Last Updated: October 2020 (v2.3)

Quick Reference Guide

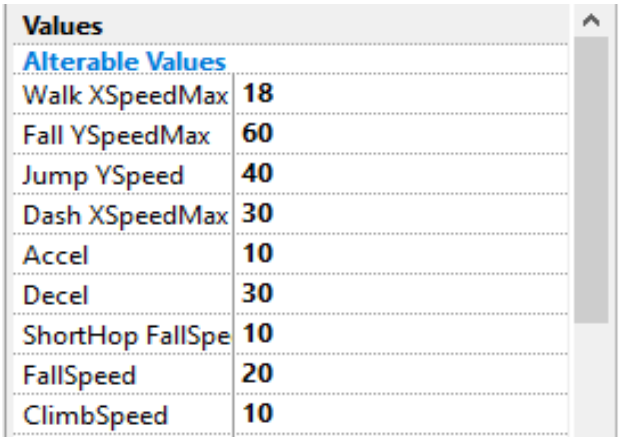
This guide assumes you're using RPM Plus. Not all features are present in the Standard or Super Basic Edition of RPM.

-You are free to use this movement for any commercial/non-commercial product or for learning purposes. The only requirements are that you do NOT share the full engine source, or use the Angela character commercially. You can share characters and custom gadgets for other people to use, preferably through object behaviors (see the Angela sprite for an example).

-This movement engine was designed to take full advantage over object behaviors. The widgets were made to allow you to manipulate certain aspects of the engine without needing to edit the engine code itself (unless there's a quirk you don't like) all from behaviors. You can still use the event editor if you prefer, however!

-To start, simply add your character to the **Group.Player** qualifier. The character will automatically be set to the mask, in addition to having basic animations. You can modify the hotspot of the mask itself if your character does not line up properly. If resizing the mask, make sure the hotspots are properly positioned.

-Use the **Player Attributes Widget** (The A box) to set the initial movement values of the mask itself. You can also determine if the mask is able to jump, slide on ceilings, fall thru platforms, and more. Additional options can be found in the Mask object's Flags



Note that while these values start off as whole numbers, they will be divided down into floating points for the Player Mask to read. More information is provided in the Player Attributes section of this document.

-All Attributes with an "_" in front of it are automatically adjusted during gameplay. In most instances you shouldn't need to these values in the Frame Editor, but you can check/modify them in the Event Editor.

-If you spawn an object that uses Float values (i.e. The Player Mask or Mask Instance obects), you must also update the Float X/Y values upon creation. The object will warp elsewhere otherwise.

-By default, **Fire 1 is Jump** and **Fire 2 is Dashing** (if enabled).

-**Climbing animations requires a minimum of 4 frames to loop properly**. If you character only has 2, simply duplicate them.

-You can make objects frame-independent by using the **_deltaTime** value in the **Engine Control Widget.** When you need to make an object move at a steady pace, instead of adding 1 to its position, add the _deltaTime value instead (or multiply by FS, if the base movement is greater than 1 pixel). You can also create a fastloop that always runs based on the Frame Sync value, and use that loop instead. By doing this, the object will travel at the same speed regardless of frame rate. This only applies if you're intending to use Frame Skip.

-**The mask relies on floating point values for movement.** You'll notice that the Player Attributes starts off with whole numbers, like 30. When the frame starts it gets converted down to 3.0 (for X/Y Speed) or 0.30 (for Acel/Decel). Keep this in mind when wanting to change the values of the mask during play. You can alter the values of the mask directly from that point, using the values in the Attributes as a way of returning to the original set. You can activate Flag 0 on the Attributes object to re-apply all of the values back to the mask. If you plan to use multiple characters on the mask, it's a good idea to set the attributes in their behaviors/event editor instead (with float values of course!).

-Two global values are used, one for X Resolution and one for Y Resolution. These are used to lock the camera object in place once it reaches those positions.

-You can get a "Sliding" HP bar by naming any counter "HP". You can also use HP Drain Delay in the Engine Object to delay the sliding.

-**The mask object has Detect values that determine how many pixels it will check on each side**, in addition to slopes and "Aerial Momentum" (retaining forward motion in air). Experiment with different values if you notice any collision errors. The mask is set to the recommended values by default. The mask is able to return collisions in 4 directions, set in the alterable strings.

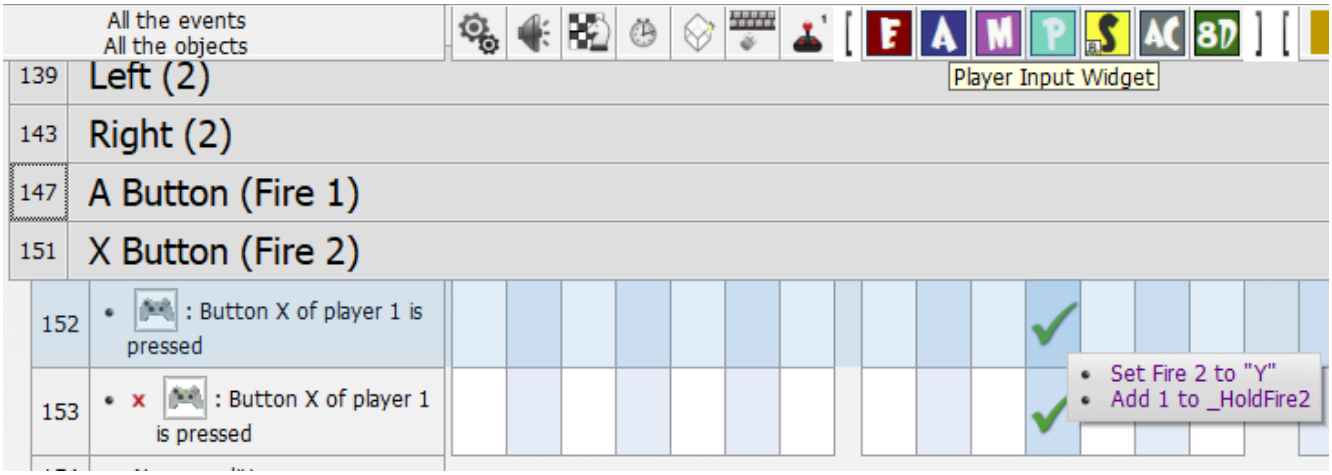
-**Objects in the Physics group need to have a second movement set to Static** to allow pausing of those objects. If you don't know how to do this, go to the object's movement properties and highlight movement. Select the +/- box on the side and use that to add an additional movement. This doesn't work for every Physics Movement for some reason, however...

-You can manipulate your character's animation through the **Animation Control Widget** (The AC box). You can set certain animations to not play depending on its actions, allowing you to play another animation in its place (1 is active, 0 is off). The All function serves as a good catch-all for any special animations you want to apply to your character. You can also "Force" an animation, which makes the current animation play at a controlled rate. Options to allow the character's walk/run animation speed or if it should run only while dashing are also located here. You can do all of this from an object's behavior as well.

-When setting a forced animation, be sure to set the amount of frames the animation has (0-based) in the Forced Length value, and whether or not it should loop. Setting the Forced Speed to 0 stops the animation, and allows you to change animation frames through alternate methods. The Climbing animation, for example, adds directly towards the Frame value when the player moves to allow forward/reverse motion.

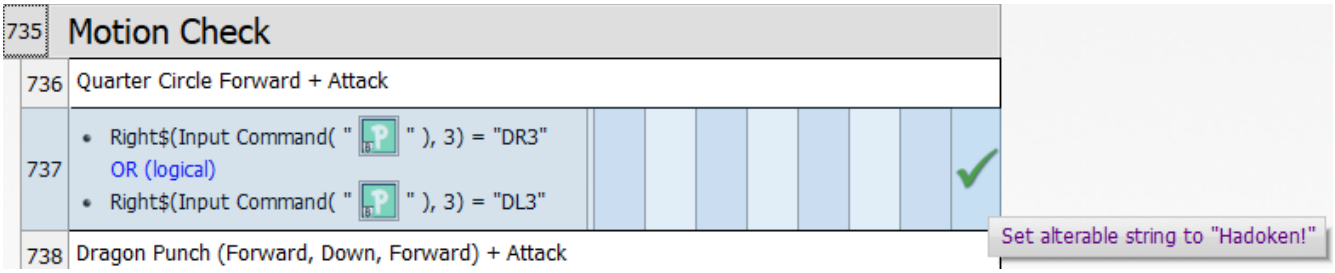
-Speaking of Climbing, **you need to set the amount of frames the animation has in the Player Attributes Widget** to make it loop properly.

-The **Player Input Widget** serves as the basis for converting any player input into something the movement engine can read. By default it uses basic keyboard and XBOX keys, but you can modify anyway you want, as long as you're using the Alterable Strings for Up/Down/Fire !/etc for the mask to read.



In this instance, when the Player is holding down the X Button on their XBOX controller, RPM will read it as Fire 2, and will add to the HoldFire2 value for as long as the button is pressed. Once the player lets go the values are reset.

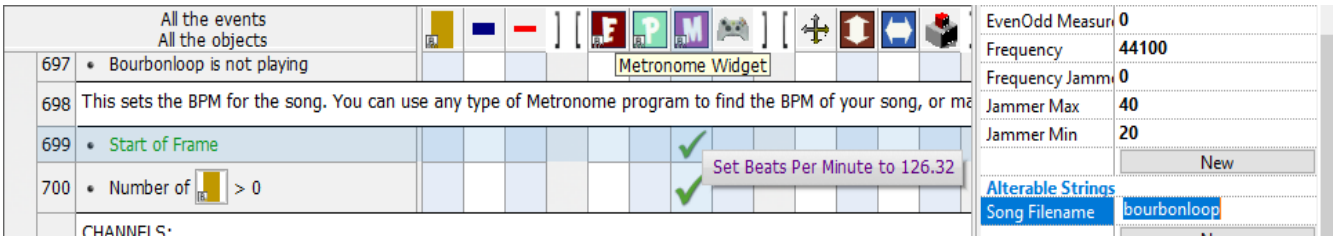
-The **Player Input Widget can also read command strings**. For example, if you wanted to check for a Hadoken motion (Quater Circle Foward + Punch), you would make an event (using "Compare two general values)



*Using the Right$ string on the Input Command string in the Input object, we can check **3** inputs to look for **D**own, **R**ight, Fire **3** (or **D**own, **L**eft, Fire **3**). If done correctly a separate alterable string will say "Hadoken!"*

Note that the Input Command clears based on the Global Input Buffer (default 20 frames) in the Engine Widget after the final input.

-To link a song towards the **Metronome** (The M Box), enter the song's filename in the "Song Filename" Alterable String on the object itself. This seems to work best if the song is in OGG format, where MP3 seems to be hit or miss. Not entirely sure why. From there you'll likely want to either look at the metronome in the debugger or use an outside metronome program to determine the BPM. It's highly recommended that you use floating point values for the BPM for extra precision. Flags 3 and 4 will turn on when a beat or off-beat occurs.



In the Yoshi's Island demo, I used a song called "The Bourbon Theater" by Orange Tulip Conspiracy under the file name "bourbonloop" (an .OGG file). With some trial and error the BPM of the song was determined to be 126.32, which is set in the event editor at the start of the frame.

-For a more in-depth look on what each value does for the objects, check the event editor! The engine itself also includes comments to give you a better understandimg on how it

Widgets

Player Mask

This is the primary objectthat moves the Player in the game world. Objects set to **Group.Player** will be attached to this object and will be automatically assigned a basic set of animations to play.

Values

XSpeed*:* The current moving speed of the mask. Acceleration and Deceleration changes this value when the player presses left/right.

YSpeed: The current jumping/falling speed of the mask. Jump Strength and Weight changes this value when the player jumps or falls off.

Acceleration: How fast the mask speeds up when the player moves left/right. Works best with floating values.

Deceleration: How fast the mask slows down when the player lets go of left/right. Works best with floating values.

XSpeedMax: The maximum moving speed of the mask.

YSpeedMax: The maximum fall speed of the mask.

Climb Speed: How fast the masks climbs up and down ladder objects.

Dash Speed: How fast the masks moves when "Dashing" is active. This changes the XMax value, and resets using the Player Attributes Object.

FallYSpeed: How fast the mask falls down. This is not used when the mask currently has Jump Time remaining.

Jump YSpeed: How fast the mask jumps up while Jump Time is in play. This, along with Jump Height, affects the overall jump height of the mask.

FloatPositionX: The "true" value of the mask's X Position. The difference in the visible X Position and this value allows the mask to move forward and back. Resets when the mask is not moving.

FloatPositionY: The "true" value of the mask's Y Position. The difference in the visible Y Position and this value allows the mask to move up and down. Resets when the mask is on the ground.

HP: How much HP the mask has. Can be used to make a "sliding" HP bar if a counter is named "HP"

Absoulte XSpeed: The absolute value of the mask's XSpeed. Used for checking how fast the mask is moving for animations and other assorted things you may want.

ShortHop FallSpeed: How fast the mask falls down if the player releases jump before reaching its apex. This value works in conjunction with FallYSpeed and is applied via the Player Attributes Widget at the start of the frame.

Hitstun: How long the player cannot react after taking a hit. When changed, the value always subtracts to zero.

Mercy Invunerability: How long before the player can take another hit. This value always subtracts back to zero, and makes the player flash by default.

Jump Height: How long the mask will be unaffected by Fall YSpeed when it jumps up. This works in conjunction with Jump Time.

Jump Time: How long the mask has been jumping. This value increases as long as the player is holding Jump. Once it reaches Jump Height, weight will bring the mask down.

XDir: Controls how the mask moves Left/Right. This should not be modified.

YDir: Controls how the mask moves Up/Down. This also should not be modified.

Ground Detect: How many pixels to check down for the ground. Increasing the value improves detection when the mask lands on a sloped platform (if Slope Detect is active) or a moving platform. Making this value too high can make the mask to "float" a bit before landing.

Top Detect: How many pixels to check up for a ceiling. Increasing this value makes it easier to detect a solid object coming towards the mask, and hitting the ceiling at high speeds. Making this too high could effect slopes and jumping, if the ceiling is low enough.

Sides Detect: How many pixels to check for walls on both sides. Increasing this allows the mask to travel into walls while moving at high speeds, in addition to detecting solid objects that may push it. Making this too high could effect how the mask can move when close to a wall. Slopes are slightly influenced by this value as well.

Slope Detect: How many pixels to check to go up a slope. Increasing this allows the mask to climb up steeper slopes. Making this too high may prevent the mask from entering tight spaces. A Slope Check object can be used to automatically reset the value to 0.

_BoostX: An additional layer of speed to move the mask. This works independent from Xspeed, and has its own deceleration value set in the Player Attributes widget.

Strings

OnGround/LeftCollision/RightCollision/TopCollision: Determines what kind of collision the mask is currently experiencing. You can check against these values to perform various actions depending on the mask's current state.

Climbing: Determines if the mask is climbing something.

Dashing: Determines if the mask is currently dashing.

_TriggerJump: When set to “Y”, this will trigger the mask's jump routine. This will automatically reset to “N” once activated.

JumpingUp: Determines if the mask is currently jumping upwards, and then disables once it begins to fall.

Flags

Force Stop: When active, this immediately sets the XSpeed and YSpeed of the mask to 0.

Acceleration/Deceleration/Jumping/Crouching/Weight: Simply controls whether the mask will be able to accelerate, slow down, jump and so on.

Crawling: When active, this allows the mask to move forward and back while crouched. Note that you will need to manually alter the XMax of the mask, and disable/enable the Crouch animation in the Animation Control Widget when it moves.

Crouch Jump: When enabled, the mask will retain its crouching state when the player jumps and is still holding down.

FallThru Platforms: When enabled, this allows the mask to fall through a jump-thru platform when the player holds Down and presses the jump key.

Jump Off Ladder: When enabled, the mask will jump off of a ladder, rather than simply drop down.

Ceiling Slide: When enabled, mask will slide around angled ceilings.

Quick Turn: When enabled, this allows the mask to turn quicker if the player presses the opposite direction. Disable this group if you need an outside force to change the character's XSpeed (like a wind tunnel).

Aerial Momentum: When enabled, the mask will retain its current XSpeed in the air even if the player makes no input.

Enable 8Dir: When enabled, the mask will be able to travel in 8 directions. More options are located in the 8 Direction Widget

Flash on Hit: When enabled, the character sprite will flash once a value to Mercy Invulnerability is applied.

Angle Detection: When enabled, this will allow the angle detector objects to determine the current angle of the ground.

Pushing: When enabled, the mask will retain its XSpeed while moving forward (although it won't physically go forward). You can use this to play a pushing animation (disable walk/run animations, check for Left/Right Collision and XSpeed) or have other objects be pushed by the mask's XSpeed.

Jump Buffer: When enabled, this allows the mask to jump just after walking off of a ledge. This relies on the Floor Detector overlapping the background to work. You can determine how many frames the player is allowed in the Player Attributes Widget.

SlopeUp SlowDown: When enabled, this will allow slopes to alter the XspeedMax to slow down the player.

Engine Control Widget

The Engine Control Widget provides various functions that need to be present in every frame. By default this is set as a Global object, so all changes made on one frame will be present in another.

Values

Ground Correction Loops: Adjusts how many times the Ground Correction group is ran, which helps keeps the mask leveled. This changes automatically based on FPS.

Hitstop: How many frames to temporarily pause the action using the Gameplay Pause feature. This always subtracts down to 0.

Slope Check Offset: How many pixels the Slope Check object is above the mask. Once Slope Check overlaps an object, the mask's slope detect is set to 0.

_Stored Slope Value: The initial value of the mask's Slope Detect at the start of the frame. When Slope Check is no longer overlapping an object, the Slope Detect value is reset.

Global Input Buffer: The amount of frames an input is considered "active" in Input Command from the Player Input Widget.

Stick to Ground: Affects how steep a slope the mask can go down. Higher values allows steeper slopes, but may cause a "jump" if the value is too high when the mask walks off a ledge. Increasing this value also requires you to increase the "Ground Correction Loops" value.

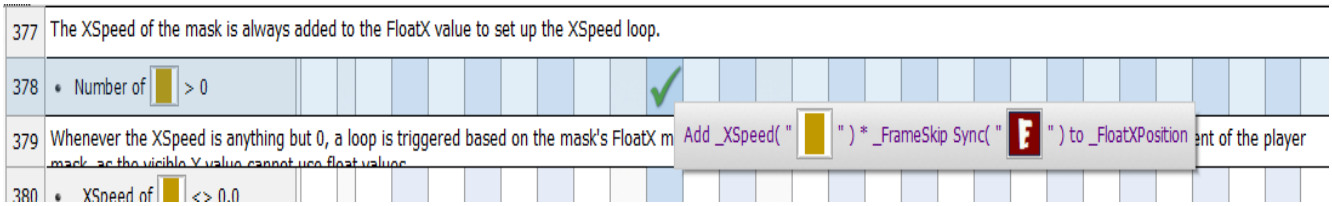
HP Drain Delay: How many frames before the HP bar slides down to the mask's HP value. This only applies to any counter named "HP". This always subtracts to 0.

Pushout: This determines how many times the “Pushout” loop is performed to move the mask out of scenery if it gets stuck.

MovingPlat Ledge Fix: This checks for how many pixels to check for a ledge while on a moving platform, before "shoving" the mask off. When inactive, the mask can be stuck on a moving platform if its currently moving faster than the mask. This usually does not need to be modified.

_X Limit/_Y Limit: These values automatically resets the mask's Float X/Y if the visible position goes beyond this value. This value is determined by the mask's current position plus its X/YSpeed.

_deltaTime: This keeps objects in sync when the framerate drops. To sync an object, you need to multiply values adjusting an objects position or counters using _deltaTime. For example, instead of simply setting to adding 1 to a counter, you would instead add the _deltaTime value. This would allow the object to add at the same speed if the game where to slow down. **At normal speeds, _deltaTime is always 1!**



In this instance, the FloatXPosition used to move the player mask adds the current XSpeed value multiplied by the current FrameSkip Sync value. At normal speeds the value is multiplied by 1, essentially keeping XSpeed the same. If the game were to slow down, _deltaTime would increase and multiply XSpeed to maintain its pace at the slower speed.

Not using this usually results in the object traveling slower during frame drops. You can also always run a fastloop based on _deltaTime instead, but too many fastloops could negatively effect performance. Ideally you'd only want to use _deltaTime if the object needs to move or you have a counter that must add or subtract at a steady pace.

Note that Frame Skip MUST be enabled in the Engine widget for this to work!

Game Speed: A base value used in the calculation of _deltaTime. Altering this value can potentially alter the game speed.

Offset Sprite X/Y*:* This determines how many pixels the player sprite will be away from the center of the Mask.

Surface Distance Check: This determines how far down the Surface Finder will look for the ground while the player is airborne.

_MaskWidth/_MaskHeight: This records the initial size of the player mask at the start of the frame. By default, this is used in determining the size of the Ceiling detector while crouching, and figuring out how to reposition the mask while climbing up a ledge.

_Ticks: The timer used by _deltaTime used to determine how far objects should move based on the current frame rate.

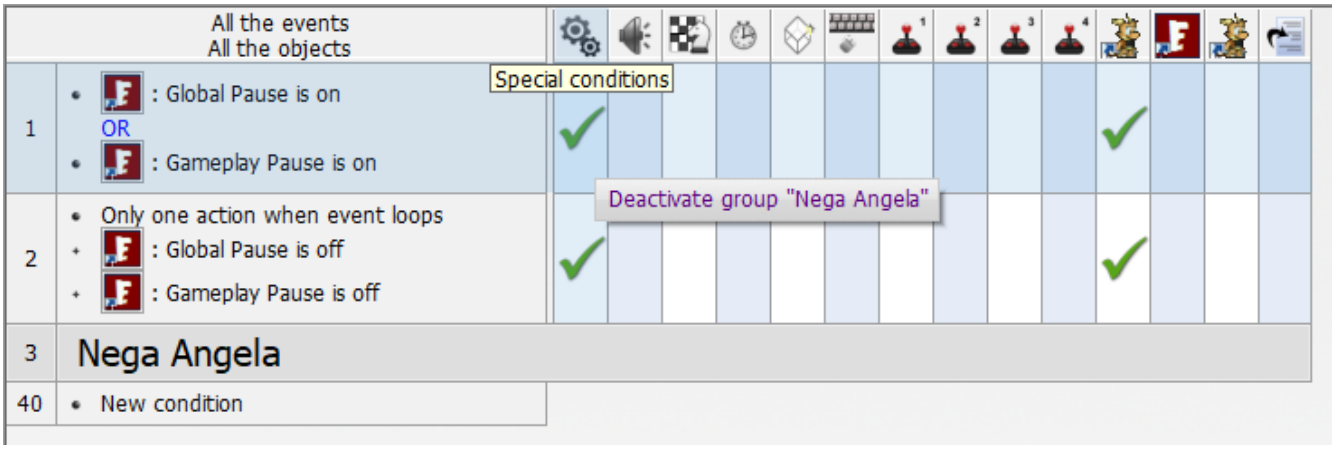
Target FPS: The intended frame rate for the game. At the start of the frame, Fusion will set its internal frame rate to this value. If Fusion's frame rate is lower than this value, _deltaTime will adjust its value to compensate the loss of speed.

Strings

Global Pause - When active, the entire game is considered paused. Ideally this is where you'd want to program in any kind of pause menus.

Gameplay Pause - This also pauses the game, but is not considered a Global Pause for the entire game. Instead this is used to momentarily halt the gameplay for another reason, such as the Hitstop function.

Pause NPCs - This will freeze any object in the NPC group. Note that you'll need to make the objects respond to the actual command, by first programming all of its behavior under one group, and allowing it to deactivate when the Global and Gameplay Pause commands are active.



Ignore Player Pause - This pauses the entire gameplay EXCEPT for the player, including various physics objects.

Use Detectors - When enabled, the Player Mask will use the detector objects to help verify collisions. Disabling it will instead use the player mask itself to determine collisions (the original system RPM used).

_CrouchJumpCheck: This checks to see if “Crouch Jump” was enabled at the start of the frame in the Player Mask. RPM checks for this because it will automatically enable Crouch Jump on the player mask if it enters a tight space.

_JumpThru Slopes - This always pushes the mask upward while on a jump-thru object to ensure it goes up a slope. Note that the mask will also go up if the floor detector overlaps two jump-thru objects simultaneously when this is active.

Player Collisions - This allows the mask to detect scenery and other objects it may interact with.

Player 1/2 Input - This allows the player to make inputs.

Force Float Position: When enabled, this forces the mask to follow it's internal Float values, allowing you to reposition with those values instead. Note that some features may not work as intended, so only use when needed.

Offset Sprite by Angle: When enabled, the character sprite will re-position itself to be inbetween the Angle Detector objects, reducing the gap when the player is on a slope. If your character's hotspot is in the center, this feature won't be needed.

Frame Skip: When enabled, _deltaTime will adjust its value in order to maintain pacing. Note that if you use this option, it's highly recommended that you only allow the player to change the resolution in an options menu. Depending on the display it may cause a huge lag spike that can cause collision errors.

Automatic Detection: When enabled, the mask's Detect values will change based on how fast the player is moving in either direction. Works well if the object is moving fast and collides, but not recommended for tight spaces, as Sides Detect work in conjunction with Slope Detection.

Use volCAMERA: When enabled, RPM will disable its own camera system in favor of volCAMERA. You will need to paste in code from your copy of volCAMERA before it can be used in RPM.

Disable RPM Animations: When enabled, RPM's animation system will become inactive. This will allow you to use your own animation system if you choose.

Use VACCiNE 2: When enabled, VACCiNE 2 will integrate itself into RPM's Player Input Widget. You will need to paste in code from your copy of VACCiNE 2 before it can be used in RPM.

Enable Debug Keys: When enabled, this will allow you to use RPM's Debug Keys. Look in the Global Event Editor in its group to learn more.

Unlink Sprite to Mask: When enabled, the player sprite will no longer follow the mask, and can be positioned manually.

Player Attributes Widget

The Player Attributes Widget helps store the initial movement attributes for the player when the frame starts, as well as enabling and disabling various moves the mask may perform.

Walk XSpeed/Fall YSpeed/JumpUp YSpeed/Dash XSpeed/etc.: These values are applied to the mask once the frame starts, and serves as a reference if the mask changes attributes throughout the game. Note that while the values are whole numbers at first, they're automatically divided into float values for the mask to use. Acceleration and Deceleration are divided by 100 for extra precision (i.e. 40 becomes 0.04) while the rest are divided by 10 (40 becomes 4.0).

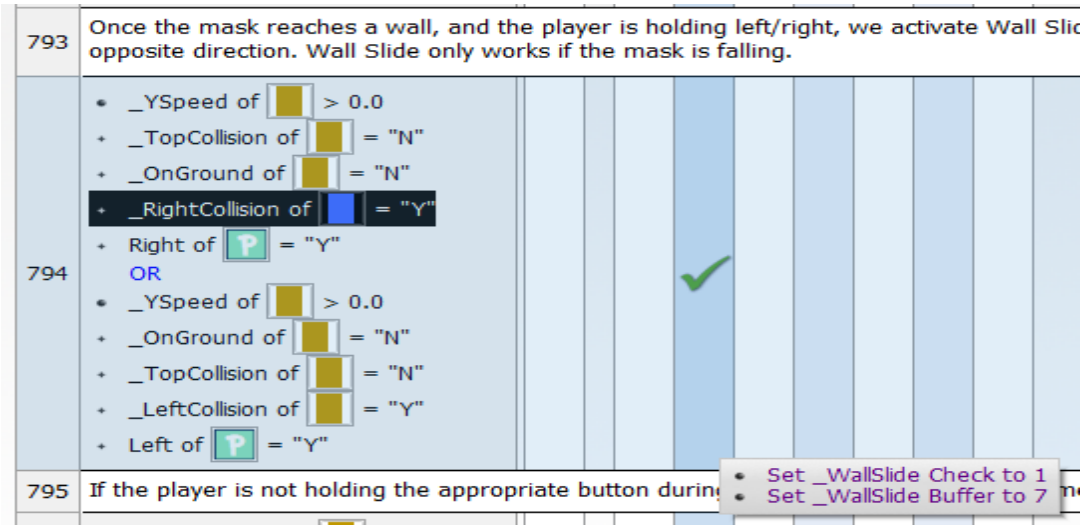
This is primarily to get around Fusion's limitation of not allowing the use of Float Values in the Frame Editor itself. Just note that any further modifications to the numbers in the Event Editor must use float values instead.

RePosX/RePosY: These reposition the mask like "Set X of Object to X + 1", but factors in scenery to prevent overlaps.

ShortHop YSpeedLimit: Determines what YSpeed the "ShortHop" value should stop applying towards the mask. For example setting this value to 3 will prevent ShortHop YSpeed from applying once YSpeed 3 is reached in the Mask. Higher values generally lead to the Mask falling fast after short hopping, while lower/negative values can make the mask "hang" in the air for a bit (depending on the Fall YSpeed set in the mask).

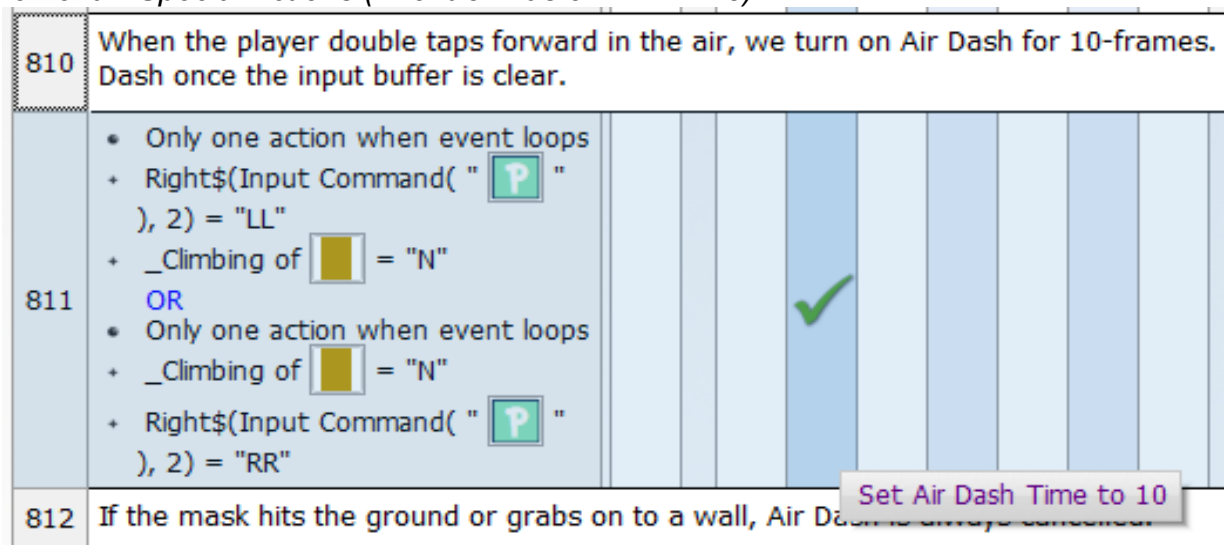
WallSlide Check: Checks to see if the Mask is sliding onto a wall.

WallSlide Buffer: Determines how many frames the player as before the mask will release from the wall.



Dash Type: Determines how dashing is performed. 0 disables dashing, 1 requires holding Fire 2 (or another button if you change it), 2 requires double-tapping and holding forward.

Air Dash Time: Determines how many frames the player will remain in the air dash state. By default this is hard-coded to 10, but like WallSlide Buffer can be changed in *Player Mask* > *Movement* > *Special Actions (Event 811 as of RPM v1.8)*



Climbing Type: This determines how the mask will climb an object. Type 1: Up/Down; Type 2: Left/Right; Type 3: All Directions

***MultiJumpCheck*:** This checks how many jumps the player has performed in mid-air

MaxJumps: This determines how many additional jumps the player can perform in mid-air. Setting to 0 will disable Multi-Jumps. Note that the player get less height as they continue to jump.

SlopeUp WalkXSpeed/DashXSpeed: This determines how fast the player will move when on

a slope. This uses the value **SlopeSlowDown Incline** in the Angle Detection Widget to determine which angle will cause this effect. Note that Angle Detection MUST be enabled in the Player Mask!

AirDashLimit: This determines how many frame Air Dash will remain active.

AirDashXSpeed/YSpeed: This determines how fast and how high/low the player will move during an Air Dash.

JumpBuffer: This determines how many frames the player has to jump after walking off of a ledge. The Floor Detector will change its Jump Buffer value to this to determine if the player can still jump.

QuickTurn Speed: If Quick Turn is enabled, this determines how fast the player will move in the opposite direction if they're currently running.

BurstSpeed Decel: If the Mask's Xspeed exceeds the current MaxXSpeed (i.e. BurstSpeed), this will determine how fast it will return to the MaxXSpeed.

BoostX Decel: This determines how fast BoostX will go back to 0.

LedgeGrab HeightOffset: This determines how high or low the mask will sit while grabbing a ledge.

WallSlideJump XSpeed/YSpeed: This determines how far the player will move once they jump off of a wall during a Wall Slide

WallSlideDown YSpeed: This determines how fast the player will slide down a wall.

Strings

Reapply Attributes: Once active, the Attributes values will be restored to the player mask. This automatically disables again once the values are set.

Enable Wall Slide/Multi Jump/Air Dash: This simply determines if the player is able to perform these actions. Uncheck to prevent the player from disabling them.

Hold RePos Value: When enabled, the RePosX/RePosY values will stay the same instead of resetting back to 0.

Enable Hold Jump: When enabled, the player will continue to jump when the Jump button is held down.

Enable Aerial Crouch: When enabled, this allows the player to perform a crouch in mid-air

Enable Moving Platform Momentum: When enabled, this will move the player mask at the same speed as the moving platform when they jump from one. Aerial Momentum needs to be enabled for this to work.

Enable Ledge Grab: When enabled, this allows the player to grab on to the edges of any backdrop or Solid object.

Disable BoostX Decel: When enabled, BoostX will no longer reset to 0.

Disabled Ledge ClimbUp Reposition: When enabled, the player mask will no longer reposition itself after the climb-up delay in the Ledge Confirm is done. Instead, it must be manually positioned during this time.

Use Ladder Buffer: When enabled, the player mask will need to walk closer to the center of any object under the Ladder group before they can climb it. When disabled, you can climb if any part of the mask or floor detector is overlapping the ladder.

One Button LedgeDrop: When enabled, the player mask will drop from the ledge from a single button press (by default Down). Normally, RPM requires Down + Fire 1 to drop from a ledge.

Animation Control Widget

This controls all of the animations played by the character sprite set in Group.Player. You can use it to determine which animations can play, or allow the use of Custom Animations outside of the defaults.

Standing/Walking/Running/etc.: These control when an animation should play. Deactivating (set to 0) will prevent that animation from playing. This is useful for allowing your character to play other animations.

ActivateDefaultAnimations: This allows all of the default animations to play as intended. Setting to 0 will disable any of these from playing.

Alter WalkSpeed with XSpeed: When enabled, the Walk/Run animation will change speed based on how fast the mask is moving. If disabled, it will use the standard animation settings.

Min Walk Speed: If Walk Speed is enabled, this sets the lowest possible speed the animation will play.

Walk Speed Multiplier: This multiplies the speed of the Walk/Run animation. If the character appears to be animating too slow, increase this value.

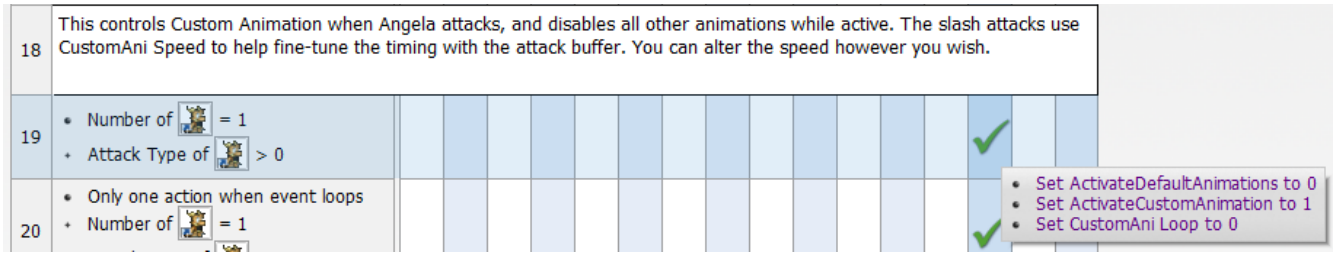
CustomAni Frame: This is the current frame of an animation set to be Forced. This value is manipulated by the CustomAni FrameSpeed value.

CustomAni Length: This is the total amount of frames the current Custom Animation. This value needs to be set in order for the animation to properly loop/end (unless there's an outside value controlling that).

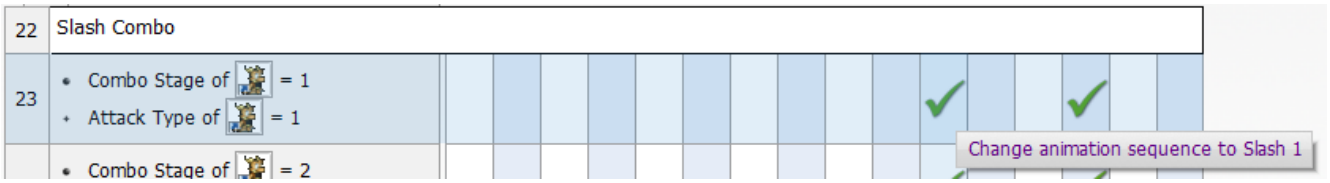
CustomAni FrameSpeed: This is the speed of the current Custom Animation. You can use float values to get more varied speeds. Forced Speed can also be set to 0 to pause an animation.

CustomAni Loop: When active, this causes the current Custom Animation to loop back, based on what's set in the Forced Length value.

ActivateCustomAnimation: When active, the current animation becomes "Forced", allowing you to manually control how the animation plays out.



In this instance we want to play Anglea's attack animation. To achieve this we look to see if the attack type value is greater than 0 (which means the player pressed the attack button). Deactivating all default animations (set the value to 0) allows our attack animation to play regardless of Angela's state. By activating Custom Animation we'll be able to directly control the speed of the animation, and we won't need to make the animation loop on the off chance a looping Custom Animation was playing beforehand.



Once we determine what attack type was performed, We then chose what animation will be played based on that. The CustomAnimationSpeed was set to 0.5 in the animation widget, but you can play around with it to get the speed you want.

When the attack sequence is over (a counter automatically sets Attack Type to 0 after a period of time), we reactivate the default animations and turn off Custom Animation.

Run at Dash: When active, the character will play their running animation only while dashing.

If disabled, the character will play it when they reach their top speed.

Climbing Frames: The amount of frames in your character's climbing animation (0-based). This is needed to properly loop the character's animation.

Player Input Widget

The Player Input Widget is used to convert all keyboard and gamepad inputs into a universal standard that RPM can understand. This eliminates the need to make a bunch of separate events for keyboard and gamepad inputs. XBOX controllers are supported by default, and you can use the format found in its events to create or change inputs as needed.

Deadzone [XBOX]: The distance the analog stick needs to exceed to register an input.

Hold Left/Right/Fire 1/etc.: How many frames the input has been held down.

***L/R Stick Angle [XBOX]*:** The current angle of the Left and Right Sticks of the XBOX controller

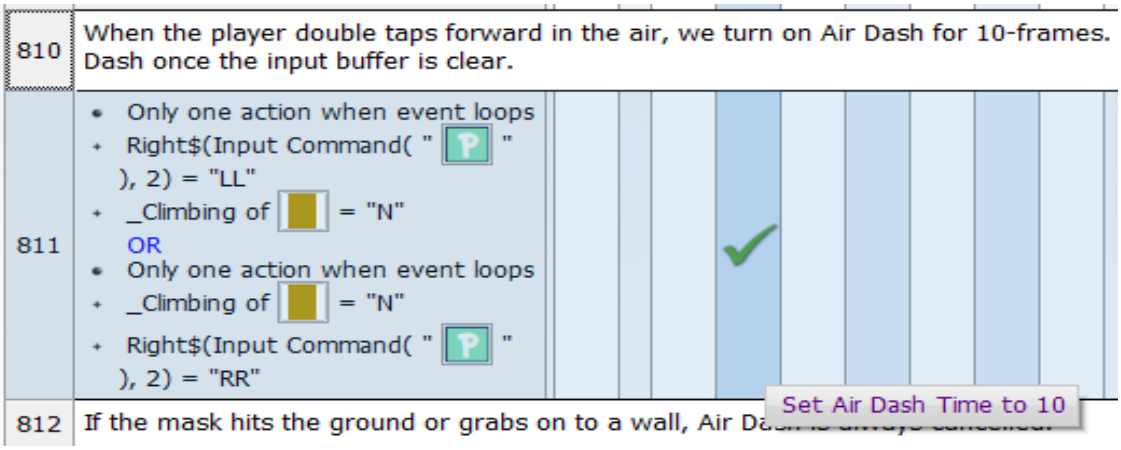
Input Buffer: The amount of frames an input is considered active in the Input Command string. This takes the Global Input Buffer set in the Engine Widget when an input is registered, then subtracts to 0.

Jump Input Buffer: The maximum amount of frames allowed to register a jump input. This is made separate from the standard input buffer as high input buffers can make the mask jump over and over if held. By default, RPM looks for 6 frames of Fire 1 input once pressed.

Block Pause: When active, the player will no longer be able to pause the game.

Fire 1/Fire 2/Left/Right/etc: Confirms that an input is being held down by the player.

Input Command: This registers all the inputs the player has made. This automatically clears when the Input Buffer value reaches 0. Use the Right$ command on the "Compare two general values" action to read inputs.



In our previous air dash example, pressing "LL" (Left x2) or "RR" (Right x2) will activate air dash.

Metronome Widget

This creates a Metronome that can create a sustained beat. Useful if you want objects to react in time with music or follow a specific pattern. Not essential for RPM to function, but its there should you want it.

Beat: The current beat of the song. This resets to one if its greater than the Beats Per Measure value.

Offbeat: The current offbeat of the song. Occurs between each beat.

Total Beats: The total amount of beats added towards the song.

Beats Per Bar: The amount of beats before a bar is added.

Bar: The current bar of the song. Useful for determining position.

Time: The current position of the song based on the Song Filename string. If there is no song, this defaults to the Fusion timer.

Total Beat Delay: The amount of milliseconds the time needs to reach to add a beat. Determined by the Beat Delay times the Total Beats.

Beats Per Minute: The current tempo of the song. Use float values to help sync up your song more accurately.

EvenOdd Beat: Determines if the current beat is either an even or odd value.

Start Delay: The amount of milliseconds before the metronome starts.

Single Beat Delay: The amount of milliseconds between each beat.

Sample Pos: An artifact from another time. Let's just call it a free value.

Accuracy: How close it is before its considered on beat.

EvenOdd Bar: Determines if the current bar is even or odd.

Frequency: The base frequency for a sample. This value is manipulated by the Jammer, so you need to set the frequency of a sample to this value.

Frequency Jammer: Manipulates the frequency to create a distortion effect.

Jammer Min/Max: Determines how high or low the frequency jammer will go.

Accuracy Beats: How many beats Accuracy will stretch.

Sub Beat: The "beats" counted between each actual beat. Can be used to check for notes.

SB Per Beat: How many SBs to check in-between each beat

Total SB/Single SB/Total SB: Similar to it's full beat cousin, just for the SBs instead.

Reset: Resets the Metronome back to its initial values

Jammer Shifting: When enabled. This makes the Frequency Jammer value go up and down.

Apply Frequency: When enabled, the attached sample with play at the current Frequency value.

OnBeat Confirm: Checks to see if an OnBeat has occurred.

OffBeat Confirm: Checks to see if an OffBeat has occcured.

<u>Angle Detection Widget</u>

This helps determine the angle of the surface the mask is standing on, utilizing special detectors around the bottom of the mask. Note that Angle Detection MUST be enabled in the Player Mask for this to work!

Ang: The current calculated Angle based on the positons of the angle detectors. This uses the Advance Direction object.

Detection Loops: The amount of loops the "Angle" fastoop will run. Specifically this value determines how far each angle detector will go down to loop for a slope.

Spacing: How many pixels each detector will be separated from the mask.

Verify X/Y Offset: How many pixels the Verify detectors will be spaced from the mask. Note that Verify X is based on its position from the Angle Detectors.

SlopeUp Incline: Determines how high the angle must be before the Mask is slowed down by a slope. Angle Detection in the player mask MUST be active first!

Allow JumpThru: When enabled, the angle detectors will check angles on jump-thru platforms.

8 Direction Widget

This helps move the mask in 8 directions, going from a side-scroller to more of an overhead type movement. Note that RPM is still focused on traditional side-scrolling, but this will allow you to move your character in a few more directions if needed.

ZSpeed - The current jump speed of the Player Sprite. Note that that player mask will not actually use this value, as it will remain in the same spot to serve as the "ground" for the player sprite to land on. This uses fuctions from the regular jumps in standard Platforming mode.

ZDir - Which direction the Player Sprite will go based on ZSpeed.

Enable Smoothing: When active, this smooths out some of the movement while the player uses the Left Stick on the XBOX Gamepad. Slight changes in direction will cause the mask to move slightly as well

Turn Speed: The minimum speed needed for the mask to turn its sprite around. This is primiarly used for directions like NE or SW.

On Ground 8Dir: Determines if the Player Sprite is currently on the ground.

Player States Widget

This object consolidates many of the actions RPM does into simple to understand values and strings. You can use these to determine the current state of the Player Mask and perform new actions for the player, instead of trying to search through a bunch of values hidden across multiple objects.

Values

StoppedTime: How long the player has not been moving.

WalkingTime: How long the player has been walking.

DashTime: How long the player has been dashing.

JumpTime: How long the player has been jumping up. This resets once the player lands on the ground

FallTime: How long the player has been falling.

AirTime: How long the player has been in the air.

MaxSpeedTime: How long the player has been at their maximum Speed

CrouchTime: How long the player has been crouching

PushTime: How long the player has been pushing an object.

DistanceFromGround: How far the player is from the closet platform below them. The Surface Finder objects are used to determine this value

GroundTime: How long the player has been in the air.

Strings

PlayerMoving: Checks to see if the player is moving, either on the ground or in the air

PlayerAtDashSpeed: The mask's Xspeed is at their dash speed.

PlayerAtTopSpeed: The mask's Xspeed is at the MaxXSpeed.

PlayerOnGround: The player is standing on the ground.

PlayerJumping: The player is jumping up in the air.

PlayerFalling: This player is falling down.

PlayerMaxFallSpeed: The mask's Yspeed is equal to YspeedMax.

PlayerPressedJump: The player pressed the Jump button

PlayerMovingLeft: The mask's Xspeed is lower than 0.

PlayerMovingRight: The mask's Xspeed is greater than 0.

PlayerCrouching: The player is crouching.

PlayerMaxJumpSpeed: The mask's Yspeed is equal to JumpYSpeed

PlayerSlowingDown: The player is decelerating, but Xspeed isn't 0.

PlayerMultiJump: The player jumped again in midair.

PlayerPushing: The player is pushing an object.

PlayerHurt: The mask's hitstun is greater than 0.

PlayerInvunerable: The mask's iFrame's is greater than 0.

Player WallSliding: The player is sliding on a wall.

PlayerAirDash: The player is performing an Air Dash

PlayerTeleporting: The player touched a teleport object and is being moved.

PlayerSwimming: The player is underwater.

PlayerStopped: The player is making no inputs.

PlayerShortHopped: The player tapped the jump button, or didn't complete a full jump.

PlayerOnLadder: The player is climbing something.

PlayerAtWall: The Side Detectors are overlapping an obstacle.

PlayerBufferJumped: The player jumped just after walking off of a ledge.

PlayerBoosting: BoostX is different from 0.

PlayerOnLedge: The Player is grabbing a ledge.

PlayerBurstSpeed: The mask's Xspeed is greater than XspeedMax.

PlayerClimbUpLedge: The player is currently climbing up the ledge.

Master Camera Object

This object dictates the movement and position of the Camera Object. Whenever this object is far enough way, the Camera object will begin to follow it to scroll the screen. Note that this simply a starter camera set, and can be replaced with your own if you prefer. Just delete any groups with Camera in them.

X/Y Distance: How many pixels the camera will move foward in either direction. By default the camera shifts left/right when moving and up/down when the player looks in those directions.

Max X/Y Distance: How far the camera will go in either position.

Camera Shake X/Y: How far the camera will shake in either direction. Larger values give you bigger shakes.

Shake Timer: How long the camera will shake. This always subtracts to 0.

Camera Mode: 0 sets the camera to Smooth Scrolling (default), while 1 sets the camera to Straight Scrolling

Null: Something that no longer does anything at the moment.

X/Y Offset: How many pixels to re-center the camera. By default, this is changed by the mask's X/Y Speed.

Enable Look: If enabled, the camera will shift up or down if the player holds up/down.

Enable RStick Look: If enabled, the camera can be adjusted by the right stick on the XBOX gamepad.

Disable X/YScroll: If enabled, the camera will no longer follow the mask on that axis.

Manual Positioning: If enabled, the Master Camera will remain in place and instead can be manipulated by outside means.

Camera Object

This object directly controls the screen itself, and follows the Master Camera object around to allow for smoother scrolling. If this object is not present, RPM will instead follow the player mask. VolCAMERA also has a series of values present here. View the documentation for volCAMERA for more information on these work.

Camera Speed: This controls how fast the Camera object will move. RPM hard-codes this value to 0.1, but can be modified under the Camera Setup group (Event 1006 as of RPM v1.8 and Event 1385 for RPM Plus v2.3)

FloatX/Y: The float position of the Camera Object

Manual Positioning: If enabled, the Camera will remain in place and instead can be manipulated by outside means.

Room Camera Widget

This object determines the boundaries in which the Camera object will scroll. Used in conjunction with the “Group.Area” objects, this can be used to create multiple “rooms” within the same frame. This object will reposition itself to the top left corner of the room (i.e. wherever your Group.Area objects are placed) as an origin point for the camera. More info about Group.Area objects will be further down. To help determine the size of an area, simply create a box and resize it to fit the boundaries of your “room”. Record the size of the Width and Height to determine your room size.

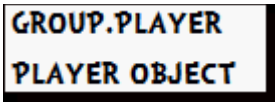
Window Width/Height: The resolution of the current game. This value is used to help calculate the Camera Limits on where to stop. By default RPM uses Global Values A & B for the game resolution, but you can manually type in the values.

Room Width/Height: The size of the current "room" in the frame. This defines the boundaries on where the camera is allowed to move. By default, the widget will pick up the values set by the "Area" objects that define each room.

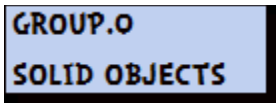
RoomID: The current room selected. Once this value matches one of the Area objects, the widget will reposition itself to that object, which in turn moves the camera to this area as well. This value changes whenever a "select" ForEach loop is triggered on the Doors group, which picks up that object's value.

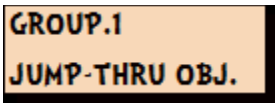
Cam Limits: This defines where the camera is not allowed to scroll. These values are generated based on what was set in both the Window and Room Sizes. Generally, half of the game's resolution is used as a deadzone for each edge of the room.

Groups



The player character itself. Add your character sprite to this group to link it to the mask. RPM will automatically play any basic animations already set up in the character sprite, and will move it along with the player mask.





Group.0 are active objects that cannot be passed through, while **Group.1** are active objects that can be jumped through the bottom. For Jump-Thru Platforms, ideally you'll want to create a small, invisble box at the top of the scenery to prevent potential positioning errors.

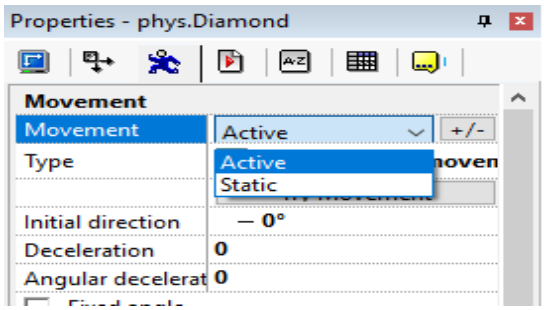
In this instance the actual scenery itself provides no collisions, but the small blocks added to Group.1 provide the player with something to stand on.

***Flag 2* - Player is On Object** : Once triggered this allows the player to stand on it. This relies on the Floor Detector to work.

Flag 0 and 1 are reserved for the Moving Platform Group, along with Values A-E.

GROUP.2
PHYSICS OBJECTS

Objects that use Physics Engine object should be added to this group. Has pre-determined collisions with background objects and can pause certain physics types.. Must have an additional "Static" movement (**NOT** the Physics–Static version, but plain ol' Static) to pause with the game.



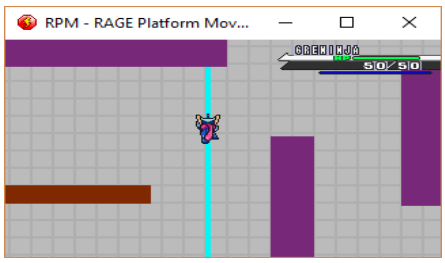
Use the +/- button in the Object's properties to add an additional movement and set the movement to be Static.

These values are stored in the object once the engine is in a paused state. They're then reapplied when the engine unpauses. If you have an object in this group you need to keep these values available for RPM to use. *Values A thru E are reserved for the Moving Platform Group.*

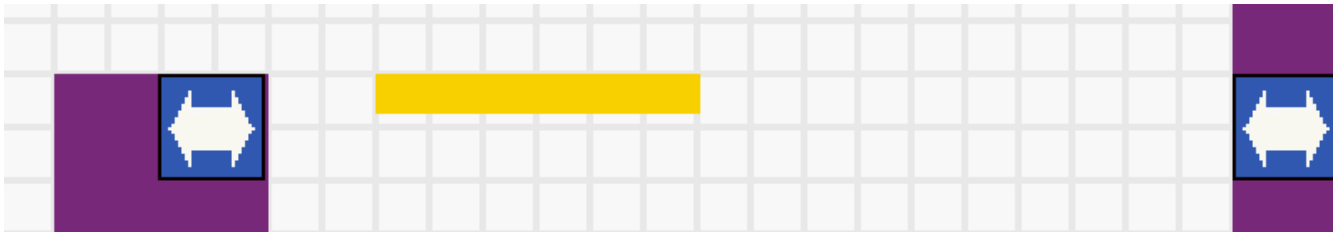
Alt. Value F - *Velocity*
Alt. Value G- *Angle*
Alt. Value H *- Velocity Angle*

GROUP.3
LADDER OBJECTS

Objects in this group can be climbed up or down. Has 3 climbing types (Up/Down, Left/Right/ and 8Directions) in Player Attributes Object. When the mask begins to climb an object, it snaps to the objects X Position as a way of centering it. Like Jump-Thru objects its best to make a small object that the mask will actually climb on instead of the scenery itself. You can also set the Player Mask to either fall off a ladder or jump off with Fire 1 in the Mask properties.



Objects in this group can be stepped on by the mask and will allow it to move with them. This works by determining how fast the object moves on an X/Y basis, and then applying those values to the Player Mask. Group.4 objects must also be set as either Group.0 or Group.1. Otherwise the player will simply go through it. This group also comes with a function to create a simple Horizontal or Vertical moving platform, which can be manipulated with the two arrow boxes shown above.



The yellow platform will toggle its direction each time it collides with one of the Left/Right Arrow boxes.

(Alt Value A) XSpeed: The calculated X position speed of the platform. Works by checking the previous X position. Only activates when the player is standing on it. [Free Movement]

(Alt. Value B) YSpeed: The calculated Y position speed of the platform. Works by checking the previous Y Position. Only activates when the player is standing on it. [Free Movement]

(Alt. Value C) Previous X Position: The X position of the platform from the previous frame. The difference in value is the XSpeed, which is applied to the mask.

(Alt. Value D) Previous Y Position: The Y position of the platform from the previous frame. The difference in value is the YSpeed, which is applied to the mask.

(Alt. Value E) Movement Speed: If horizontal/vertical is active, the platform will move this many pixels every frame.

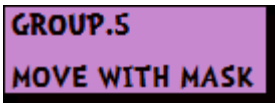
(Alt. String A) Horizontal: If set to Y, the platform moves left/right based on movement speed.

(Alt. String B) Vertical: If set to Y, the platform moves up/down based on movement speed.

If both strings are blank, the platform enters Free Movement

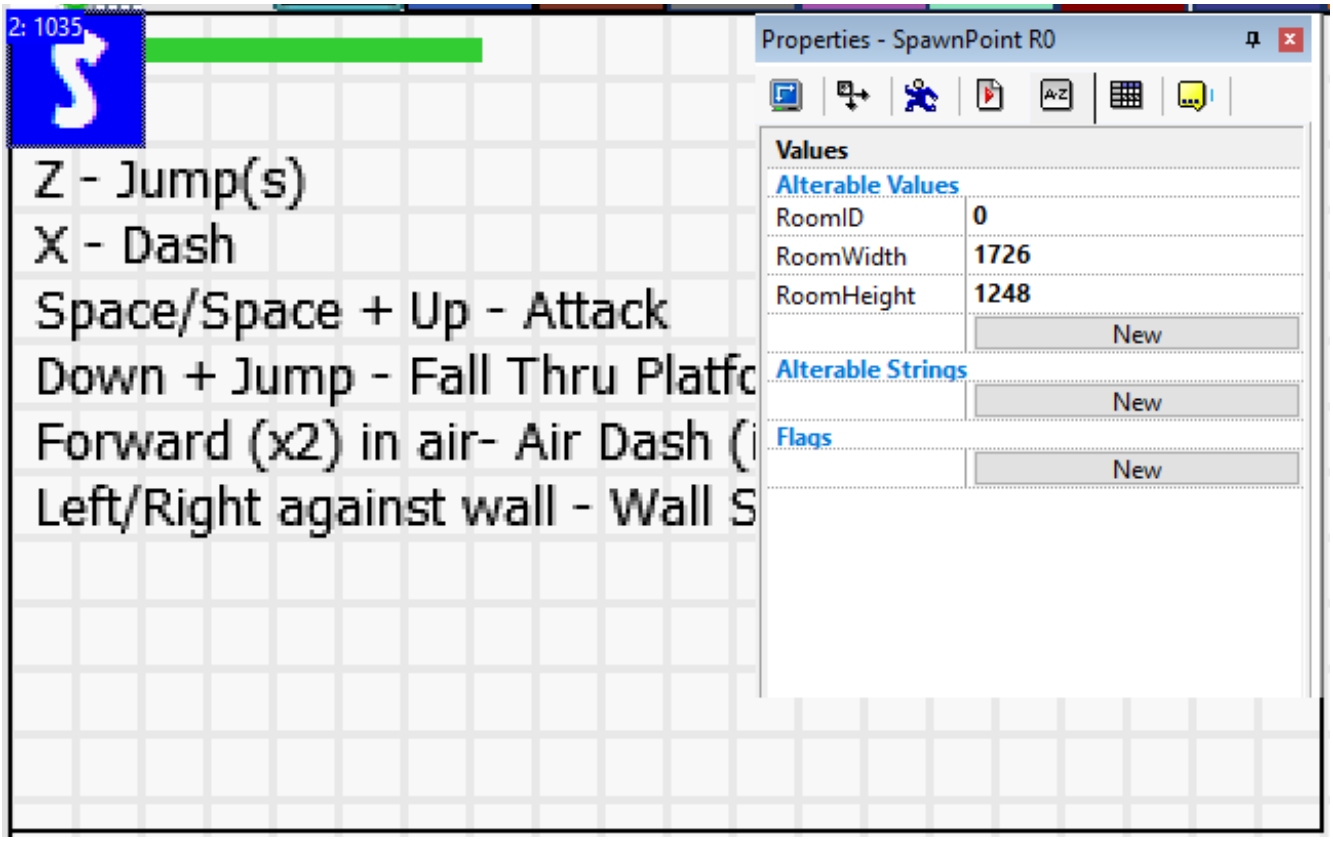
Flag 0 - Toggle Direction (L/R) [Horizontal]

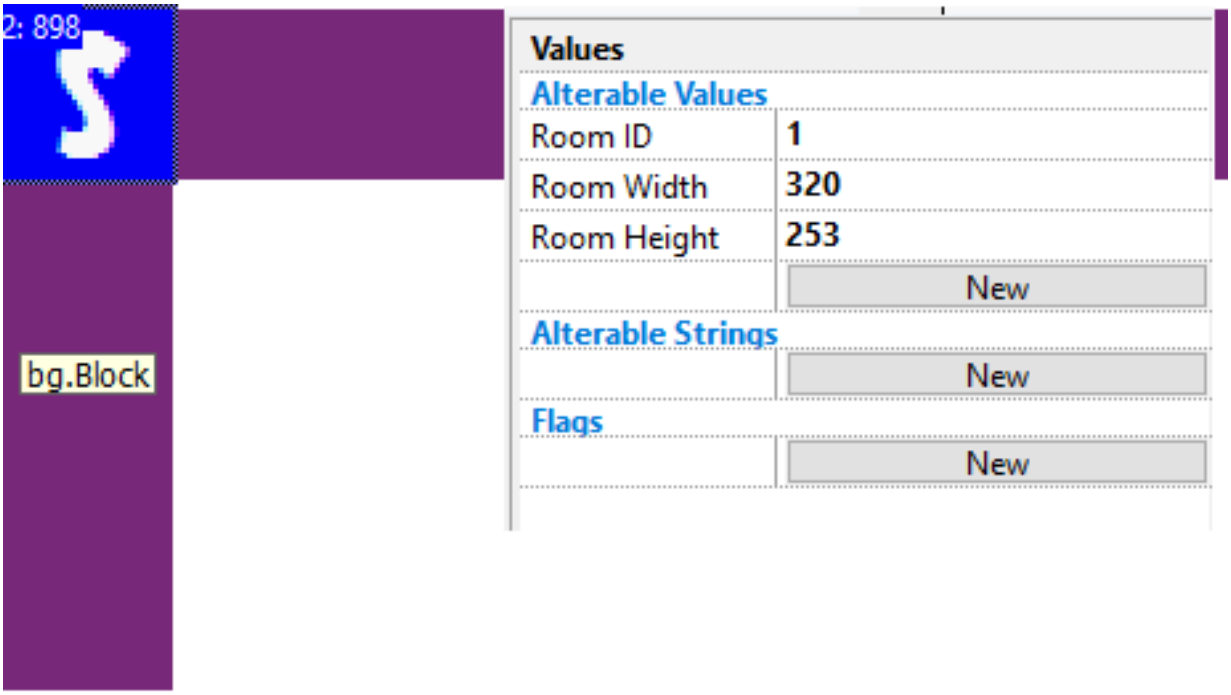
Flag 1 - *Toggle Direction (U/D) [Vertical]*



Objects in this group will move around using the Player Mask's current X/YSpeed. This is usually reserved for detector objects to allow them to keep pace with the mask.

Objects in this group determine the origin for the Room Camera widget to base its values from to create a “Room”. Each of these objects should be placed on the top-left of each Room you want to make, and the size of the room and RoomID must be determined. Each object in this group MUST be a unique instance with different values, and the Room Camera widget must exist in the current frame to use this feature





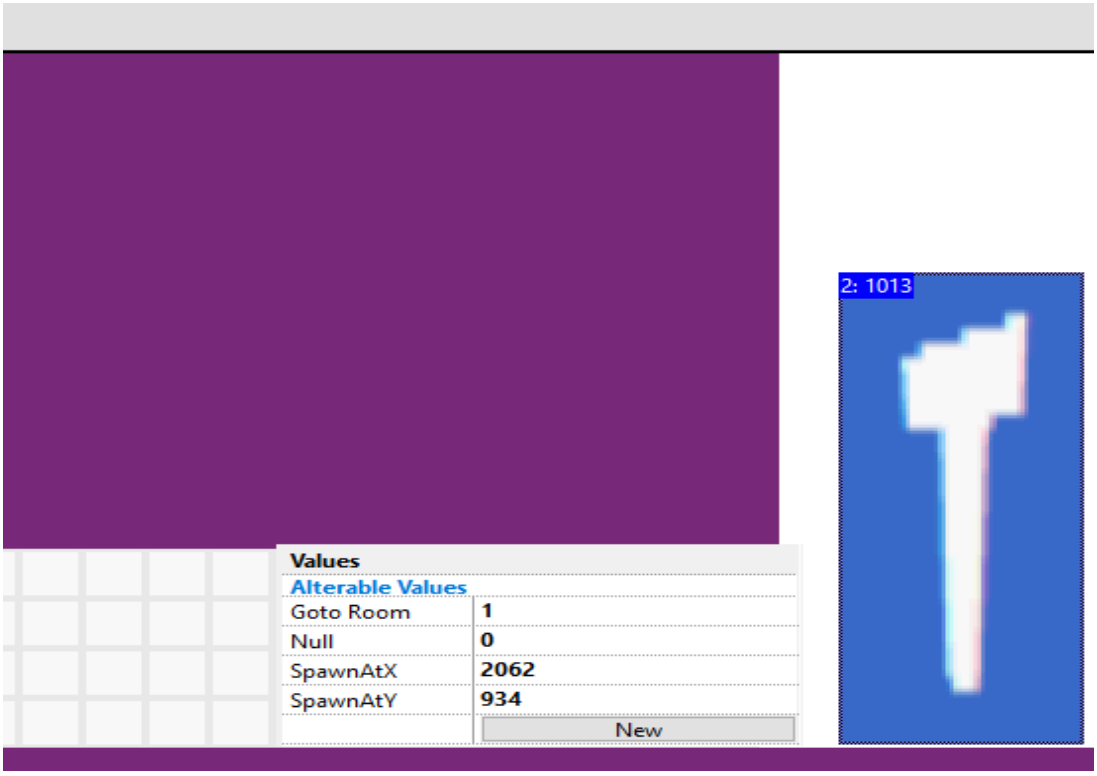
In this instance the Super Basic Level has two Rooms, the very large area at the start, and a small room off to the right. Each room has its own unique Group.Area object (The S Box, but it could be anything) with a separate ID and Width/Height located on the top-left corner of each room. An easy way to determine the size of a room is to create a box object and resize it to match the boundary of your room. Then, record the Width and Height of the box to get your Room size.

Notice how when you reach the end of a Room's boundary, the camera stops despite there being more level off to the side.

RoomID: This identifies each room, which the Room Camera widget selects during a "select" ForEach loop triggered by a Group.Doors object. If the Room Camera selects this value, it will warp to the Area's location.

Room Width/Height: This defines the camera boundaries for the room.

Objects in this group will teleport the mask to specific locations, in addition to determining which RoomID the Room Camera widget will go to. When the mask collides with an object in this group, a Fade Out will occur obscuring the teleportation. Once it fades in, the mask will be in a new room. Like Group.Area, these objects must be a unique instance to work properly and relies on the Room Camera widget existing.

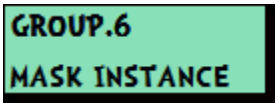


In this instance, a large Group.Door teleporter object exists just outside of the room boundary. Once the player walks into it, the Group.Door object will move the mask to the specified location and set the Room Camera Widget to go to the Group.Area S Box listed with RoomID 1.

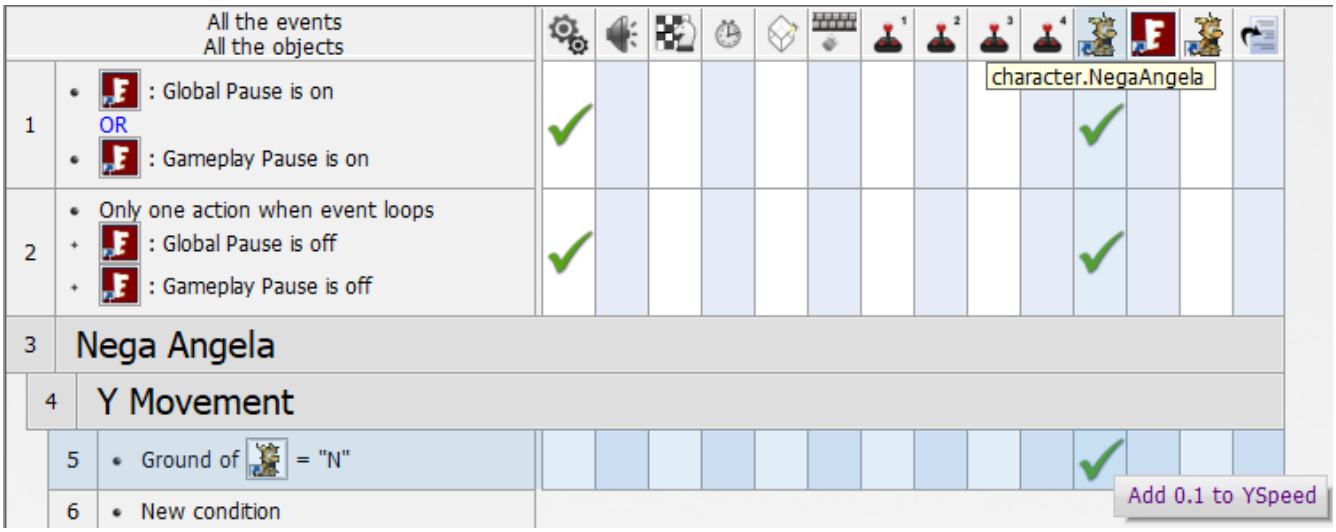
Goto Room: This determines which RoomID (set by any Group.Area object) the mask will teleport to once it collides with this object.

SpawnAtX/Y: This is where the mask will respawn during the "select" ForEach loop. You can use the values in the bottom right-hand corner of the Frame Editor to determine the X/Y of where you want the mask to spawn. Trial and Error is also an option.

Flag 0 *- When the mask collides with a door object, this flag is turned on to trigger a fade in. This will trigger the "select" ForEach loop which will reposition the mask and the Room Camera object.*



Objects in this group will have its own Xspeed/YSpeed and collision detection, loosely based on how the player mask itself works. The Nega Angela objects use this feature to move around, go up and down slopes, and have gravity. **Alt. Values A,B,Y, Z, and Alt. Strings A thru G must be reserved for objects in this group**.



In this example, when the Nega Angela character is not on the ground. We add 0.1 to its YSpeed to make it fall down. Also note how the character's main programming is all in one group. You'll need to do this to allow it to be affected by the Pausing mechanics.

(Alt. Value A) XSpeed: The Left/Right Speed of the instance. Unlike the Player Mask this value isn't changed automatically, and must be set manually.

(Alt. Value B) YSpeed: The Up/Down Speed of the instance. Like with XSpeed this must be changed manually.

(Alt. Value Y) FloatX & (Alt Value Z) FloatY: The floating point positions of the instance. Like the Player Mask these allow you to use floating point values to precisely move the mask around the area with the Xspeed/YSpeed values.

(Alt. Strings A thru D) Ground/Left/Right/Top – These determine the current collision state of the mask.

(Alt. String E) Use Float – This allows the instance to read Float values in XSpeed/YSpeed.

(Alt. String F) Ignore Collisions – This will let the instance pass through scenery.

(Alt. String G) Allow Slopes – This will let the instance travel up and down some slopes.

Known Issues

-Other exporters (UWP, Android, etc.) can produce mixed results. RPM was in development before those were acquired. Using a ton of tiles can potentially cause slowdown on those exporters. **RPM works best with Windows EXE.**

-The mask may not position itself properly on Free Moving Platforms with Machine-Independent Speed active. Instead, you can use the Frame Skip option in the Engine Widget.

-Collision errors are likely to happen with Jump-Thru Platforms at low framerates, while Frame Skip is active. The Floor Detector resizes itself by default to help alleviate this problem (You can remove this feature under the Mask Setup group) . This can also occur if you switch to full screen while standing on one, depending on how your setup changes resolution.

-Not all Physics objects types will pause properly.

-Slope Detect may prevent the mask from entering a very tight space (i.e. something as narrow as the mask itself). You can modify the included Slope Check for this, or make a "No Slope Zone" that sets Slope Detect to 0. At least until a better solution is made.

-The mask may lose momentum if it lands on a very steep slope. You can modify Ground Detect/Slope Detect/Ground Correction Loops to help, though this sometimes can lead to the mask detecting the ground before physically touching it.

-Angle Detection may cause slowdown, especially on other exporters.

-Mask Instance doesn't support Jump-Thru Platforms or slopes for Solid Actives. Angled and very low ceilings can also throw off its collisions.

-Probably something else I missed that you'll eventually find. :)