Super Mario 64 - Mario's Jump Height

Chase Bradley

December 13th, 2025

Contents

1 Calculating Jump Height

To find how big any given jump goes, we could spend the next week trying various jumps in-game and seeing how high they go, but that sucks and ain't nobody got time for that. Instead, we're going to derive a function which, given an initial vertical velocity and a gravity constant, can find the peak heigh of Mario's jump.

1.1 Velocity Function

Let $v_0 \in \mathbb{R}^+$ be the velocity of Mario right after entering an airborne state. We assume a positive or zero velocity for now to simplify our height function.

Let $g \in (\mathbb{R}^+ - \{0\})$ be a gravity constant. This should remain constant throughout the jump, but it can change based on action, such as long jumps.

We'll define a function $v_g(t) : \mathbb{R}^+ \rightarrow \mathbb{R}$ which represents Mario's velocity after t in-game ticks since entering an airborne state as the following:

$$v_g(t) \quad = \quad max(v_0 - \lfloor t \rfloor, -75)$$

We take the max because Mario's terminal velocity is 75 units downward per frame. We also take the floor of t so we have a domain over $\mathbb{R}^+$ instead of over $\mathbb{Z}^+$. This will be useful for deriving the position function.

1.2 Position Function

Next, we're going to derive the position function, $r_g(t) : \mathbb{R}^+ \rightarrow \mathbb{R}$ which calculates Mario's vertical displacement from his original jump location after t in-game ticks with a gravity of g.

To solve this, we will use calculus, but with some minor liberties. Since we're using the floor function and the max function, our function is not easy to integrate. In particular, the floor function means we can't use regular integration to solve this because the function isn't continuous. In addition, the max function secretly turns our function into a piecewise function. To solve both these problems, we're going to do the following:

- Define a continuous function which represents the antiderivative of the floor function
- Make $v_g(t)$ piecewise so we can integrate each component seperately

For the first point, we will define the following antiderivative of the floor function:

$$\int_0^x \lfloor t \rfloor dt \quad = \quad \frac{1}{2}\lfloor x \rfloor(2x - 1 - \lfloor x \rfloor)$$

This function has the nice property that it's continuous, as well as having a linear slope which more accurately interpolates the non-integer values. This also ensures the function is monotonic for $x \geq 0$. To explain why both these points are important, consider the following alternative antiderivative:

$$\int_0^x \lfloor t \rfloor dt = \frac{x(x-1)}{2}$$

This function works fine for positive integer values, but it begins to show issues with non-integer values. On the interval $[0, 1]$, our function actually goes negative. This doesn't make any sense, because we're accumulating strictly positive quantities. In addition, quadratic interpolation doesn't make sense because each interval of the floor function is analygous to integrating a constant, which should be linear. Thus, we'd expect linear interpolation between each integer point, not quadratic interpolation. Thus, this function isn't suitable.

Next, we'll redefine $v_g(t)$ to be piecewise, free of the max function. This will allow us to integrate each component of the function independently, then combine together as a new piecewise function after integration.

$$v_g(t) = \begin{cases} v_0 - g\lfloor t \rfloor & v_0 - g\lfloor t \rfloor > -75 \\ -75 & v_0 - g\lfloor t \rfloor \leq -75 \end{cases}$$

With these restrictions lifted, we can now compute $r_g(t)$.

$$\begin{aligned} r_g(t) &= \int_0^t v(w)dw \\ &= \begin{cases} \int_0^t (v_0 - g\lfloor w \rfloor)dw & v_0 - g\lfloor t \rfloor > -75 \\ \int_0^t (-75)dw & v_0 - g\lfloor t \rfloor \leq -75 \end{cases} \\ &= \begin{cases} v_0 t - g\int_0^t \lfloor w \rfloor dw & v_0 - g\lfloor t \rfloor > -75 \\ -75t & v_0 - g\lfloor t \rfloor \leq -75 \end{cases} \\ &= \begin{cases} v_0 t - \frac{g}{2}\lfloor t \rfloor(2t - 1 - \lfloor t \rfloor) & v_0 - g\lfloor t \rfloor > -75 \\ -75t & v_0 - g\lfloor t \rfloor \leq -75 \end{cases} \end{aligned}$$

With this, we now have derived the following position function:

$$r_g(t) = \begin{cases} v_0 t - \frac{g}{2}\lfloor t \rfloor(2t - 1 - \lfloor t \rfloor) & v_0 - g\lfloor t \rfloor > -75 \\ -75t & v_0 - g\lfloor t \rfloor \leq -75 \end{cases}$$

However, this is not entirely accurate. Since the game runs in discrete ticks, we should also have the position function be discrete. This allows us to make some simplifications.

$$\begin{aligned} r_g(t) &= \begin{cases} v_0 t - \frac{g}{2}\lfloor t \rfloor(2t - 1 - \lfloor t \rfloor) & v_0 - g\lfloor t \rfloor > -75 \\ -75t & v_0 - g\lfloor t \rfloor \leq -75 \end{cases} \\ &= \begin{cases} v_0 \lfloor t \rfloor - \frac{g}{2}\lfloor t \rfloor(2\lfloor t \rfloor - 1 - \lfloor t \rfloor) & v_0 - g\lfloor t \rfloor > -75 \\ -75\lfloor t \rfloor & v_0 - g\lfloor t \rfloor \leq -75 \end{cases} \\ &= \begin{cases} \lfloor t \rfloor(v_0 - \frac{g}{2}(\lfloor t \rfloor - 1)) & v_0 - g\lfloor t \rfloor > -75 \\ -75\lfloor t \rfloor & v_0 - g\lfloor t \rfloor \leq -75 \end{cases} \end{aligned}$$

This gives us our final position function:

$$r_g(t) \quad = \quad \begin{cases} \lfloor t \rfloor (v_0 - \frac{g}{2}(\lfloor t \rfloor - 1)) & v_0 - g\lfloor t \rfloor > -75 \\ -75\lfloor t \rfloor & v_0 - g\lfloor t \rfloor \leq -75 \end{cases}$$

1.3 Height Function

We will now find a function $h_g(v) : \mathbb{R}^+ \to \mathbb{R}^+$ which computes the maximum value of the position function, given the starting velocity v and constant gravity g.

We will first makes some simplifications to $v_g(t)$ and $r_g(t)$. Since we are concerned with the maximum values, we aren't interested in negative velocities. Thus, we will ignore terminal velocity in our modified velocity and position functions.

$$\begin{aligned} v_g(t) &= v_0 - g\lfloor t \rfloor \\ r_g(t) &= \lfloor t \rfloor (v_0 - \frac{g}{2}(\lfloor t \rfloor - 1)) \end{aligned}$$

Next, we want to find the tick which has the maximum height. We can accomplish this by finding when the velocity is zero.

$$\begin{aligned} v_g(t) &= 0 \\ v_0 - g\lfloor t \rfloor &= 0 \\ -g\lfloor t \rfloor &= -v_0 \\ g\lfloor t \rfloor &= v_0 \\ \lfloor t \rfloor &= \frac{v_0}{g} \end{aligned}$$

Here, we will make another technical decision. Since the floor function is not surjective, it may not be possible to find a value for t which satisfies this equation without the floor function.

To solve this, we're going to choose to use the ceil function. This must be used since we may have no individual frame where $v_g(t) = 0$. However, the first tick where $v_g(t) < 0$ will always maximize $r_g(t)$. Thus, we'll get the following value for t:

$$t \quad = \quad \lceil \frac{v_0}{g} \rceil$$

We can now plug this in to $r_g(t)$ to get our height function $h_g(v)$.

$$\begin{aligned} h_g(v) &= r(\lceil \frac{v}{g} \rceil) \\ &= \lfloor \lceil \frac{v}{g} \rceil \rfloor (v - \frac{g}{2}(\lfloor \lceil \frac{v}{g} \rceil \rfloor - 1)) \\ &= \lceil \frac{v}{g} \rceil (v - \frac{g}{2}(\lceil \frac{v}{g} \rceil - 1)) \end{aligned}$$

This cannot be simplified further. Given $x, y \in \mathbb{R}$, $y\lceil\frac{x}{y}\rceil = \lceil x \rceil$ is not generally true. Thus, our final height function is the following:

$$h_g(v) \quad = \quad \lceil\frac{v}{g}\rceil(v - \frac{g}{2}(\lceil\frac{v}{g}\rceil - 1))$$

1.4 Inverse Height Function

Lastly, we will find an inverse for the height function $h_g^{-1}(r) : \mathbb{R}^+ \to \mathbb{R}^+$. This will be useful when we want to find the required velocity to travel a given height with a given gravity constant.

We're also going to make the following modification and define v in terms of an arbitrary function. This will be invaluable later when we want to find h_g^{-1} with initial velocity as a function. For example, initial vertical velocity can be a function of forward running speed. Thus, we'll define the following functions:

$$\begin{aligned} v \quad &: \quad A \to \mathbb{R}^+ \text{ such that } A \subseteq \mathbb{R}, 0 \in \mathsf{Im}(v), v \text{ is injective} \\ v^{-1} \quad &: \quad \mathbb{R}^+ \to A \text{ such that for } \forall a \in A, v^{-1}(v(a)) = a \end{aligned}$$

We will now consider given $r \in \mathbb{R}^+$, $g \in (\mathbb{R}^+ - \{0\})$, the function $h_g(v(a))$ attempt to find $a \in A$ such that $h_g(v(a)) = r$. Rearranged, we are trying to find $a \in A$ such that $a = v^{-1}(h_g^{-1}(r))$.

To solve this, we're going to show that $h_g(v(a))$ can be written as a piecewise function such that for certain intervals of h_g, it can be written in the form of $mv(a) + b$ such that $m, b \in \mathbb{R}$. We will then use this fact to find m and b when given r, then finally solve for a.

First, we're going to find all the intervals where $h_g(v(a))$ can be written as $mv(a) + b$.

Proposition. Given $g \in (\mathbb{R}^+ - \{0\})$ and $n \in \mathbb{Z}^+$, then $\exists m_n, b_n \in \mathbb{R} : h_g(v(a)) = m_n v(a) + b_n$ on the interval $(v^{-1}(gn), v^{-1}(g(n+1))]$.

Proof. Recall that for $\forall n \in \mathbb{Z}^+$, $\lim_{\alpha \to 0^+} \lceil n + \alpha \rceil = n + 1$.

Now look at our function $h_g(v(a)) = \lceil\frac{v(a)}{g}\rceil(v(a) - \frac{g}{2}(\lceil\frac{v(a)}{g}\rceil - 1))$. In particular, focus on the $\lceil\frac{v(a)}{g}\rceil$ term. If we assume $\lceil\frac{v(a)}{g}\rceil$ to be constant C, then we get the following:

$$\begin{aligned} h_g(v(a)) \quad &= \quad \lceil\frac{v(a)}{g}\rceil(v(a) - \frac{g}{2}(\lceil\frac{v(a)}{g}\rceil - 1)) \\ &= \quad C(v(a) - \frac{g}{2}(C - 1)) \\ &= \quad Cv(a) - C\frac{g}{2}(C - 1) \\ &= \quad Cv(a) - \frac{gC(C-1)}{2} \\ &= \quad Cv(a) + (\frac{-gC(C-1)}{2}) \end{aligned}$$

This is in the form of $h_g(v(a)) = m_n v(a) + b_n$. Therefore, if $\lceil \frac{v(a)}{g} \rceil$ is constant over the interval I, then $\exists m, b \in \mathbb{R} : h_g(v(a)) = mv(a) + b$ over the interval I.

Next, we're going to show that $\lceil \frac{v(a)}{g} \rceil$ remains constant over the interval $(v^{-1}(gn), v^{-1}(g(n+1))]$.

To do this, we're going to show that $\lim_{\alpha \to 0^+} \lceil \frac{v(v^{-1}(gn))}{g} + \alpha \rceil = \lceil \frac{v(v^{-1}(g(n+1)))}{g} \rceil$.

$$\begin{aligned}
\lim_{\alpha \to 0^+} \lceil \frac{v(v^{-1}(gn))}{g} + \alpha \rceil &= \lim_{\alpha \to 0^+} \lceil \frac{gn}{g} + \alpha \rceil \\
&= \lim_{\alpha \to 0^+} \lceil n + \alpha \rceil \\
&= n + 1
\end{aligned}$$

$$\begin{aligned}
\lceil \frac{v(v^{-1}(g(n+1)))}{g} \rceil &= \lceil \frac{g(n+1))}{g} \rceil \\
&= \lceil n + 1 \rceil \\
&= n + 1
\end{aligned}$$

Since $\lim_{\alpha \to 0^+} \lceil \frac{v(v^{-1}(gn))}{g} + \alpha \rceil = \lceil \frac{v(v^{-1}(g(n+1)))}{g} \rceil$, we know that $\lceil \frac{v(a)}{g} \rceil$ is constant over the interval $(v^{-1}(gn), v^{-1}(g(n+1))]$.

As a result, this means that $\exists m_n, b_n \in \mathbb{R} : h_g(v(a)) = m_n v(a) + b_n$ over the interval
$(v^{-1}(gn), v^{-1}(g(n+1))]$.

$\square$

With this knowledge, we now know if we can find n in terms of r, we can then rewrite $h_g(v(a)) = m_n v(a) + b_n$ over the interval $(v^{-1}(gn), v^{-1}(g(n+1))]$. Assuming we already have m_n and b_n, this is trivial to solve.

$$\begin{aligned}
h_g(v(a)) &= r \\
m_n v(a) + b_n &= r \\
m_n v(a) &= r - b_n \\
v(a) &= \frac{r - b_n}{m_n} \\
h_g^{-1}(r) &= \frac{r - b_n}{m_n} \\
v^{-1}(h_g^{-1}(r)) &= v^{-1}(\frac{r - b_n}{m_n})
\end{aligned}$$

Next, we're going to solve for n in terms of r such that the interval for n contains our solution. To do this, we're going to plug in the start point of each interval and then try to solve for n in terms of r.

$$\begin{aligned}
h_g(v(v^{-1}(gn))) &= r \\
h_g(gn) &= r \\
\lceil\frac{gn}{g}\rceil(gn - \frac{g}{2}(\lceil\frac{gn}{g}\rceil - 1)) &= r \\
\lceil n\rceil(gn - \frac{g}{2}(\lceil n\rceil - 1)) &= r \\
n(gn - \frac{g}{2}(n-1)) &= r \\
gn(n - \frac{1}{2}(n-1)) &= r \\
\frac{gn}{2}(2n - (n-1)) &= r \\
\frac{gn}{2}(2n - n + 1) &= r \\
\frac{gn}{2}(n+1) &= r \\
n(n+1) &= \frac{2r}{g} \\
n^2 + n &= \frac{2r}{g} \\
n^2 + n - \frac{2r}{g} &= 0 \\
(1)n^2 + (1)n + (\frac{-2r}{g}) &= 0 \\
n &= \frac{-(1) + \sqrt{(1)^2 - 4(1)(\frac{-2r}{g})}}{2(1)} \\
n &= \frac{-1 + \sqrt{1 + \frac{8r}{g}}}{2} \\
n &= \frac{-1}{2} + \frac{1}{2}\sqrt{1 + \frac{8r}{g}} \\
n &= \frac{1}{2}\sqrt{1 + \frac{8r}{g}} - \frac{1}{2} \\
n &= \sqrt{\frac{1}{4}(1 + \frac{8r}{g})} - \frac{1}{2} \\
n &= \sqrt{\frac{1}{4} + \frac{8r}{4g}} - \frac{1}{2} \\
n &= \sqrt{\frac{1}{4} + \frac{2r}{g}} - \frac{1}{2} \\
n &= \sqrt{\frac{2r}{g} + \frac{1}{4}} - \frac{1}{2}
\end{aligned}$$

This will give us a real number which is on the interval $[n, n+1)$. We get the exact value for n by taking the floor of the above. This results in our final solution for n:

$$n = \lfloor\sqrt{\frac{2r}{g}+\frac{1}{4}}-\frac{1}{2}\rfloor$$

We will now solve for m_n and b_n in terms of n. This is actually pretty simple. We just plug in for the endpoint of the n'th interval for all the ceil terms and write it in the correct form.

To help clean things up, we're going to first simplify $\lceil\frac{v(v^{-1}(g(n+1)))}{g}\rceil$.

$$\begin{aligned}\lceil\frac{v(v^{-1}(g(n+1)))}{g}\rceil &= \lceil\frac{g(n+1)}{g}\rceil \\ &= \lceil n+1\rceil \\ &= n+1\end{aligned}$$

Let's now plug in to $h_g(v(a))$ on the n'th interval.

$$\begin{aligned}h_g(v(a)) &= \lceil\frac{v(a)}{g}\rceil(v(a)-\frac{g}{2}(\lceil\frac{v(a)}{g}\rceil-1)) \\ &= \lceil\frac{v(v^{-1}(g(n+1)))}{g}\rceil(v(a)-\frac{g}{2}(\lceil\frac{v(v^{-1}(g(n+1)))}{g}\rceil-1)) \\ &= (n+1)(v(a)-\frac{g}{2}((n+1)-1)) \\ &= (n+1)(v(a)-\frac{gn}{2}) \\ &= (n+1)v(a)-\frac{gn(n+1)}{2} \\ &= (n+1)v(a)+\frac{-gn(n+1)}{2} \\ h_g(v(a)) &= (n+1)v(a)+\frac{-gn(n+1)}{2}\end{aligned}$$

With this, we now have m_n and b_n in terms of n.

$$\begin{aligned}m_n &= n+1 \\ b_n &= \frac{-gn(n+1)}{2}\end{aligned}$$

Now, let's revisit our previous solution for $v^{-1}(h_g^{-1}(r))$ and plug in for m_n and b_n.

$$\begin{aligned}
v^{-1}(h_g^{-1}(r)) &= v^{-1}(\frac{r-b_n}{m_n}) \\
&= v^{-1}(\frac{r-\frac{-gn(n+1)}{2}}{n+1}) \\
&= v^{-1}(\frac{r+\frac{gn(n+1)}{2}}{n+1}) \\
&= v^{-1}(\frac{r}{n+1}+\frac{\frac{gn(n+1)}{2}}{n+1}) \\
&= v^{-1}(\frac{r}{n+1}+\frac{gn}{2}) \\
v^{-1}(h_g^{-1}(r)) &= v^{-1}(\frac{r}{n+1}+\frac{gn}{2})
\end{aligned}$$

Finally, we write n in terms of r to get our final solution to $v^{-1}(h_g^{-1}(r))$.

We could just plug in for n, but that would result in a butt ugly equation with no way to simplify. Instead, we're going to define $n_g(r) : \mathbb{R}^+ \to \mathbb{Z}^+$ to be our solution for n given r, then use this function inside our definition for $h_g^{-1}(r)$. This gives us our final answer for $h_g^{-1}(r)$.

$$\begin{aligned}
n_g(r) &= \lfloor\sqrt{\frac{2r}{g}+\frac{1}{4}}-\frac{1}{2}\rfloor \\
h_g^{-1}(r) &= \frac{r}{1+n_g(r)}+\frac{gn_g(r)}{2}
\end{aligned}$$

Notice that there is no dependence on v^{-1} to compute h_g^{-1}. Therefore, it's redundant to specify $v^{-1} \circ h_g^{-1}$, and we can treat v as if it's a regular variable in terms of our solution.

1.5 Function Index

For reference, here is a table of all relevant functions for this section:

Name	Identifier	Domain and Codomain	Definition
Velocity Function	$v_g(t)$	$\mathbb{R}^+ \to \mathbb{R}$	$v_g(t) = max(v_0 - g\lfloor t \rfloor, -75)$
Position Function	$r_g(t)$	$\mathbb{R}^+ \to \mathbb{R}$	$r_g(t) = \begin{cases} \lfloor t \rfloor (v_0 - \frac{g}{2}(\lfloor t \rfloor - 1)) & v_0 - g\lfloor t \rfloor > -75 \\ -75\lfloor t \rfloor & v_0 - g\lfloor t \rfloor \leq -75 \end{cases}$
Height Function	$h_g(v)$	$\mathbb{R}^+ \to \mathbb{R}^+$	$h_g(v) = \lceil \frac{v}{g} \rceil (v - \frac{g}{2}(\lceil \frac{v}{g} \rceil - 1))$
Inverse Height Interval	$n_g(r)$	$\mathbb{R}^+ \to \mathbb{Z}^+$	$\lfloor \sqrt{\frac{2r}{g} + \frac{1}{4}} - \frac{1}{2} \rfloor$
Inverse Height Function	$h_g^{-1}(r)$	$\mathbb{R}^+ \to \mathbb{R}^+$	$h_g^{-1}(r) = \frac{r}{1+n_g(r)} + \frac{g n_g(r)}{2}$

2 Jumping Actions

We will now analyze how different "jumping" actions set Mario's starting vertical velocity and affect gravity, which can be used with the previously derived height function to compute the peak height for each jumping action.

2.1 Jumping Velocities

We will classify a "jumping" action as a Mario action with the following criteria:

- Either the ACT_GROUP_AIRBORNE or ACT_GROUP_SUBMERGED flag is set
- Vertical velocity is set to a positive value just before entering the action
- Vertical velocity is only affected by gravity while in the action
- Mario can be purposefully controlled with the analog stick throughout the jump

Some actions set initial vertical velocity based on Mario's forward speed (m->forwardVel). Mario's forward speed will be denoted with 'f'. In addition, some use Mario's vertical speed. This will be denoted with 'v'.

In addition, some actions use set_mario_y_vel_based_on_fspeed() instead of directly setting m->vel[1]. These actions check if Mario is either in quicksand (m->quicksandDepth ≥ 1) or squished (m->squishTimer $\neq 0$). If he is, the intended initial vertical velocity is halved. These actions have their vertical velocity colored blue. The given initial vertical velocity assumes Mario is neither in quicksand or squished.

This leaves us with the following table of actions:

Name	Identifier	Initial Vertical Velocity (units/tick)	Gravity (units/tick^2)
Single Jump	ACT_SINGLE_JUMP	$42 + \frac{f}{4}$	4
Double Jump	ACT_DOUBLE_JUMP	$52 + \frac{f}{4}$	4
Triple Jump	ACT_TRIPLE_JUMP	69	4
Flying Triple Jump	ACT_FLYING_TRIPLE_JUMP	82	4
Special Triple Jump (initial jump)	ACT_SPECIAL_TRIPLE_JUMP	69	4
Special Triple Jump (landing bounce)	ACT_SPECIAL_TRIPLE_JUMP	42	4
Jump With Object	ACT_HOLD_JUMP	$42 + \frac{f}{4}$	4
Jump From Poll (middle)	ACT_WALL_KICK_AIR	62	4
Jump From Poll (top)	ACT_TOP_OF_POLL_JUMP	62	4
Jump While Riding Shell	ACT_RIDING_SHELL_JUMP	$42 + \frac{f}{4}$	4
Side Flip	ACT_SIDE_FLIP	62	4
Back Flip	ACT_BACKFLIP	62	4
Dive (from ground)	ACT_DIVE	20	4
Forward Rollout	ACT_FORWARD_ROLLOUT	30	4
Backward Rollout	ACT_BACKWARD_ROLLOUT	30	4
Sliding Kick (initial kick)	ACT_SLIDE_KICK	12	2
Sliding Kick (landing bounce)	ACT_SLIDE_KICK	$-\frac{v}{2}$	2
Jumping Kick	ACT_JUMP_KICK	20	4
Wall Kick	ACT_WALL_KICK_AIR	62	4
Long Jump	ACT_LONG_JUMP	30	2
Lava Boost	ACT_LAVA_BOOST	84	3.2
Crazy Box Bounce (1st bounce)	ACT_CRAZY_BOX_BOUNCE	45	4
Crazy Box Bounce (2nd bounce)	ACT_CRAZY_BOX_BOUNCE	60	4
Crazy Box Bounce (3rd bounce)	ACT_CRAZY_BOX_BOUNCE	100	4
Underwater Metal Cap Jump	ACT_METAL_WATER_JUMP	32	1.6
Underwater Metal Cap Jump With Object	ACT_HOLD_METAL_WATER_JUMP	32	1.6

We will also include the following table of actions, which consist of exceptions to the 4th criteria for a jumping action:

Name	Identifier	Initial Vertical Velocity (units/tick)	Gravity (units/tick^2)
Steep Jump	ACT_STEEP_JUMP	$42 + \frac{f}{4}$	4
Jump From Water	ACT_WATER_JUMP	42	4
Jump From Water With Object	ACT_HOLD_WATER_JUMP	42	4
Jump While Burning	ACT_BURNING_JUMP	31.5	4

2.2 Jumping Heights

We will use the height function $h_g(v)$ from section 1.3 to calculate the peak jump height for each jumping actions.

We will split jumping actions into two tables, based on whether their peak jump height is constant or a function of either Mario's forward or vertical speed.

Actions which have their initial vertical velocity affected by either being in quicksand or squished will have the peak jump height while affected also provided.

Name	Identifier	Peak Jump Height (units)	Squished Peak Jump Height (units)
Triple Jump	ACT_TRIPLE_JUMP	630	166.5
Flying Triple Jump	ACT_FLYING_TRIPLE_JUMP	882	231
Special Triple Jump (initial jump)	ACT_SPECIAL_TRIPLE_JUMP	630	166.5
Special Triple Jump (landing bounce)	ACT_SPECIAL_TRIPLE_JUMP	242	
Jump From Poll (middle)	ACT_WALL_KICK_AIR	512	136
Jump From Poll (top)	ACT_TOP_OF_POLL_JUMP	512	136
Side Flip	ACT_SIDE_FLIP	512	136
Back Flip	ACT_BACKFLIP	512	136
Dive (from ground)	ACT_DIVE	60	
Forward Rollout	ACT_FORWARD_ROLLOUT	128	
Backward Rollout	ACT_BACKWARD_ROLLOUT	128	
Sliding Kick (initial kick)	ACT_SLIDE_KICK	42	
Jumping Kick	ACT_JUMP_KICK	60	
Wall Kick	ACT_WALL_KICK_AIR	512	136
Long Jump	ACT_LONG_JUMP	240	64
Lava Boost	ACT_LAVA_BOOST	1144.8	
Crazy Box Bounce (1st bounce)	ACT_CRAZY_BOX_BOUNCE	276	
Crazy Box Bounce (2nd bounce)	ACT_CRAZY_BOX_BOUNCE	480	
Crazy Box Bounce (3rd bounce)	ACT_CRAZY_BOX_BOUNCE	1300	
Underwater Metal Cap Jump	ACT_METAL_WATER_JUMP	336	
Underwater Metal Cap Jump With Object	ACT_HOLD_METAL_WATER_JUMP	336	
Jump From Water	ACT_WATER_JUMP	242	66
Jump From Water With Object	ACT_HOLD_WATER_JUMP	242	66
Jump While Burning	ACT_BURNING_JUMP	140	

Name	Identifier	Peak Jump Height (units)	Squished Peak Jump Height (units)	Domain (units)
Single Jump	ACT_SINGLE_JUMP	$h_4(42 + \frac{f}{4})$	$h_4(21 + \frac{f}{8})$	$f \geq -184$
Double Jump	ACT_DOUBLE_JUMP	$h_4(52 + \frac{f}{4})$	$h_4(26 + \frac{f}{8})$	$f \geq -224$
Jump With Object	ACT_HOLD_JUMP	$h_4(42 + \frac{f}{4})$	$h_4(21 + \frac{f}{8})$	$f \geq -184$
Jump While Riding Shell	ACT_RIDING_SHELL_JUMP	$h_4(42 + \frac{f}{4})$	$h_4(21 + \frac{f}{8})$	$f \geq -184$
Sliding Kick (landing bounce)	ACT_SLIDE_KICK	$h_2(-\frac{v}{2})$		$-75 \leq v \leq 0$
Steep Jump	ACT_STEEP_JUMP	$h_4(42 + \frac{f}{4})$	$h_4(21 + \frac{f}{8})$	$f \geq -184$

Notice that we add a domain restriction for the non-constant jump heights. This is because the height function requires the initial vertical velocity to be either positive or zero. In addition, it's impossible to have a sliding kick landing bounce with a vertical velocity smaller than terminal velocity, which is -75 for this action.

Also notice that we give variable jump height in terms of $h_g(v)$ without plugging in and simplifying. This is because there are not many simplifications to be made, thus it's redundant to plug in for each function seperately. Plugging in the velocity function and simplifying the height function is left as an exercise for the reader.

We're now going to show the required speed to reach a jump height of $r \in \mathbb{R}^+$ for all the variable jumping actions. Like above, we will give equations in terms of $h_g^{-1}(r)$ from section 1.4 to avoid a nightmare of horrendous equations which can't be simplified. Also note that we don't provide a function-specific domain here. This is because it's redundant, as the domain of h_g^{-1} is always $\mathbb{R}^+$ by definition.

Name	Identifier	Required Speed (units/tick)	Squished Required Speed (units/tick)
Single Jump	ACT_SINGLE_JUMP	$4h_4^{-1}(r) - 168$	$8h_4^{-1}(r) - 168$
Double Jump	ACT_DOUBLE_JUMP	$4h_4^{-1}(r) - 208$	$8h_4^{-1}(r) - 208$
Jump With Object	ACT_HOLD_JUMP	$4h_4^{-1}(r) - 168$	$8h_4^{-1}(r) - 168$
Jump While Riding Shell	ACT_RIDING_SHELL_JUMP	$4h_4^{-1}(r) - 168$	$8h_4^{-1}(r) - 168$
Sliding Kick (landing bounce)	ACT_SLIDE_KICK	$-2h_2^{-1}(r)$	
Steep Jump	ACT_STEEP_JUMP	$4h_4^{-1}(r) - 168$	$8h_4^{-1}(r) - 168$

2.3 Jumping Height Scales

A minor point, but something interesting to note. There are two complexity classes jumping action heights can be classified into. Given height function $h^j \in \{h_g : g \in \mathbb{R}^+ - \{0\}\}$ for jumping action j and initial velocity $v \in \mathbb{R}^+$, h^j can be classified into one of the following complexity classes:

$$\begin{aligned} h^j(v) &\in \Theta(1) \\ h^j(v) &\in \Theta(v^2) \end{aligned}$$

This means the jump height is either always constant, or increases proportionally with the square of the input. This is especially interesting for the second case, as this means as we increase the input, the jump height begins to rapidly grow.

For example, if we perform a double jump with f running speed, then a jump with $2f$ running speed is proportional to 4 times the height as the previous jump. With $3f$ speed, the jump height is proportional to 9 times the height as the previous jump.

All of the jumping actions with a constant peak height belong to $\Theta(1)$, and all jumping actions which are a function of either Mario's forward speed or vertical speed belong to $\Theta(v^2)$.

3 Applications

This is all interesting, but how is this useful (other than being cool, of course!)?

3.1 Running Speed Required To Grab A Hangable Ceiling

Mario's vertical collision hitbox is 160 units tall. That is, if a ceiling's height is 160 units above Mario or more, he will be allowed to fit underneath it. In the case of grabbable ceilings, if Mario's next position is less than 160 units below the ceiling's hitbox, he will grab the ceiling. However, ceilings can only be grabbed from either a single jump or a double jump state.

What if there is a ceiling we want to grab, where we know how high off the ground is and we can build enough speed to grab it? How fast do we need to be moving for a double jump to reach high enough to grab the ceiling?

Most people would probably try some random movement, fail the jump, and give up. More big brain people would hack the game to give mario some amount of speed before a double jump, then try various speed values until one works. However, we can do better! We can calculate the *exact* speed required for such a jump!

Let's take for example the hangable ceiling just above the main entrance to the castle in castle grounds. Let's try doing a double jump from the very edge of either of the bridge's banisters. If we look at the model data, the height of the banister closest to the castle is 957, and the height of the hangable ceiling is 1692. Since Mario's vertical hitbox height is 160 units, we need to jump over $(1692 - 160) - 957 = 575$ units high from a double jump.

We solve this by plugging in $r = 575$ into the formula for double jumps from the inverse height function table and only consider running speeds greater than that as solutions.

$$\begin{aligned} v^{-1}(h_g^{-1}(r)) &= 4h_4^{-1}(575) - 208 \\ &= 4(\frac{575}{1 + n_4(575)} + \frac{4n_4(575)}{2}) - 208 \\ &= 4(\frac{575}{1 + n_4(575)} + 2n_4(575)) - 208 \end{aligned}$$

To be cleaner, we're going to simplify $n_4(575)$ independently then plug back in once simplified.

$$\begin{aligned} n_g(r) &= \lfloor\sqrt{\frac{2r}{g} + \frac{1}{4}} - \frac{1}{2}\rfloor \\ n_4(575) &= \lfloor\sqrt{\frac{2(575)}{4} + \frac{1}{4}} - \frac{1}{2}\rfloor \\ &= \lfloor\sqrt{\frac{575}{2} + \frac{2}{4}} - \frac{1}{2}\rfloor \\ &= \lfloor\sqrt{\frac{575 + 2}{2}} - \frac{1}{2}\rfloor \\ &= \lfloor\sqrt{\frac{577}{2}} - \frac{1}{2}\rfloor \end{aligned}$$

No pressure here, just use a calculator.

$$n_4(575) \quad = \quad 16$$

We can now plug in to our original equation.

$$\begin{aligned}
v^{-1}(h_g^{-1}(r)) &= 4(\frac{575}{1+n_4(575)} + 2n_4(575)) - 208 \\
&= 4(\frac{575}{1+16} + 2(16)) - 208 \\
&= 4(\frac{575}{17} + 32) - 208 \\
&= 4(\frac{575}{17} + \frac{544}{17}) - 208 \\
&= 4(\frac{575+544}{17}) - 208 \\
&= 4(\frac{1119}{17}) - 208 \\
&= \frac{4476}{17} - 208 \\
&= \frac{4476}{17} - \frac{3536}{17} \\
&= \frac{4476-3536}{17} \\
&= \frac{940}{17}
\end{aligned}$$

Thus, we need over $\frac{940}{17}$ forward speed into a double jump to grab the hangable ceiling just above the castle entrance from the close end of the bridge's banister. Written as a decimal approximation, this is approximately 55.294118 units per frame of forward speed.

If you experiment with this in-game, you may notice that there is some inaccuracy. For example, 55.293 forward speed may work. This is due to floating-point rounding errors accumulating through the game's physics logic over several frames. Our solution assumes zero precision issues, which does not reflect reality. Thus, the exact floating-point bits will differ based on version of the game, platform, processor, rounding modes, and compiler. For example, on my laptop running an Intel Core i7-1165G7 with nearest rounding and testing with commit `d7ca2c0` of `sm64ex` compiled with `gcc-15.2.1`, the exact minimum floating-point value for Mario's forward speed is `0x425d2d05`, or approximately 55.293964 units per tick. This is a difference of -0.000154 from the expected result.

Despite this, our method is still preferable because it removes a huge amount of unreliable experimentation, and we can instead focus around testing the range of floating-point error around our expected result. We can also do many other things, since we have Mario's peak jump height and required speed as mathematical functions, not random numbers from experiments.

3.2 Optimizing For Multiple Variables

Imagine the case where we are jumping up a steep slope. Our forward speed decreases the higher we run up the slope, but our starting height increases the higher we go up the slope.

If we are looking to maximize height in this circumstance, we could try jumping every frame we're on the slope to maximize this value. What we can do is set up a function which represents our jump height after running a given distance up the slope, then use Calculus to optimize this function.