Character Statistic Spreadsheet Companion Document

By chumpatrol1

Introduction

This is a document to act as a companion to this spreadsheet, and its goal is to supplement knowledge of the game for both data miners and more competitive players. Each entry in this document corresponds to a column present in the spreadsheet, and the entries are grouped similarly to how the spreadsheet displays the columns. Each entry on the spreadsheet can be looked up by typing it with inside of square brackets. For example, [Weight] will lead to the entry for weight. Additionally, many stats will note what these values look like in the code for people looking to mod the game.

The Character Statistic Spreadsheet has two spreadsheets for each version. The sheets that end with ".HS" represent Hard Statistics, and are acquired directly from the game's source code. Examples of these are weights and run speeds. The sheets that end with ".DS" represent Derivative Statistics, and are created using some Python code I cobbled together in my free time. Examples of these are SHFFL timings and full hop heights.

If there are questions or you would like to make a revision, feel free to contact me on Discord (chumpatrol1#9247) or on Twitter (@chumpatrol1). I'll get back to you as soon as possible and make changes as necessary. I also plan to add new sheets to the CSS as updates get released. I'll also accept reasonable requests for creation of new Derivative Statistics.

Table of Contents

[Grounded Stats]

[Walk Speed]: The walk speed of a character. The units for this stat are in “units per frame”, which is relatively arbitrary without stage information. Final Destination’s total length is 9386 units from ledge to ledge (arbitrary as well). A character with a walk speed of 5 will walk for 94 frames along the stage from one side of FD to the other without falling off the edge. This means that a speed of “1” will cover 20 of these stage units per frame. Coded as “norm_xSpeed”.

[Initial Walk Acceleration]: The acceleration of a character as they begin to walk from a stand still. A higher walk acceleration means that they will get up to speed faster. Seems to accelerate each frame until the character gets up to speed. Coded as “accel_start”.

[Run Speed]: The run speed of a character. The units are the same as walk speed and the rest. Coded as “max_xSpeed”.

[Initial Dash Acceleration]: The acceleration of a character during their initial dash frames. Initial dashes tend to be short, after which the character automatically switches to their default run speed. Does not affect dash turnaround speeds. The extremely high Initial Dash Acceleration on Sandbag is not a typo. Coded as “accel_start_dash”.

[Turn Acceleration]: The acceleration of a character as they turn around from a walk, or a run. The higher the walk turn acceleration, the faster that character will swing their momentum. It’s overall extremely similar to the initial walk acceleration. Coded as “accel_rate”.

[Crawl Speed]: The speed of a character when they are crawling. A little over 10 characters in the game can crawl, and crawling can preserve horizontal momentum. This is especially useful for Luigi, whose long skid animation makes it impractical to simply stop running to do a move. Coded as “crouchWalkSpeed”.

[Traction]: A character’s traction. A number that is closer to 0 suggests that the character will slide more in general, while a number further away from 0 means that the character will stop more quickly. Traction is always a negative number. Traction applies after ending a dash, run or walk. Traction also applies after landing with horizontal momentum (such as after a Luigi Misfire) or after doing a standard ledge climb. Coded as “decel_rate”.

[G2A Multiplier]: A multiplier that is always less than or equal to 1. Multiplies the character's horizontal velocity when jumping off the ground. This can be circumvented using an "RC Dash". Coded as "groundToAirMultiplier".

[Aerial Stats]

[Jumpsquat]: The number of frames after a jump input before a character leaves the ground. Most characters have a 2 frame jumpsquat, though others have a longer jumpsquat. Holding space through a jump squat will result in a Full Hop, while releasing it before then will result in a Short Hop. Inputting an Up Tilt, Up Smash, Up Special, Upwards Item Throw, Grab, or Z-Drop will cancel the jumpsquat into that respective animation. If Jumpsquat happens to be larger than a character's jump.fla file, that character will be unable to jump off the ground. Conversely, if Jumpsquat is set to 0 that character will leave the ground immediately and will be unable to cancel their jump into a move or short hop. Coded as "jumpStartup".

[Gravity]: The downwards force that pulls on airborne characters. This is subtracted from the Y-velocity every frame. Having a larger gravity value will mean that the character will return to the ground faster after a jump, may be combo'd harder, and will have great vertical endurance. Not to be confused with Terminal Velocity/Fall Speed. Coded as "gravity".

[Short Hop Force]: The initial Y-velocity of a Short Hop, with the same unit of measurement as the grounded movement options. A higher Short Hop Force means that the character will have a higher initial Y-Velocity, but due to the force of gravity this may or may not result in a higher jump. As a side note, all jumping actions are slightly broken in this game and actual height gained may vary. Coded as "shortHopSpeed".

[Full Hop Force]: The initial Y-velocity of a Full Hop. It also affects how high a ledge jump is, with all ledge jumps being as high as a full hop. As a side note, all jumping actions are slightly broken in this game and actual height gained may vary. See Short Hop. Coded as "jumpSpeed".

[Midair Jump Force] [Midair Jump Force List]: The initial Y-velocity of a midair/double jump. Overwrites the current Y-velocity. In the case of the Jump Force List, each double jump will read the next unused jump on the list until it reaches the last one. For example, Meta Knight has 5 midair jumps and a Jump Force List with 4 elements in it (13, 12, 11 and 10). Meta Knight's first jump will have a force of 13, his second 12, his third 11, and his last two 10. Having a Jump Force List will cause the Midair Jump Force

to be ignored entirely. As a side note, all jumping actions are slightly broken in this game and actual height gained may vary. Coded as “jumpSpeedList”.

[Air Speed]: This is the maximum horizontal velocity a character can reach normally (through drifting left or right). It solely impacts horizontal velocity and has no effect on vertical velocity. See also “Air Friction”. Coded as “max_jumpSpeed”.

[Air Acceleration]: This is the character’s air acceleration. Having a high air acceleration will make the character change their horizontal velocity more easily in the air. Coded as “accel_rate_air”.

[Air Friction]: This is the character’s air friction. It only applies when no left and right are not held during drift. It also applied when a character’s horizontal velocity is greater than air speed (such as after knockback, or Luigi Misfire). If air friction is close to 0, then that character will drift longer. If air friction is a larger negative number, the character will come to a stop sooner. Coded as “Decel_rate_air”.

[Terminal Velocity] [Fall Speed]: This is the character’s fastest normal fall speed. Characters will normally accelerate downwards until they reach this number, after which their fall speed is set to the Terminal Velocity. Coded as “max_ySpeed”.

[Fast Fall Speed]: This is the character’s fall speed after a fast fall is inputted. It changes a character’s y-velocity to this, ignoring the terminal velocity. If a fast fall speed is slower than the terminal velocity, fall falling will do nothing. In general, fast falls can be inputted as soon as the character has moved downwards a frame and is in most normal states. Coded as “fastFallSpeed”.

[Midair Jumps] [Double Jumps]: The amount of double jumps that a character has. Most characters have 1 double jump, while a character like Jigglypuff has 5. Double jumps are restored upon touching the ground, grabbing ledge, or after certain moves like Falcon Kick. Coded as “max_jump”.

[Midair Turn?]: Whether or not a character turns around after a double jump. Characters with multiple double jumps tend to have this ability. Yoshi technically does not have this flag set to true, but is capable of turning around in midair anyway. Coded as “midAirTurn”.

[Midair Hover]: The amount of time a character can float, in frames. Only Goku (and his Kaioken Variant) and Peach are capable of floating. Programmed as “midAirHover”.

[Midair Jump Time]: The amount of time a character spends in a delayed double jump, in frames. A higher value means that the character will spend more time in their double jump animation. Coded as “midAirJumpConstant”.

[Midair Jump Delay]: The amount of delay, in frames, before a character starts moving upwards. Think of this as a jumpsquat that lets characters continue falling before starting their ascent. Coded as “midAirJumpConstantDelay”.

[Midair Jump Acceleration]: How quickly a character accelerates to their Midair Jump Force during a delayed double jump. Once the Midair Jump Force is reached, the upwards velocity is capped and will remain that way until a Double Jump Cancel or the Midair Jump Time limit is reached. Coded as “midAirJumpConstantAccel”.

[Glide Speed]: An artifact from version 0.9b of SSF2, when Meta Knight’s up special allowed him to glide after completion of the move. Coded as “glideSpeed”.

[Roll Stats]

[Dodge Roll Speed]: How fast a character moves when starting their dodge roll. Dodge rolls are the kind of rolls done when in a shielding state. Coded as “dodgeSpeed”.

[Dodge Deceleration]: How fast a character decelerates during a dodge roll. Having a higher deceleration means that the character will stop moving sooner after initiating their roll. Even if a character has stopped moving, they may still be in the roll animation or even still have invincibility. Programmed as “dodgeDecel”.

[L/GU Roll Speed]: This is similar to Dodge Roll Speed. It affects Ledge Roll, Get Up Roll, and Tech Roll. This refers only to the initial roll speed. Coded as “roll_speed”.

[L/GU Roll Decay]: Affects get up roll, tech roll and ledge roll. A number closer to 0 makes rolls decelerate to 0 very quickly. A number at 1 or above does not cause rolls to decelerate, and have the chance to crash or freeze the game. It probably is multiplicative, with some rounding. Coded as “roll_decay”.

[L/GU Roll Delay]: Probably is unused. Might be applied if Ledge Roll Delay, GU Roll Delay and Tech Roll Delay don’t have any set values. If it were used, it would add a delay to a character’s movement during their respective roll animations. Coded as “roll_delay”.

[Ledge Roll Delay]: Affects how long it takes for a character to start moving horizontally after inputting a roll at ledge. A longer delay means that more time is spent before moving, and in general makes that roll more reactable. Delay has no effect on distance rolled as a rule of thumb (exceptions occur if the delay happens to be longer than a character's tech roll animation). Coded as "climb_roll_delay".

[GU Roll Delay]: Affects how long it takes for a character to start moving horizontally after inputting left or right while lying prone on the ground. A longer delay means that more time is spent before moving, and in general makes that roll more reactable. Coded as "GU Roll Delay".

[Tech Roll Delay]: Affects how long it takes for a character to start moving horizontally after inputting a roll while teching on the ground. A longer delay means that more time is spent before moving, and in general makes that roll more reactable. Delay has no effect on distance rolled as a rule of thumb (exceptions occur if the delay happens to be longer than a character's tech roll animation). Coded as "tech_roll_delay".

[Misc Stats]

[Weight]: How "heavy" a character is. A higher weight means that the character will take less knockback with each attack, allowing them to survive strong blows. Coded as "weight".

[Width] [Height]: Affects stage collisions. An equal width and height forms a rectangle of sorts. A character with large proportions cannot get as close to a stage as a character with smaller proportions. Also affects the distance a character is displaced when they let go or fastfall from ledge. Does not impact movement on platforms. On a stage like Crateria, a small width/height will allow a character like Marth to squeeze through the gap on the right side of the stage, while a larger character like Bowser cannot do the same. Coded as "width" and "height" respectively.

[Shield Scale]: A multiplier that affects how large a character's shield is at max health. Defaults to 1, and most characters have a shield exactly this size. Having a larger shield will make it better at blocking attacks, particularly at low shield health. However, a larger shield can also be hit from further away and lock the defending character in shieldstun, which may be undesirable. Another thing to note is that all shields have the same effective health, and will break at the exact same time. Coded as "shield_scale".

[Shield Offset]: How far a shield is offset from the typical “center” of a character. There is an X offset and a Y offset, both of which default to 0. Changing the offset by 1 unit has little effect, shifting the shield over about a character sprite pixel. Shield offsets are usually used to cover exposed hurtboxes when the shield is at full health. Coded as “shield_x_offset” and “shield_y_offset”.

[Shield Break Power]: The y-velocity that the character flies up to upon having his or her shield broken. Default unknown, but is likely to be 10. When the character has his or her shield broken, gravity will act on them as normal. A ludicrously high Shield Break Power is detrimental since being in the Shield Break Jump animation and passing the top blast zone will result in a KO. Coded as “shieldBreakPower”.

[Hurtframes]: The amount of different hitstun sprites a character has. These hitstun sprites are caused by low knockback moves (such as ZSS jab at 0%) that don’t cause reeling/tumbling. Hurtframes are mostly aesthetic, but some slight hurtbox shifting can occur (best viewed by pummeling Bowser with Link using Monte’s Training Mod). Coded as “hurtframes”.

[Max Projectiles]: The amount of projectiles produced by the character that can exist at a time. Most characters have this set to 10 (this may be due to some effect trails being considered “projectiles”). If the maximum number of projectiles from that character is on screen, no new projectile will be created. Coded as “max_projectiles”.

[Tether Grab]: Internal flag that shows if a grab is considered a tether grab or not. Some grabs which seem like Tether Grabs are not treated as such for whatever reason. Coded as “tetherGrab”.

[Pummel %]: The amount of damage a single pummel does. Coded as “grabDamage”.

[Hold Jump]: Purpose unknown, and is set to false on every character. Setting it to true seems to do nothing. Coded as “holdJump”.

[Tilt Toss Multiplier] [Smash Toss Multiplier]: Impacts the force of a thrown projectile multiplicatively. Few characters have a non-1 smash throw multiplier, and all characters have a tilt throw multiplier of 1.

[Wall Jump]: A flag that determines if a character can wall jump or not. Since wall jumps are actually not in the game (yet), we can only guess that an implementation was intended before the release of Beta 1.1 since Pichu has this flag set to true. Coded as “wallJump ”.

[Derivative Calculations]

[Short Hop Height]: The calculated short hop height of a character. A larger number means that the character will jump higher on a short hop. As a side note, all jumping actions are slightly broken in this game and actual height gained may vary (best seen with Ganondorf).

[SH Airtime]: The amount of frames that a character is airborne during a short hop. Adding 1 to this number will result in the expected frame that the character will land. As a side note, all jumping actions are slightly broken in this game and actual airtime may vary (for example, Fox is expected to be in the air for 12 frames, but sometimes he is in the air for 13 frames). **TODO: Add a link demonstrating this**

[SH Apex Frame]: This is the frame that the character is at the apex of their jump. At this point, the character is expected to be at their Short Hop Height. Jumping on this frame will allow for a character to gain more height than jumping before or afterwards. As a side note, all jumping actions are slightly broken in this game and the actual apex frame may vary. **TODO: Add a link demonstrating this**

[SHFFL Input Frame]: This is the first frame that the character can input a fast fall. If a fast fall is registered on this frame, then the character will SHFFL optimally. As a side note, all jumping actions are slightly broken in this game and the actual SHFFL input frame may vary.

[SHFFL Airtime]: The amount of frames that a character is airborne during a SHFFL. It seems to be inaccurate at the moment, and the program which was used to do the calculations will be updated at some point in order to remedy this. As a side note, all jumping actions are slightly broken in this game and actual airtime may vary.

[SHFFL Land Frame]: The frame that a character will land on after successfully performing a frame-perfect SHFFL. It seems to be inaccurate at the moment, and the program which was used to do the calculations will be updated at some point in order to remedy this. As a side note, all jumping actions are slightly broken in this game and the actual landing frame may vary.

[Full Hop Height]: The calculated full hop height of a character. A larger number means that the character will jump higher on a full hop. As a side note, all jumping actions are slightly broken in this game and actual height gained may vary.

[FH Airtime]: The amount of frames that a character is airborne during a full hop. Adding 1 to this number will result in the expected frame that the character will land. As a side note, all jumping actions are slightly broken in this game and actual airtime may vary.

[FH Apex Frame]: This is the frame that the character is at the apex of their jump. At this point, the character is expected to be at their Full Hop Height. Jumping on this frame will allow for a character to gain more height than jumping before or afterwards. As a side note, all jumping actions are slightly broken in this game and the actual apex frame may vary.

[FHFFL Input Frame]: FHFFL is short for Full Hop Fast Fall Land. This is the first frame that the character can input a fast fall. If a fast fall is registered on this frame, then the character will FHFFL optimally. As a side note, all jumping actions are slightly broken in this game and the actual FHFFL input frame may vary.

[FHFFL Airtime]: The amount of frames that a character is airborne during a FHFFL. It seems to be inaccurate at the moment, and the program which was used to do the calculations will be updated at some point in order to remedy this. As a side note, all jumping actions are slightly broken in this game and actual airtime may vary.

[FHFFL Land Frame]: The frame that a character will land on after successfully performing a frame-perfect FHFFL. It seems to be inaccurate at the moment, and the program which was used to do the calculations will be updated at some point in order to remedy this. As a side note, all jumping actions are slightly broken in this game and the actual landing frame may vary.

[0 to TV in Frames]: The amount of frames that it takes for a character to reach their terminal velocity in frames. In game, this is usually applicable when letting go of the ledge.

[0 to TV in Fall Distance]: The amount of distance a character will fall before hitting their terminal velocity. The units aren’t particularly helpful, and stage labbing would be helpful.