A (Rough) Guide to the Platformer+ Settings

Updated for the release of P+ 1.44

The Platformer+ engine plugin for GBStudio has been out for about a month, and has gone through some pretty drastic expansions and modifications in that time. So I thought I would document what the different settings do, where there are small issues, and where some future developments might take place. If you have questions about the plugin, or discover any bugs, I regularly check the GBStudio discord and usually can get a patch together in a day or two. Thanks for all your support, and please drop me a line when you make something with P+, I always love to see what people create!

New Settings

New Events

Platforms

The first two settings both concern new ways of interacting with platforms. There is a variable you can store using a Platformer+ event plugin to check whether the player is currently attached to a moving platform.

Drop Through Platforms: This allows the player to drop through collision tiles that are set to collide only from the top. The options controls what button causes the drop to happen.

Off:	Disables drop-through.
Down Button (default):	Pressing the down arrow drops through.
Jump and Down:	Pressing these two keys together drops through.

Platform Actor Collision Group: This option allows you to make actors into platforms while maintaining the full functionality of actors (they can move, and run on_hit events). These are actors that you set to a specific collision group, and then specify that collision group here. The plugin will attach the player to the top of any actor in that group, when the player descends from above. No other directions are affected. Controlling the actual movement of the platforms, disabling them, or any other actions are all handed through scripting.

None (default):	Disables platform actors.
Collision Group X	Enables, and sets the collision group for moving platforms

Solid Actor Collision Group: This option transforms an actor into a solid object that the player cannot pass through. The actor can move, push the player, and will run collision scripts. The solidity applies to all directions, so the player can land on the actor or bump their head while jumping. Currently the wall slide ability does not work with actors however. To enable this setting, specify the collision group of the actors that you want to be solid.

Jumping

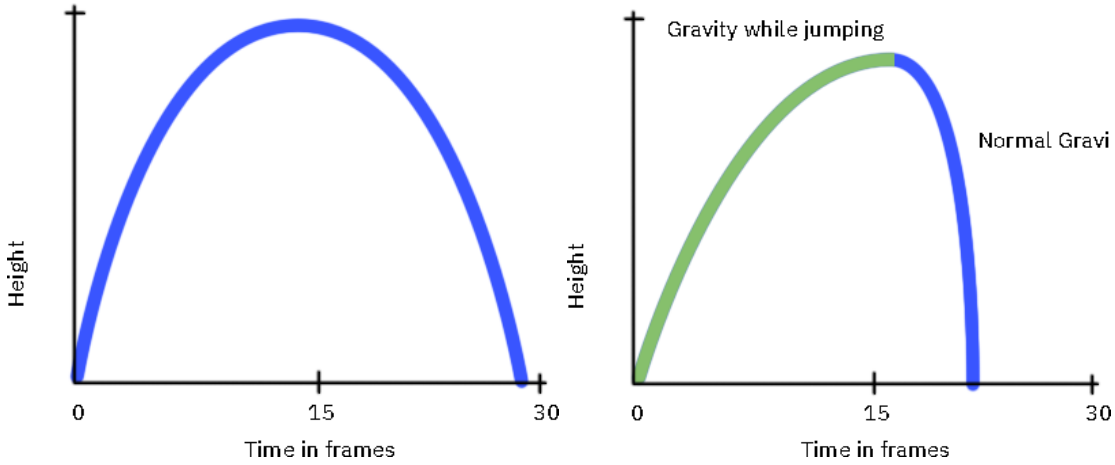
The options in jumping are interconnected with many of the default options from the platformer engine to ensure compatibility. Unfortunately that means you'll have to move between the two panels. Most of these options are subtle tweaks that change the 'game-feel' of a jump. The Platformer+ event plugin allows you to track whether the player is grounded or is currently jumping. The jump state lasts for the number of jump frames you have set while the player is holding the jump button (but not, for instance, while falling after a jump).

Minimum Jump Height,

Jump Frames
Jump Velocity,
Gravity While Jumping

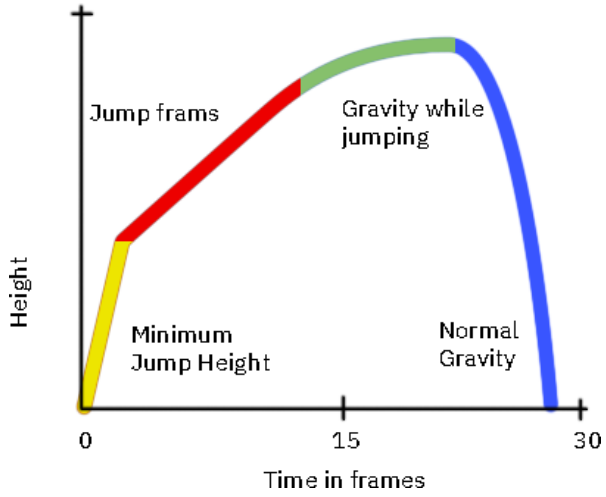
To understand jumping in Platformer+, we need to take a look at how these four elements all fit together.

In the original GBS platformer engine, Jump Velocity adds a certain amount of upward force to the player at the moment they press the jump button. Gravity while jumping applies a different amount of downward force throughout the upward part of the jump, as long as the player is holding the jump button. This gives the player some control over the height of the jump (as long as the gravity is set to a lower number than regular gravity. To visualize this difference we can graph the player's height over time like this:



In the first graph, we have consistent gravity; in the second, the gravity is less during the upward part–so the player has more time to maneuver, before accelerating quickly.

That extra control is good, but we can do better! Steve Swink, in his book Game Feel, uses the terminology of musical instrument ADSR envelopes to describe how game mechanics react to input. ADSR stands for Attack, Decay, Sustain, and Release. We don't need to get too deep into this, but the new jump model in Platformer+ adds some more controls to shape the jump. First, the minimum jump height replaces jump velocity as the force applied on the first frame. Jump velocity now has a different role: it is applied for a fixed number of frames–set with the Jump Frames input–at the start of the jump. So we have two kinds of upward motion: the first boost, followed by 1-100 extra frames of force. We also keep the two kinds of gravity. That means there are four phases to the jump with different dynamics: the first frame, the boost from the jump frames, the time after the jump frames while the lower gravity kicks in, and finally the descent. It looks something like this:



Note 1: If you want to approximate the older style GBStudio gravity in this new model, set the jump frames to 1, the Jump Velocity to 0, and copy your old jump velocity to the Minimum Jump Height field.

Note 2: Because the variable that holds vertical velocity can't go any higher than 32,767 the engine has to do some tricks to make sure that the combined jump forces don't exceed that number. If you max out Jump Height Min, and Jump Velocity, while reducing the frames to 1, the engine will overwrite your settings during run-time.

Extra Jumps: This setting allows the player to jump a second time in mid-air. That second jump uses the same dynamics as the ground jump. You can give the player more than one extra jump by setting the number higher. The counter resets every time the player touches the ground.

0	Disable air-jumping
1-254	Sets the number of jumps the player can perform consecutively
255	Infinite jumps

Height Reduction on Subsequent Jumps: This reduces the overall jump power of each consecutive air-jump. This is the effect of multi-jump in Kirby games, for instance. The numbers need to be relatively high (~4000) to be noticeable. This option only takes effect if the Extra Jumps setting is on.

Coyote Time: A common problem in platforming games is that the player presses the jump button just after the character has left a ledge. Because the character isn't touching the ground, the default GBS Platform Engine won't allow them to jump. Coyote time is a simple counter that keeps track of how long ago the player last touched the ground, and if it was recently enough it allows the player to jump even though they are in mid-air. This variable is measured in frames.

0	No coyote time.
1-10	Number of frames to allow jumping after being on a platform.

Jump Buffer: The flips side of coyote time. Sometimes a player will hit the jump button too soon, right before they land. Again, the GBS engine treats this as the player still being in the air. The jump buffer keeps track of how long ago the jump button was pressed, and if the player lands soon afterwards then it automatically triggers a jump.

0	No jump buffer.
1-20	Number of frames to buffer the jump button for.

Wall Jumps: This allows the player to jump when they are pressed up against the wall. For this to trigger, the player must be directly against the wall and pushing into it when they hit jump. By itself, this can be a little tricky to get the timing right. By enabling wall-slide, it becomes much easier, and the player gets a wall-jump buffer like the regular jump buffer. The variable is the number of consecutive wall jumps the player can perform before touching the floor again.

0	Disable wall-jumping
1-255	Sets the number of consecutive wall jumps.

Enable Wall Slide: This is a simple toggle that turns on or off the wall slide ability. The speed is set below with the wall-slide gravity. Turning this on also changes the way a player interacts with a wall, so that it change the direction the player is facing to show a different animation, and can help with dashing in the correct direction.

Off	Disables wall-sliding
On	Enables wall-sliding

Wall Slide Gravity: Controls the speed at which the character descends while attached to a wall. The player will descend at a constant rate (whereas normal gravity is cumulative).

0	Stick to the wall
>0	Slowly slide down the wall while pressing into it.

Wall Kick Off: When wall-jumping, the player often has to rapidly switch from pushing the directional pad one way to pushing it the other, and this can make wall-jumping frustrating. Additionally, if the character remains pushed up against the wall, then the vertical component of a jump can get cut off. The Wall Kick Off variable controls how much outward force is applied to the avatar whenever they wall jump.

Note that this variable cannot be turned off entirely when wall-jumping. There is a minimum amount of kick-off necessary.

Float Input: This option enables the Float mechanic and determines what key the player uses for it. Floating allows the player to descend at a constant, usually slower, rate than gravity.

None	Disable the float mechanic
Hold Jump	Float automatically if you hold jump after you start descending
Hold Up	Float while holding the up arrow and descending.

Float Fall Speed: This variable sets the speed at which the float mechanic descends.

Horizontal Motion

The horizontal motion controls are a bit of a varied category. These options generally change how the player accelerates and decelerates to create slightly different feelings while walking and running. The Platformer+ event plugin allows you to track when the player is running, and what the intensity of that run-state is (from 0-4).

Air Control: When people jump in real life, they can't change direction mid-air. Platforming avatars, however, often can and we call this air-control. Some game designers like to disable this to create more realistic or cinematic effects, as in games like Prince of Persia or Another World. By disabling air-control, the player's horizontal velocity is set by the speed they are traveling when they press jump, and it doesn't change until they land or hit a wall.

Change Avatar Direction in Air: By default, GBStudio keeps your avatar oriented in the direction you were facing when you started the jump. This option updates the avatar to face in the direction of the most recent key-press.

Air Deceleration: By default, the GBS Platform Engine does something similar to disabling air control because it doesn't slow the character down in the air. So if the player doesn't press left or right, their character will keep the same momentum. Air deceleration adds a bit of friction so that the character will come to a stop unless the player actively presses a button.

Run Style: By default, the GBS run command adds a small amount of acceleration to the character every frame, smoothing accelerating it from a stand-still to a full run. That is often good, but there are other ways of conceptualizing how running should feel and work. This option allows you to select from additional styles.

No Running: Disables the run ability entirely.

Default Smooth Acceleration: GBStudio's standard run model

Enhanced Smooth Acceleration: Uses walk acceleration until the player gets to their full walk speed. Also uses turn acceleration (below).
Immediate Run Speed: Player instantly accelerates to full speed.
Two Run Speed Levels: Keeps track of acceleration behind the scenes. The player walks at a normal speed until they've built up enough acceleration to match their full run speed, at which point it switches over.
Three Run Speed Levels: As above, but adds a middle tier in between walking and running.

Acceleration when Turning: By default, GBStudio allows your character to turn instantly while running at full tilt. Some of the run options above now allow you to keep some of that momentum when the character turns. This option sets the speed with which the player bounces back.

0 Disable turn acceleration

Jump Boost from Run Speed: This option allows you to add more height to a jump based on the speed that the player is moving horizontally. It calculates the height based off the speed, rather than the accumulated acceleration–so the different run speed tiers will give consistent boosts, while the boost from smooth acceleration will vary each frame.

Note: the code for this has been re-written in v1.4 to offer more dramatic changes in height. You may need to adjust your prior settings.

Dashing

The dash is the most complex new mechanic implemented by Platformer+ and there are a number of different options for players to tweak about how the dash functions. However there are still many ways that you might want to tweak the dash, and you can do so by using the Platformer+ event plugin to read the plat_state variable and check if the player is dashing. That will allow you to tweak the animation, add effects during the dash, and tweak its final moments.

Dash Input: This option enables or disables dashing in your game, and determines what key press it is assigned to. There are three options: double-tap (left or right), the interact button, and down and jump. Down-and-jump is primarily meant for games that use dashing to implement a slide mechanic, and isn't recommended if you allow for air dashing.

Dash Style: Allows you to select if the player can dash only when touching the ground, only while in the air, or both. Note that the ground dash only requires you to be touching

the ground when you start, and will send the character over the edges of platforms. A future update may add an option to ground dash only to the edge of a platform.

Dash Momentum: Dashing, unlike regular movement, tries to travel a fixed distance from its start point to its end point. This option allows you to control what other forces come into play on that trajectory. If you enable horizontal momentum, the character will end a dash moving at their full running speed. If you enable vertical momentum, the character will be able to jump while dashing (think Mega Man X) and gravity will pull them down. If you turn off dash momentum, the character will travel a straight line and stop at the end of it.

Note: The option to wall-dash below requires that the dash calculate whether or not there is an open space at the end of its trajectory. Adding gravity and vertical collisions makes it impossible to predict that end-point (at least in GBStudio). So if you enable vertical momentum, you cannot dash through walls.

Note 2: The code for dash-jumping has been re-written. The results are much more pleasing!

Dash Through: This setting allows you to change how the dash interacts with objects along its path. It doesn't change anything about the interactions at the landing site–so if you dash into an enemy and land on them, you will take damage as normal. However, you can disable interactions on the path between the start and the finish. There are three levels of interactions you can disable. You can turn off actor interactions, which means that the player won't trigger any scripts on actors in their path. You can turn off actor interactions AND trigger interactions, so that the player will not run any scripts in trigger areas. Finally, you can turn off wall collisions as well. If you turn off collisions, the dash still calculates whether its ultimate end position is within a wall or not, and if it would put the player in a wall it chooses instead to dash normally. Sometimes this means the player thinks they should be able to dash through something but it doesn't work. Future versions may have a fix for this, but currently it is working as designed.
Note: Actor interactions are expensive to calculate, so P+ only makes use of the one check that is typically run in the basic engine. That means that if you set Dash Time to a low enough number, and the distance high enough, you will dash through actors even if Dash-Through is set to None.

Dash Distance: The total distance covered by the dash. Divide this by Dash Time to get the distance traveled each frame. Setting this really high and Dash Time really low means that the camera has to quickly catch up to the player. There is some basic camera smoothing to help with that jitter, but it can still be a little odd.

Dash Time (frames): The time it takes for the dash to cross its total distance. This is measured in frames, and Game Boy games run at 60 frames per second. I've capped this number at 30, but if you feel like you want to increase the cap for any reason, it's easy to edit in the engine.json.

Dash Recharge Time: The time (in frames) before the player can dash again after previously using a dash. Currently the dash doesn't have any recharge setting for touching the ground–so you can air-dash multiple times in a row if the recharge time is low enough. In the future there will probably be an option for resetting the dash when landing.

Engine Field Events with Platformer Plus

One of the cool things about combining Platformer+ with GBS 3.1 is that we now have events that can change engine fields. That means that you can have power-ups that enable double-jumping, wall-jumping, dashing, or floating. It also means that you can change the gravity or other aspects of the physics on the fly to create slippery zones or shorter jumps.

However, there are a couple of caveats about making changes to some of the jumping and dashing numbers in Platformer+. The amount of jump force applied during each jump frame is calculated when the scene starts, as is the amount of dash distance applied during each frame of dashing. That means that any changes you make to the Jump Velocity, Jump Frames, Dash Distance, or Dash Time, won't register until you move into the next scene. The initialization phase also has some safety checks to make sure the variables all stay within a valid range–however there are some creative ways to bypass those checks when changing things during gameplay. If your character's jump ever stops working when you update an engine variable, try dialing back the amount that you're increasing that variable.

Platformer+ Player Fields

In addition to the engine fields, Platformer+ comes with some additional custom events for monitoring its unique engine fields.

Store Platformer+ Fields in Variable:

This event allows you to check the value of some useful variables.

Player on Moving Platform: True if the player is atop to a platform actor or solid actor.

Current Run Stage: Useful for tracking how much acceleration the player has accumulated from running.

-1 = The player is moving in the opposite direction from the keypress.

0 = The player is not running

1 = Top speed for immediate and default smooth running.

2 = Top speed for enhanced smooth running.
3 = Top speed for Two-tier running
4 = Top speed for Three-tier running.

Current Jump Type: Tracks the type of jump the player is performing.
0 = Not jumping
1 = Jumping from the ground
2 = Jumping in the air (ie. double-jump)
3 = Jumping from a wall
4 = Floating

Frames of Coyote Time Left: let's you know if a player can jump even if they aren't grounded.

Dashing is Frozen: Not currently implemented.

Store Platformer+ State in Variable:

Platformer+ uses a state machine to track the player's behavior. This means that the player can only ever be in a single 'state' at a time. You can find out which state using this event. The values are as follows:

0 = Entering the falling state (1 frame)
1 = Falling
2 = Entering the grounded state
3 = On the ground
4 = Entering the jumping state
5 = Jumping
6 = Entering the dashing state
7 = Dashing
8 = Entering the climbing state
9 = On a Ladder
10 = Entering the wall-slide state
11 = On a Wall
12 = Entering the knock-back state
13 = Knockback state
14 = Entering the blank state
15 = Blank state

Set Platformer + State:

You can now directly set the player's state in Platformer+. Some of the most useful cases include:

Set to Fall State: You can use this to interrupt a dash or a jump.

Set to Jump State: You can use this to create a jump through code, attach it to a button, or add it to the end of another state like dashing.
Set to Dash State: If you want the player to dash in circumstances that aren't covered by Platformer+, you can use this state to create new dashes. For instance, dashing backwards.
Set to Knockback State: In the knockback state, the player will still feel the effects of gravity and other forces, and will still collide with walls in physical ways. However the player cannot input any commands and the animation state will not change by itself.
Set to Blank State: The blank state is even more dramatic. It zeros out any prior velocity that the player had, and removes collisions with walls (though it keeps collisions with triggers and enemies). Its useful for cinematic events where you want the rest of the game to keep moving while the player is being directly controlled.

Enable Actor Gravity:
Platformer Plus now allows you to give any actor to check whether they are grounded, to fall at the speed of gravity when they aren't, and to collide properly with floors. The ground checks happen once every 8 frames.

Disable Actor Gravity:
Gravity on actors can hog some system resources. When you're no longer using it, turn it back off again.