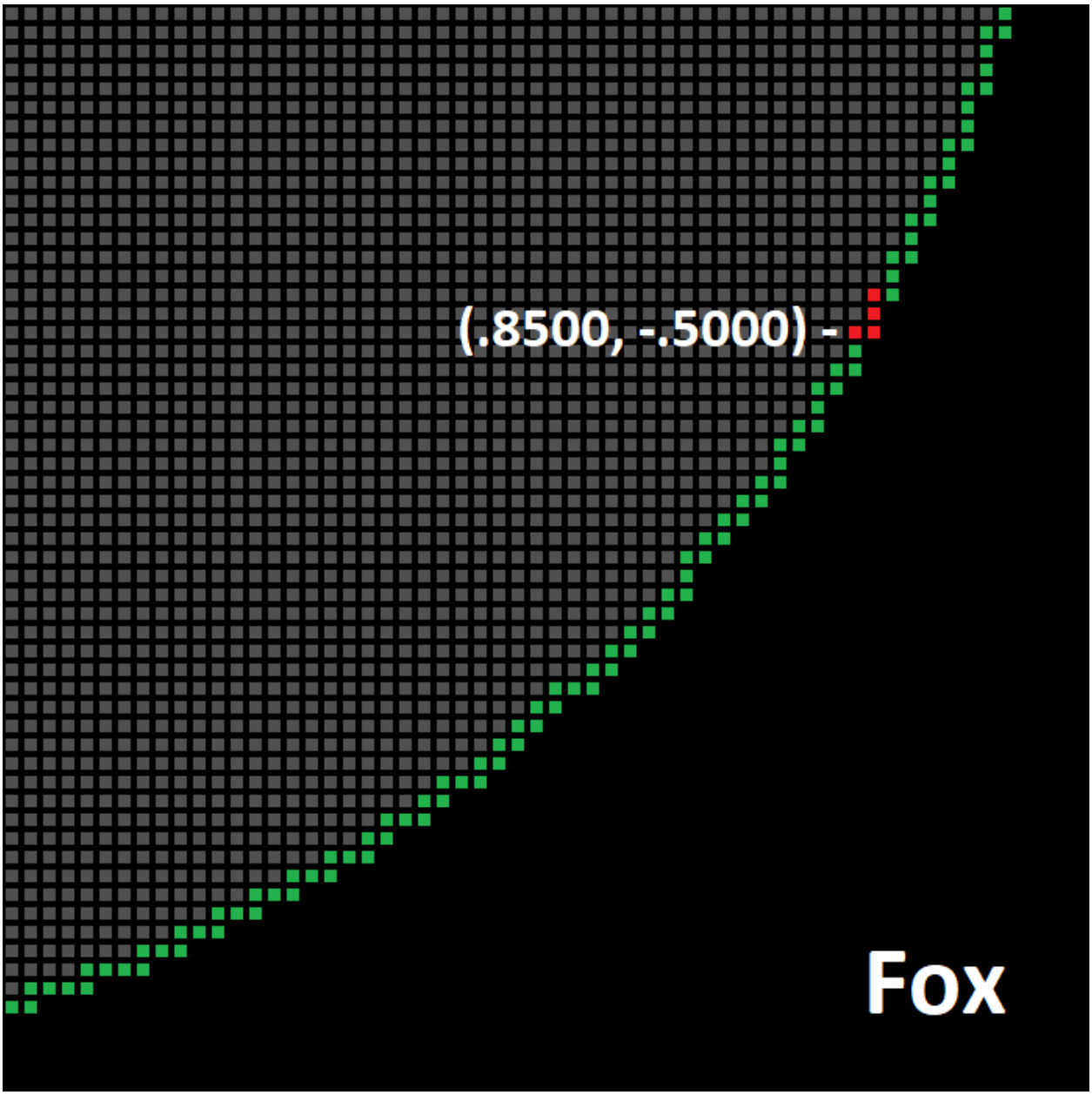
Conventional wisdom has it that the shallower the airdodge, the further the wavedash will travel. This is not universally true. When a character's velocity exceeds their maximum walk speed (a seemingly unrelated stat), their traction is doubled, reducing their effectiveness at sliding across the floor. **For most characters, this creates a breakpoint where certain angles aren't shallow enough to make up for doubling the character's traction, resulting in a net loss in distance traveled.** It is appropriate that I use Fox to demonstrate this anomaly, since the airdodge coordinates I've given the B0XX cause him to be impeded by it.

Fox's maximum walk speed is 1.6 units/frame. His base traction is .08. This means that if Fox's velocity ever exceeds 1.6 units/frame, his traction will double to .16.

This is how far some of the shallower angles cause Fox to travel:

X +/-.8375 Y -.5375 (32.7°) = 25.6 units
X +/-.8375 Y -.5250 (32.1°) = 26.3 units
X +/-.8500 Y -.5250 (31.7°) = 26.6 units
X +/-.8500 Y -.5125 (31.1°) = 26.9 units
X +/-.8500 Y -.5000 (30.5°) = 25.8 units*
X +/-.8625 Y -.5000 (30.1°) = 26.0 units*
X +/-.8625 Y -.4875 (29.5°) = 26.4 units*
X +/-.8625 Y -.4750 (28.8°) = 26.8 units*
X +/-.8750 Y -.4750 (28.5°) = 27.0 units

*Despite being shallower, these angles are actually *worse* than the ones directly before them. This is because they cause Fox's velocity to exceed 1.6 units/frame, but not by enough to be worth it. Here is a diagram of these gimped angles:



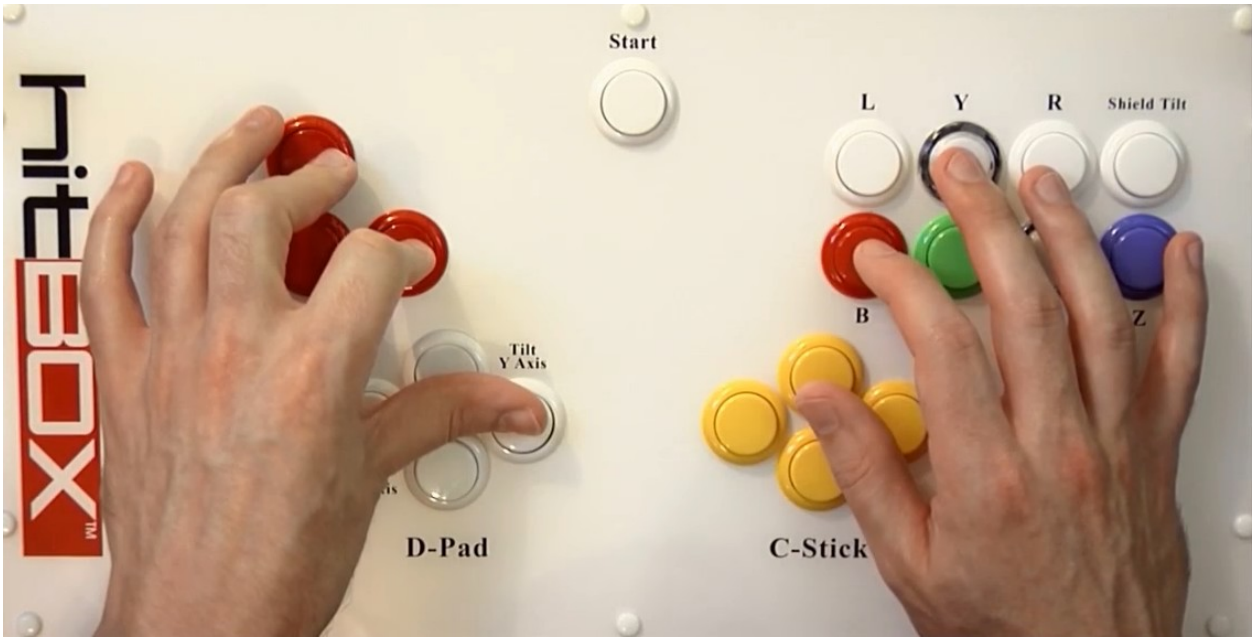
Interestingly, the first airdodge angle along the rim that enables Fox's frame 6 ledgedash on the harder stages (X +/-.8500 Y -.5000) is also the first one that doubles Fox's traction. This means **Fox's wavedash on the B0XX is a bit weaker than it would seem (since the B0XX uses these coordinates)**. Whereas the airdodge angle itself is 13.6° shy of perfect, Fox's wavedash is effectively 15° to 16° shy, since airdodges ~2° steeper than this one cause him to travel further.

****

[7] Modifier Buttons

****

Modifier buttons are the glue that hold the B0XX together. They are what make it viable to play Melee with a completely digital controller. That being said, modifier buttons take up valuable real estate, which is why one of my highest priorities was needing as few of them as possible.

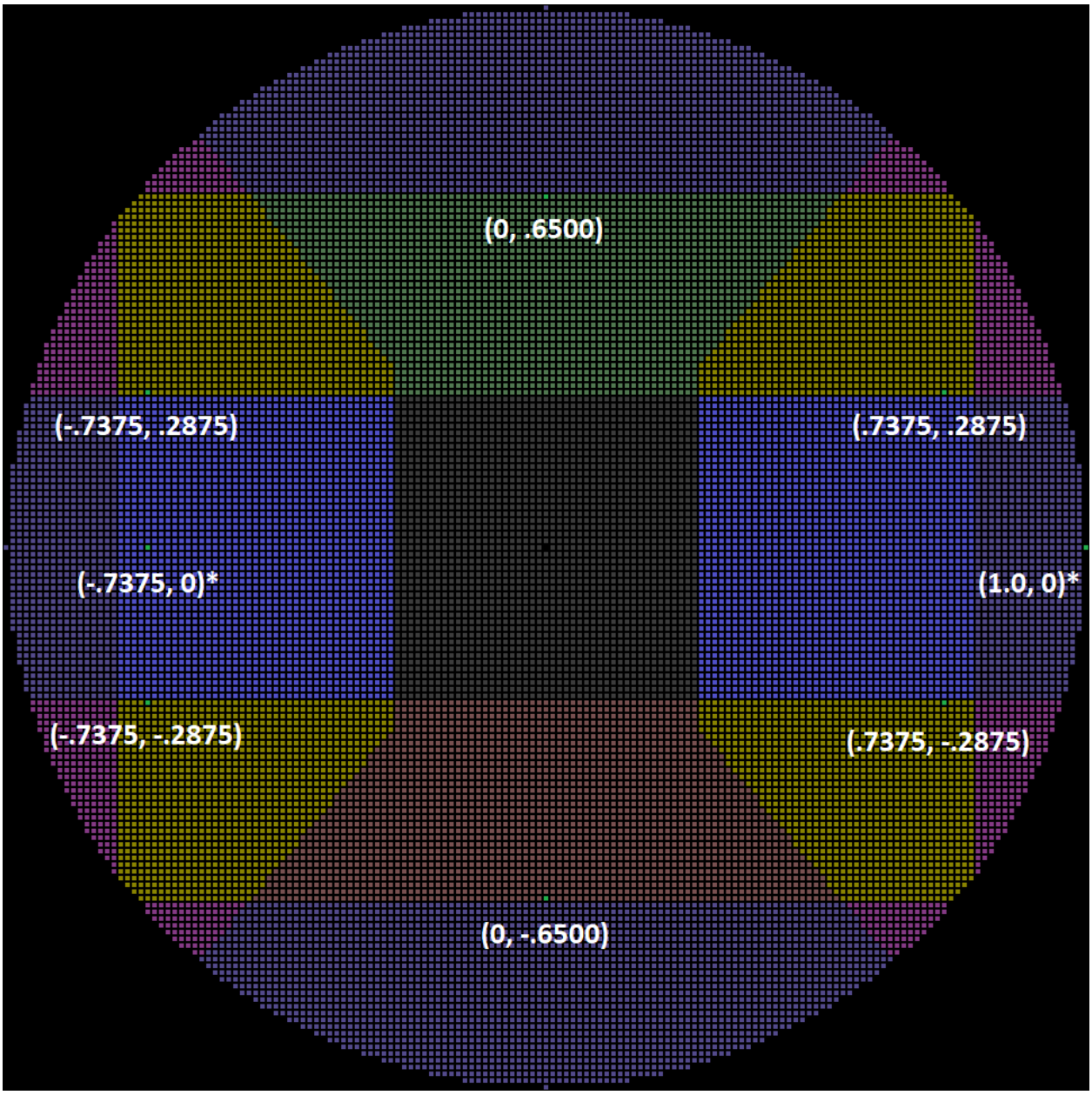


4 modifier buttons (and a WASD arrangement) is a no-go.

The downfall of most digital Melee controllers is the inclusion of Modifiers 3 and 4. Though these might seem necessary at first, the problems they cause are just as readily apparent. Like poison to the controller, Modifiers 3 and 4 push other buttons out of good locations only to station themselves in lackluster ones. Then, they force you to memorize several new button combinations, exacerbating an already uninviting lack of intuition.

From early on, I knew that **the B0XX had to be kept to two modifier buttons**. Fortunately, two is all it needs. The secret is to fill each of them to the brim with functionality.

[7.1] Modifier 1



When Modifier 1 is pressed in conjunction with the cardinal directions, you will receive:

Horizontals: X +/-.7375*
Verticals: Y +/-.6500
Quadrants: X +/-.7375 Y +/-.2875

On the B0XX, both modifier buttons enable slight presses of all 4 cardinal directions as well as all 4 quadrants. This is their most basic function.

Along the X-axis, Modifier 1 will produce X .7375. While X-tilt spans up until X .7875, X .7375 is the upper extremity a digital controller should use. This is because **X .7375 is the greatest X-value that does not break teeter, the mechanic that protects your character from falling when they walk near the end of the stage.** X-values that do not exceed .7375 are therefore most efficient, since they not only let you slight DI when thrown, but also safely walk and perform tilts.

It is worth mentioning that X .7375 isn't ideal for intentionally teetering (to set up a teeter-drop, for example). The best way to do that is to initiate walk (X-tilt), then immediately release the modifier button to continue walking at X 1.0 (maximum) speed. This allows you reach the end of the stage as fast as possible.

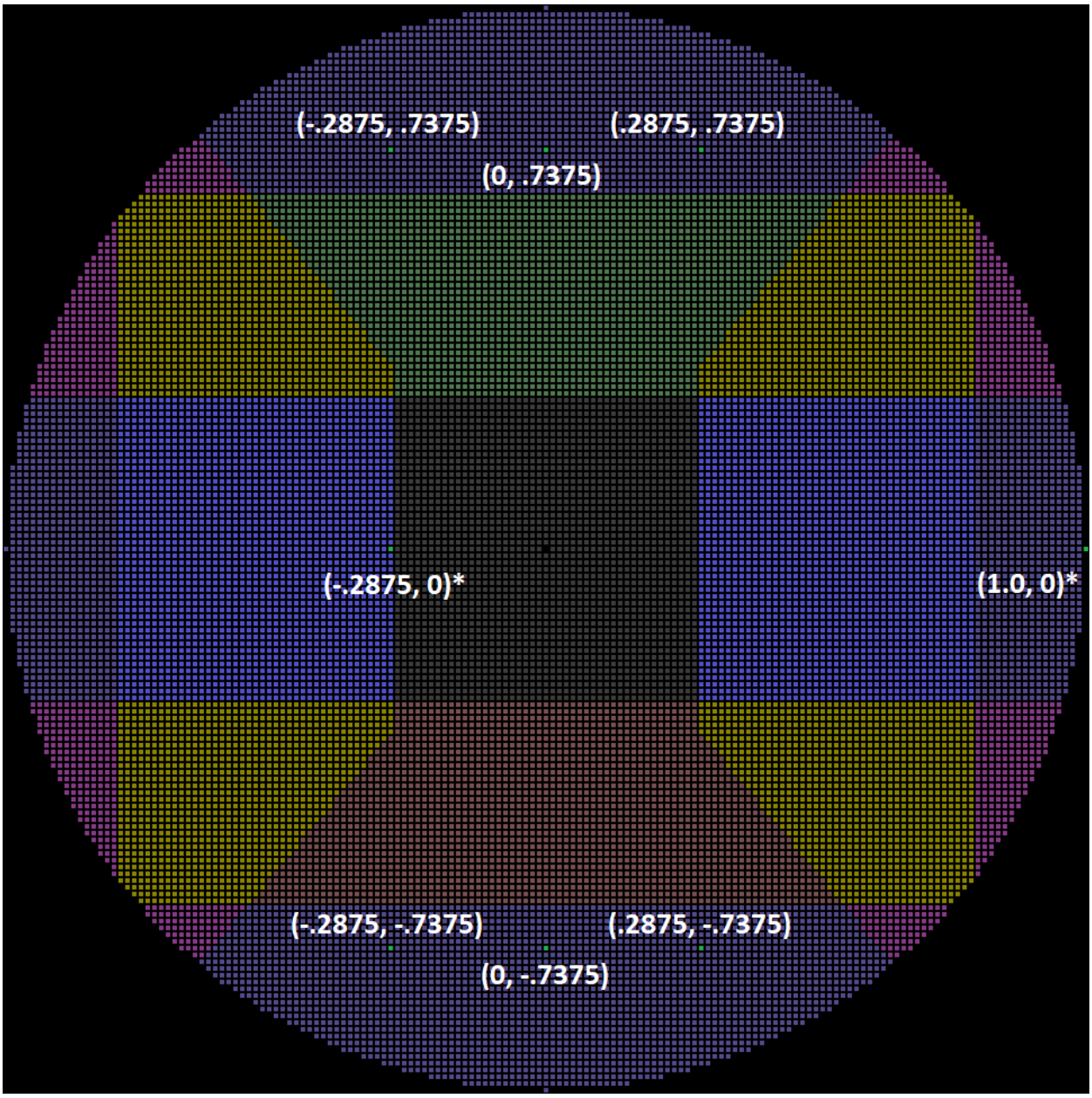
*Modifier 1 won't always cause the horizontals to be modified to X .7375 (see Section 7.3).

Along the Y-axis, the only worthwhile modification is Y .6500. The lower extremity (Y .2875) is obsolete in comparison, since it causes you to stand up from crouch (Y => -.6125), performs neutral-B (|Y| <= .5375) instead of vertical-B, etc.

Within the quadrants, Modifier 1 will produce X .7375 Y .2875. X is kept consistent with the horizontals so that alternating the horizontals and quadrants doesn't awkwardly change your movement speed, while Y is given the non-arbitrary value of .2875. These coordinates allow you to point in < 50° territory and perform various techniques, such as UF/DF-tilt and **the shallowest Firefox angle**. In particular, it was imperative for me to assign this side of the 50° line to the modifier button in the best resting position for the user's left thumb. This is because **Modifier 1 enables the 30.5° wavedash, making it the most important modifier button.** I will explain how it does this in Section 8.3.3.

Lastly, **when pressed in conjunction with the L button, Modifier 1 (or Modifier 2) will produce analog L 49.**

[7.2] Modifier 2



When Modifier 2 is pressed in conjunction with the cardinal directions, you will receive:

Horizontals: X +/-.2875*
Verticals: Y +/-.7375
Quadrants: X +/-.2875 Y +/-.7375

Originally intended to mirror Modifier 1, Modifier 2 is a watered down version of what it could have been. Within the quadrants, Modifier 2 is banned from pinpointing Y-tilt + > 50°, and must settle for pinpointing Y-smash + > 50° instead. Still, this is useful for performing turnaround U-tilt in buffered situations (Y <= -.7000 will cause crouch and prevent turnaround D-tilt). In the cardinals, X +/-.2875 usually serves as another slight DI option, while Y +/-.7375 is chosen for the sake of symmetry with Modifier 1 (this allows Modifier 2 to enable **the steepest Firefox angle**).

*Modifier 2 won't always cause the horizontals to be modified to X .2875 (see Section 7.3).

[7.3] Horizontal Modification Conditionals

While this mechanic is neither a macro nor a button bind, it still falls under a separate category. On the B0XX, the 2 modifier buttons don't always modify the horizontals to X .7375/.2875. **Instead, they present the option to modify the horizontals when doing so would be useful (based on the number of horizontals currently held).** Without this mechanic, the modifier buttons' X-axis modification can actually be a hindrance in certain scenarios.

Only the horizontals are subject to this mechanic (the quadrants are exempt). This is also the only instance on the entire controller of an overriden cardinal mattering (if 2 horizontals are held).

1 Horizontal Held

X is modified to .7375/.2875. If only 1 horizontal is held, this will always match the player's intention.

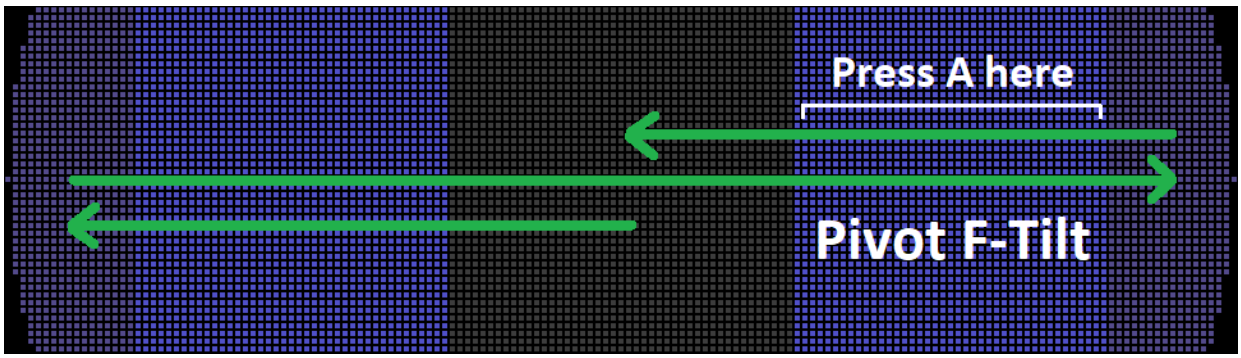
2 Horizontals Held

X is not modified (it remains 1.0). This is because when 1 horizontal is already held, modifying the second horizontal is almost always unwanted.

The best example would be ledgedashing with Modifier 1. Modifier 1 enables the 30.5° wavedash, meaning it's safe to assume the player's intention is to travel as far as the B0XX permits. In this situation, an experienced B0XX user would want to jump with X 1.0 trajectory to gain as much distance as possible, *but Modifier 1's modification to X .7375 would normally interfere.* By making it so that the modifier buttons do not cause an X-axis modification when the second horizontal is pressed, the option to press Back to fall from the ledge, Forward (X 1.0) + jump, then Down-Forward + airdodge is no longer arbitrarily compromised. This is the most difficult (and rewarding) ledgedash motion that can be performed on the B0XX.

Modifier is Pressed When 2 Horizontals are Already Held

As mentioned in Section 2.2.3, **this is the first of two instances of order-dependency on the B0XX.** It is implemented to account for the one scenario in which the player *would* want to modify the X-axis while (already) holding 2 horizontals: pivot F-tilt.



In Section 5.1.2, I showed the best pivot F-tilt method on the Gamecube controller. This method involves flick pivoting, followed by pressing A while the stick is in X-tilt during its return. **Since digital inputs do not traverse X-tilt upon release, they cannot use this method, which means they need an alternative.**

This alternative is dashing twice, followed by modifying the X-axis to X-tilt (valid territory for the pivot as well as the F-tilt). In order to support this sequence without compromising the ledgedash sequence shown a moment ago, this X-axis modification

will only occur if a modifier button is pressed when 2 horizontals are *already* held.

Even without this interaction, the B0XX would still be able to perform pivot F-tilt (since X-tilt would still be producible by holding 1 horizontal). This interaction merely makes it so that you do not have to arbitrarily release the second horizontal in order to do so. **The B0XX should recognize that whenever a modifier button is pressed *after* a horizontal, the player's intention is always to modify the X-axis to X-tilt.**

****

[8] Non-Dedicated Modifiers

****

Squeezing as much functionality as possible into Modifiers 1 and 2 was only the first step. **The real breakthrough that kept the number of modifier buttons on the B0XX to two was allowing A/B/C/L/R/X/Y/Z to possess modifier properties.** I realized this was a necessary feature when I couldn't find ways to accommodate certain coordinates with two modifier buttons alone, but I couldn't crowd the layout of the controller, either. At that point, the solution was to use existing buttons to pinpoint the remaining coordinates the B0XX needed. This removed the need to install additional modifier buttons.

Once I established that non-dedicated modifiers were necessary in some capacity, I quickly realized that **A/B/C/L/R/X/Y/Z's action input had to remain actuated at all times (even while the button served as a non-dedicated modifier).** At first, I experimented with the A/B/C/L/R/X/Y/Z buttons influencing the analog X/Y-coordinates without generating an action input at all (just like modifier buttons). This seemed like the obvious way to accommodate the coordinates the B0XX needed without turning A/B/C/L/R/X/Y/Z into action-direction button binds, but it resulted in several sources of conflict. Most notably, it had become possible to "flub" certain action inputs if the buttons they interacted with were being held. **For example, if Up + Left + Modifier 2 were held and C-Up was pressed with the intention of U-smashing, the B0XX would interpret this as an attempt to angle Firefox and fail to generate a C-Up input.** I dismissed this idea almost as it came for this reason alone.

The need to accommodate niche coordinates without compromising their corresponding action inputs led to my next epiphany: **action-direction button binds didn't have to be harmful.** Modifiers already weren't allowed to generate an Up, Down, Left, or Right input on their own, which took most of the concerns raised by action-direction button binds out of the picture (i.e. U-tilt button, Shine button, etc.). This meant that the remaining concerns laid *within* the cardinals and quadrants themselves. Melee's coordinate plane was bound to have more than just eight

meaningful thresholds (4 cardinals/4 quadrants); however, if I could manage to define the rest of these thresholds and deem it illegal for non-dedicated modifiers to traverse them, then the concerns raised by action-direction button binds would be fully alleviated. **At that point, the B0XX's "action-direction button binds" would merely be buttons that interchangeably served as action inputs or directional inputs, but never both at the same time in a meaningful way.** This would allow the controller to operate on the minimum # of buttons possible, a quality of life improvement that cannot be overstated.

[8.1] Restrictions

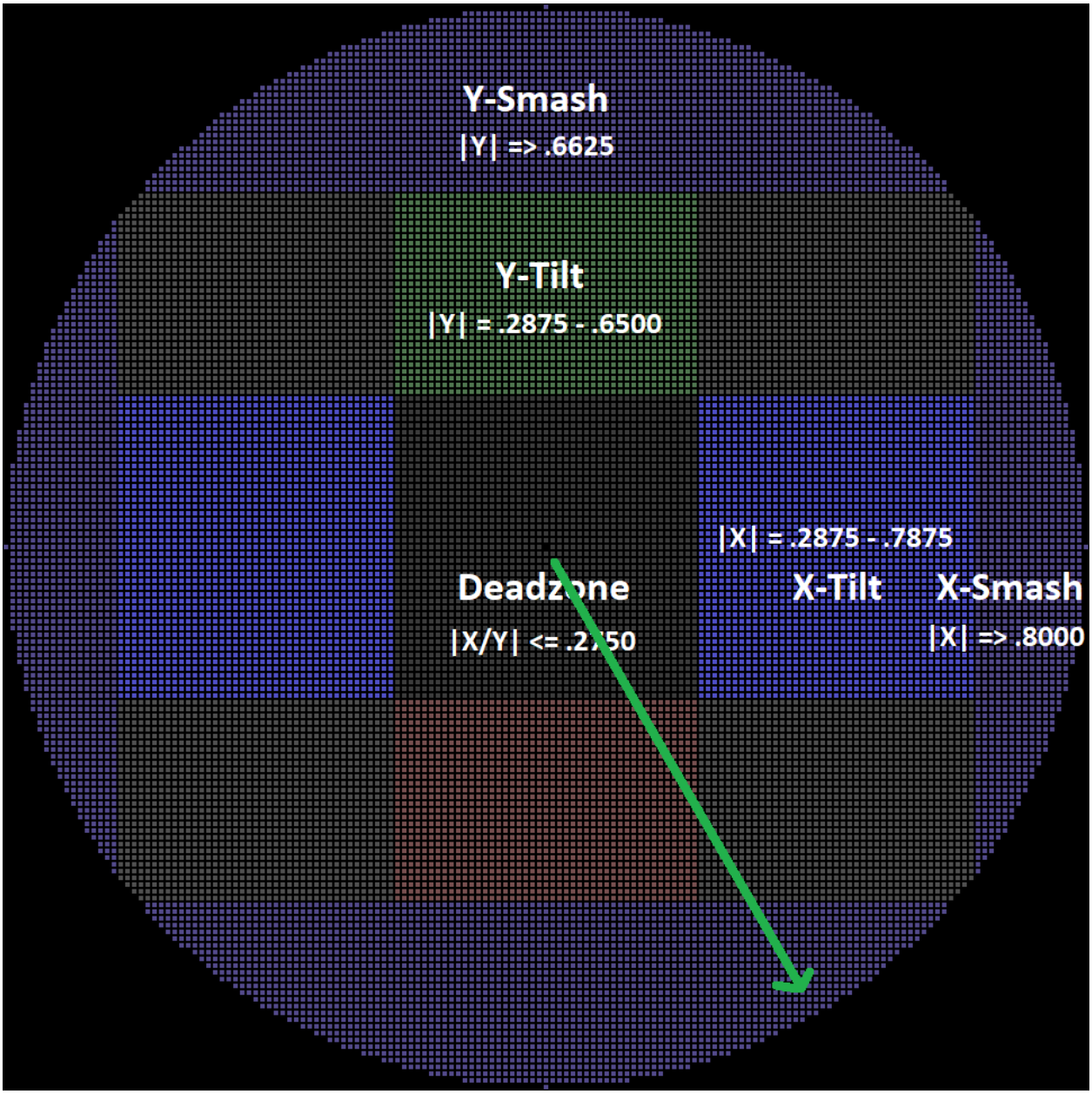
This section will define the thresholds within the cardinals and quadrants that dictate what constitutes meaningful execution, and then deem it illegal for non-dedicated modifiers to traverse them. For the most part, this means paying respect to the zones of the grid I covered in Section 3.2.

[8.1.1] Jump Trajectory Integrity

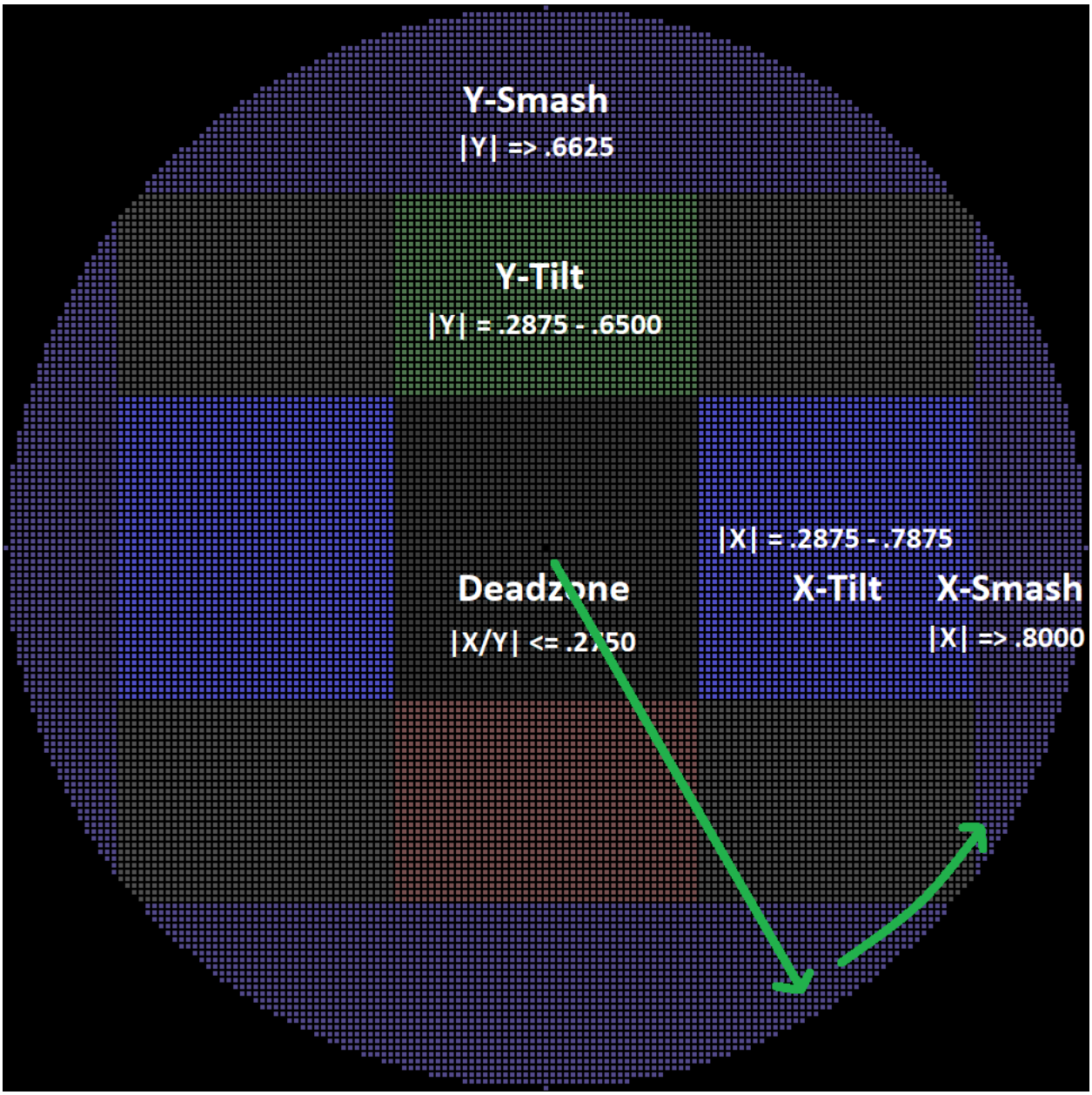
Of the four non-dedicated modifier restrictions, this is the only one that pertains to an aspect of Melee's analog control rather than specific zones of the coordinate plane. Since the X and Y buttons are directly tied to jump trajectory, allowing them to serve as non-dedicated modifiers can only lead to disingenuous behavior (i.e. jumping with a different X-value than your current walk/run speed). For this reason, **it is illegal for the X and Y buttons to serve as non-dedicated modifiers.**

[8.1.2] Tilt/Smash Integrity

X .8000 and Y .6625 serve as the dividing lines for many techniques. As a result, manually entering, then leaving these zones is commonly required. I will use fastfalling (Y <= -.6625), followed by wavelanding with two different airdodge angles to illustrate the importance of the tilt/smash thresholds.



Fastfalling, then wavelanding with airdodge coordinates that contain Y <= -.6625 can be performed with a single stick motion.

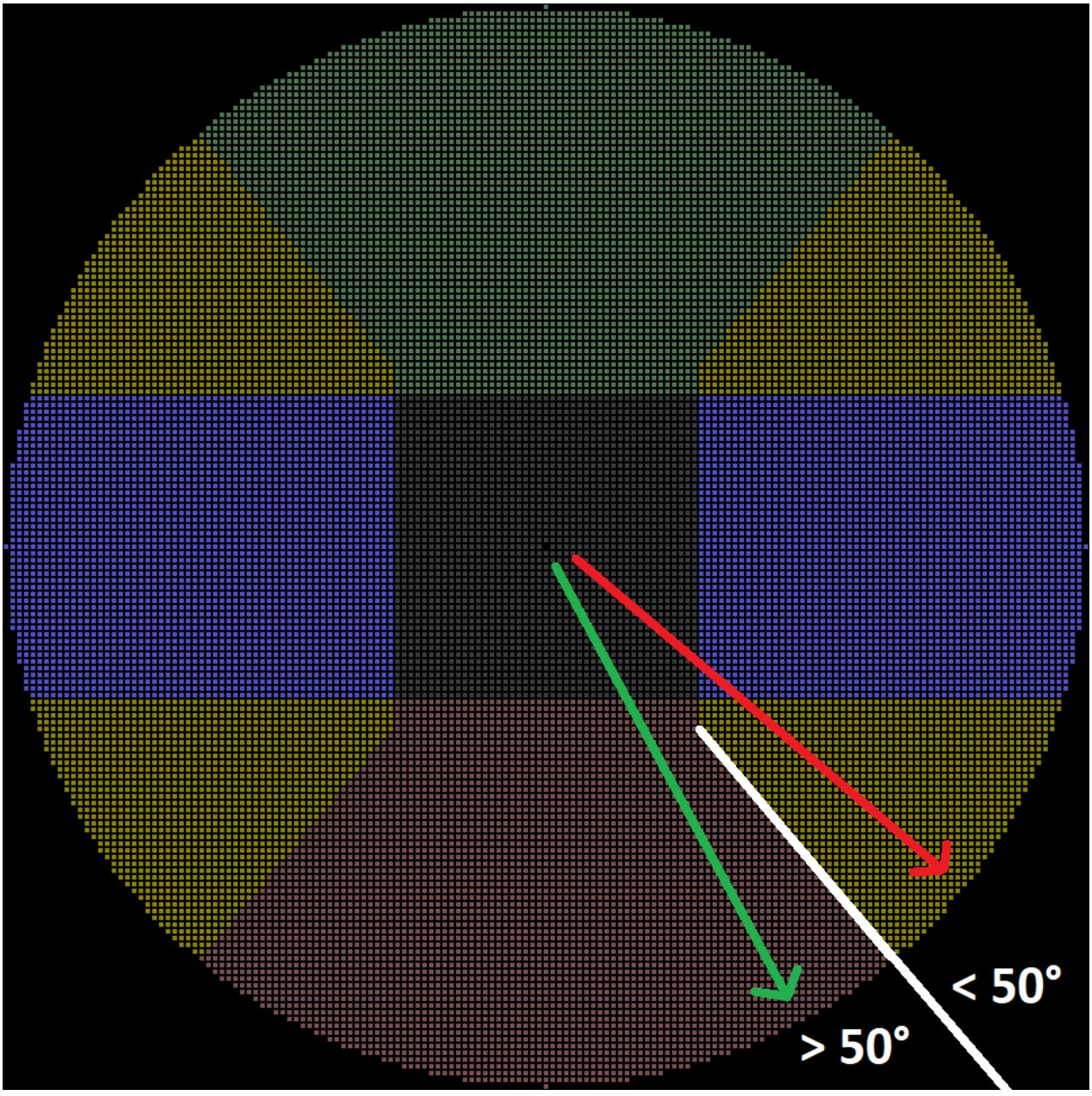


Wavelanding with Y > -.6625, however, requires Y -.6625 to be manually traversed with a second stick motion. This concept applies to various techniques along either axis.

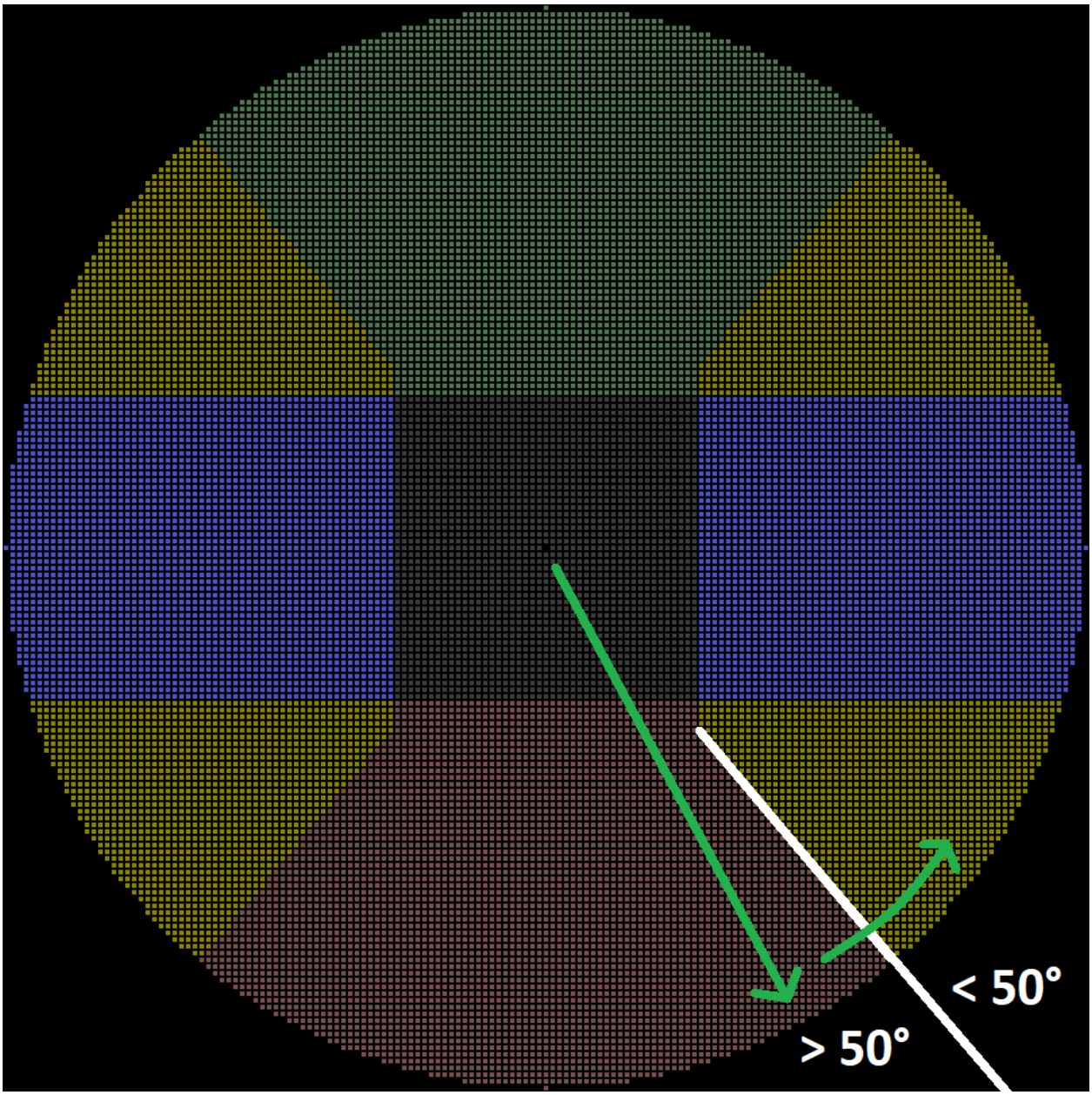
The B0XX must respect these thresholds as well. **A/B/C/L/R/Z may not cause the analog stick to traverse X .8000 or Y .6625** ***in a manner that meaningfully circumvents having to perform a stick motion***. The clause at the end is included because several of the non-dedicated modifiers *can* traverse the tilt/smash thresholds in situations where they are of no relevance (i.e. angling up-B).

[8.1.3] 50° Line Integrity

The division caused by the 50° line is most notable for dictating what constitutes genuine ledgedash behavior.

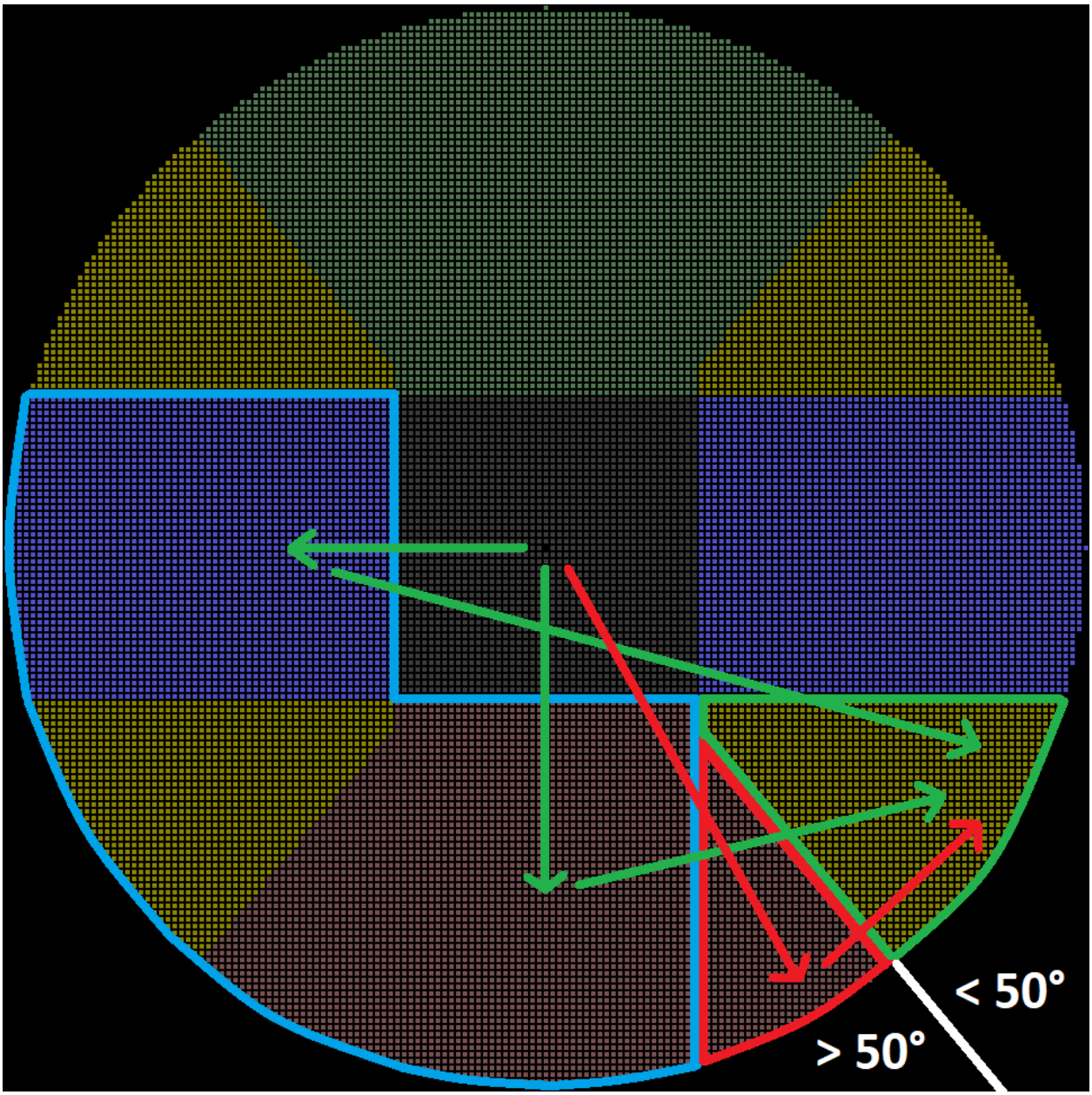


When attempting to ledgefall with the Gamecube controller, < 50° territory must be avoided (as it will cause regular get-up). > 50° territory, however, is valid for a ledgefall. This means that a > 50° ledgedash (ledgefall -> jump -> airdodge at > 50°) can all be performed with a single stick motion.



Ledgedashing at < 50°, however, requires the 50° line to be manually traversed with a second stick motion.

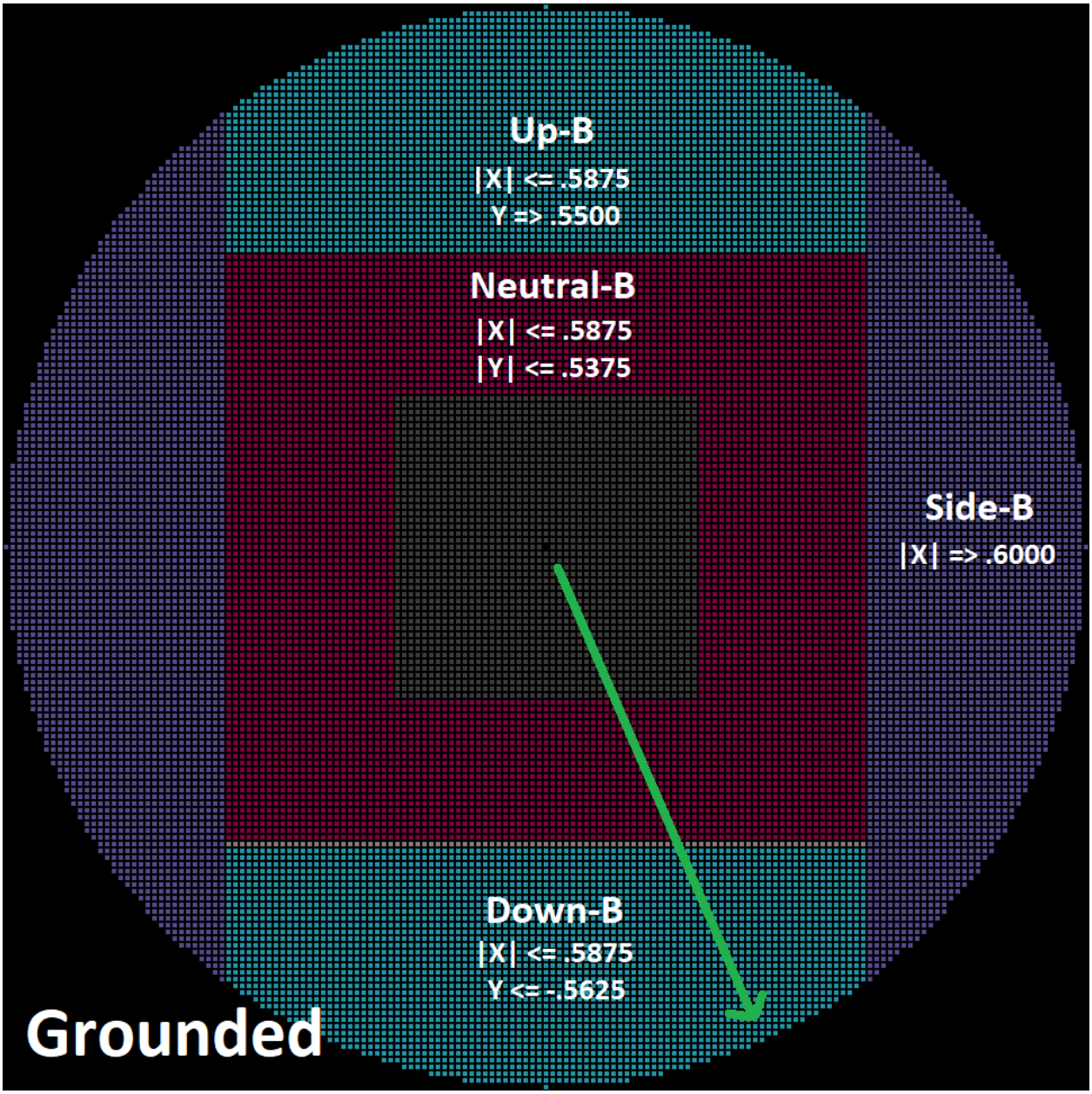
As far as the B0XX goes, this means that pointing in > 50° territory must be paired with actions that are in > 50° territory as well (and vice versa). In other words, **A/B/C/L/R/Z may not cause the analog stick to traverse the 50° line in a manner that meaningfully circumvents having to perform a stick motion.**



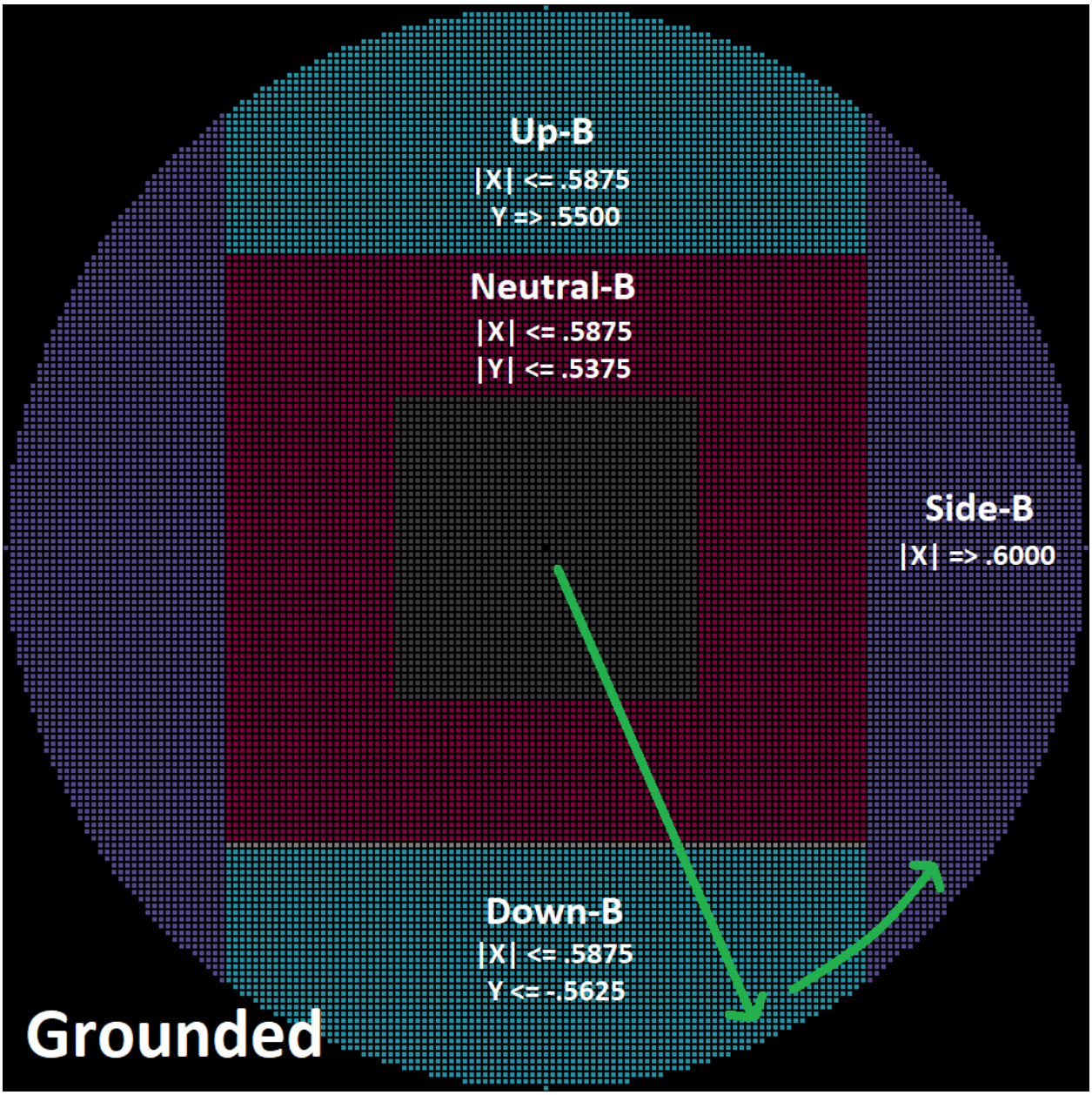
This diagram embodies 50° Line Integrity. The two green sequences are examples of valid ledgedash motions: the player ledgefalls in either west, southwest, or south, then manually points at southeast in order to airdodge at < 50°. The red sequence is an example of an illegal ledgedash: the player ledgefalls in southeast, then, the L/R button (airdodge) automatically traverses the 50° line for them.

[8.1.4] Down-B/Side-B Integrity

Pointing in either down-B or side-B territory means being unable to perform certain other actions. This usually isn't relevant (since most special moves are not cancellable), except for with Fox and Falco's Shine.



If you want to Shine, then wavedash at an angle in down-B territory, a single stick motion will do the job.



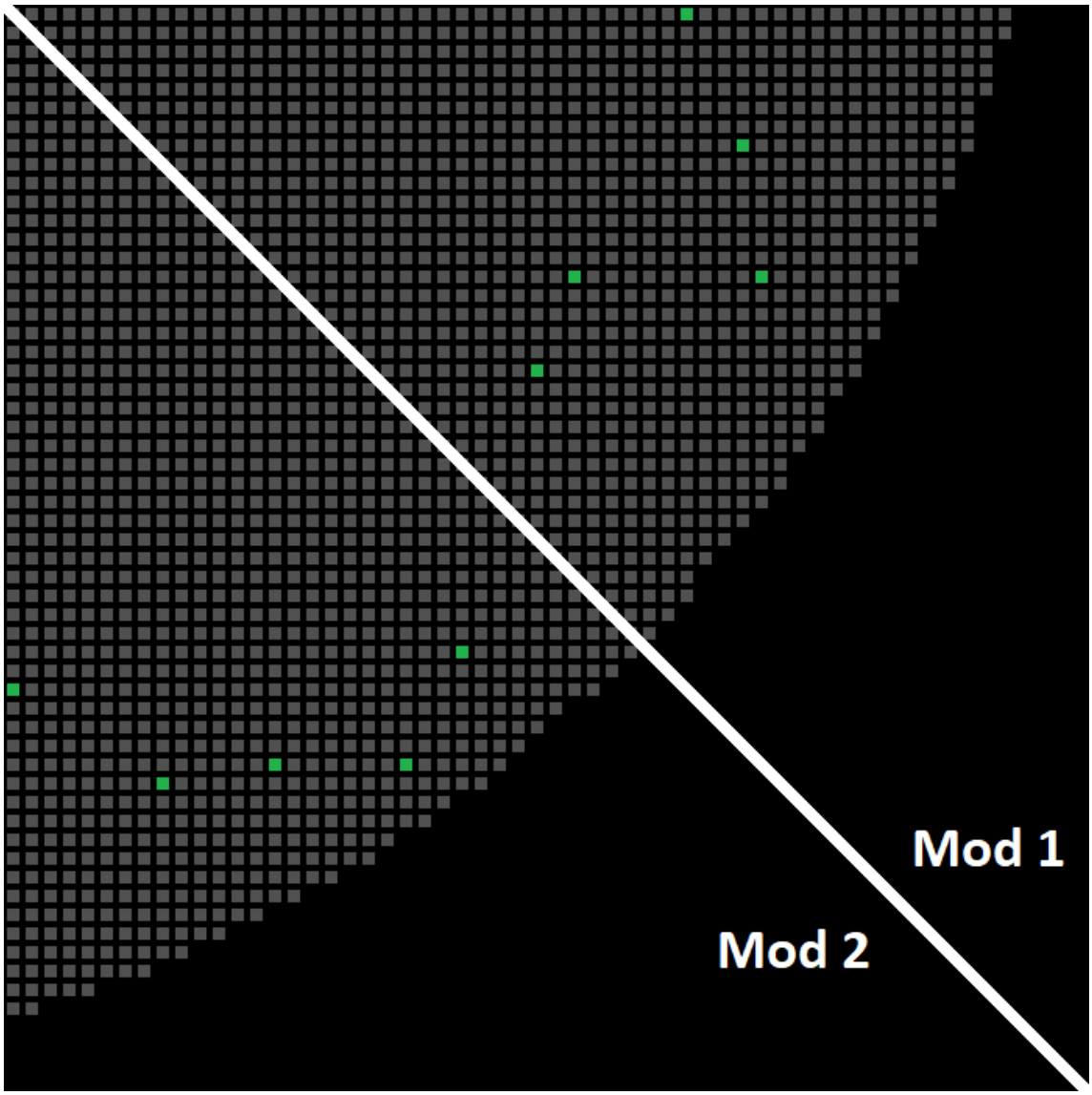
However, if you want to Shine, then wavedash at an angle in side-B territory, a second stick motion will be needed; therefore, **if pressing the B button would produce a down-B, A/C/L/R/Z may not redirect the analog stick to side-B territory (and vice versa)*.**

*The only exception to this rule is shield (not airdodge) pointing at ~45° regardless of the sitation. This is because the B0XX cannot support a variety of shield angles.

[8.2] Definitely Not Action-Direction Button Binds

With Section 8.1's rules in effect, there is very little that non-dedicated modifiers can still do. Nonetheless, they are needed to tap into a few aspects of Melee's analog control. **Section 8.2 covers cases in which it is not only irrelevant, but strictly disadvantageous that A/B/C/L/R/Z must be actuated in order to prompt its corresponding directional modification.** These action inputs are inconsequential, however, since your character isn't actionable in any of these situations.

[8.2.1] Firefox



To angle Firefox (and other up-B's), two modifications must be made to the diagonal directional inputs. First, Modifier 1 or 2 is used to denote whether the X or Y-axis is being hugged (this also produces the shallowest/steepest Firefox angle). Then, the C-stick buttons allow you to choose from four additional angles approximately 4.6° apart from each other.

Firefox Angles:

Modifier 1: X +/-.7375 Y +/-.2875 (21.3°)
Modifier 1 + C-Down: X +/-.7750 Y +/-.3750 (25.8°)
Modifier 1 + C-Left: X +/-.7875 Y +/-.4625 (30.4°)
Modifier 1 + C-Up: X +/-.6625 Y +/-.4625 (34.9°)
Modifier 1 + C-Right: X +/-.6375 Y +/-.5250 (39.5°)

Modifier 2 + C-Right: X +/-.5875 Y +/-.7125 (50.5°)
Modifier 2 + C-Up: X +/-.5500 Y +/-.7875 (55.1°)
Modifier 2 + C-Left: X +/-.4625 Y +/-.7875 (59.6°)
Modifier 2 + C-Down: X +/-.3875 Y +/-.8000 (64.2°)
Modifier 2: X +/-.2875 Y +/-.7375 (68.7°)

These coordinates were obtained through the use of a program that accounted for several restrictions:

-Coordinates in Y-tilt + > 50° territory are illegal. Y -.6625, -.6750, and -.6875 are also illegal.

-Coordinates that cause Ice Climbers desyncs are illegal.

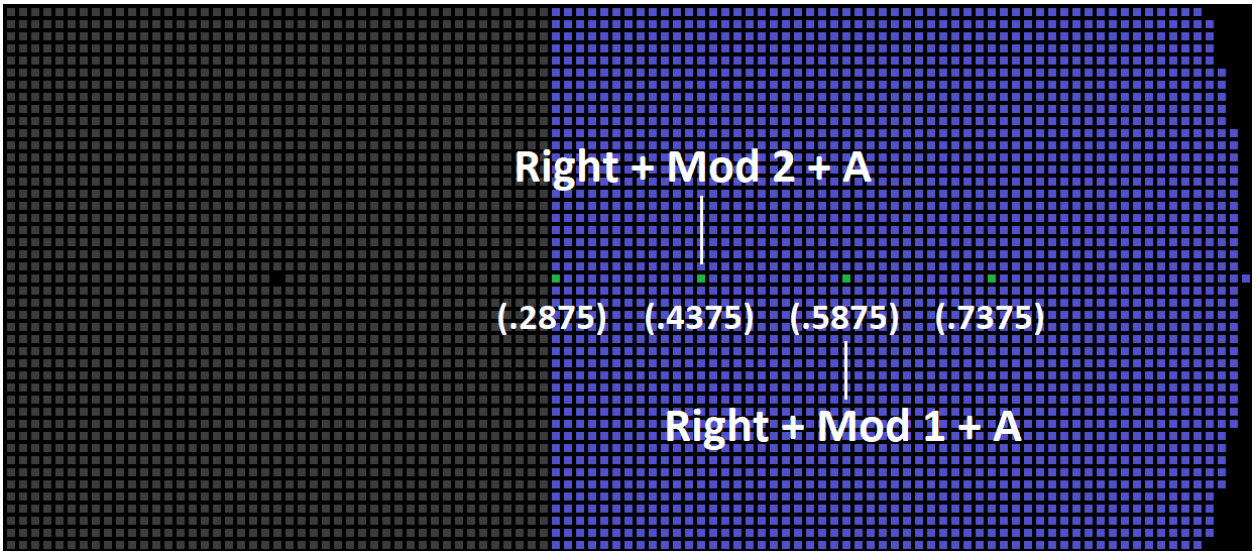
-These coordinates do not contain X => .8000, which means they cannot be exploited to perform dash back out of crouch.

-None of these coordinates are in neutral-B territory (so that Neutral-B Integrity is automatically enforced).

-None of these coordinates are in the sections of the grid that shift between side-B and vertical-B depending on whether your character is airborne or grounded (so that pressing B results in a consistent outcome).

-All of these coordinates are on the appropriate side of the 50° line (< 50° for Modifier 1 and > 50° for Modifier 2). X +/-.5875 Y +/-.7125 (50.5°) are the closest coordinates to the 50° line that Modifier 2 can use. X +/-.6375 Y +/-.5250 (39.5°) were chosen afterwards for symmetrical purposes.

[8.2.2] Slight DI



Normally, Modifiers 1 and 2 will modify the horizontals to X .7375 and .2875 respectively. To access X .5875 and .4375, hold the A button as well. These coordinates will still produce an F-tilt in the neutral game, but they'll let you DI less predictably when being thrown.

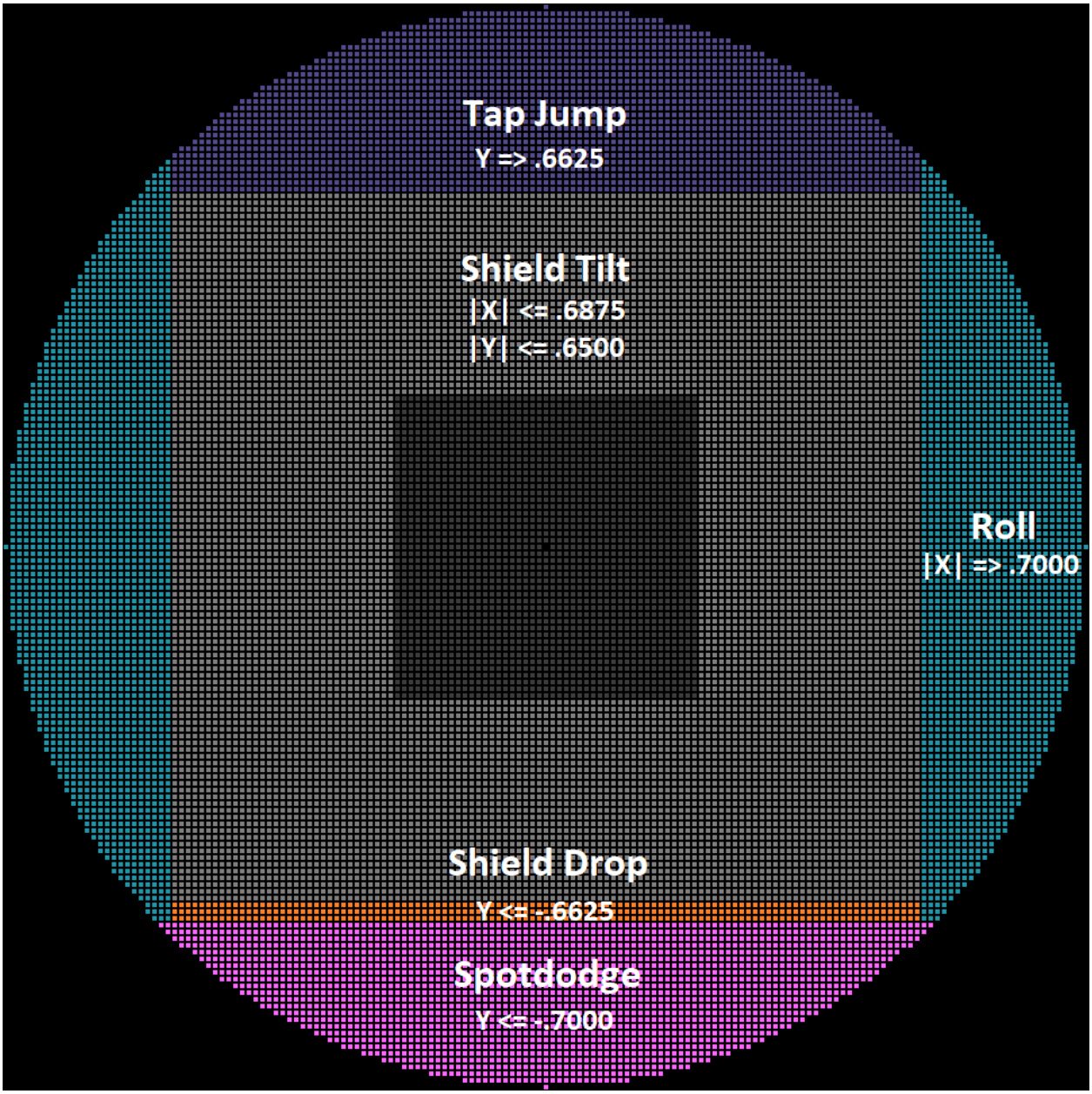
[8.3] Sort of Action-Direction Button Binds

In Section 8.2, there was no relationship between the action inputs and their corresponding directional modifications. C (Firefox) and A (slight DI) were entirely chosen based on their button locations.

In Section 8.3, this is not the case: these modifications are prompted by action inputs that directly need them to occur. While this is worth distinguishing, it ultimately doesn't make a difference. The ability to influence the analog X/Y-coordinates as needed without circumventing meaningful execution remains consistent with these modifications.

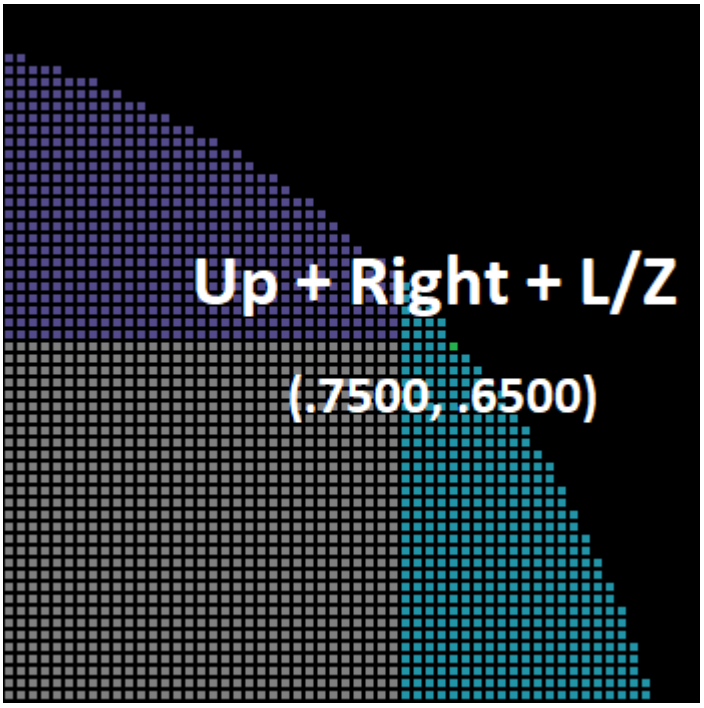
All of the modifications within this section pertain to the L and R buttons, which are inherently problematic for digital controllers. Since shielding, wavedashing, *and* airdodging defensively – three very different techniques – are all linked to these buttons, several interactions must be created in order for the B0XX to offer its user the correct options to choose from.

[8.3.1] Shield Tilt (Automatic)



Shield tilt can always occur up until the thresholds for roll, spotdodge, and tap jump, meaning |X| <= .6875 with |Y| <= .6500 is always valid. On the ground floor specifically, shield can be tilted down to Y -.6875, since shield drop cannot be performed.

With digital inputs, making the most of shield tilt requires two separate features. This section will focus on **automatic shield tilt, which looks to correct the B0XX's default quadrant coordinates of X +/-.7000 Y +/-.7000**. These coordinates are strictly undesirable when shielding because they'll cause tap jump or spotdodge. Had the player intended to perform these actions, they would have simply pointed in a cardinal direction; therefore, it is redundant for the quadrants to perform them. **The L button will make these corrections, as it is the B0XX's primary shield trigger. Z is sometimes useful as a shield, so it will make these corrections as well.**

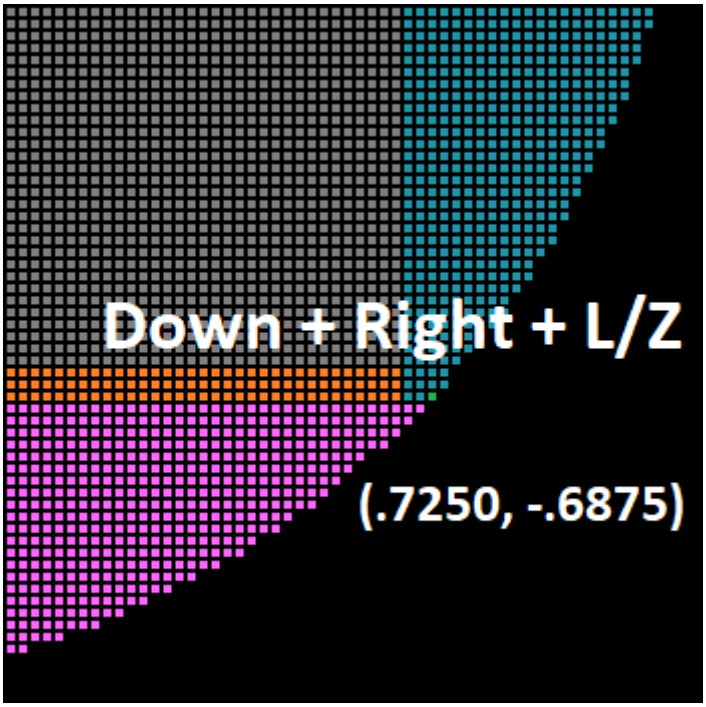


When L or Z is pressed in conjunction with quadrant 1 or 2, you will receive:

X +/-.7500 Y .6500*

Automatic shield tilt is intended to be used in situations where your ability to roll has been shut off (i.e. dashing then buffering a shield horizontally). Since these situations encompass the vast majority of those that call for shield tilt, automatic shield tilt is usually all you need. In quadrants 1 and 2, the values X +/-.7500 Y .6500 are obtained by maxing out the Y-axis (such that it does not cause tap jump), followed by the X-axis. Because we're operating under the assumption that roll has been shut off, it is fine for X to be equal to or greater than .7000.

*L in conjunction with quadrant 1 or 2 won't always produce X +/-.7500 Y .6500 (see Section 8.3.4).



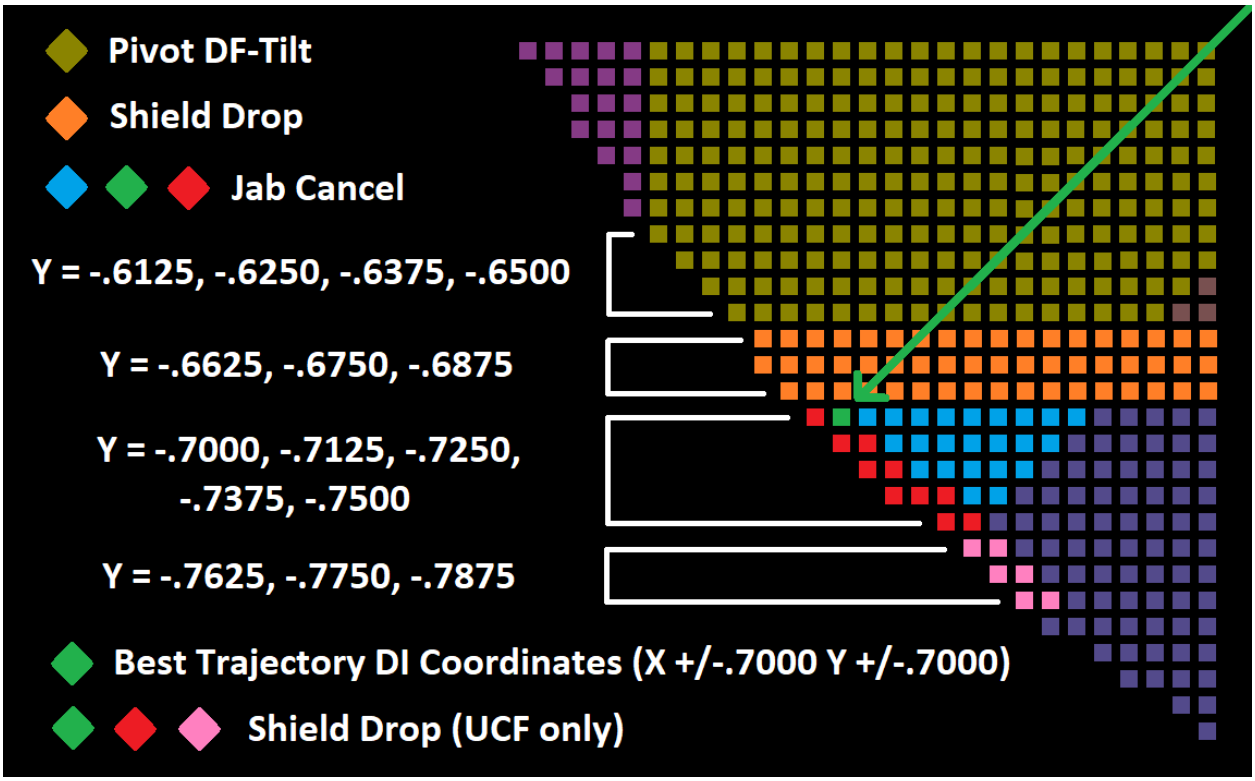
When L or Z is pressed in conjunction with quadrant 3 or 4, you will receive:

X +/-.7250 Y -.6875

In quadrants 3 and 4, it is best for shield to default to Y -.6875. **Not only does this Y-value avoid spotdodge on the ground floor, but it also causes shield drop on platforms.** From there, the X-axis is maxed out. These coordinates (X +/-.7250 Y -.6875)

comply with the shield drop down nerf from Section 5.2.3, which stated that all shield drop Y-values on the B0XX needed to be paired with roll X-values.

For transparency's sake, there is a small complication involving the coordinates X +/-.7250 Y -.6875 that has to do with the dynamics of notches (precision, physical construction, etc.) as explained in Section 5.1.3. Based on the logic conveyed within that section, one could make the argument that pinpointing shield drop in quadrants 3 and 4 requires a degree of precision that is only permissible if the B0XX opts to use its SW/SE "notches" on one of shield drop's 3 Y-values. To be exact, this is to say that the B0XX should be able to pinpoint either Y -.6625 through -.6875 (shield drop) *or* Y -.7000 through -.7500 (jab cancel), but not both. Therefore, L and Z modifying to Y -.6875 (shield drop) is problematic, since the B0XX is already capable of pinpointing X +/-.7000 Y -.7000 (jab cancel). **This argument would have been sound, *had the B0XX been geared for vanilla Melee.***



On UCF, shield drop and jab cancel overlap.

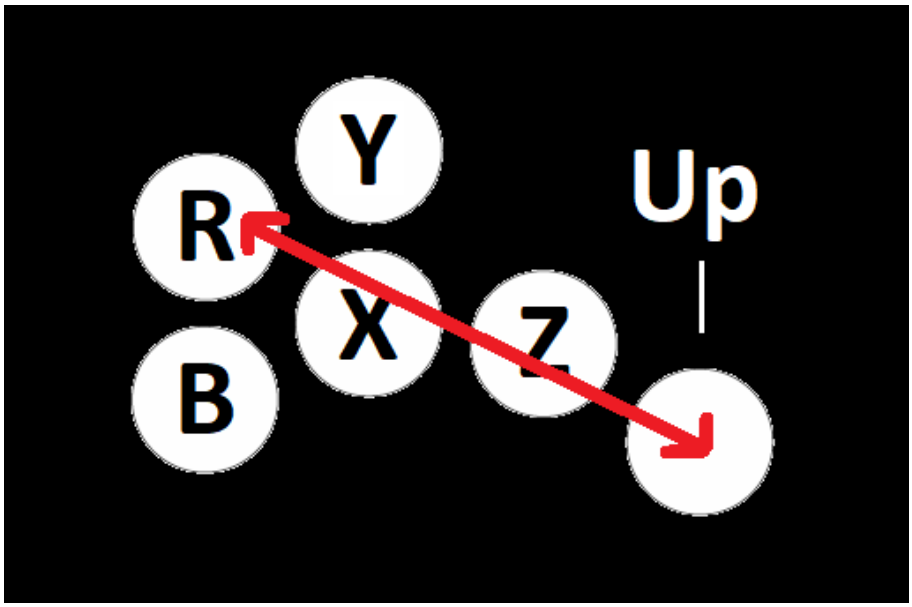
As shown in the diagram, **an unavoidable consequence of UCF's modified shield drop range is shield drop overlapping with jab cancel, making it possible to notch for both techniques on the same controller.** This is one of the more subtle ways in which UCF surpasses traditional Gamecube controller hardware.

Under these circumstances, Y -.6625 through -.6875 are reduced to generic shield tilt coordinates, as they are no longer needed for shield drop. It is, therefore, fine for the B0XX to retain X +/-.7000 Y -.7000 (jab cancel/UCF shield drop) as its unmodified quadrant values while modifying to X +/-.7250 Y -.6875 when L or Z is held (to avoid spotdodge). To put this in perspective: had UCF not been the tournament standard, the B0XX would have had to use X +/-.7250 Y -.6875 as its unmodified quadrant values in order to retain shield drop, at which point it would not have been able to jab cancel.

[8.3.2] Shield Tilt (Manual)

Manual shield tilt, which completes the shield tilt duo by blocking actions such as roll, spotdodge, and shield drop, is far less straightforward to implement than its counterpart. In a perfect world, either Modifier 1 or 2 would have performed manual shield tilt, but they are both occupied with analog L 49. To make matters worse, most non-dedicated modifiers aren't viable candidates. Unlike Firefox (C) and slight DI (A), manual shield tilt is performed in situations where your character is actionable; therefore, if A/B/C/X/Y was required for manual shield tilt, its action input would surely conflict. **This leaves R as the only eligible button for manual shield tilt, since its action input is a shield itself.**

The first thing to know about manual shield tilt is that R's modifications will override L and/or Z's if R is held alongside them. Since manual shield tilt's coordinates are by all means secondary to those of automatic shield tilt, the intention to use manual shield tilt is clear whenever R is held. Giving R priority allows it to not only serve as a tilted shield on its own, but also a "modifier button" of sorts when used with L or Z.



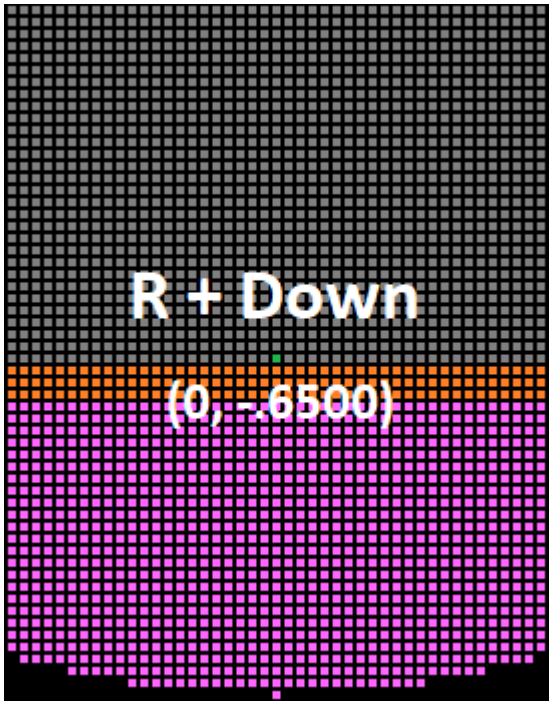
Having to press Up and R at once would have been poor design.

The second thing to know about manual shield tilt is that it cannot be performed in north, quadrant 1, or quadrant 2. These three sections of the grid *could* have been supported, but I chose not to do so due to Up and R being on separate rows. From a design standpoint, it would have been a mistake to encourage the player to tilt their wrist diagonally to simultaneously press these buttons. Conveniently, shield tilt in north is useless, while automatic shield tilt suffices for quadrants 1 and 2 in almost every situation.

When R is pressed in conjunction with Left or Right and no modifier buttons are held, you will receive:

X +/-.6875 Y 0

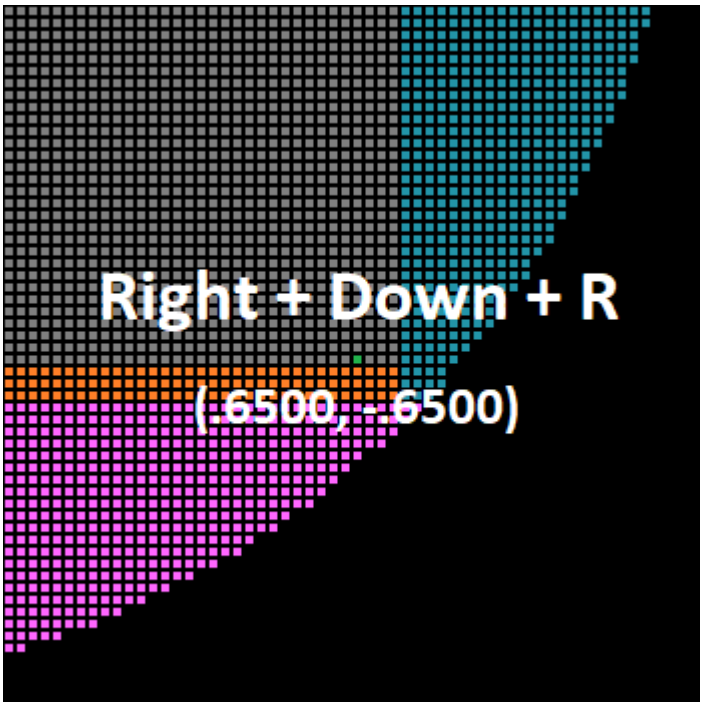
Of R's three manual shield tilt modifications, this one is most straightforward: X +/-.6875 avoids roll in the horizontals.



When R is pressed in conjunction with Down and no modifier buttons are held, you will receive:

X 0 Y -.6500

Aside from avoiding spotdodge, X 0 Y -.6500 allows you to perform a hidden shield drop method that I mentioned briefly in Section 5.2.3. By tilting shield downwards in Y-tilt (the light gray area) for 4 frames, the entirety of spotdodge (pink) will turn into shield drop on specifically frames 5 and 6. While there isn't much reason to use this method over the "Axe method," it is still nifty that the B0XX is able to accommodate it.



When R is pressed in conjunction with quadrant 3 or 4 and no modifier buttons are held, you will receive:

X +/-.6500 Y -.6500

Even though the ideal set of coordinates to use within quadrants 3 and 4 is X +/-.6875 Y -.6500 (since X can extend up until .6875), **X +/-.6500 Y -.6500 is used so that R can produce a 45° vector. This is an unnoticeable hindrance to manual shield tilt that allows R to excel at its other duties (these are wavedash-related; see Section 8.3.3).** As far as manual shield tilt goes, these coordinates allow you to tilt your shield diagonally downwards on a platform without shield dropping.

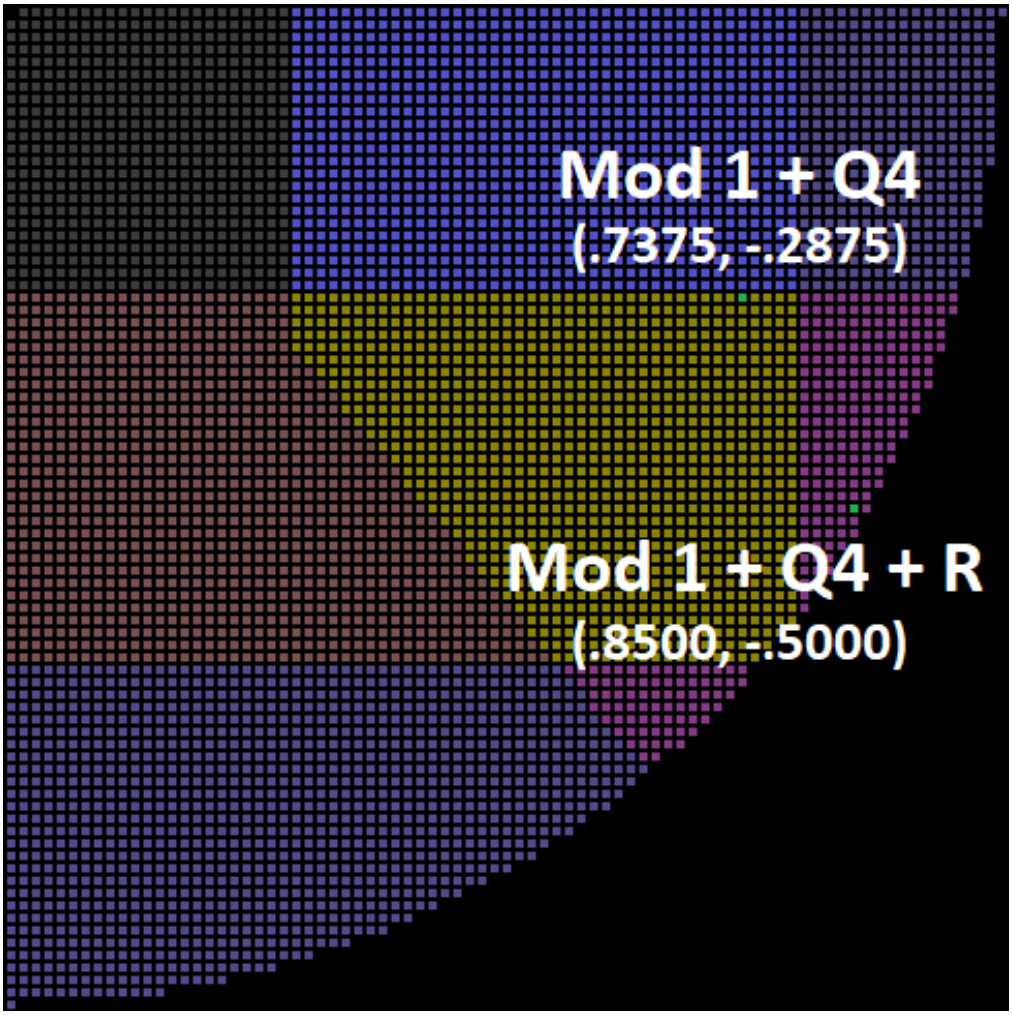
Throughout this section, you may have noticed that manual shield tilt does not occur when either Modifier 1 or Modifier 2 is held. The primary reason for this is **holding an analog trigger and a digital trigger at once causes the digital trigger to take priority**. Since manual shield tilt is tied to digital R, a digital shield would always override analog L 49 and defeat the purpose of tying them to this feature.

[8.3.3] Wavedash

The altered wavedash angles (30.5° especially) are the most frequently used modifications on the B0XX. As with everything else in this chapter, the goal is to incorporate these angles in a manner that doesn't involve installing additional buttons. Since there are two non-45°* angles and two modifier buttons to work with, achieving this is relatively straightforward.

Modifier 1 is associated with < 50° territory, and Modifier 2 is associated with > 50° territory, which makes distributing the 30.5° and 59.5° wavedashes easy enough. The only complication stems from the fact that the modified quadrants have to preserve certain X and Y coordinates (+/-.2875 and +/-.7375) in order to perform other duties. **Since the wavedash angles don't consist of these coordinates, another modification must be prompted by their action button: R.**

*As shown in Section 8.3.2, R (in conjunction with no modifier buttons) modifies the analog X/Y-coordinates from X +/-.7000 Y -.7000 to X +/-.6500 Y -.6500 on its own so that it can assist with shield tilting in addition to wavedashing at 45°.



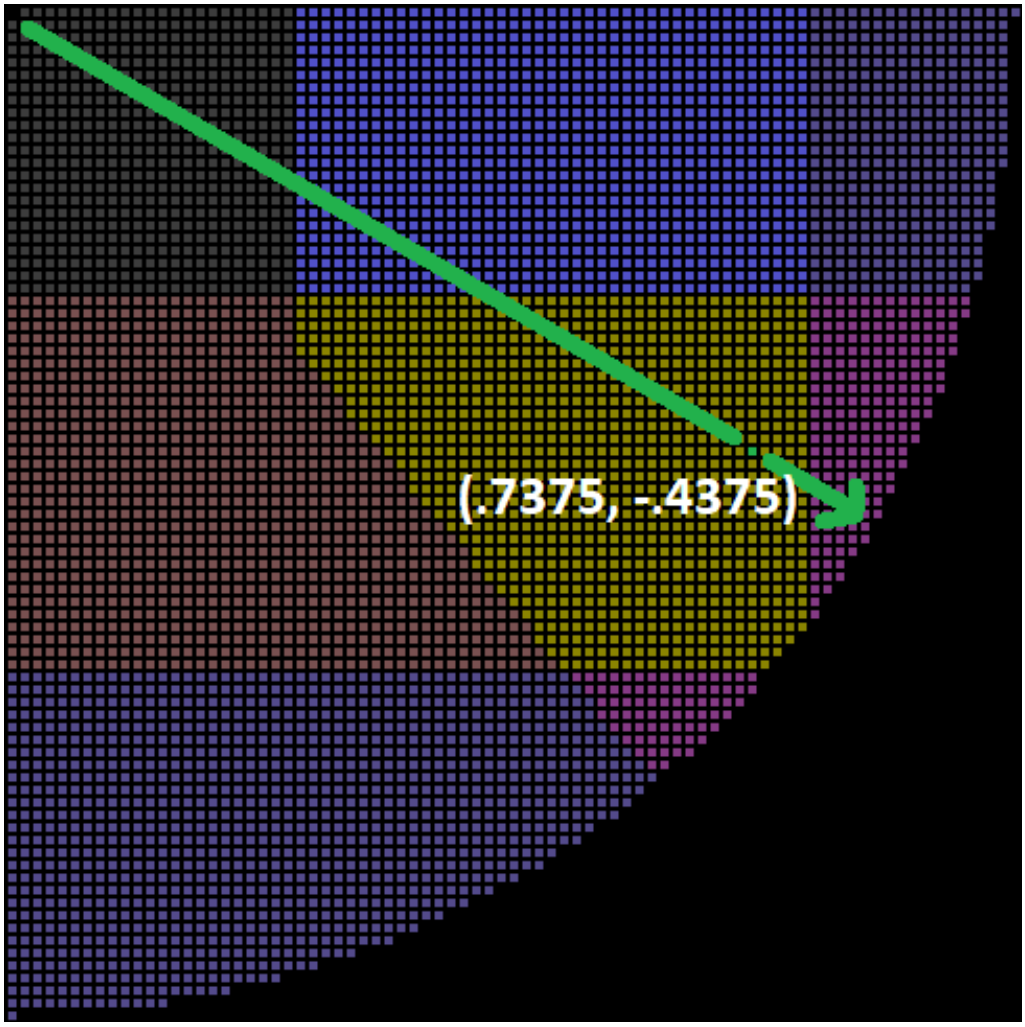
When the modifier buttons and R are pressed in conjunction with quadrant 1, 2, 3 or 4, you will receive:

Modifier 1: X +/-.8500 Y .5000 (30.5°)* ** *(pictured)*
Modifier 2: X +/-.5000 Y .8500 (59.5°)**

At first glance, this would appear to be a blatantly disingenuous action-direction button bind; however, looks can be deceiving. Once this diagram is viewed from the perspective of what these coordinates actually *do*, it is clear that **there was no directional modification of any significance.**

*These coordinates cannot be exploited to perform dash back out of crouch because the R button will generate a shield.

**These coordinates aren't used in quadrants 1 and 2. X +/-.8750 Y .4750 (28.5°) and X +/-.4750 Y .8750 (61.5°) are used instead (see Sections 5.2.9 and 8.3.4)



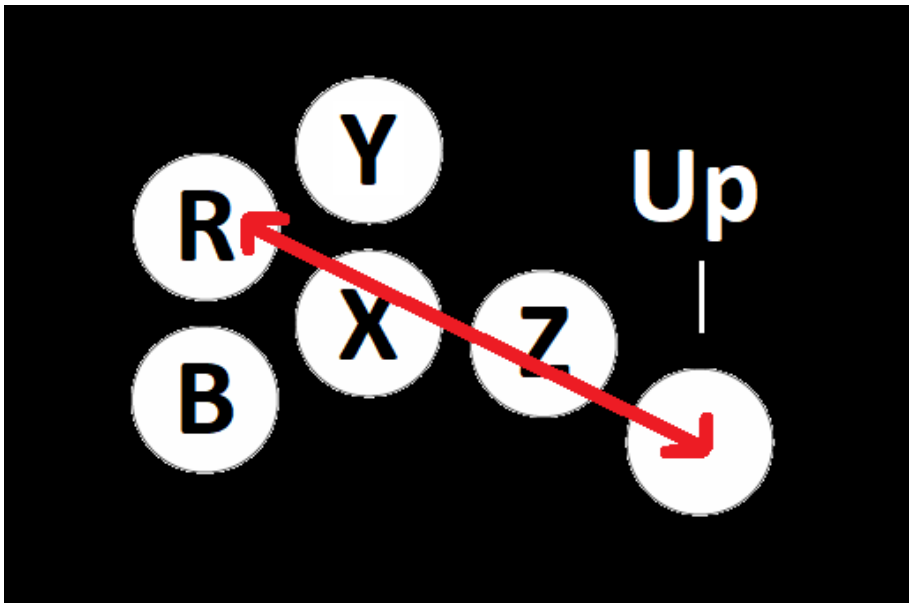
Theoretical coordinates the B0XX could have used.

When an airdodge is calculated, the magnitudes of X and Y are ignored (only the angle between them is relevant). This means that, in theory, **Modifier 1 could have given coordinates that produce a 30.5° vector, point in X-tilt + Y-tilt territory, *and* walk at X .7375 to begin with.** X +/-.7375 Y +/-.4375, for example, satisfies all of this criteria (its wavedash angle is 30.6°, but that's besides the point). These coordinates can therefore be considered the analog stick's effective location whenever the modified quadrants (with or without R) are pinpointed on the B0XX.

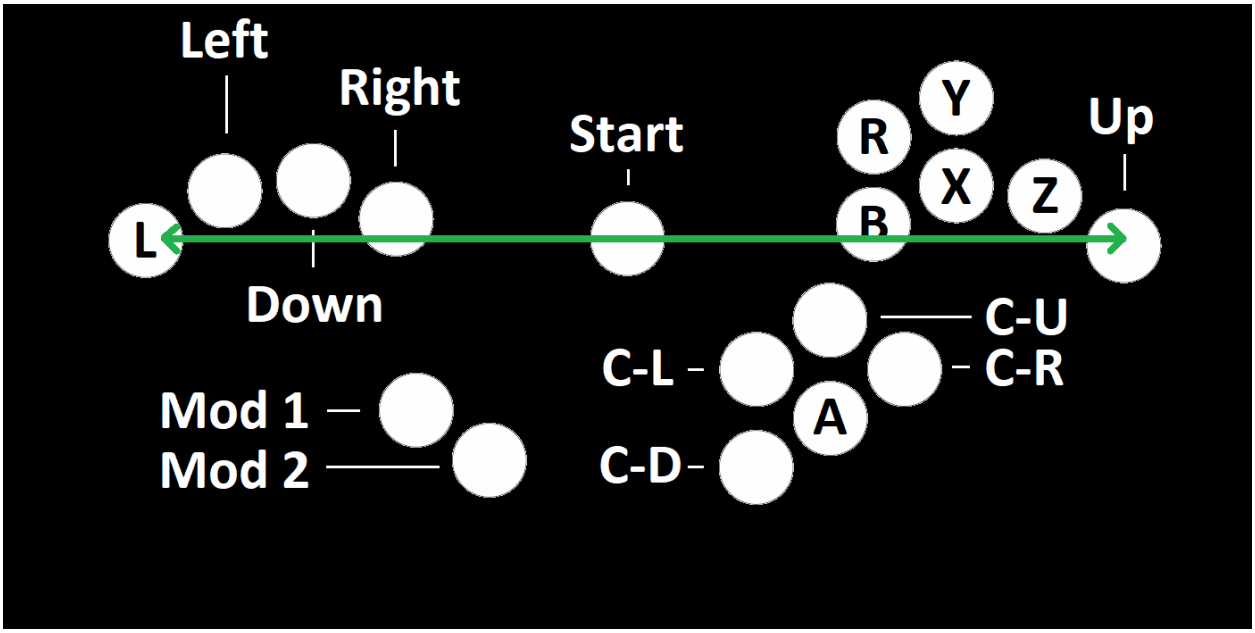
I opted not to use coordinates like X +/-.7375 Y +/-.4375 for the sole purpose of design elegance. Wavedashes are usually performed with the analog stick pressed against the rim, which I wanted the airdodge coordinates I gave the B0XX to stay true to (not to mention how ridiculous testing all the coordinates that aren't along the rim would have been). I also wanted the raw modified quadrants to contain X or Y +/-.2875, as hugging the X or Y-axis seemed least arbitrary. R's modifications should be understood as a small touch I gave the controller so that it could consist of relatable coordinates like these. As far as gameplay goes, they do not make a difference.

[8.3.4] Home Row Upwards Airdodge

Throughout Section 8.3.3, it was implied that R was only meant to be used for wavedashing (which pertains to airdodging in quadrant 3 or 4 specifically). This may have come across as an oversight, but it was very much intentional. While R *does* cause its modifications in quadrants 1 and 2, these modifications are only meant to restrict the B0XX's airdodges to less shallow/steep angles (X .7375 Y .2875 and X .2875 Y .7375 would have airdodged at 21.3° and 68.7° respectively). As far as actually performing upwards airdodges goes, using R is discouraged.



Once again, this is because it would have been a mistake to formally support the simultaneous use of buttons that are on different rows with the same hand. R was placed in this location because the index finger's strength is needed for swift and accurate wavedashes (similar reasoning went into B's placement). When it comes to upwards airdodges, R is unideal.

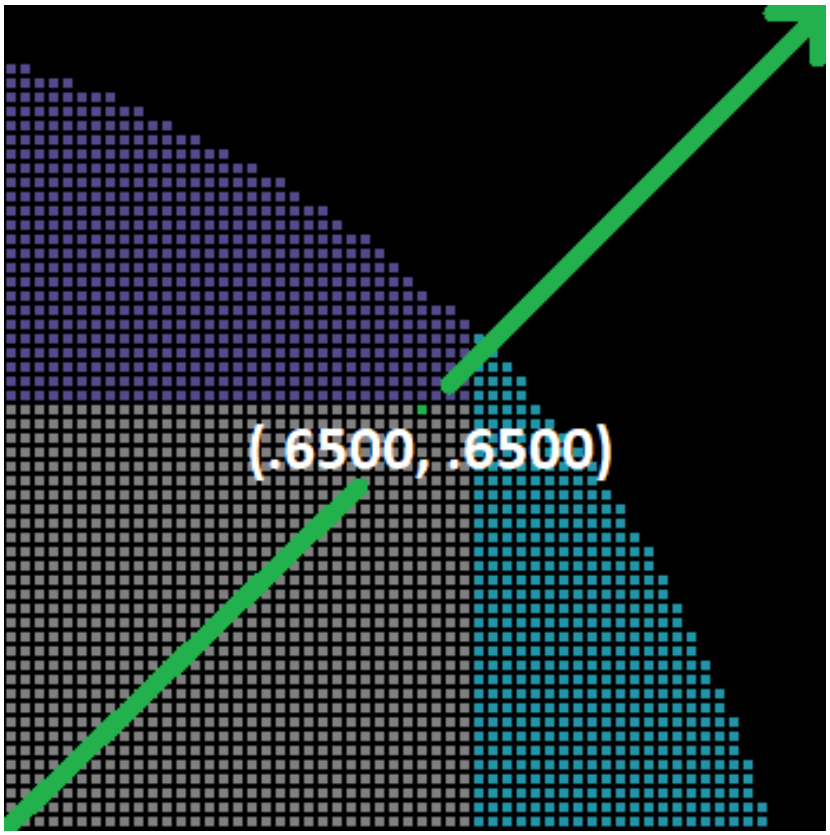


L, on the other hand (no pun intended...), is in a great location for upwards airdodges.

The potential to correct this design flaw lies in the L button, which is in an eligible location for upwards airdodges. The only hurdle is that analog L 49 and automatic shield tilt cannot be compromised in the process. **In order for L to accommodate upwards airdodges while preserving the rest of its interactions, the second and last instance of order-dependency is needed.**

As to comply with the Peach's ledgedash ban from Section 5.2.9, the goal is to give the B0XX the ability to comfortably airdodge at 28.5°, 45°, and 61.5° in quadrants 1 and 2. **For the L button to make this possible, it must adjust for these duties when it is pressed *after* Up (this is inclusive of a simultaneous Up + L press).** Since directional airdodges can only be performed by pressing L/R after (or alongside) the directional input, these adjustments will always match the player's intention to airdodge.

This means that **the only drawbacks to L's adjustments involve shielding (with L) and shield tilting in quadrant 1 or 2 *on specifically the same frame*.** Due to the inclusiveness of "after" inputs, the B0XX can mistakenly interpret quadrant 1/2 + L on the same frame as an upwards airdodge when the intention was to shield tilt. Luckily, there is a way to heavily mitigate this interference when it occurs.



When L is pressed in conjunction with no modifier buttons and quadrant 1 or 2 and Up is already held, you will receive:

X +/-.6500 Y .6500 (45°)

Once again, X +/-.6500 Y .6500 are invaluable in that they are compatible with airdodge *and* shield. These coordinates not only allow an upwards airdodge to take place at 45°, but they also nullify the most common occurrence of the same-frame shield tilt interference. When a digital shield is simultaneously actuated and tilted in quadrant 1 or 2, the only consequence is pointing

at slightly less optimal coordinates (X +/-.6500 Y .6500 instead of X +/-.7500 Y .6500). This is unnoticeable.

When L is pressed in conjunction with Modifier 1 and quadrant 1 or 2 and Up is already held, you will receive:

Digital L + X +/-.8750 Y .4750 (28.5°)

Aside from the airdodge angle being modified to 28.5°, **L will now produce a digital press (for airdodge) instead of analog L 49.**

This time around, the drawbacks to the same-frame shield tilt interference are more severe, since this outcome is radically different from the expected lightshield. Due to the infrequency of tilting a lightshield in quadrant 1 or 2 (let alone simultaneously), however, this isn't a concern.

When L is pressed in conjunction with Modifier 2 and quadrant 1 or 2 and Up is already held, you will receive:

Digital L + X +/-.4750 Y .8750 (59.5°)

Again, L will produce a digital press (instead of L 49), and the airdodge angle is modified to 59.5° as expected. The same-frame shield tilt interference is infrequent to the point that it isn't a concern.

[8.4] Summary

Jump Trajectory Integrity
The X and Y buttons cannot serve as non-dedicated modifiers.

Tilt/Smash Integrity
Non-dedicated modifiers cannot traverse X .8000 or Y .6625 in a manner that meaningfully circumvents a stick motion.

50° Line Integrity
Non-dedicated modifiers cannot traverse the 50° line in a manner that meaningfully circumvents a stick motion.

Down-B/Side-B Integrity
If the B button would yield a down-B, non-dedicated modifiers cannot redirect the analog stick to side-B territory (and vice versa).

Firefox
Modifiers 1 and 2 are used to hug the X or Y-axis. The C-stick buttons are then used to angle Firefox.

Slight DI
Modifiers 1 and 2 in conjunction with the horizontals will normally produce X .7375/.2875. To access X .5875/.4375, hold the A button as well.

Shield Tilt (Automatic)
L and Z correct the quadrant coordinates so that they default to shield tilt (and shield drop).

Shield Tilt (Manual)
R allows you to shut off roll, spotdodge, and shield drop.

Wavedash
Strictly for aesthetic purposes: R modifies Modifier 1 and 2's quadrant values to wavedash coordinates along the rim.

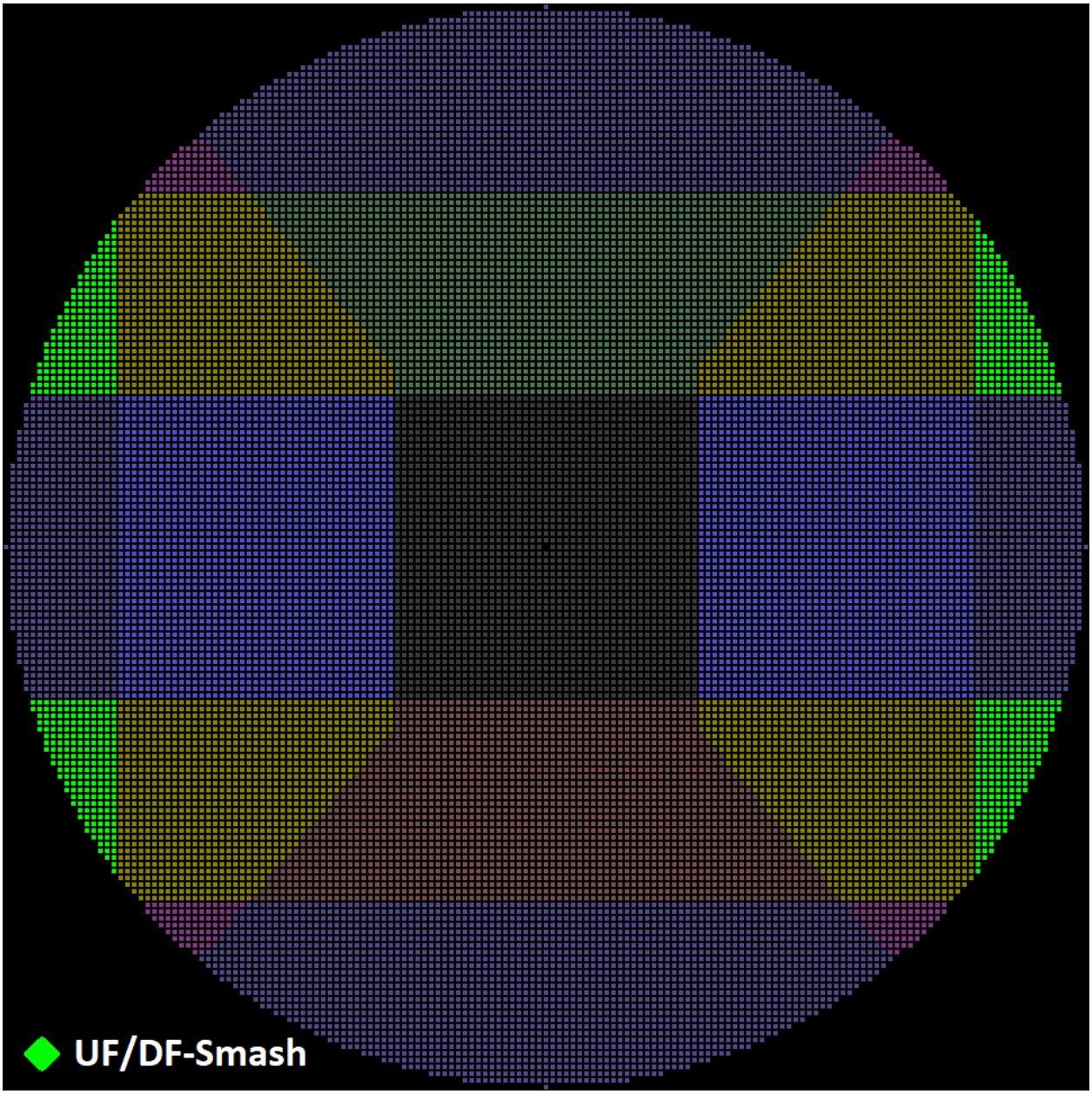
Home Row Upwards Airdodge
L allows you to comfortably airdodge upwards (without having to tilt your wrist diagonally to press R).

****

[9] Other Interactions

****

[9.1] UF/DF-Smash



X-smash + Y-tilt is UF/DF-smash territory on both the analog stick and C-stick.

Throughout this document, I failed to show a way to pinpoint the regions where X-smash overlaps with Y-tilt (aside from Firefox/airdodge angles, which require C/L/R to be actuated). This is because **the B0XX does not possess one.** These parts of the grid are responsible for very few functions: they can be used to dash/run with ASDI down, but the C-stick is better for that. They can also be used to guarantee that dash back out of crouch succeeds, but that was banned in Section 5.1.4. The only technique the B0XX truly needs these regions for is UF/DF-smash, which can be performed by 8 characters in the game. Even still, neither Modifier 1 nor 2 can afford to accommodate these regions, as the ones they pinpoint are more important. **This leaves the option of using the C-stick to perform UF/DF-smash.**

With the analog stick, the B0XX inherently has to hold two cardinal directions in order to produce diagonal vectors. Due to the long list of techniques the analog stick has to do with (smash DI comes to mind), it is essential that this remains true. With the C-stick, however, we can afford to break this rule since there is nothing remotely exploitative that could result from doing so. This is especially necessary considering the arrangement of the C-stick buttons; since they are meant for the right thumb, having to press two of them at once isn't ideal.

To prompt these diagonal C-stick vectors, **Modifier 1 must first be held in conjunction with Up or Down on the analog stick**. This will produce X 0 Y +/-.6500, which won't inform your opponent that you are pointing in either direction (since these coordinates cause neither tap jump nor crouch). **Then, press C-Left or C-Right.** Instead of the usual C X +/-1.0 Y 0, this will produce C X +/-.8125 Y +/-.2875, which are valid coordinates for UF/DF-smash. These coordinates comply with the restrictions on Popo's F-smash desync, and the 4th/5th F-smash angles.

[9.2] D-Pad

When Modifiers 1 and 2 are held simultaneously and the C-stick has yet to be actuated, the C-stick transforms into the D-pad. Holding Modifiers 1 and 2 simultaneously also shuts both of their directional, analog L, and C-stick modifications off.

****

[10] B0XX Advantages

****

[10.1] Travel Time

Most of the remaining advantages on the B0XX stem from the ability to press certain arrow keys in succession without experiencing physical recoil. These can fall under:

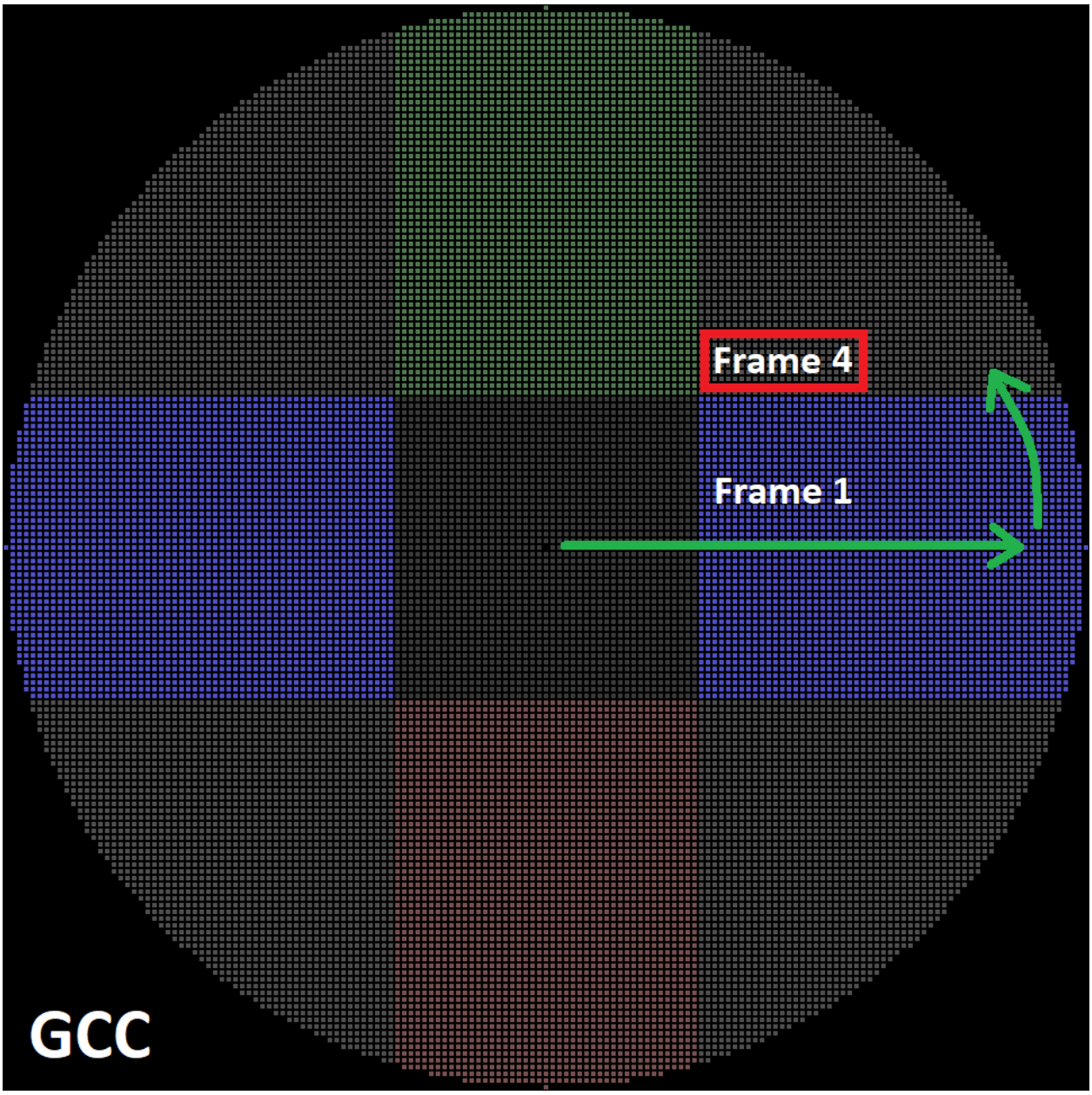
-**Cardinal/diagonal to diagonal**. The B0XX can perform these inputs on frame 1 then 2. This is 100% unremovable, since any sort of timing-based lockout would conflict with the ability to perform basic functions that involve pointing in the quadrants.

-**Cardinal to opposite cardinal**. The B0XX can perform these inputs on frame 1 then 2 as well. Even though this sequence *could* have been toned down through the use of lockouts, I opted not to do so for a few reasons. For one, unlike the SDI nerf (which bans the 3rd SDI input for 6 frames in order to fill up <= 9 frame hitlag windows), enforcing a lockout on opposite cardinals would have been arbitrary, since there isn't a definitive number of frames in this case. Furthermore, enforcing a lockout would have been overkill. Whereas the SDI nerf completely disincentivizes the button sequence it targets, an opposite cardinal lockout would have *incentivized* performing a button sequence as quickly as permitted. This would have caused the player to compete against their own controller's lockouts, an entirely artificial diversion.

It should also be noted that all of the aforementioned sequences are risky to attempt so quickly (frame 1 then 2), as they won't be sequences at all if both arrow keys are input on the same frame. Frame 1 then 3 generally gives much better risk/reward.

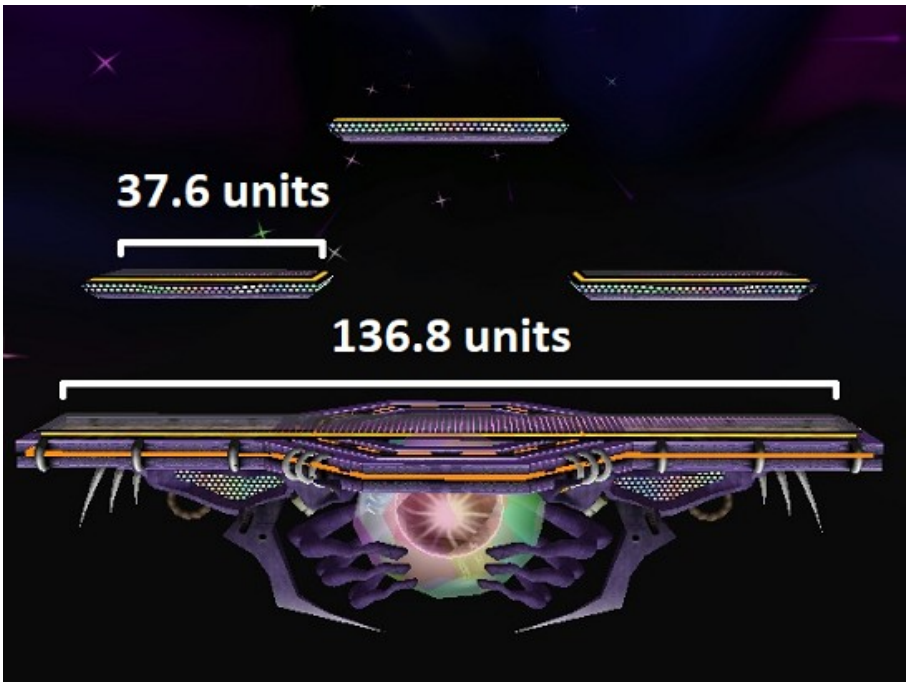
Lastly, the following analog stick motions/difficulties should be kept in mind for Sections 10.1.6, 10.1.7, 10.1.8, and 10.1.9:
-**Frame 1 West (X -1.0), Frame 5 East (X 1.0):** Easy
-**Frame 1 West (X -1.0), Frame 4 East (X 1.0):** Difficult
-**Frame 1 West (X -1.0), Frame 3 East (X 1.0):** Humanly unrealistic
-**Frame 1 West (X -1.0), Frame 2 East (X 1.0):** Impossible

[10.1.1] Quarter-Circle Smash DI



With an analog stick, quarter-circle SDI inputs should take place on frame 1 then either 3 or 4. With digital inputs, frame 1 then 2 is possible. This can sometimes result in an additional SDI input within a hitlag window. For example, if a 9-frame hitlag window was about to expire, it would be possible for digital inputs to squeeze in two SDI inputs on frames 8 and 9, whereas an analog stick would only able to perform one SDI input (on frame 8 or 9).

An image of in-game “units” (for reference):



Furthermore, the B0XX is guaranteed to receive SDI inputs of 1.0 within the cardinals, and .7000 within the quadrants. This means quarter-circle SDI inputs will always cause your character to travel 10.2 units in the desired direction (SDI = X or Y * 6. 1.7 * 6 = 10.2).

On the Gamecube controller, it is possible to receive SDI inputs less than 1.0 within the cardinals. Since the formula for a valid SDI input is X^2 + Y^2 => .7125, it is possible for the cardinal to generate an SDI input between .7125 and .9875 (depending on when your controller is polled). 1.0 is the most frequent outcome, however, due to the velocity the analog stick accumulates causing it to traverse .7125 through .9875 rather quickly. The subsequent diagonal SDI input is also almost always greater than .7000.

[10.1.2] Wiggle

When your character is airborne and in tumble, you must first wait for hitstun to expire. Then, to regain actionability, one of several options can be chosen. Aerials, special moves, and jumps will break your character out of tumble 100% of the time. Wiggling, on the other hand, is unreliable with an analog stick.

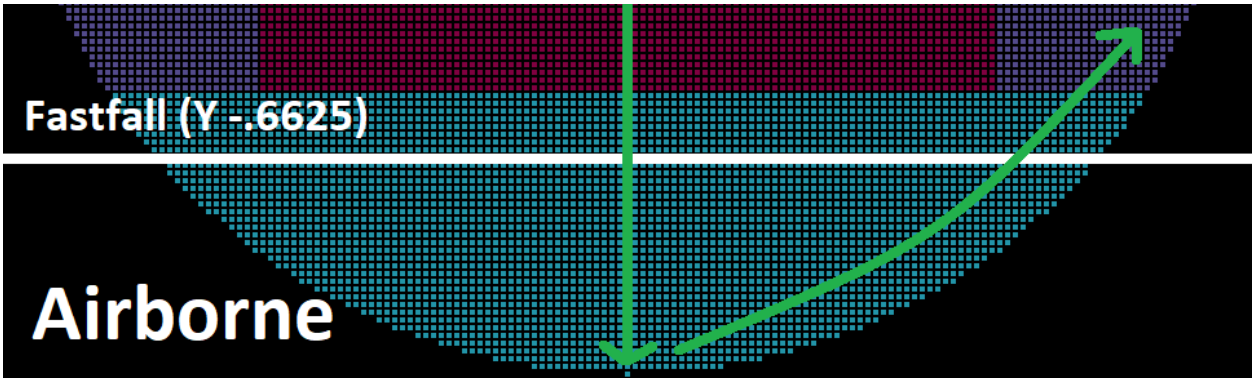
To perform a wiggle, the analog stick must skip from the deadzone to X-smash *without being polled in X-tilt*. X 0 -> X 1.0, for example, will successfully wiggle, while X 0 -> X .7875 -> X 1.0

will fail to wiggle. Since polling sequences like these are impossible to account for, it is usually better to break out of tumble by performing an aerial, special move, or jump with the Gamecube controller.

These options (jump especially) are usually adequate, however, there are a few situations where a wiggle specifically is the best choice. For example, jumping then airdodging consumes your double jump, whereas wiggling then airdodging preserves it. A wiggle can also be used to prevent yourself from having to tech on a platform, or to enable grabbing the ledge.

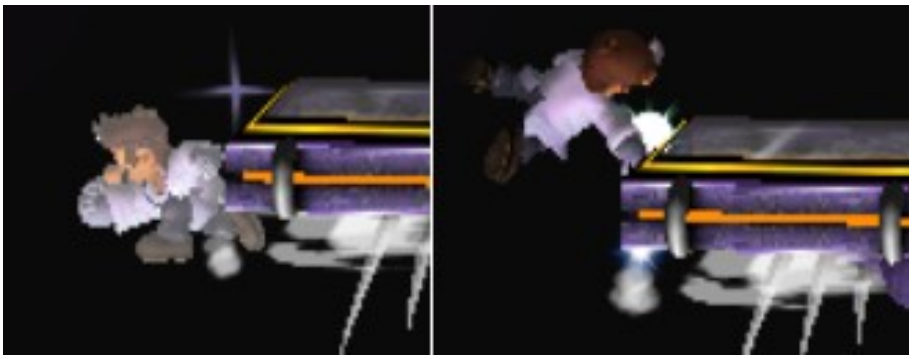
Since digital inputs skip directly from X 0 to X 1.0, they make wiggling a reliable option.

[10.1.3] Samus' Short Hop Fastfall Missile



Digital inputs make it easier to fastfall on frame 1 then side-B on frame 2. This is strictly relevant for Samus, who only has a 2-frame window to perform short hop fastfall missile.

[10.1.4] Dr. Mario's Reverse Up-B Cancel



Dr. Mario's up-B cancel can be used to cause a ledgesnap (can be done facing either direction).

While Dr. Mario has a few up-B cancel variants, his reverse up-B cancel is particularly troublesome. To perform the above ledgesnap sequence, Dr. Mario must input X => .6375 to the left on either frame 2 or 3, followed by X => .2875 to the right on frame 4. This technique is so difficult with an analog stick that it can be considered a digital inputs-exclusive.

[10.1.5] Crouch -> U/UF-Tilt

Crouch can be cancelled into any attack on the very next frame. Whereas U/UF-tilt are physically far away from crouch on an analog stick, they are immediately accessible with digital inputs. These tilts can be useful after cancelling run into crouch, or after crouch cancelling an attack.

[10.1.6] Moonwalk

Digital inputs allow you to alternate the horizontals on frame 1 then 2 to perform the best moonwalk in the game.

[10.1.7] Dash Back -> Dash Back

Once a dash forward is initiated, it can be cancelled into a dash back on frame 5. A dash back, however, can be cancelled into another dash back on frame 3. Digital inputs make a single repetition of dash back (frame 1) -> dash back (frame 3) entirely possible (the rate of subsequent repetitions will be on par with that of an analog stick).

[10.1.8] Dash -> Jump With Backwards Trajectory

With characters who have 3 frames of jumpsquat, it is difficult to dash in one direction (frame 1) then reach the opposite direction in time for trajectory to be calculated on the final frame of jumpsquat (frame 4) with an analog stick. Digital inputs make this a breeze, since they can already be there by frame 2.

[10.1.9] Aerial Drift

The impact digital inputs have on aerial drift is largely misunderstood. While they are better overall, this is not nearly to the degree some people believe, nor for the reasons that might appear to be true.

Most assessments of aerial drift fail to account for jump trajectory, which is an entirely separate mechanic. As shown in Section 10.1.6, jump trajectory is calculated based on your X-value on the final frame of jumpsquat. This locks your character into an arc that cannot be exceeded once it has been determined. For example, X .2875 trajectory followed by X 1.0 aerial drift (on every airborne frame) will travel significantly less far than X 1.0 trajectory followed by X 1.0 aerial drift. In competitive play, it is usually best to takeoff with either X 0 or X +/-1.0 trajectory. The former allows you to assess the situation before making a decision, while the latter gives you the most potential to drift in either direction.

Neither digital inputs nor an analog stick ever struggle to takeoff with X 1.0 trajectory (excluding the situation I covered in Section 10.1.6); however, **preparing to perform aerials with the A button can arbitrarily cause conflict.** The coordinates with the greatest X-value that produce a D-air, for example, are X

+/-.6375 Y -.7625; therefore, pointing in D-air territory prior to takeoff results in your jump trajectory being stunted.

For this reason, **it is always best to C-stick the 4 directional aerials**. The analog X/Y-coordinates can then be aimed as desired, leaving N-air as the only aerial that conflicts with jump trajectory. This is pertinent to the matter at hand because **the B0XX's button layout encourages you to always C-stick your directional aerials, whereas the Gamecube controller's does not**. I mentioned this in Section 2.1.2 when I criticized the Gamecube controller for its C-stick being inaccessible without a grip most players aren't willing to employ. This tends to result in B0XX users having a jump trajectory advantage over Gamecube controller users despite the option being there to nullify this in full. The B0XX's Nunchuk-compatibility will likely open people's eyes to just how much of an inconvenience this is.

As far as aerial drift itself goes, both controllers have their advantages. Unlike with jump trajectory, which favors polarizing coordinates like X 0 and X +/-1.0, an analog stick has merit when it comes to aerial drift. Through its intuitive design and full range of X-values, an analog stick allows you to effortlessly point at where you want to go. This is usually much more efficient than repeatedly tapping X -1.0 and X 1.0 to station yourself with digital inputs (since using the X-values in between is even more difficult).

On the flipside, an analog stick cannot compare to how effective digital inputs are at alternating one horizontal, then the other. Whereas an analog stick will typically complete this sequence by frame 4 or 5, digital inputs can complete it by frame 2. This shines when aerial drifting, since it is the only movement-related area of the game where directional inputs will always produce immediate results.

Currently, I am inclined to believe that **the Gamecube controller is advantaged at aerial drifting in one direction (i.e. East only), while the B0XX is clearly advantaged at alternating two directions.** Even though it doesn't always come into play, the B0XX's advantage is less replaceable, making it more valuable overall. This is because digital inputs *can* simulate fine analog

control (though this requires a high degree of skill), whereas their lack of recoil truly cannot be replicated with an analog stick.

[10.2] Precision

In most cases, it is easy to remove precision-related advantages from the B0XX. This is usually a one-step process that involves banning the coordinates in question. Despite this tendency, there remains a notable instance of the B0XX inherently being more precise. This pertains to a basic function that cannot be removed.

[10.2.1] No-Fastfall from Ledge

When ledgedashing, it is theoretically always best to fall from the ledge (on frame 1) without fastfalling on frame 2. This causes your character to remain at nearly the same elevation on frame 2, which can be used to create 2-frame windows on your fall, jump, and airdodge inputs (as opposed to 1-frame windows had you fastfell).

Fox's Ledgedashes:

Frame 1	Fall	Fall	Fall
Frame 2	**Jump**	No-Fastfall	Fastfall
Frame 3	--	**Jump**	Jump
Frame 4	--	--	--
Frame 5	**Airdodge**	--	--
Frame 6		**Airdodge**	--
Frame 7			Airdodge

With Fox, for example, ledgedashing is most consistent* if you time your no-fastfall for frame 1.5, jump for frame 2.5, and airdodge for frame 5.5. This is because Fox's frame 5 and frame 6 ledgedashes are interchangeable (since they both require the airdodge to take place 3 frames after the jump), which makes it best to time your inputs for in between frames (to pad yourself with leniency in either direction). A no-fastfall is needed on frame 2 to make the frame 6 ledgedash a part of this equation.

***These are the most consistent inputs *timing-wise*.** Jump trajectory and airdodge shallowness are separate variables that this chart fails to account for.

With the Gamecube controller, there are three ways a no-fastfall from ledge can be performed. Since each of these methods have unique characteristics, I'll go over them one at a time.

Analog Stick / Back

Despite being widely perceived as the best way to ledgefall, **pointing the analog stick backwards is the worst of the three no-fastfall from ledge methods, since it defeats its own purpose.** This is because it compromises your ability to jump with neutral or forwards trajectory on frame 2, making it nearly impossible to perform the most intangible ledgedash. Since jumping with backwards trajectory automatically results in a SD when ledgedashing (because it prompts the backflip animation), this method usually forces you to wait until frame 3 to jump safely, resulting in only 1-frame leniency on your jump and airdodge.

Analog Stick / Down (Y -.2875 through -.6500)

Slightly pressing downwards on the analog stick is probably tied for the best way to no-fastfall from ledge but isn't without its downsides. Since overshooting into Y <= -.6625 will cause you to fastfall, there is a learning curve to this method. **Jump trajectory is also compromised by this method due to the gentle nature of the motion it requires** (your analog stick probably won't be extended that far horizontally by the time you jump).

C-Stick / Back or Down

A claw grip-exclusive: using the C-stick to ledgefall is also perfectly viable but suffers from similar problems. **When using this method, you must be wary of alternating between the C-stick's cardinals and quadrants, which will cause your character to perform an aerial and (most likely) SD.** For example, if you fall with C-Left on frame 1, then shift to C-Down-Left on frame 2, a B-air or D-air will come out. As a result, this method requires a degree of precision comparable to that of gently

pressing the analog stick downwards. **Jump trajectory is also compromised by this method, since you'll have to refrain from jamming the analog stick horizontally until frame 2 or later** (if you simultaneously point in ledge get-up territory on the analog stick and ledgefall territory on the C-stick on frame 1, the ledge get-up takes priority).

All in all, there are two viable no-fastfall from ledge methods on the Gamecube controller, both of which incur risk. **On the B0XX, all three no-fastfall from ledge methods are viable, and none of them incur risk.** Pressing the analog stick backwards becomes viable since physical recoil no longer exists, while the other two methods cannot fail (since overshooting into undesirable territory is impossible). This makes the B0XX more consistent at no-fastfall from ledge than the Gamecube controller.

It is also worth noting that a byproduct of the Nunchuk B0XX is the third method becoming risk-free. Since its C-stick is comprised of digital buttons, the Nunchuk B0XX isn't subject to the challenge of not shifting between the C-stick's cardinals and quadrants. This allows a "Gamecube controller" to guarantee itself no-fastfalls from ledge.

Despite this, ledgefalling with the C-stick on the Nunchuk B0XX isn't always the best option. For that matter, ***none* of the no-fastfall from ledge methods (on the Gamecube controller, B0XX, or Nunchuk B0XX) covered in this section are unequivocally the "best" way to ledgedash.** This is because the most potent strategy is to disregard not fastfalling and jam the analog stick into X-value territory in order to perform the most intangible ledgedash with as much jump trajectory as possible (see Section 11.1.1).

****

[11] Gamecube Controller Advantages

****

[11.1] Hardware

The Gamecube controller has scattered advantages stemming from the physical construction and/or inner workings of its analog stick. There is no common denominator among these other than that they are hardware-related.

[11.1.1] Most Intangible Ledgedash

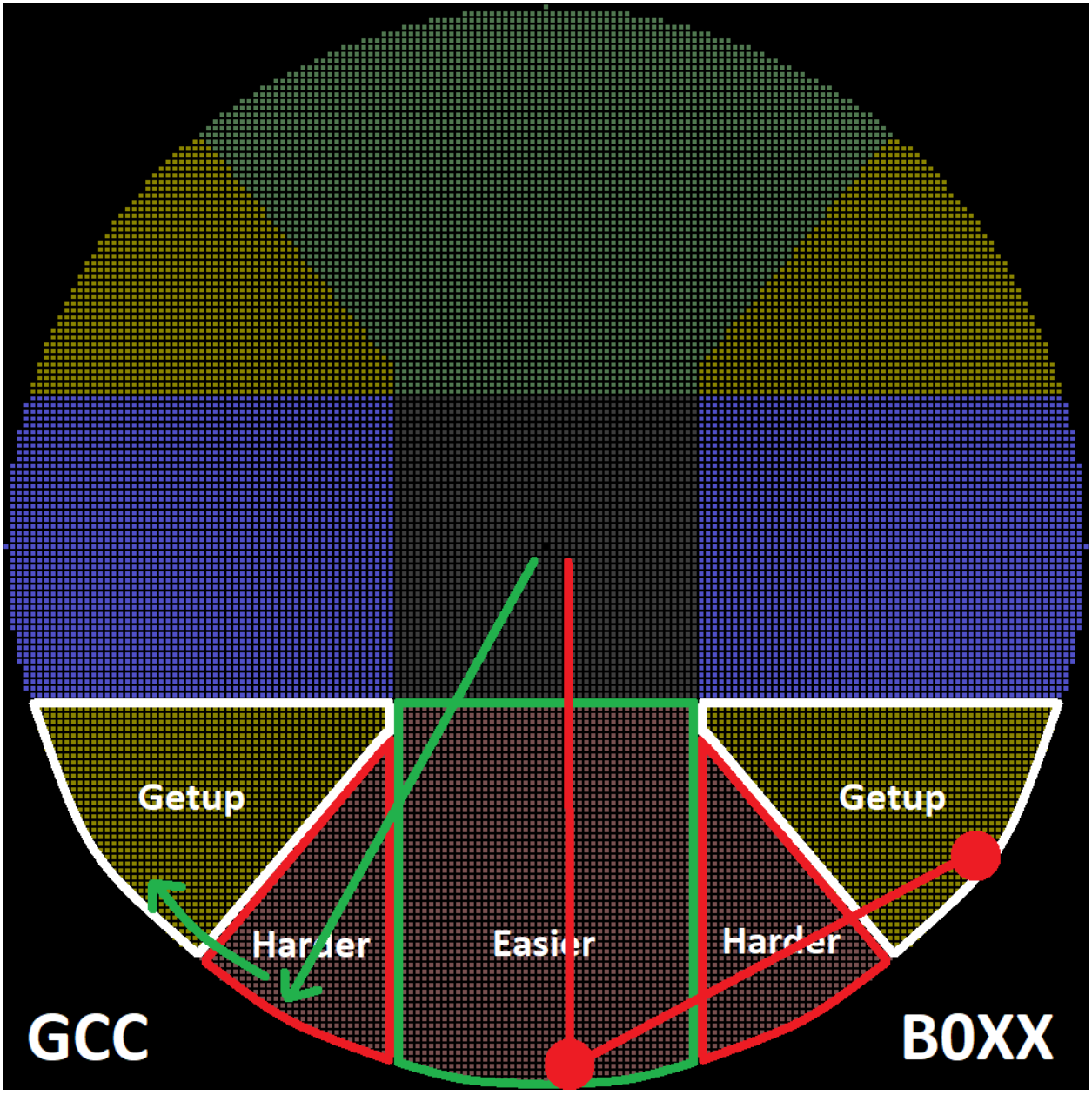
In Section 10.2.1, I examined the theoretical best timings for the (no-fast)fall, jump, and airdodge of a ledgedash. **While following the steps provided in Section 10.2.1 will increase your ledgedash consistency*, doing so will stunt your ledgedash potency.** This is because, on a human level, all of the no-fastfall from ledge methods (on the Gamecube controller, B0XX, and Nunchuk B0XX) are suboptimal for jumping with trajectory (refer to Section 10.2.1); therefore, the most potent ledgedash method does not look to no-fastfall from ledge.

As explained in Chapter 6, jump trajectory is a vital component of ledgedashing. Jump trajectory will always contribute to the distance of a ledgedash, but in some cases it can even determine a ledgedash's success. For these reasons, it is best to prioritize jumping with trajectory (preferably as much as possible) in some situations. **This is done by jamming the analog stick into > 50° territory in order to avoid ledge get-up (< 50°) while also picking up an X-value either prior to, or alongside the jump frame of your ledgedash. In the process, you will most likely traverse Y -.6625 (fastfall) due to the forceful nature of this motion, which means the second most intangible ledgedash will not be an option.** This is inconsequential, however, if you are confident in your ability to perform the most intangible ledgedash.

*Sometimes, a lack of jump trajectory will cause a ledgedash to fail; therefore, the various no-fastfall from ledge methods only

increase the consistency of ledgedashes that don’t require jump trajectory to succeed.

In particular, it is crucial that you jump with trajectory on the harder stages (FD, YS, PS) in order to reduce the shallowness needed on your airdodge. On the B0XX, this correlation still exists, but it behaves in a much more fixed manner. **Since the B0XX’s airdodge angles are set in stone, jumping with trajectory isn’t just recommended, but required for certain ledgedashes.**



Whereas the Gamecube controller can always initiate a fall in > 50° territory, the B0XX must fall elsewhere to perform < 50° ledgedashes. This makes the B0XX less effective than the Gamecube controller at jumping with trajectory.

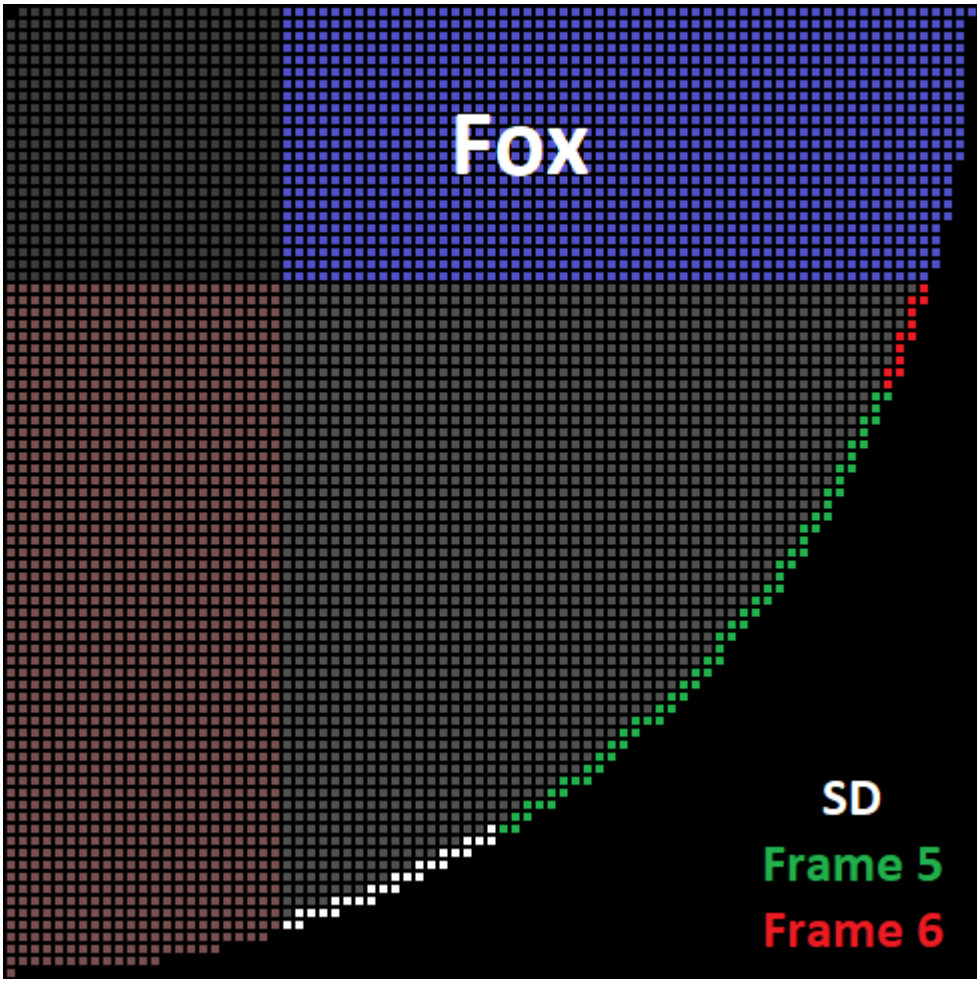
The dilemma with this is the B0XX is worse at jumping with trajectory than the Gamecube controller. With an analog stick, you're always able to aim for X-value territory as early as the fall frame (frame 1) of a ledgedash. Since > 50° territory is valid for both the fall *and* the X-value, it is possible to satisfy all of your criteria before the jump frame (frame 2 or later) even takes place. **This effectively creates a 2-frame window (at minimum) to jump with trajectory.**

On the B0XX, falling in > 50° territory during a < 50° ledgedash sequence is impossible. 50° Line Integrity (one of the non-dedicated modifier restrictions) ensures this in order to preserve the challenge of jumping with trajectory; however, this results in it becoming more difficult to jump with trajectory on the B0XX than on the Gamecube controller. Since the B0XX *cannot* fall with the analog stick pointed forward and perform a < 50° ledgedash, it *must* point the analog stick forward on the jump frame. **This gives the B0XX only a 1-frame window to jump with trajectory when performing the most intangible < 50° ledgedashes.**

Fox in particular is crippled by this 1-frame window because his frame 5 30.5° ledgedash *requires* him to jump with trajectory on the harder stages (see Section 6.2.1). This is mitigated by the fact that **the 30.5° airdodge allows Fox to perform his *second** most intangible ledgedash on frame 6.** The jump frame for this ledgedash takes place on frame 3, which gives the player a much more reasonable 2-frame window (frames 2 and 3) to jump with trajectory. It is a must that the B0XX is permitted the airdodge coordinates X +/-.8500 Y -.5000 (30.5°), as Fox's ledgedash capabilities on the harder stages are astonishingly poor without them. **In general, the B0XX is more dependent on the second most intangible ledgedash than the Gamecube controller.**

*Technically, a frame 6 ledgedash is Fox's *third* most intangible ledgedash (and a frame 5 ledgedash is his second). Through ECB manipulation, Fox is able to ledgedash on frame 4. This ledgedash

has not been mentioned throughout this document because it cannot be performed in most situations.



When performing Fox's frame 5 ledgedash, don't airdodge in the final ~7°!

So long as a Gamecube controller user is aware of their character's intangibility thresholds, their controller is inherently better than the B0XX at performing the most intangible ledgedashes. Jumping with trajectory will not only cause these to travel further, but in some cases, make it on-stage.

[11.1.2] Actuation Time

One of the biggest misconceptions about digital inputs is that their lack of a travel route grants them an actuation time advantage over an analog stick. This would appear to be the case based on a comparison of the two input methods' analog X/Y readings. Whereas an analog stick motion gets polled at several points along the way, digital inputs skip directly from point A to point B. This creates the illusion that digital inputs actuate faster.

The error in this reasoning is that **travel time and actuation time aren't the same thing**. Travel time specifically pertains to how long it takes to skip from point A to B *once the analog X/Y-coordinates have been actuated*. Actuation time, on the other hand, measures how long it takes for the decision to influence the analog X/Y-coordinates to occur in-game in the first place. The latter not only can't be measured in-game, but most certainly has a travel route; it just happens to be on a physical level.

In this case, the actuation time in question is the duration of a B0XX button being physically reached and pressed. While there are tools that can be used to measure an individual controller's actuation time (such as an oscilloscope), this isn't a relevant statistic. Since we are only interested in *comparing* two controllers, the best method is to simply have the same person actuate both of them simultaneously.

In December 2017, I conducted several Gamecube controller vs. B0XX actuation time tests to compare how quickly the two controllers initiated a dash (X-smash) (50 attempts per trial). To ensure accurate results, the following assumptions were made:

> -The Gamecube controller did not have P.O.D.E. (a potentiometer malfunction that causes wonky X/Y-axis readings).
>
> -The B0XX uses Sanwa OBSF-24 buttons. These have a 30g actuation force (the lowest of any arcade button on the market).

-Starting positions were standard. My left thumb rested atop the center of the Gamecube controller's analog stick, while my right index finger hovered 1cm* above the B0XX buttons.

*Maintaining this distance is necessary when playing on the B0XX. Without it, you cannot alternate the horizontals effectively.

-Both controllers were actuated swiftly and comfortably.

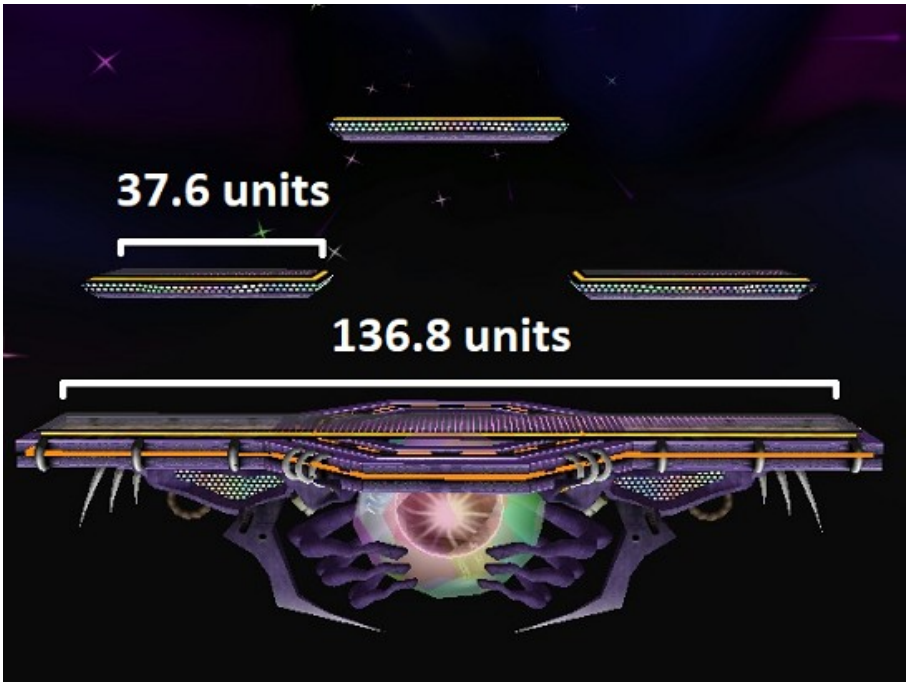
There are also some key pieces of information to convey:

-Only the Gamecube controller can be polled in X-tilt when attempting a dash. This is because an analog stick must traverse X-tilt to reach X-smash, whereas digital inputs skip from the deadzone to X-smash. X-tilt actuates faster than X-smash, since it is closer to the center of the analog stick. X-tilt inputs are relevant, and will be tallied.

-Only dash forward was performed. Dash forward actuation time was then extrapolated to assess dash back actuation time with 100% accuracy (since dash forward actuation time tells us X-tilt and X-smash actuation time).

-Our dash back assessments will operate under the assumption that UCF is on. UCF makes X-tilt valid for the turnaround frame of a dash back, effectively reducing dash back's actuation time on the Gamecube controller.

To put these actuation time differences in perspective, the advantages in units traveled along the X-axis will be listed (image for reference):



Lastly, Fox was the character used. Here are the the possible outcomes he can receive (the units traveled are approximations):

Dash forward (X-tilt)
+.2 units on frame 1
+.2 units on frame 2
+2 units on frame => 3

Dash forward (X-smash)
+0 units on frame 1
+2 units on frame => 2

Dash back (UCF) (X-tilt)
+0 units on frame 1
+2 units on frame => 2

Dash back (UCF) (X-smash)
+0 units on frame 1
+2 units on frame => 2

Here are the results from the first trial:

GCC X-tilt 2 frames before B0XX X-smash: 7 occurrences
Dash forward: GCC 2.4 units advantage
Dash back (UCF): GCC 4 units advantage

GCC X-smash 1 frame before B0XX X-smash: 5 occurrences
Dash forward: GCC 2 units advantage
Dash back (UCF): GCC 2 units advantage

GCC X-tilt 1 frame before B0XX X-smash: 23 occurrences
Dash forward: GCC .4 units advantage
Dash back (UCF): GCC 2 units advantage

GCC / B0XX X-smash on same frame: 5 occurrences
Dash forward: Tie
Dash back (UCF): Tie

GCC X-tilt / B0XX X-smash on same frame: 10 occurrences
Dash forward: B0XX 1.6 units advantage
Dash back (UCF): Tie

In this particular trial, the Gamecube controller averaged a 1.04 units advantage on dash forward, and a 1.68 units advantage on dash back (UCF).

All trials were consistent with these results. While this can't be seen on-screen, **the Gamecube controller has a clear actuation time advantage over the B0XX on UCF.** At tournaments running on Arduino adapters (which fix dash back by killing 1 frame of X-tilt), this advantage is mostly neutralized.

[11.1.3] Wank Smash DI

Popular within the Smash 64 community, wank SDI is an aptly named technique that allows you to generate SDI inputs as rapidly as humanly possible. Recently, Wizzrobe has shown that while it may not be as necessary, wank SDI is just as effective in Melee.

The only way to survive in Smash 64.

Wank SDI is performed by situating the analog stick against the rim with your left thumb, then vibrating the controller back and forth with your right hand. This requires you to slightly adjust your grip on the controller, but the reward is worth it. Wank SDI will generate SDI inputs every 3-4 frames (i.e. frame 1, 5, 9, 12, 16...) with the Gamecube controller, and can be performed indefinitely. In terms of speed, this is on par with a burst of quarter-circle SDI, but it comes with the advantage of removing the need to time your SDI inputs.

The B0XX does not have a wank SDI equivalent.

[11.2] Analog

The Gamecube controller has higher potential in several areas due to it having the full range of analog X/Y, C X/Y and L/R-values.

[11.2.1] Lightshield

Whereas the Gamecube controller contains the full range of analog L/R 43 through 140, the B0XX is only able to pinpoint L/R 49.

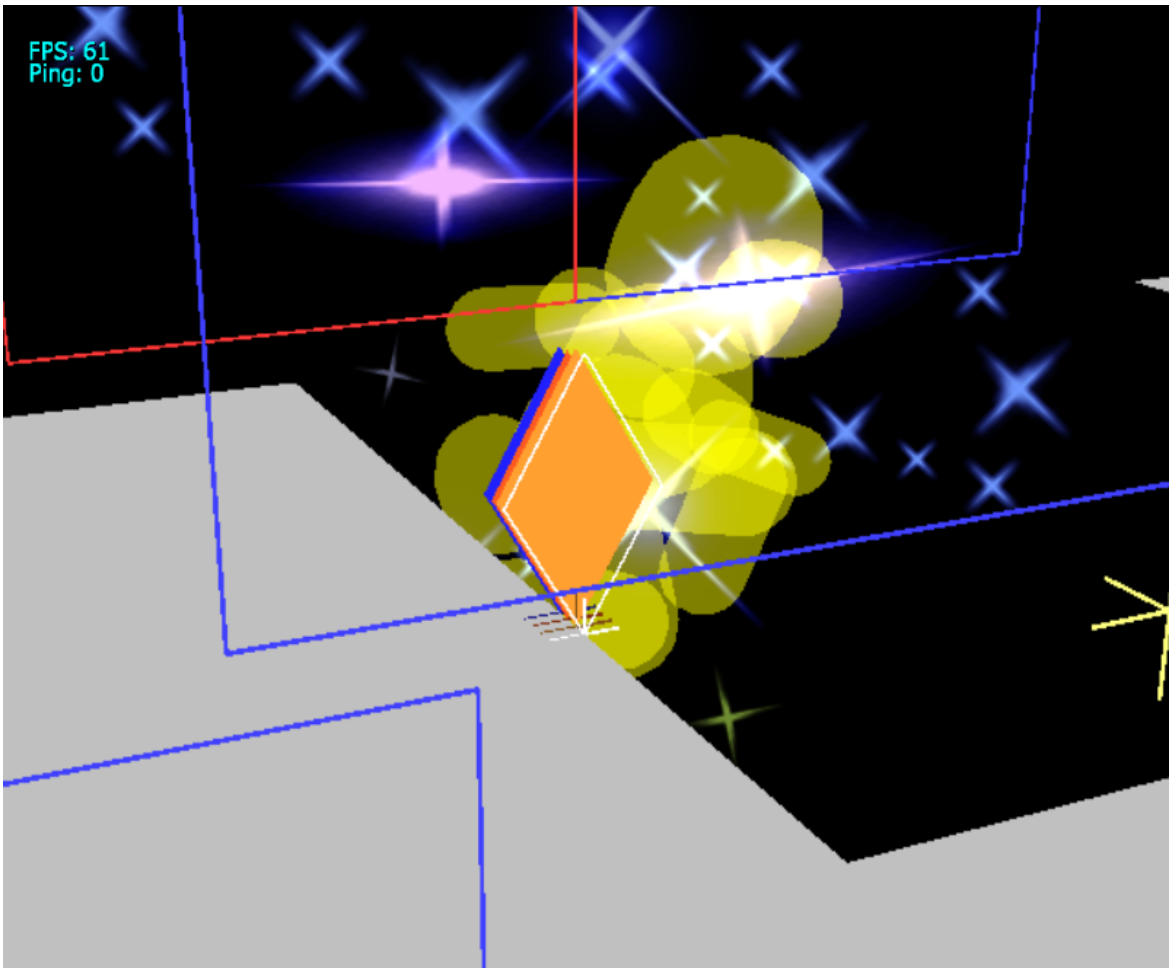
[11.2.2] Trajectory DI

While the B0XX *can* TDI (trajectory DI) at angles that aren't 45° (through the use of Firefox coordinates, wavedash coordinates, etc.), this feature not only isn't formally supported, but suffers from a massive intuition disadvantage. Overall, this isn't a concern, as TDI'ing in increments of 45° is surprisingly adequate.

The B0XX's main TDI-related disadvantages stem from its horizontal TDI capabilities (or lack thereof). While these are similarly unintuitive, the real problem is the shortage of options. Since the B0XX only contains X .2875, .4375, .5875, .7375, and 1.0, **it cannot perform ambiguous DI mix-ups**. Perhaps even more importantly, **slide-off DI (influencing your character to slide off the end of a stage or platform in order to regain actionability) is often impossible**. These two disadvantages simplify the punish game for your opponent very frequently.

[11.2.3] Automatic Smash DI

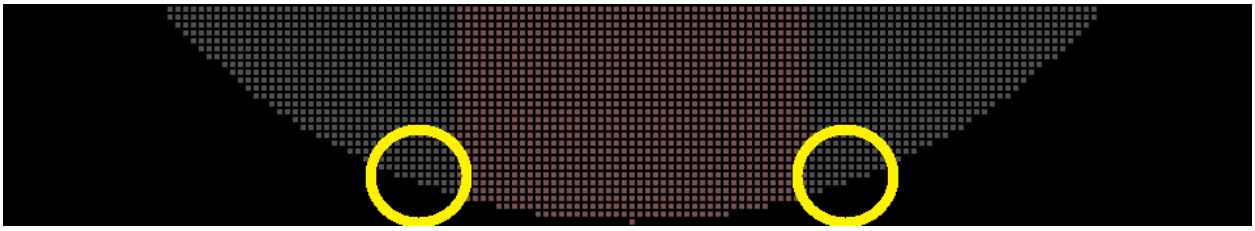
The B0XX's C-stick can only pinpoint C X or Y +/-1.0, C X +/-.7000 Y +/-.7000, and C X +/-.9500 Y +/-.3000 (for UF/DF-smash), which means it is missing the vast majority of its coordinate plane. This limits the B0XX's ASDI (automatic smash DI) options.



Fox's ECB (the orange diamond) is not above the platform during frames 31-40 of his forwards techroll.

For the most part, this isn't an issue, as it is usually best to ASDI in one of the four cardinal directions with C X or Y +/-1.0. However, there is a notable exception that pops up frequently in competitive play. In techroll situations like the one shown in the image, it is common to use DSDI (double-stick DI, a technique that utilizes TDI from the analog stick and ASDI from the C-stick

at once to prompt a platform slide-off) in response to being attacked by your opponent. Usually, C Y -1.0 (straight down) is the ideal ASDI coordinate for this, but in certain cases this won't work. With Fox's forwards techroll, for example, Fox's ECB shifts slightly off the platform on frames 31-40. If C Y -1.0 is used here, a platform slide-off will not occur.



ASDI'ing in these regions (with the C-stick) will shift Fox's ECB back onto the platform while preserving as much downwards influence as possible.

A few years ago, tauKhan deduced that downwards diagonal ASDI could be used to DSDI with Fox (and potentially other characters) in this situation. C X +/-.2875 Y -.9500, for example, causes downwards influence in addition to X .2875 horizontal influence. This horizontal influence is usually enough (more is needed at higher %'s) to shift Fox's ECB onto the platform and cause DSDI to succeed.

Although the B0XX can pinpoint C X +/-.7000 Y -.7000, these coordinates are not ideal for this technique, since their Y-value of -.7000 only shifts Fox by 2.1 units along the Y-axis (ASDI = X/Y * 3) when he is attacked. The maximum diagonal Y-value of -.9500 (which shifts Fox by 2.85 units) causes DSDI to succeed until much higher percents.

As with several other areas of the game, the B0XX would most likely be too precise if given the ability to perform DSDI with steep ASDI coordinates. This technique will probably always remain a Gamecube controller-exclusive.

[11.2.4] Walk/Run

Whereas the Gamecube controller can walk with X .2875 through 1.0, the B0XX can only walk with X .2875, .7375, and 1.0 (without the use of A/B/C/L/R/X/Y/Z).

Additionally, X <= .6125 will cause runbrake (the mechanic that causes run to cease) once your character is in the run state. Whereas the Gamecube controller can vary its run between X .6250 and 1.0, the B0XX can only modify run to X .7375.

[11.2.5] Firefox

With the Gamecube controller, you are able to choose from hundreds of Firefox angles. These allow you to weave around edgeguards and sweet-spot the ledge whenever you are in range.

With the B0XX, you are only able to choose from 48* Firefox angles. Since these angles are approximately 4.6° apart from each other, weaving around edgeguards usually means settling for an imperfect one. Similarly, sweet-spotting the ledge is hindered, although measures can be taken to mitigate this. Most of the time, the best strategy is to begin your up-B in a location that offers you the option to sweet-spot the ledge. This solves one problem, but not without creating another: the act of getting to this location can give your opponent the extra few frames they need to set up a successful edgeguard.

Additionally, the B0XX is incapable of selecting a Firefox angle within 4.5° of the shallowest/steepest angle in the game.

*Technically, the shield tilt coordinates (X +/-.7500 Y .6500 and X +/-.7250 Y -.6875) and airdodge coordinates (X +/-.8500 Y +/-.5000, X +/-.5000 Y +/-.8500, X +/-.8750 Y .4750, and X +/-.4750 Y .8750) make for a total of 64 Firefox angles.

[11.2.6] Airdodge

The biggest disparity between the Gamecube controller and B0XX is their airdodge capabilities. Between inherent disadvantages and the need for a hard cap on its upper/lower limits, airdodge brings out the worst in a controller that lacks analog control.

Usually, I am of the opinion that the B0XX's intuition-related disadvantages become nonfactors as one gains mastery over the controller. Wavedashing is the main exception to this rule. Since a wavedash must be aimed instantaneously, it is extremely difficult to determine when a 45° or 59.5° airdodge should be chosen over 30.5° (which most players will find themselves defaulting to). **An arbitrary disadvantage also comes in the form of being unable to change your mind once you've committed to a wavedash angle.** On the Gamecube controller, a spontaneous decision can be made to readjust the analog stick during jumpsquat. This cannot be done on the B0XX, since it is usually impossible to shift between the modifier buttons so quickly.

Similar to sweet-spotting the ledge with Firefox, certain starting points are preferable when airdodging with the B0XX. When recovering, the especially useful 61.5° (steepest) airdodge must begin at a specific elevation relative to the stage in order to minimize the amount of time your character hovers in the air before landing. Grapple characters (Samus, Link, and Young Link) are hindered in their ability to align their recoveries with the ledge for similar reasons. Most notably, microspacing with wavedashes does not exist. Since the B0XX is only capable of airdodging at three different angles, the ideal wavedash length usually isn't present. It is recommended that you adapt a neutral game centered around microspacing with dashes instead.

Finally, none of the aforementioned disadvantages would have mattered had the B0XX been allowed to wavedash at 16.8°. In giving the B0XX its most necessary nerf, its shallowest wavedash angle goes from being humanly unrealistic to underwhelming. 30.5° may be viable, but any pro would have faith in their ability to consistently beat this benchmark with the Gamecube controller.

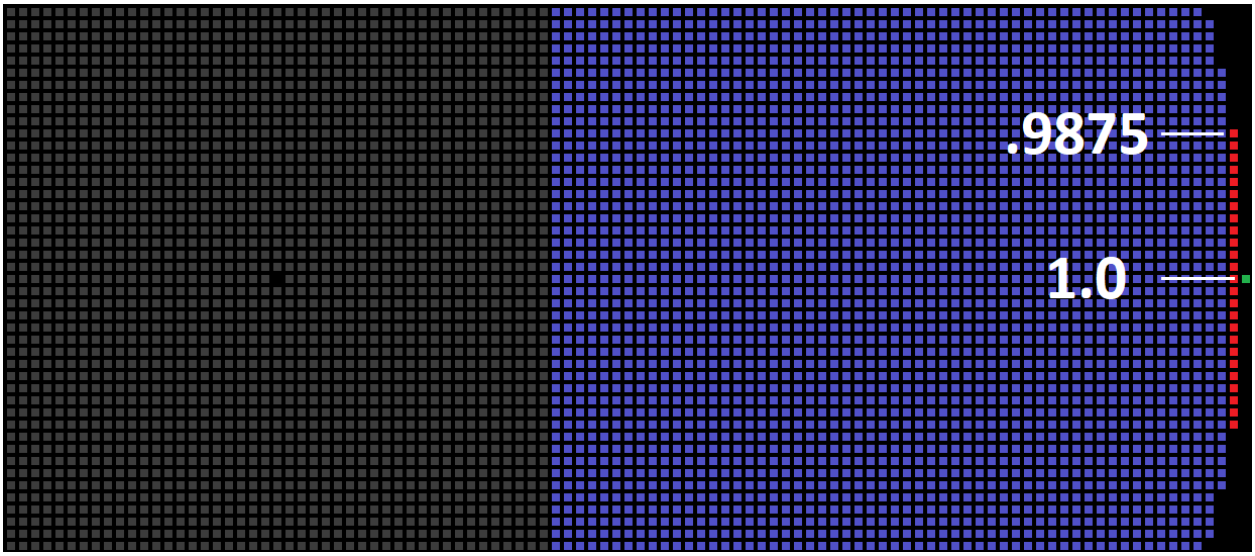
****

[12] 1.0 Cardinal

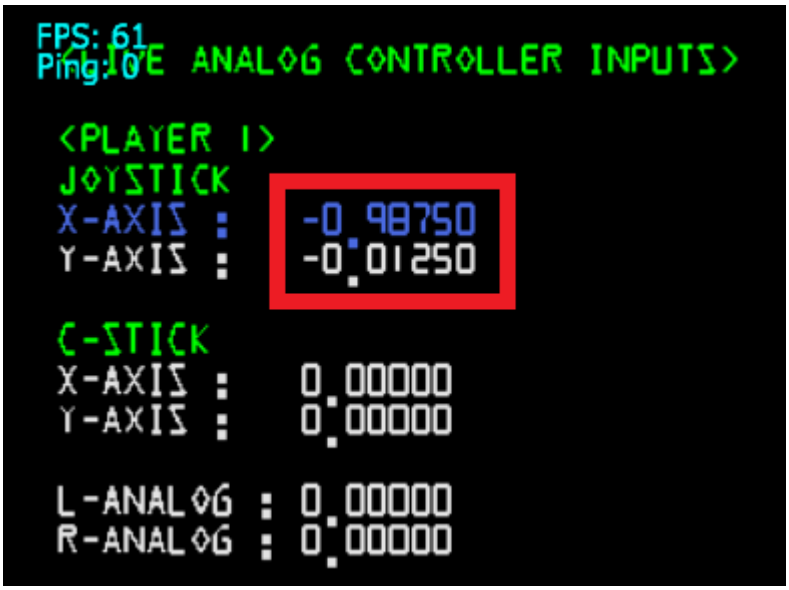
****

[12.1] Overview

Throughout this document, I refrained from acknowledging the 1.0/.9875 cardinal disparity due to how insignificant, yet lengthy of a subject it is. Ultimately, this is a game mechanic where digital inputs are *marginally* better than the ideal analog stick as it stands. It is unfair to isolate them, however, due to the glaring differences in 1.0 cardinal efficacy among Gamecube controllers.



The root of the problem is a decision made by the game developers that can be considered questionable at best. Within each cardinal (on both sticks), there is a generous range assigned to the value .9875, yet only a single set of coordinates assigned to 1.0. Being the greatest X/Y-value in the game, the latter is naturally more desirable in most situations. When running, for example, Fox will reach a peak acceleration of 2.17 units/frame with .9875, as opposed to 2.20 units/frame with 1.0. While this is an almost unnoticeable difference, the fact remains.



A Gamecube controller's analog X/Y-values being read in 20XX 4.07.

In a perfect world, the 1.0 cardinal could have added an element of skill to the game. In reality, however, this isn't the case. Due to how tiny the 1.0 cardinal's range is, pinpointing it is almost entirely dependent on hardware. The Gamecube controller shown in the picture, for example, will consistently receive X -.9875 Y -.0125 when its analog stick is pointed to the left. This consistency stems from a plastic case with sharp corners (these help situate the stick in a specific location), while X -.9875 Y -.0125 stems from misalignment due to manufacturing variance. **Had it not been for this misalignment, this controller could have consistently pinpointed the 1.0 cardinal.**

Based on this information, it is clear that finding the 1.0 cardinal varies from controller to controller. The next issue, ironically, has to do with *another* developer's decision.

At the moment, the competitive Melee scene has displayed widespread acceptance of Universal Controller Fix, a game mod intended to fix inconsistencies in Gamecube controller performance. Generally, these inconsistencies stem from poorly designed game mechanics. Dash back, which gives a 1-frame window for an analog input, is an oversight by any developer's standards. Likewise, shield drop, which has a miniscule range of

3 Y-values, was presumably shafted late into Melee’s development by the inclusion of spotdodge. These are the Gamecube controller’s two most notorious sources of inconsistency, as well as the ones most desperately in need of repair.

UCF fixes these two mechanics then closes its doors, despite it not necessarily being correct to do so. Lesser sources of inconsistency, namely the 1.0 cardinal, remain unaddressed as of the current version. This decision appears to have been made for two reasons.

For one, fixing the 1.0 cardinal simply isn’t in demand. Unlike with high dash back % and shield drop notches, there has never been widespread incentive to seek out and/or vend controllers that possess 1.0 cardinals (despite the fact that they can be notched). Those who don’t believe UCF should operate on an objective metric tend to dismiss 1.0 cardinals on this basis alone. In patching a game, it is common practice to target mechanics that necessitate the patch in the first place rather than make every conceivable improvement. From this perspective, 1.0 cardinals do not make the cut.

The issue with this view is that it practically concedes that the 1.0 cardinal meets UCF's criteria, yet dismisses the 1.0 cardinal due to a lack of demand, an entirely arbitrary factor. This decision is then justified by a philosophy that applies to modern-day games that often have *thousands* of imperfections to attend to, which should not apply to UCF, a mod intended to serve a niche purpose within a game that has a handful of controller-related problems at most. The lack of demand is especially meaningless given the lack of knowledge surrounding the 1.0 cardinal. Had any effort been made to educate the public about this mechanic, demand for it very well could have been there.

This leads to the second counterargument to the 1.0 cardinal’s inclusion, which is founded on falsehoods. Many proponents of UCF, including the dev team themselves, have propagated the belief that UCF merely causes controllers to tie the theoretical best hardware (without exceeding it in any capacity). Under this premise, the 1.0 cardinal receiving a redesign can be dismissed on the basis of the 1.0 cardinal becoming easier to pinpoint than

on vanilla. The issue with this is that **UCF already exceeds the best vanilla controller in several ways.**

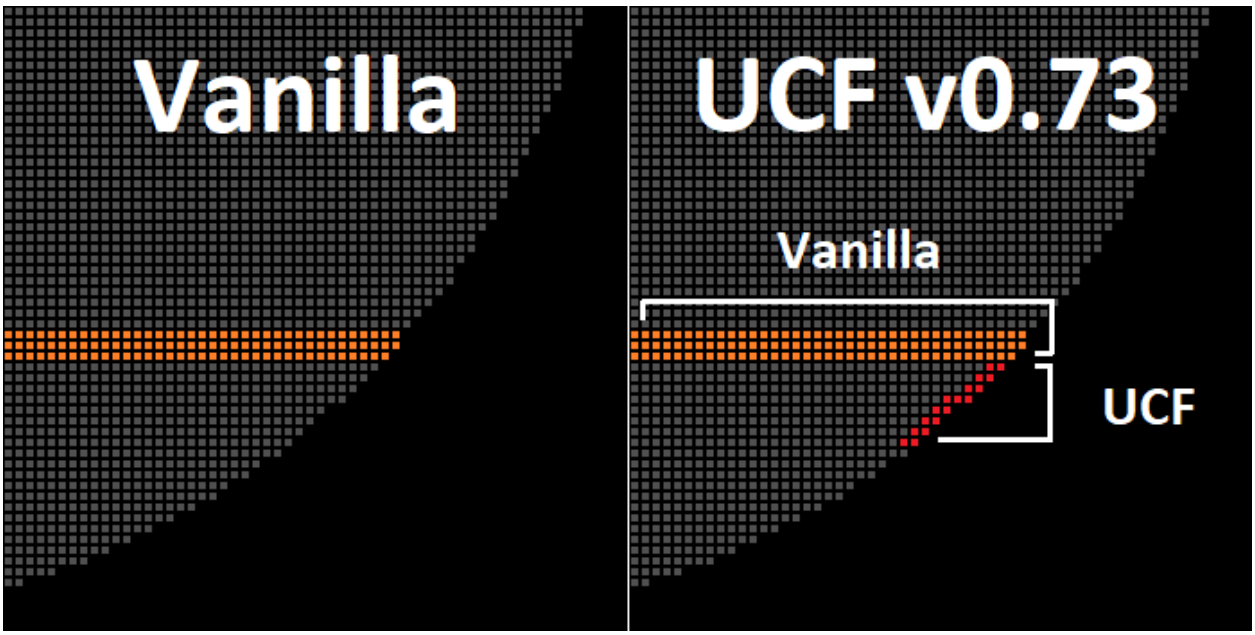
One way to debunk the claim that UCF doesn't exceed the best hardware is to examine the unavoidable byproducts of fixing dash back and shield drop. In Section 5.1.3, for example, I showed why it is impossible for shield drop notches and jab cancel notches to coexist on vanilla. **By increasing shield drop's range, it becomes possible to notch for both on UCF.** Even more striking are the implications of fixing dash back, a change that flips the controller lottery on its head. On vanilla, highly elusive controllers "suffer" from a potentiometer malfunction known as P.O.D.E. This gives them high dash back % at a cost. P.O.D.E. compromises pivoting, dashing out of crouch, and several other areas of the game. **On UCF, no such trade-off is necessary. Since dash back is distributed on a software level, it is possible to reap the benefits of P.O.D.E. without actually having it.** This not only devalues P.O.D.E., but also lets you have the best of both worlds.

While these byproducts are telling on their own, there is no need to delve so deeply. The easiest way to debunk the UCF team's claim is to examine UCF's redesigns of dash back and shield drop themselves. **Formerly, no Gamecube controller could successfully dash back 100% of the time or perform shield drops with as little finesse as UCF requires.** Even though UCF isn't *that* big of a jump from a high P.O.D.E controller (95-98% dash back) with Y -.6750 notches (extremely easy shield drops), this is more so meant to illustrate a point. **In order to fix inconsistencies, UCF exceeds the best possible hardware.**

How *else* could UCF fix inconsistencies? If a mechanic is broken, its redesign inherently has to set the bar higher. Once this has been established, UCF's rationale must be re-evaluated entirely. **The essence of this mod isn't to tie the best hardware, but to redesign mechanics that make it impossible to achieve equal performance through hardware.** Under this premise, the fact that a controller with eight 1.0 cardinals (four on each stick) is nearly impossible to maintain is no longer a relevant detail. The bottom line is that **by not redesigning the 1.0 cardinal, we are knowingly accepting a less fair version of Melee.**

[12.2] Redesign

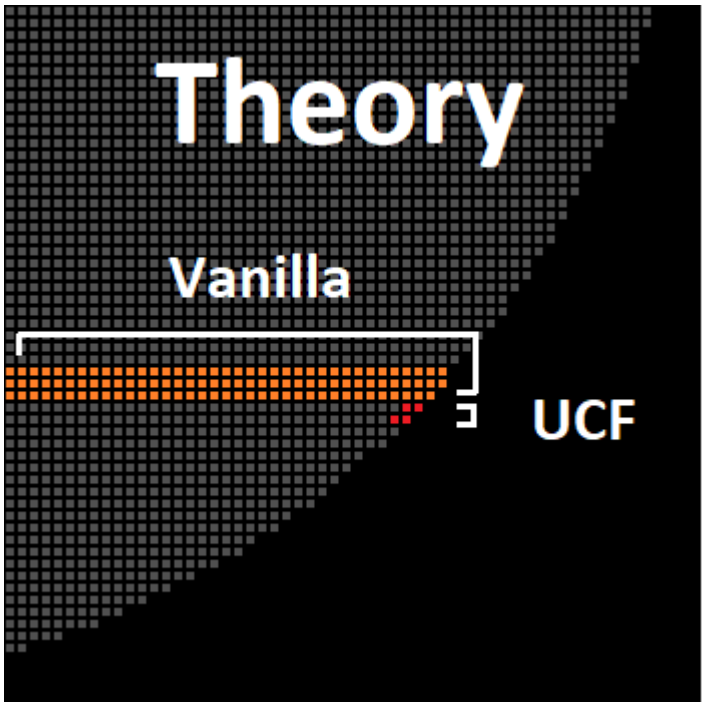
Before I can show how the 1.0 cardinal should be redesigned, there is a crucial concept I must illustrate.



Shield drop ranges on vanilla (Y -.6625 through -.6875) and the current version of UCF (Y -.6625 through -.7875).

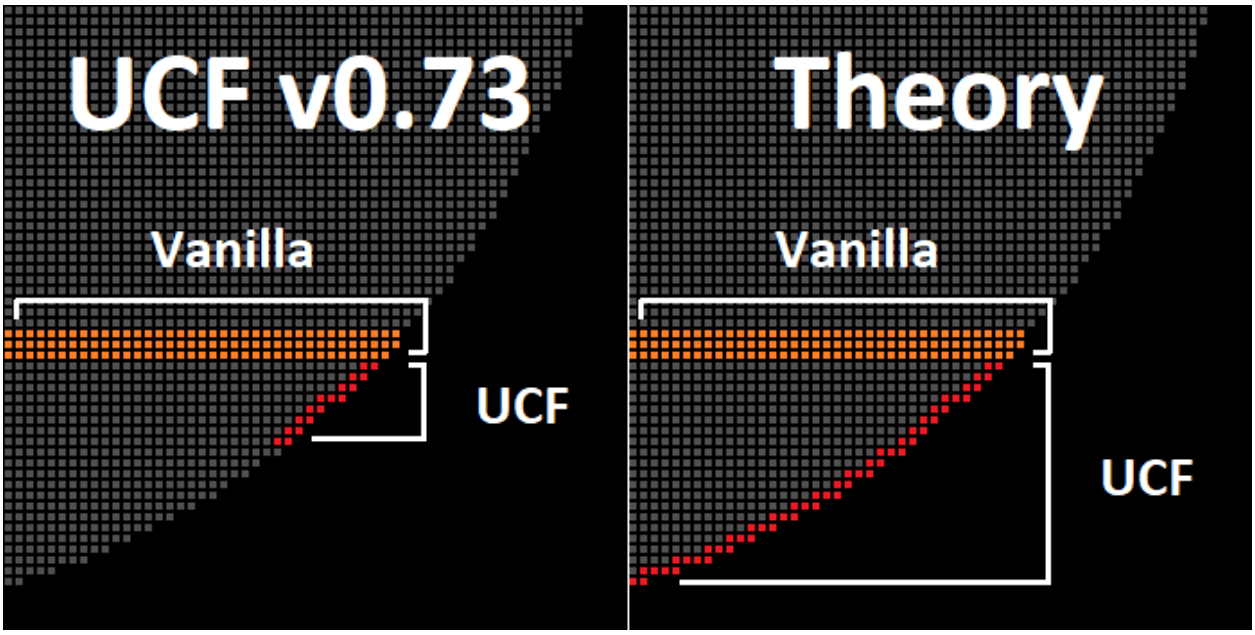
When UCF was first introduced, a common complaint was that "shield dropping [had been made] too easy." After all, UCF had given it a whopping *11* Y-values to vanilla's 3. While this change allowed everyone to experience the joy of shield dropping, it was initially subject to criticism. Many people felt that shield drop's range should have been increased, but not by this many Y-values.

The hole in this logic is that it evaluates the shield drop mod based on the number of Y-values it converts, rather than how difficult it is to shield drop under the mod. The former is an entirely superficial statistic that can be misleading at face value. This is because, just like on vanilla, two controllers that have the same shield drop range on a software level aren't necessarily on an even playing field.



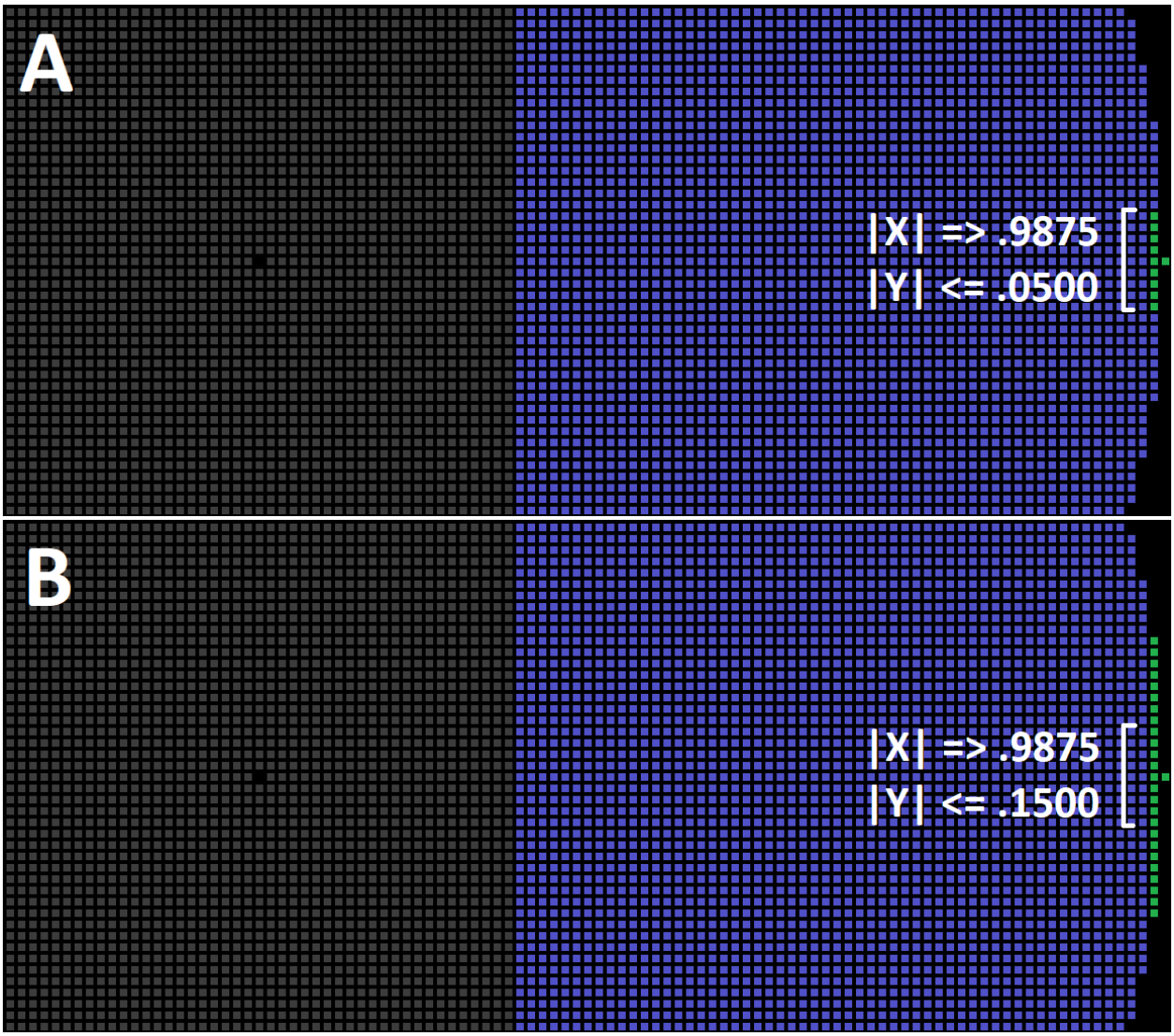
A theoretical shield drop range UCF could have used. 2 Y-values (-.7000 and -.7125) are added to vanilla's 3. This would have failed to achieve equality among controllers.

Say, for example, UCF increased shield drop's range by only 2 Y-values. This would have had a drastic effect on some controllers, but none on others. Whereas controllers with corners centered on Y -.7000/-.7125 would have gained the ability to shield drop, controllers with Y <= -.7250 would have been unaffected. This mod would have failed to encompass a significant percentage of controllers.



Despite their differences, these shield drop ranges are equally easy to pinpoint on most controllers. Either of these would have been acceptable.

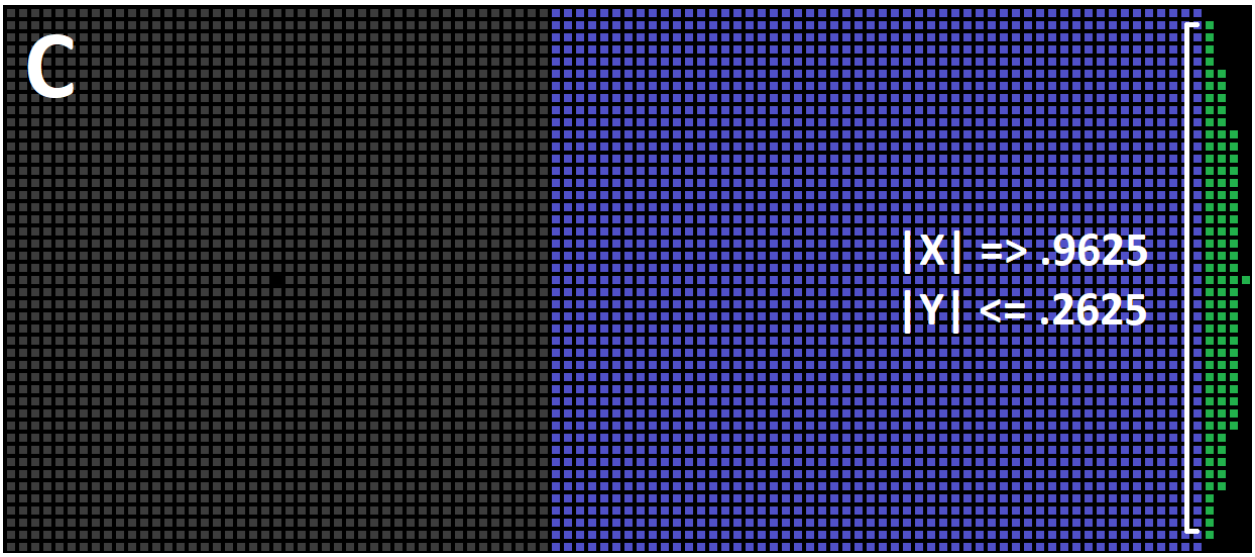
For this reason, **a redesign can only be correct if it guarantees all controllers a 100% success rate**. UCF achieves this with its overwhelmingly generous shield drop range of 11 Y-values (top left). It should be understood, however, that *there would have been nothing wrong with UCF going even further* (top right). The only thing to be wary of when redesigning shield drop is shrinking spotdodge's range in exchange. This justifies keeping shield drop's redesign to a minimum.



1.0 cardinal redesigns that I do not recommend.

I took the time to establish this concept in order to convey why **neither of the above 1.0 cardinal redesigns make sense.** It should be obvious that Redesign A, which converts an additional 9 sets of coordinates to 1.0, is a far cry from being acceptable. Sharp corners are still entirely necessary under this redesign, and misalignment remains a significant disadvantage. A controller that begins with a worn down case and X .9875 Y .0375 (to the right), for example, is far more likely to exit this modified 1.0 range than one that begins with a sharp case and X .9875 Y .0125. Redesign A would be adequate for some controllers, but not most.

Redesign B, on the other hand, is by no means *bad* - it just isn't logically consistent. Its additional 25 sets of coordinates would encompass the majority of controllers, but performance would still vary ever so slightly. Redesign B is suggestive of wanting to preserve some sort of difficulty in pinpointing the 1.0 cardinal, when in reality very few controllers would still be subject to this. Most controllers would already be well outside the realm of ever missing.



All X/Y and C X/Y-values that are => .9625 should be converted to 1.0. This is a non-arbitrary cutoff, since => .9625 is cardinal-exclusive (.9500 can be reached in the quadrants).

Eventually, things slippery slope their way into justifying Redesign C. Unlike shield drop's redesign, there is no trade-off (spotdodge's range) to be wary of when redesigning the 1.0 cardinal; Redesign C sacrifices the values .9625 and .9750, which are obsolete to 1.0. There is no reason not to do this, since it helps ensure a 100% 1.0 cardinal success rate across all controllers; therefore, **Redesign C is the 1.0 cardinal redesign I encourage the UCF team to implement.** If Redesign C (or even Redesign B) went through, the Gamecube controller's 1.0 cardinal efficacy would tie that of digital inputs, at which point the 1.0 cardinal would no longer be a point of contention.

[12.3] Plan B

While I believe the most logical course of action is for the 1.0 cardinal to receive a redesign, I am aware that this doesn't necessarily mean the UCF team will implement one. Political decisions like these aren't always about correctness. If the UCF team chose to redesign the 1.0 cardinal, it would (ironically) be viewed as the biggest update to their mod yet. This may or may not be a risk they are willing to take.

In the event that the 1.0 cardinal is not redesigned, the Gamecube controller's ability to pinpoint the 1.0 cardinal could only be equal or worse to that of a digital controller. Despite this, **there are several reasons the B0XX should be permitted 1.0 cardinals (rather than .9875) regardless of the UCF team's decision**. Some of these are relatively straightforward, while others involve more complex interactions.

The most direct way to justify the B0XX being permitted 1.0 cardinals is to assess the ideal Gamecube controller's 1.0 cardinal efficacy. This is to ask, "How much worse than the B0XX is the Gamecube controller [with the best hardware] at pinpointing the 1.0 cardinal?" **The answer is: *not much.*** Despite the 1.0 cardinal's microscopic range, lucky alignment and sharp corners will make a Gamecube controller incredibly consistent at pinpointing it. While this document does not contain video evidence, this can be surmised by verifying that a Gamecube controller with sharp corners does in fact get polled at the same set of coordinates consistently; therefore, **it is within the rules to use a Gamecube controller that most frequently receives the 1.0 cardinal in all four directions (on both sticks). If we are to then base the B0XX off of what is theoretically legal (the only rational approach to take), restricting its cardinals to .9875 would be far more extreme than guaranteeing them 1.0.**

At this point, the remaining counterargument is that the B0XX is *still* more efficient than the Gamecube controller at pinpointing the 1.0 cardinal. While this is true, this shouldn't be taken at face value. Similar to how I encouraged you to evaluate the redesigned shield drop ranges in Section 12.2 based on their

difficulty rather than their size, **the 1.0 cardinal should be evaluated based on its implications, not its magnitude.**

At first glance, giving the B0XX .9875 cardinals would appear to be consistent with the trend of making the B0XX equal to or worse than the Gamecube controller wherever possible; however, this line of reasoning fails to consider what makes 1.0 cardinals a competitive concern in the first place. Among the areas of the game affected by the 1.0 cardinals, horizontal movement speed is paramount. As shown in Section 11.1.2, the B0XX is *already* at an actuation time disadvantage in this area that outweighs the advantage gained by having 1.0 cardinals. Based on this statistic alone, it is reasonable to permit the B0XX 1.0 cardinals as compensation.

Fox's frame 6 ledgedash does not succeed on the harder stages with a 30.5° airdodge and X .9875 jump trajectory.

Furthermore, there is an interaction involving Fox that tips the scales even more heavily in favor of the B0XX being permitted 1.0 cardinals. In Section 6.1.3, I explained that the B0XX was given

the airdodge coordinates X +/-.8500 Y -.5000 (30.5°) so that Fox could ledgedash on frame 6 on the harder stages, and in Section 11.1.1 I elaborated on the necessity of this technique. Something I didn't mention in either of those sections, however, is that **Fox's frame 6 ledgedash only succeeds with a 30.5° airdodge on the harder stages if Fox jumps with X 1.0 trajectory (X .9875 jump trajectory will not work).** This is due to the inverse correlation between airdodge shallowness and jump trajectory. Since 30.5° is the steepest eligible angle for Fox's frame 6 ledgedash, it is only enabled by the strongest jump trajectory; therefore, **if Fox's jump trajectory is nerfed to X .9875, his airdodge angle must be buffed to X +/-.8625 Y -.5000 (30.1°) in order for his frame 6 ledgedash to succeed.**

This means that **nerfing the B0XX's cardinals to .9875 would have to coincide with a buff that would roughly negate this nerf's impact.** It must be understood that whether the combination of 1.0 cardinals and a 30.5° airdodge or .9875 cardinals and a 30.1° airdodge is chosen makes a negligible difference overall. Neither combination is better across the board; dash-reliant characters (i.e. Captain Falcon) would surely prefer the former combination, while wavedash-reliant characters (i.e. Luigi) would prefer the latter. It is, therefore, reasonable to endorse one or the other based on an assessment of game balance, at which point the combination of 1.0 cardinals and a 30.5° airdodge is clearly the healthier choice. This is single-handedly due to the fact that this combination encourages onstage gameplay with its stronger cardinals, whereas the combination of .9875 cardinals and a 30.1° airdodge encourages ledgedashing with its stronger airdodge angle. Naturally, the former favors the skill set we should be looking to test.

For all of the aforementioned reasons, it is clear that the B0XX should be permitted 1.0 cardinals.

****

[13] Conclusion

****

There are many genres of games in which analog inputs would never be able to coexist with digital inputs. It is a blessing that this is not the case with Super Smash Bros. Melee, the game perhaps most in need of them. As I found ways to replicate most of the Gamecube controller's intrinsic challenges over the course of the B0XX's development, it became increasingly apparent that this project was destined to happen. I can't say that this surprised me; having had this game since the day it came out, I have witnessed its divinity countless times. That being said, I am glad that this worked out.

There is no doubt in my mind that the B0XX should be tournament legal in its current iteration. Following its fine-tuning, the B0XX is a well-balanced controller that suffers from several inherent disadvantages but compensates for them with new flavor in other areas. If anything, this is the most we could have asked for from a third-party controller; with a physics engine as complex as Melee's, it is fitting that the B0XX provides players with a fresh experience.

As evidenced by its overwhelming demand, the B0XX will likely go down as one of the biggest leaps forward in the history of Melee. HAL Laboratory may have created a masterpiece back in 2001, but nothing is perfect; in recent years, we have come to embrace the fact that the future of this game lies in our hands. Software modifications and third-party controllers have gone from being radical ideas to household names as Melee remains alive as ever in 2018. It is paramount that the evolution of this industry continues to prosper with the legalization of the B0XX.

Thank you for reading,
Aziz "Hax$" Al-Yami

****

[14] F.A.Q.

****

Q: If the B0XX is legal, are any Gamecube controller PCB mods legal?
A: No.

Q: What does all of this mean for the legality of other digital controller brands?
A: This document does not endorse the legality of any controller that does not utilize the latest version of the B0XX software and abide by the B0XX ruleset.

Q: How will future updates to the B0XX's software be carried out?
A: Presumably, the Melee It On Me Competition Committee will serve as the hub for the B0XX's tournament legal parameters.

Q: How will tournament organizers verify that someone is running the latest version of the B0XX software?
A: At the moment, only a handful of people have a B0XX. Once the product is commercially available, verifying a controller's PCB will be a one-step process through a downloadable client on our website. The only tools needed will be a USB Type-C cable, a computer, and an internet connection.

Q: Should it be standard protocol to verify a B0XX user's PCB?
A: No. Unless there is reason to believe that someone is cheating, their controller should not be examined.

Q: Will analog ranges (i.e. Firefox angles) be customizable on the tournament patch?
A: No. Even if allowing for customization within the parameters I've set wouldn't raise any concerns, I believe the distributions I've chosen are most efficient from a logistical perspective.

Q: Can I rearrange my button locations?
A: Yes.

Q: Will I be able to play on an un-nerfed version(s) of the B0XX?
A: Yes. We will provide plenty of recreational patches.

Q: Will the B0XX support other games?
A: Yes. More information TBA.

Q: Will a WiiU adapter be needed to use the B0XX on a PC?
A: No. More information TBA.

Q: When will the B0XX be commercially available, and how much will it retail for?
A: July 2018 / $199.99 USD + shipping/tax. Wii Nunchuk sold separately.

Q: Will the initial run of the B0XX be sold through a Kickstarter?
A: No. The initial run will already have been manufactured by the time it goes on sale.

Q: Where can I stay tuned for B0XX-related updates?
A: All major updates will be made through my Twitter account (https://twitter.com/ssbmhax) and our website (http://20XX.gg). You can also subscribe to our newsletter through our website, or join our Discord channel (http://20XX.gg/discord).

****

[15] Patch Notes

****

04/16/2018

Initial release.

04/19/2018

-Minor grammatical revisions.
-Section 5.2.4 (lightshield nerf) added.
-Section 5.3 (nerfs summary) updated.
-Section 7.1 (Modifier 1) updated to explain that Modifier 1 enables analog L 49.
-Section 8.4 (summary of Chapter 8) added.
-Section 10.1.1 (quarter-circle smash DI) updated.
-Section 10.1.4 (Dr. Mario's reverse up-B cancel) added.
-Shield tilt removed from Chapter 11.
-Page numbers removed.

04/20/2018

-UF/DF-smash's coordinates changed to C X +/-.8125 Y +/-.2875 (see Section 9.1)
-Section 10.1.3 renamed to "Samus' Short Hop Fastfall Missile."

04/23/2018

-Section 10.1.2 (wiggle) added.

04/25/2018

-Section 5.2.8 (Firefox nerf) added.
-Section 5.3 (nerfs summary) updated.
-Sections 7.1 (Modifier 1) and 7.2 (Modifier 2) updated to explain that X +/-.7375 Y +/-.2875 and X +/-.2875 Y +/-.7375 are now intended to serve as Firefox angles.

04/27/2018

-Section 5.2.1 (Y-tilt + > 50°) updated to outlaw Y -.6625, -.6750, and -.6875. Even though these Y-values are not in Y-tilt territory, they can be used to perform turnaround D-tilt (Y <= -.7000 causes crouch, which prevents this).
-Sections 5.2.7 (airdodge) and 5.2.8 (Firefox) updated to explain that the B0XX is based on a stock Gamecube controller, not one that contains wavedash/Firefox notches.
-Section 8.1.1 (jump trajectory integrity) added.
-Section 8.2.1 (Firefox) updated.
-Section 10.1.5 updated and renamed to "Crouch -> U/UF-Tilt."
-Section 11.2.3 (automatic smash DI) updated to explain that the B0XX will probably never be given the ability to perform DSDI with steep ASDI coordinates.
-Section 11.2.5 (Gamecube controller's Firefox advantage) updated to account for Firefox nerf.

04/28/2018

-Section 8.2.1 (Firefox) updated.

05/01/2018

-Section 5.2.4 (lightshield nerf) updated to outlaw analog L/R-values in proximity of 140. The B0XX is now restricted to L 49 specifically. Several other sections (5.3, 7.1, 7.2, 8.3.2, 8.3.4, and 11.2.1) were updated to account for this change.
-Section 5.2.9 (other nerfs) updated to outlaw Peach's ledgedash. Several other sections (5.2.7, 5.3, 8.3.3, 8.3.4, and 11.2.6) were updated to account for this change.