Index

Important:

Typically the on-line version of the Spriter manual will be more up-to-date than Spriter's built in manual.

You can view the online version at http://www.brashmonkey.com/spriter_manual/
You can also download a PDF version of the manual from
https://brashmonkey.com/spriter_manual/Spriter_Manual.pdf

What is Spriter

Spriter enables the "modular" method of animating where, instead of each frame being a single complete image, it is constructed from many small, reusable images (such as body parts). Each of these images that are used to construct the full frame can be scaled and rotated to further increase the mileage an artist can get from them.
This modular method of animating offers many benefits for several aspects of a game's development and the final finished product:

Time! Because an artist will be reusing a handful of modular images to create all of the frames for a character or effect, there will be much less time spent tweaking and polishing.

Iteration! Let's say it becomes necessary to change an otherwise finished character's head design. Instead of the huge task of redrawing or editing the head in every single frame of full frame animations, the artist would only need to change the handful of the head images that are used across all frames, turning a huge task into a quick and painless one.

Tweaking... Because the modular images (body parts) can be freely nudged around or rotated, it becomes much easier for a non-artist to make tweaks that might be necessary for gameplay, and very easy for the artist to go and re-address whatever tweaks the designer needed to make.

Character variations! Not only does this method allow for super fast and painless creation of alternate characters based on the data of another character, it also allows for an extremely time and memory efficient way of creating all the variations of a character which can change throughout a game (such as collecting power-ups and new equipment).

Huge savings of file and heap space. Instead of each frame of animation being a large complete image, it's simply a tiny amount of data storing the position, rotation etc of each small and re-used "body part" image. The larger and more robustly animated your characters and effects, the greater the savings will be.

Spriter also offers a wonderfully natural and visual way of editing critical aspects of actual gameplay! Here's how:

Per frame, you'll be able to trigger multiple sound samples (with volume control) in an animation.

Per frame, you will be able to place and name an unlimited number of "action points". Perfect for telling the game where to spawn bullets or anchor other sprites etc. (Pro version)

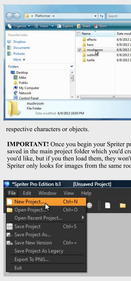
Per frame, you will be able to place, name and set an alpha or numeric value for any images, points, bones, or collision rectangles! (Pro version)

All with ease and immediate visual or audio feedback within the editor. You will also be able to create an unlimited number of "variables" for any "character" (a character is a set of animations) and trigger the change of any variable at any frame of any animation! (Pro version) These features will allow for an incredibly easy and natural way of tweaking actual aspects of game-play within this easy-to-use and highly visual editing tool.

Getting Started

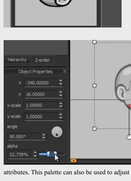
Organizing your project folders and images before you begin.

Before you start up Sprite itself, its important to understand, Sprite is not used to draw images from scratch, it's used to combine, move, rotate and stretch images you've already created in order to create fully assembled frames and animations.



Step 1) (Getting your images ready) Create a project folder which will be used for your Sprite project. Then add sub-folders in which you should organize the PNG images you'll be using to create your animations. For example: If you were creating animation sets for a platformer game, you might first create the project folder and name it "Platformer" and then inside that folder you would create other folder named "hero", "mushroom", "turtle", "effects", "sub-bosses" etc and within each of these folders you'd place the images you'll be using to create and animate those respective characters or objects.

IMPORTANT! Once you begin your Sprite project, the actual Sprite file (.scml) should always be saved in the main project folder which you'd created. You can save backup files in other folders wherever you'd like, but if you then load them, they won't know where to find the required images because Sprite only looks for images from the same root as the .scml file itself.

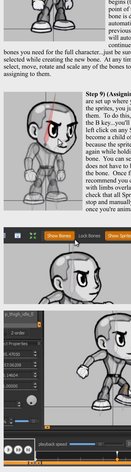


Step 2) (Starting Sprite and creating your project file) Start Sprite and from the main menu choose: File/New Project or hold Cntrl+N.

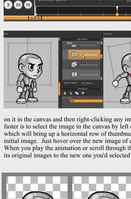
You will be prompted to choose the root folder for your project. Click on OK and then use the file dialogue to direct Sprite to the main project folder you had created. In the case o four example, this would be the folder called "Platformer". Sprite's working interface will then appear and you'll be ready to begin creating your first key frame.

Creating the initial key-frame.

Step 3) toward the lower right of your screen you should see the "animations palette" You'll see Sprite started your new project file with its first entity (character) and first animation for that character. You can double-click on the name of either to rename them to something more descriptive... for example, you could rename the entity to "hero", and the animation to "idle"

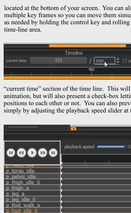


Step 4) (Setting default pivot points for Sprites) On the upper right of your screen (above the animation palette) you should see the file palette. Use this palette to browse through the image folder(s) you'd created in step one to find the images you'll be using to assemble the initial key-frame. Before you begin using the images you might want to take your time to give each image a custom pivot point. (images default to a top-left pivot point, and its often more convenient and leads to better final results to set pivot-points based on actual points of the image in question...for example, the image of an upper arm would scale and rotate more naturally around a pivot point set in the center of the shoulder). To set a default pivot point for an image, double-click on that image in the file palette and a dialogue box will appear giving you the ability to set the pivot point. Once you've set the pivot point how you'd like, click OK.



Step 5) (Adding Sprites to the canvas and manipulating them) Now that your image is ready to use, you can simply start dragging them from the file palette onto the "canvas" in the center of your screen to begin assembling the first key frame. Once on the canvas, you can select any Sprite (the selected Sprite is always indicated by left clicking, and then use the transform controls which appear around the sprite to rotate it or stretch it as you need. You can use the "object properties" dialogue on the lower left of your screen to keep track of or carefully edit any of the currently selected sprites attributes. This palette can also be used to adjust the currently selected sprites opacity.

Step 6) (adjusting the z-order of sprites) As you assemble and tweak your initial key-frame you may need to adjust the z-order of your Sprites. The order in which they are drawn on screen..in other words, which are in front and which are behind). This can be done by clicking and dragging on the sprites in the z-order palette on the upper-left of your screen or by selecting a sprite on the canvas and then holding Cntrl and pressing the up or down arrows. You can also hold Cntrl and press the left or right arrows to send the selected sprite to the absolute top or bottom of the z-order respectively.

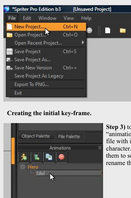


Step 7) (Aligning the assembled key-frame to the canvas crosshair) Now that your first key frame is properly assembled, you might want to make sure that the entire frame (all sprites collectively) is aligned to the "canvas crosshair" in a manner that will be most useful for game engines. The point where the vertical and horizontal lines bisect the canvas represent coordinate 0,0 for the frame. If you left a game engine to draw your frame to the screen at a given coordinate, it will place the animation based on this 0,0 point as it's pivot point or "hotspot". In the case of this example, for a platformer character, you'd likely want the 0,0 coordinate (canvas cross-hair) centered near the feet of the character. To move all sprites at once you can press Cntrl+Alt to instantly select all objects on the canvas or left click and drag a selection rectangle to select all of the objects. You can then use the arrow keys or click and drag on any of the selected objects to move everything at once to properly align the frame.

Step 8) (Creating bones to more easily animate complex objects or characters) While you are not required to use bones to animate with Sprite, and in act, for many types of animations bones would just be an inconvenience, for the case of animating complex objects or characters the initial investment of literally a minute or two to "rig" the character with bones will end up making work much easier and more natural, and save you a lot of time, even resulting in a superior final animation. To create bones, simply hold the Alt key and left click and drag from the point you want the bone to begin to the point where you want the bone to end. The point where the bone begins (the tail) acts as the pivot point of the bone. When you let go of the left mouse button that bone is done being created and is automatically selected...if you create a new bone while the previous bone is still selected, the new bone will automatically be a child of the selected bone. You can continue holding Alt and create all of the bones you need for the full character...just be sure the bone you want to be the next bone's parent is selected while creating the new bone... At any time in this process you can let go of the Alt key and select, move, rotate and scale any of the bones to perfect their position relative to the sprites you will be assigning to them.

Step 9) (Assigning Sprites as children of bones) Now that your bones are set up where you want for the whole character, properly aligned to all of the sprites, you just need assign (child) the appropriate sprites to each of the bones. To do this, simply select a bone by left clicking on it, and then hold the B key...you'll see all Sprites become more translucent. Now if you left click on any Sprite while still holding the B key, that Sprite will become a child of the selected bone. You can tell that it is now assigned because the sprite is now more opaque.. If you click that same sprite again while holding B it will disassociate that sprite from the selected bone. You can select as many sprites to each bone as you'd like. He sprite does not have to be touching the bone, or even be close in proximity to the bone. Once finished assigning all sprites to their respective bones we recommend you quickly play with your character to make sure movements with limbs overlapping the body and the other limbs so you can double-check that all Sprites are z-ordered properly. This way, you won't have to stop and manually fix the z-order of sprites across several key-frames once you're animating.

Step 10) (Animating with bones and sprites) If you've assigned sprites to bones you'll see that if you rotate, move or scale bones, the assigned sprites will be effected with them. If all sprites are assigned to bones you may not have a need to ever select or edit a sprite directly. If this is the case, you can lock and/or even hide all sprites so that you can not accidentally select or edit a sprite by clicking on the show or lock buttons along the top right of the canvas. You can do the same for bones if you have a need at any point to only edit sprites. To start animating, first edit your starting frame by selecting and moving, rotating, stretching or changing the alpha (translucency) of any of the sprites or bones. Then click on another point along the time line and then adjust the sprite or bones as necessary to create the next key frame. Editing any sprite or bone while on a new point in the time line will automatically create a key frame. You can also create a key frame at any time by clicking on the "key all" button near the bottom right of the canvas, or by clicking on the key selected button while one or more objects (sprites or bones) are selected. **(See Key All vs Key Selected for more information)**



Step 11) (Swapping a sprites Image at any time) You might have a need in some animations to actually swap out one image with another at some point. This can be done by simply scrolling to the point in the animation you'd like an image changed, selecting the sprite by left clicking on it in the canvas and then right-clicking any image in the file palette. An alternate method which can be faster is to select the image in the canvas by left clicking, then click and hold on the right mouse button which will bring up a horizontal row of thumbnails off all other images available in the same folder as the initial image. Just hover over the new image of choice and release the mouse button to make the change. When you play the animation or scroll through the time-line you'll see that the sprite now changes from its original images to the new one you'd selected at that exact moment in the animation.

IMPORTANT: When swapping, a new image will be placed in the exact location of the current image based on their respective pivot points, so it's very important to have the default pivot-point of the new image set appropriately. One way to ensure perfect continuity when swapping from one image to the next is to actually insure that each image you'll be swapped has the same exact pivot-dimensions as each other, and are arranged so that all of the default pivot points would have the same exact pivot coordinate. Look at all these head images as an example...all images are buffered with enough transparent pixels to ensure the actual head images are placed per image in perfect alignment with each other...so the exact same default pivot-point coordinates can be set for all...ensuring there will be no jarring "pop" in position as one image is switched to the next.

Step 12) (Editing the timing of key-frames or the entire animation) You'll likely find as your animation progresses that you'll need to tweak the relative distance between key-frames. To do so, you simply left click and drag on any key-frame in the time-line in order to be able to move it to

located at the bottom of your screen. You can also hold the Control key in order to be able to select multiple key frames so you can move them simultaneously. You can also zoom in and out on the time-line as needed by holding the control key and rolling the mouse wheel while your mouse pointer is in the time-line area.

It will also often be necessary to expand or reduce the total length of the animation. To do so you can simply left click in the second number box of the

"current time" section of the time line. This will allow you to not only type in a new total length for the animation, but will also present a check-box letting you either stretch the keys to maintain there relative positions to each other or not. You can also preview your animation playing back at different speeds simply by adjusting the playback speed slider at the left side of the time-line.

Step 13) (Advanced time-line editing) As your animation gets closer to being perfected, you might find the need to slightly offset the timing of one or more objects relative to the rest. Sprite actually keeps separate time-lines and key frame per object. To view and edit this you need to expand the time-line area upward by left clicking the bar in the time-line window and dragging. Now you can left click and drag to move the key-frames of any objects key-frames. You can also delete them via the delete key after selecting them.

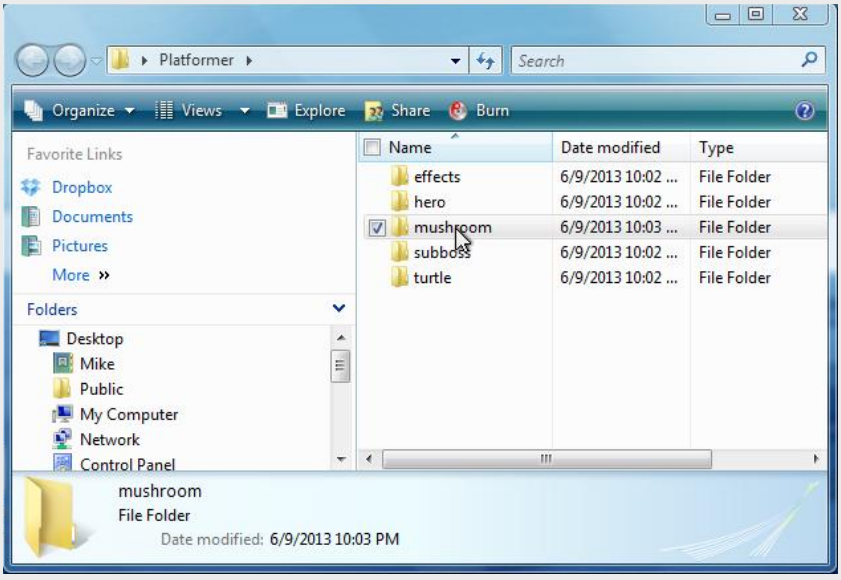
Step 14) (Duplicating entire key-frames) You can also copy and paste an entire full frame to any other place on the time-line by choosing the position the time-line to copy or the main time-line, then pressing Cntrl+Shift+C, then by going to the target location on the main time-line and pressing Cntrl+V. This even works if you are copying from a spot in the time-line that is not currently visible. This is often a fantastic way to start a new animation for a character...by finding an point in an animation you'd already created that comes somewhat close to the starting pose for the new animation. Just Control+Shift+C to copy the initial pose from an already existing animation, and then create your new animation, make sure you are at time 0 (the very beginning) on the main time-line of the new animation, and then press Cntrl+V.

Step 15) (Adding additional sprites to a finished animation) What if you've already created an entire animation, with lots of key-frames, but then decide you should add something to the character...like sunglasses for example? Sprite makes this easy. If you've animated your character with bones then Sprite makes the solution simple. Simply go the the very first key-frame at time 0, add your sunglasses image to the frame and perfect its position, scale, rotation etc to fit perfectly on the characters face. Then assign that new sunglasses sprite to the head bone by selecting the head bone, holding B and then left-clicking on the sunglasses sprite. You can test that the sunglasses are now firmly attached to the characters face by rotating the head bone, and then pressing Cntrl+Z to undo the movement. Now that the sunglasses are perfectly places and a child of the head bone, just select them, press Cntrl+C, then in the menu, choose "edit/paste to all keys", or by pressing Cntrl+Shift+V. Alternately you can do this entire action in one step by selecting the object you want copied to all key frames and just hold Cntrl+Alt. After perhaps a few seconds of processing, you can scroll through your time-line or play your animation to see that the sunglasses are now properly positioned and attached to the head on all frames.

Starting A Project

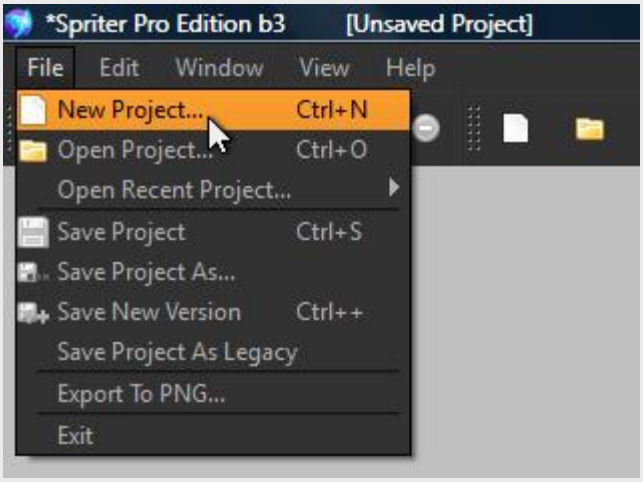
Organizing your project folders and images before you begin.

Before you start up Spriter itself, its important to understand, Spriter is not used to draw images from scratch, it's used to combine, move, rotate and stretch images you've already created in order to create fully assembled frames and animations.



Step 1) (Getting your images ready) Create a project folder which will be used for your Spriter project. Then add sub-folders in which you should organize the PNG images you'll be using to create your animations. For example: If you were creating animation sets for a platformer game, you might first create the project folder and name it "Platformer" and then inside that folder you would create other folder named "hero", "mushroom", "turtle", "effects", "sub-boss" etc and within each of those folders you'd place the images you'll be using to create and animate those respective characters or objects.

IMPORTANT! Once you begin your Spriter project, the actual Spriter file (.scml) should always be saved in the main project folder which you'd created. You can save backup files of the .scml anywhere you'd like, but if you then load them, they won't know where to find the required images because Spriter only looks for images from the same root as the .scml file itself.



Step 2) (Starting Spriter and creating your project file) Start Spriter and from the main menu choose: File/New Project or hold Cntrl+N.

You will be prompted to choose the root folder for your project. Click OK and then use the file dialogue to direct Spriter to the main project folder you had created.. In the case o four example, this would be the folder called "Platformer". Spriter's working interface will then appear and you'll be ready to begin creating your first key frame.

Setting Default Pivot Points

On the upper right of your screen (above the animation palette) you should see the file palette. Use this palette to browse through the image folder's you'd created in step one to find the images you'll be using to assemble the initial key-frame.

Before you begin using the images you might want to take your time to give each image a custom pivot point. (images default to a top-left pivot point, and its often more convenient and leads to better final results to set pivot-points based on actual aspects of the image in question...for example, the image of an upper arm would scale and rotate more naturally around a pivot point set in the center of the shoulder).

To set a default pivot point for an image, double-click on that image in the file palette and a dialogue box will appear giving you the ability to set the pivot point.

You can use the quick-set icons at the top right of the dialogue to set instantly and accurately set the default pivot point of the currently selected image to any corner, center, or the middle of any of its sides.

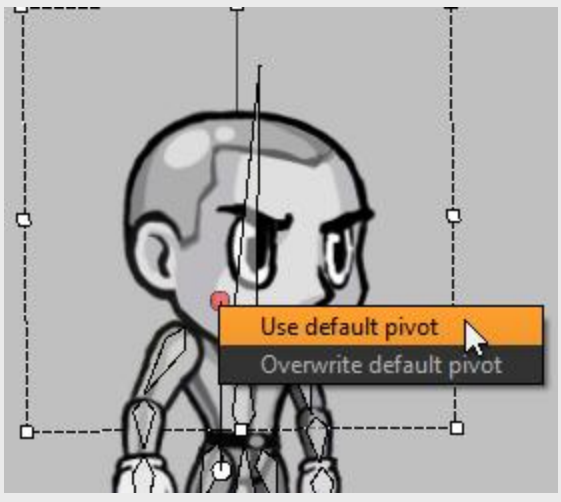
Once you've set the pivot point how you'd like, click OK.

You might end up with several images that use the same size, especially in order for image swapping purposes. In this case, don't worry, you won't have to set the same custom default pivot point ot them all manually. Just set the custom default pivot point to one of them, then click the "Copy Default Pivots" button Under the "Currently Equipped Tools" section of the Palette, then multiselect all the images you want to have the same default pivot point and click the "Paste Default Pivots" button.

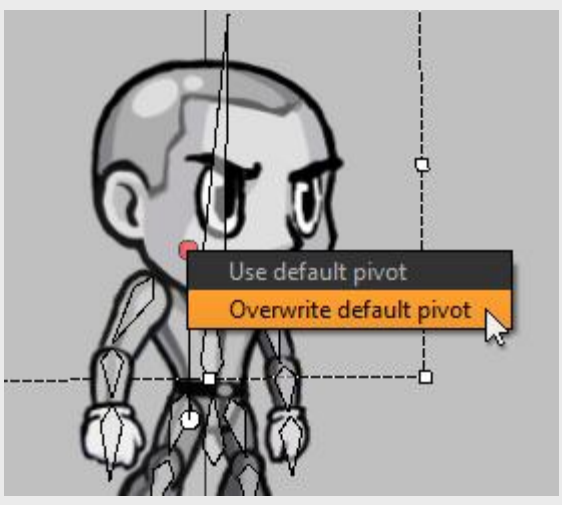
IMPORTANT: You might notice that once an image is placed on the canvas, you can left-click and drag on the round widget designating the position of the pivot point to change it's position. It's very important to know that this sets a NON-DEFAULT pivot-point, which actually tween betweens keyframes. While this feature can be useful, (for example if you need at one point in an animation for an image to rotate around a different pivot point), it can also be very confusion, and might not be well supported by certain Spriter run-times. For this reason we recommend you avoid setting pivot points in the canvas unless you not only need the special attributes of a non-default pivot point and you're sure the resulting animations will be properly supported by whichever run-time you will be using. (if any)

If you happened to set your images pivot points via the canvas accidentally, and had intended to use default pivot-points, there are two features in Spriter that can help you switch to actual default pivot points.

Example 1) Let's say you've accidentally changed your pivotpoint via dragging the round widget in the canvas, and you would rather that the image use the universal default pivot point which you had previously (or after the fact) set via the proper way mentioned at the top of this section. Just right-click on the actual round widget representing the position of the pivot point in the canvas and choose "use default pivot point" from the drop down menu which will appear. This will revert the pivot point the default pivot point, but leave the image exactly where you had placed it.

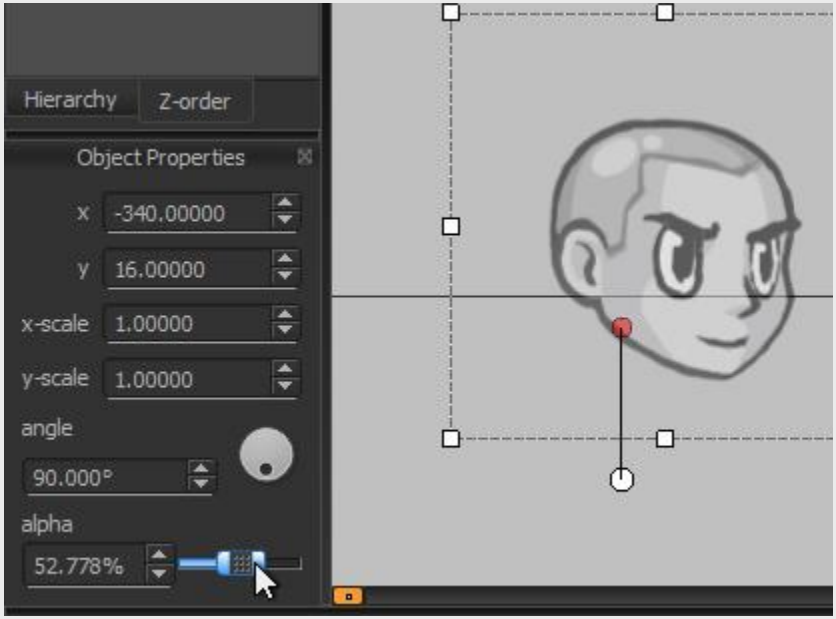


to



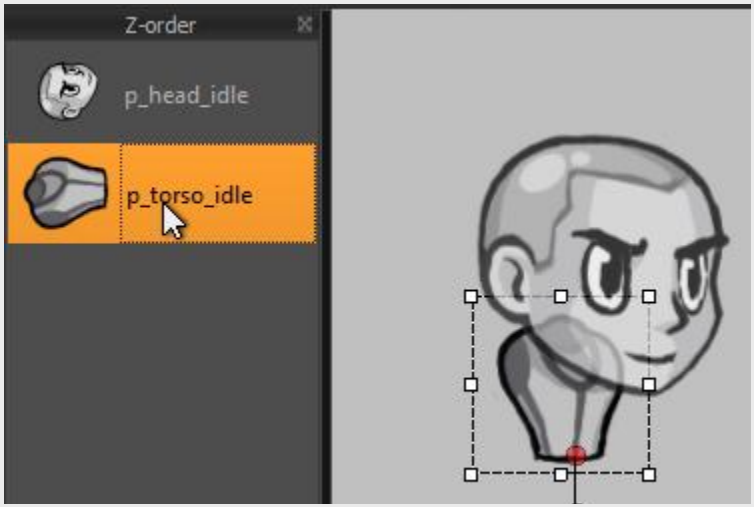
Example 2) Let's say you like the new position you've set for the pivot point via the canvas and you want to replace the current default pivot point with this new pivot point position. Just right click on the round gizmo representing the position of the current pivot point in the canvas, but this time choose "Overwrite default pivot" from the drop down menu that appears. This will make your new pivot point position the new default pivot point position, replacing whatever the old position was.

Adding Sprites To the Canvas (frame)



Now that your images are ready to use, you can simply start dragging them from the file palette onto the "canvas" in the center of your screen to begin assembling the first key frame. Once on the canvas, you can select any Sprite (the image's you've placed) by left clicking, and then use the transform controls which appear around the sprite to rotate it or stretch it as you need. You can also use the "object properties" dialogue on the lower left of your screen to keep track of or carefully edit any of the currently selected sprites attributes. This palette can also be used to adjust the currently selected sprites opacity.

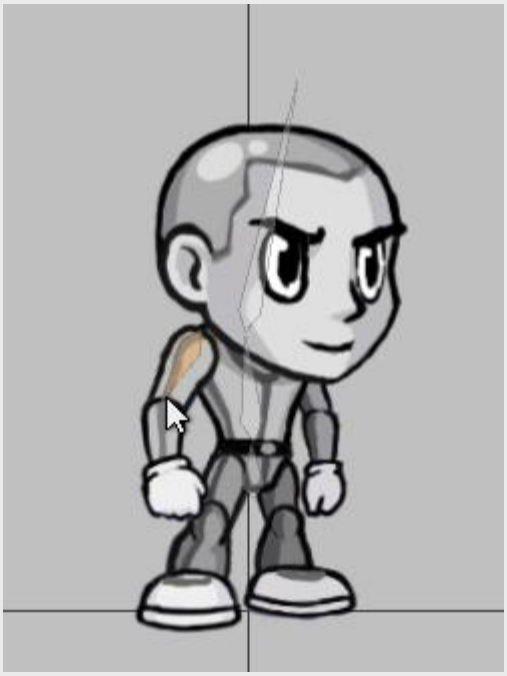
Adjusting the Z Order of Sprites



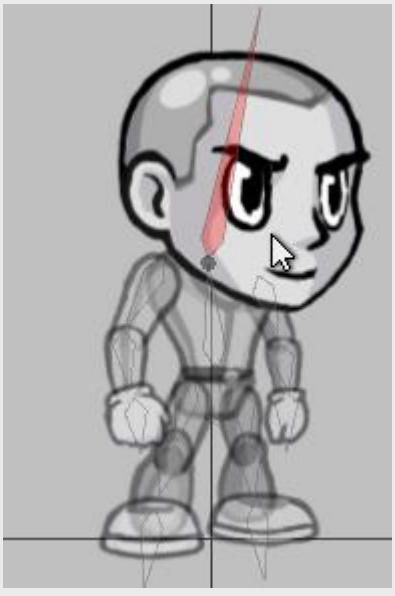
As you assemble and tweak your initial key-frame you may need to adjust the z-order of your Sprites. (the order in which they are drawn on screen..in other words, which are in front and which are behind). This can be done by clicking and dragging on the sprites in the z-order palette on the upper-left of your screen or by selecting a sprite on the canvas and then holding Cntrl and pressing the up or down arrows. You can also hold Cntrl and press the left or right arrows to send the selected sprite to the absolute top or bottom of the z-order respectively.

It's very important to remember that Spriter allows for the z-order of all images to be diferent at any point on the time-line...so, if you change the z-order on one key-frame and would like the new z-order to effect the entire animation you must be sure you're on the keyframe with the desired z-order and then choose "Edit/Copy Z-order To All Keyframes" from Spriter's menu.

Creating and Assigning Bones

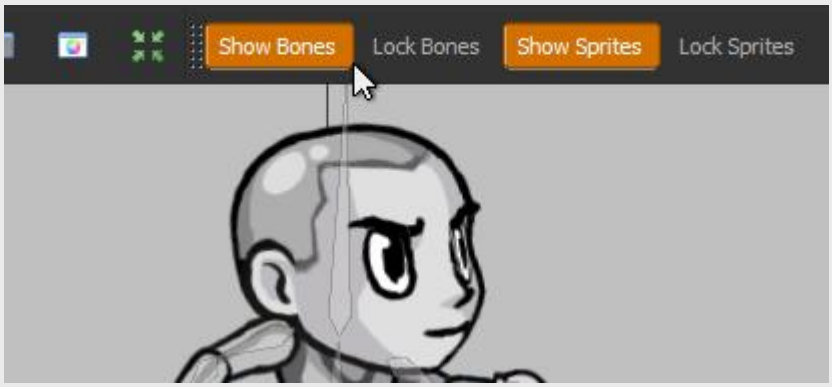


(Creating bones to more easily animate complex objects or characters) While you are not required to use bones to animate with Spriter, and in act, for many types of animations bones would just be an inconvenience, for the case of animating complex objects or characters the initial investment of literally a minute or two to "rig" the character with bones will end up making work much easier and more natural, and save you a lot of time, even resulting in a superior final animation. To create bones, simply hold the Alt key and left click and drag from the point you want the bone to begin to the point where you want the bone to end. The point where the bone begins (the thick end) acts as the pivot point of the bone. When you let go of the left mouse button that bone is done being created and is automatically selected...if you create a new bone while the previous bone is still selected, the new bone will automatically be a child of the selected bone. You can continue to hold Alt and create all of the bones you need for the full character...just be sure the bone you want to be the next bone's parent is selected while creating the new bone. At any time in this process you can let go of the Alt key and select, move, rotate and scale any of the bones to perfect their position relative to the sprites you will be assigning to them.

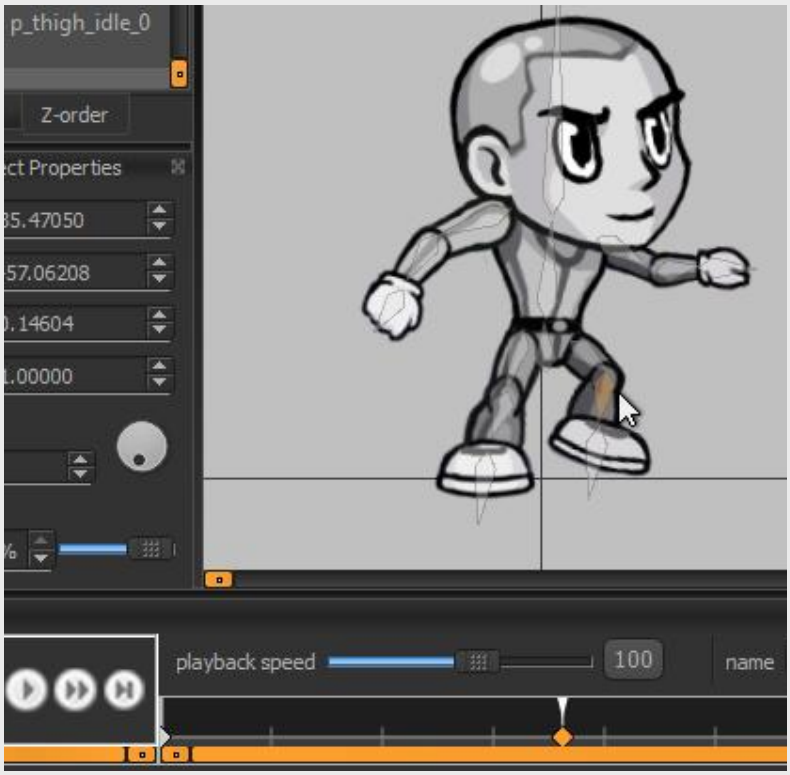


(Assigning Sprites as children of bones) Now that your bones are set up where you want for the whole character, properly aligned with the sprites, you just need assign (child) the appropriate sprites to each of them. To do this, simple select a bone by left clicking on it, and then hold the B key...you'll see all Sprites become more translucent. Now if you left click on any Sprite while still holding the B key, that Sprite will become a child of the selected bone. You will see that it is now assigned because the sprite is now more opaque. If you click that same sprite again while holding B it will disassociate that sprite from the selected bone. You can select as many sprites to each bone as you'd like. He sprite does not have to be touching the bone, or even be close in proximity to the bone. Once finished assigning all sprites to their respective bones we recommend you quickly play with putting the character in extreme poses with limbs overlapping the body and the other limbs so you can double-check that all Sprites are z-ordered properly. This way, you won't have to stop and manually fix the z-order of sprites across several key-frames once you're animating.

Animating Sprites and Bones



If you've assigned sprites to bones you'll see that if you rotate, move or scale bones, the assigned sprites will be effected with them. If all sprites are assigned to bones you may not have a need to ever select or edit a sprite directly. If this is the case, you can lock and/or even hide all sprites so that you can not accidentally select or edit a sprite by clicking on the show or lock buttons along the top right of the canvas. You can do the same for bones if you have a need at any point to only edit sprites.To start animating, first edit your starting frame by selecting and moving, rotating, stretching or changing the alpha (translucency) of any of the sprites or bones. Then click on another point along the time line and then adjust the sprite or bones as necessary to create the next key frame. Editing any sprite or bone while on a new point in the time line will automatically create a key frame. You can also create a key frame at any time by clicking on the "key all" button near the bottom right of the canvas, or by clicking on the key selected button while one or more objects (sprites or bones) are selected. **(See Key All vs Key Selected for more information)**



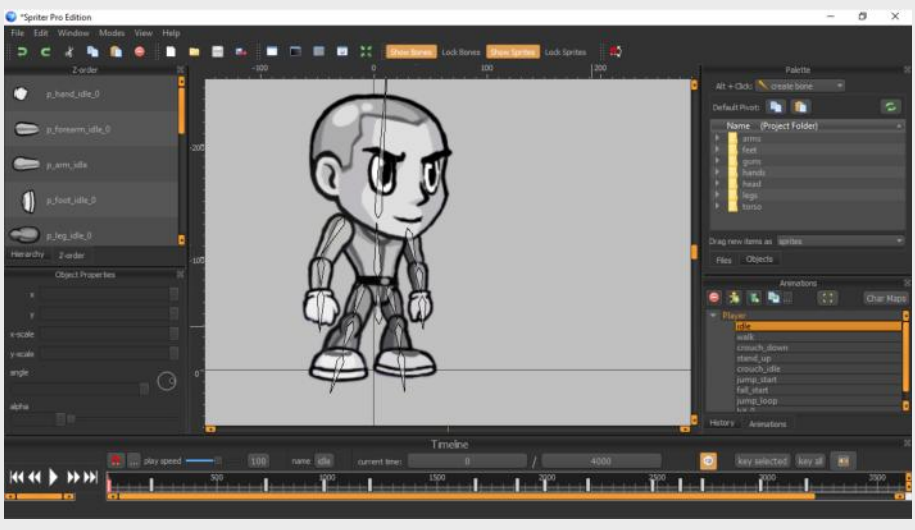
Copying Individual Attributes of an Object to All Frames

Let's say you've nearly completely an animation but then decide you want to change some aspect of the character's appearance… for example, longer torso, longer forearms, etc. This could be a very problematic and time consuming situation to deal with. Luckily Spriter Pro offers a feature to help make changing individual aspects of any given object easy.

For this example, We'll be taking the free example Spriter file called "Grey Guy" and changing the forearm images and bones in the idle animation to give him very long forearms.

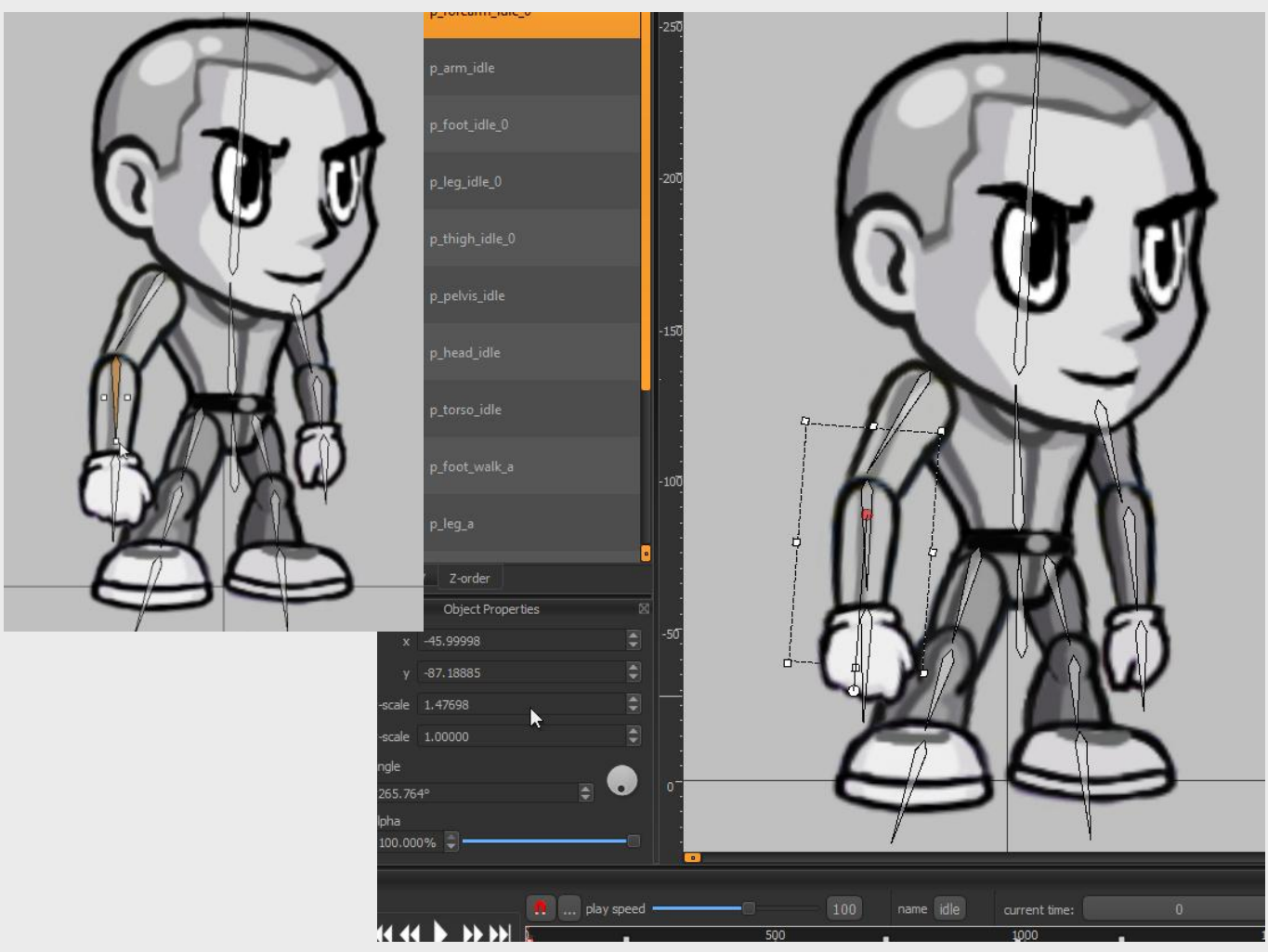
Here's how.

1) Load up the Spriter file in Spriter and click on the Idle animation so we can start editing on the fist keyframe at zero seconds in the timeline.



2) Select each forearm bone and stretch the forearm until the new length is to your liking. Now click the forearm images and remember the number in the x scale attribute in the properties palette for each of them, as you'll need this later.

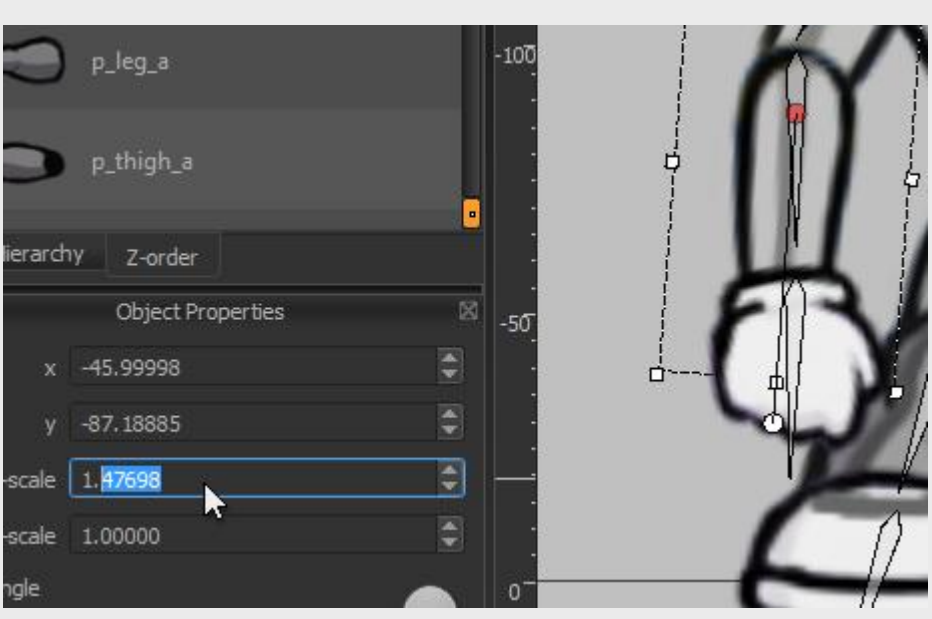
note: In this case we need to be very exact because we'll be editing the image files to match the new scale. but if this is not needed, you don't need to use the numerical settings in the objects properties palette.



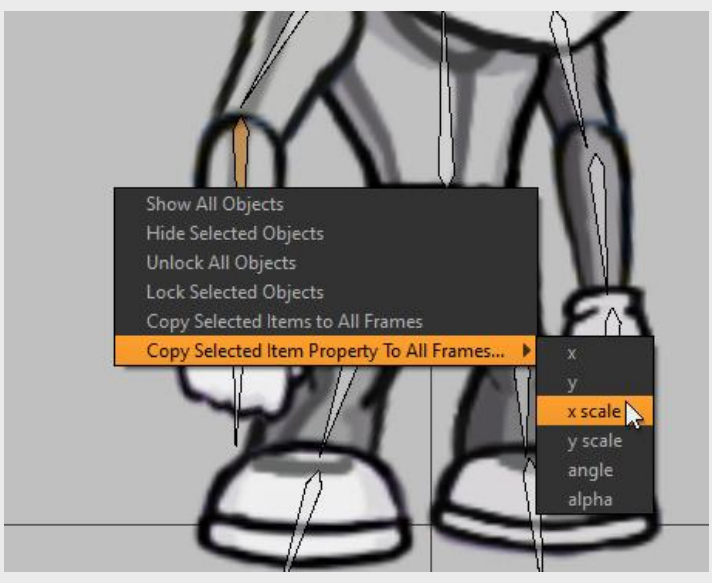
3) If you don't want the forearm images to be stretched to match the new size, and you want them to actually be the new size, then, using your graphics program of choice, find the forearm images that are being used in the arms folder of the Spriter project, and scale them to the same percentile and in the same direction that you did in Spriter in step 2. For example, if the new scale of the forearm image in Spriter is 1.47, that means its 147 percent of the original size.

4) Once both forearm images are their new scale and saved over the originals, go back into Spriter and click the reload images button near the top right of the "files palette". You'll see the new forearm images, but they will now look too long.

5) Now just select each forearm image ans switch their x-scale to 1 in the properties palette.



6) Now your changes look correct, BUT are only effecting the first keyframe at zero in the timeline. If you scrub to other key frames you'll see things are a bit messed up. Don't worry… here's comes the part were we copy the new properties to all key frames. Go back to the very beginning of the timeline (the very first key frame that has your proper changes.) Select each forearm BONE by left clicking it, then hold the right mouse button on a blank part of the canvas and select "Copy Selected Item Property to All Frames/ x scale". Then do the same thing for each forearm image.



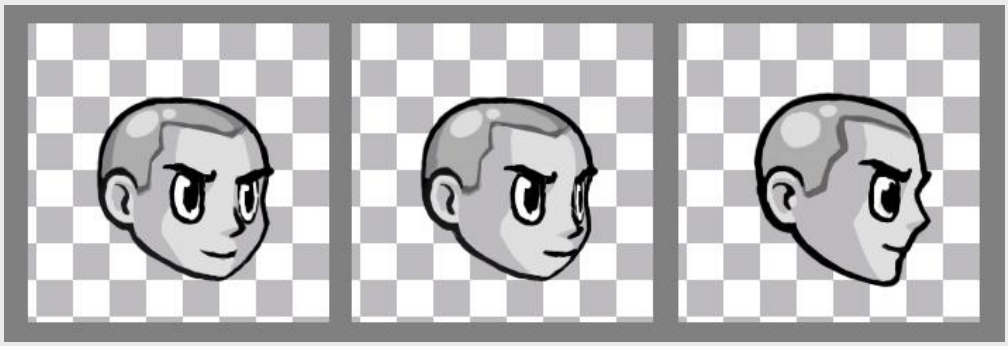
7) We're not quite done yet. Because each forearm bone also effects the child bones (hands) and therefore the hand images indirectly, we must do the same for the hand bones. Select them at zero in the timeline, one at a time, and choose the same option via holding the right mouse button on a blank part of the canvas and selecting "Copy Selected Item Property to All Frames/ x scale"

Once you finish this, if you play the animation you should see the new gibbon-like forearms are correct throughout the animation.

NOTE: For this example, we almost could have multi-selected the forearm bones, hand bones, and forearm images once the first keyframe was perfect and then right clicked on a blank part of the canvas and selected "Copy Selected Items to All Frames", because they don't change any other attributes throughout this specific animation. BUT, for animations where the forearms change angle or scale or image being used throughout the animation, we would have lost those changes in the respective keyframes.

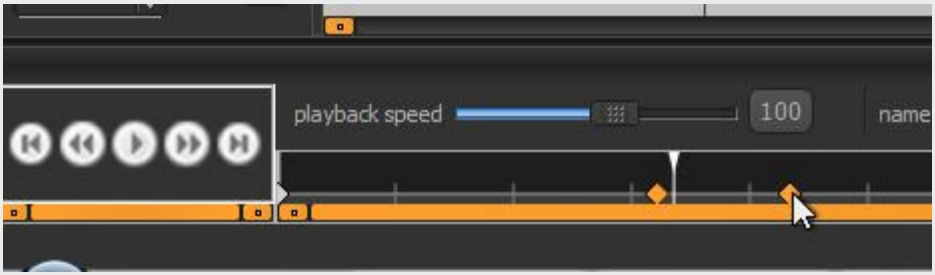
Swapping The Image of a Sprite

Step 11) (Swapping a sprites Image at any time) You might have a need in some animations to actually swap out one image with another at some point. This can be done by simply scrolling to the point in the animation you'd like an image changed, selecting the sprite by left clicking on it in the canvas and then right-clicking any image in the file palette. When you play the animation or scroll through the time-line you'll see that the sprite now changes from its original images to the new one you'd selected at that exact moment in the animation.

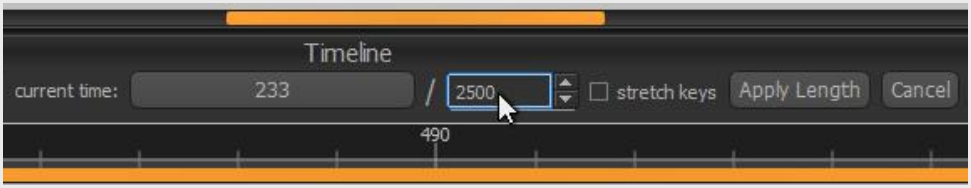


IMPORTANT: When swapping, a new image will be placed in the exact location of the current image based on their respective pivot points, so it's very important to have the default pivot-point of the new image set appropriately. One way to ensure perfect continuity when swapping from one image to the next is to actually insure that each image to be swapped has the same exact pixel-dimensions as each other, and are arranged so that all of their default pivot points would have the same exact pixel
coordinate. Look at all these head images as an example... all images are buffered with enough transparent pixels to ensure the actual head images are placed per image in perfect alignment with each other... so the exact same default pivot-point coordinates can be set for all...ensuring there will be no jarring "pop" in position as one image is switched to the next.

Editing the Timing of Key-frames or the Entire Animation

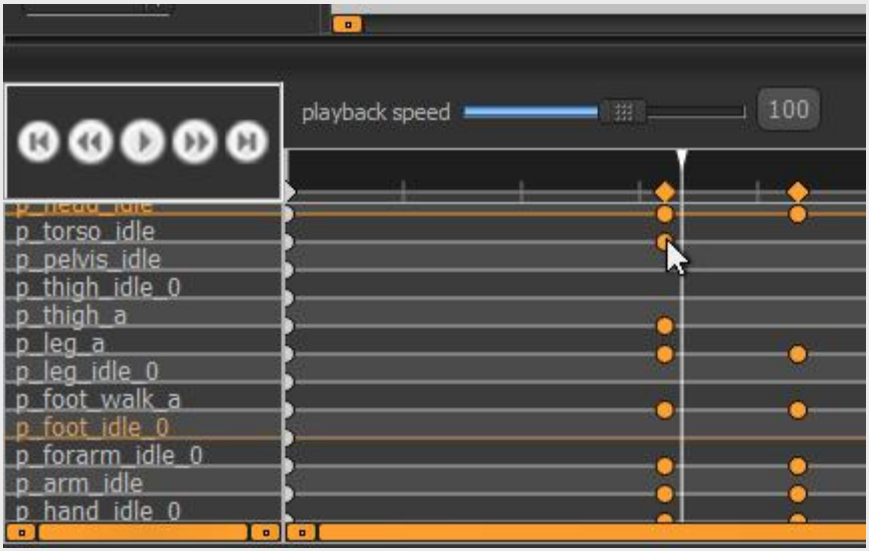


You'll likely find as your animation progresses that you'll need to tweak the relative distance between key-frames. To do so, you can simple click and drag on any key-frame in the time-line located at the bottom of your screen. You can also hold the Cntrl key in order to be able to select multiple key frames so you can move them simultaneously.



It will also often be necessary to expand or reduce the total length of the animation. To do so you can simply left click in the second number box of the "current time" section of the time line. This will allow you to not only type in a new total length for the animation, but will also present a check-box letting you either stretch the keys to maintain there relative positions to each other or not. You can also preview your animation playing back at different speeds simply by adjusting the playback speed slider at the left side of the time-line.

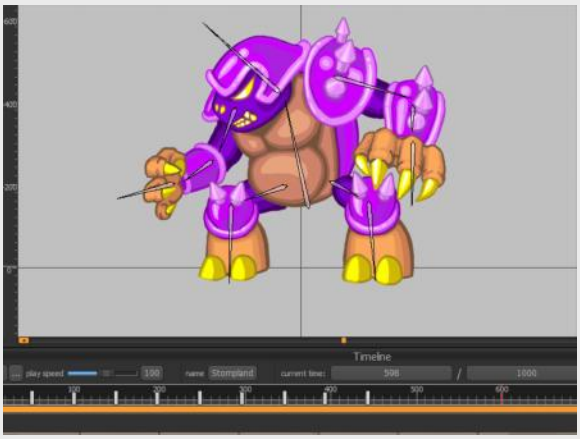
Advanced Time-line Editing



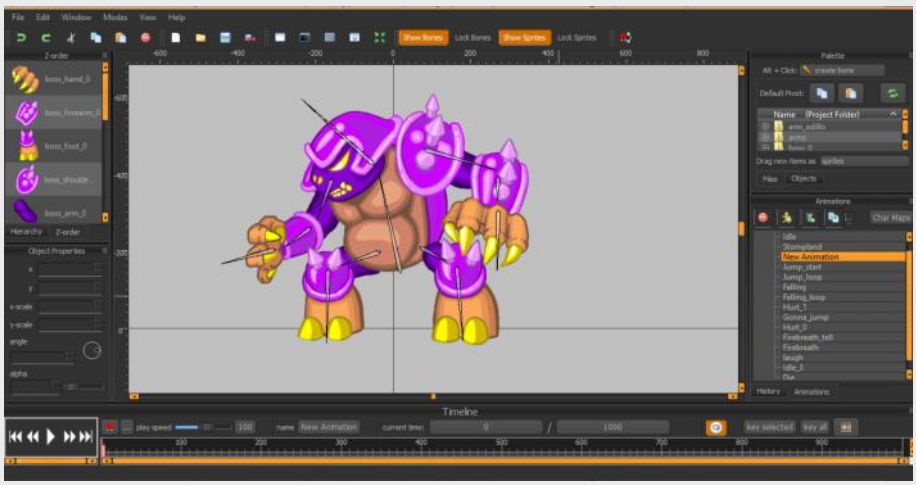
As your animation gets closer to being perfected, you might find the need to slightly offset the timing of one or more objects relative to the rest. Spriter actually keeps separate time-lines and key frame per object. To view and edit them you need to expand the time-line area upward by left clicking the top of the time-line window upward. Now you can left click and drag to change the time position of any objects key-frames. You can also delete them via the delete key after selecting them.

Duplicating Entire Key-frames

You can also copy and paste an entire full frame to any other place on the time-line by choosing the position the time-line to copy on the main time-line, then pressing Cntrl+Shift+C, then by going to the target location on the main time-line and pressing Cntrl+V.

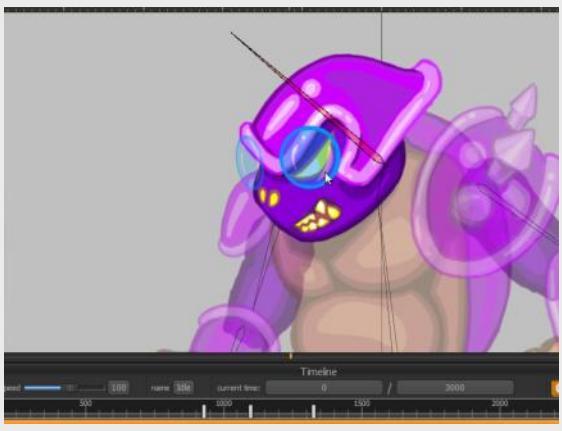


This even works if you are copying from a spot in the time-line that is not key-framed. This is often a fantastic way to start a new animation for a character...by finding an point in an animation you'd already created that comes somewhat close to the starting pose for the new animation. Just Cntrl+Shift+C to copy the initial pose from an already existing animation, and then create your new animation, make sure you are at time 0 (the very beginning) on the main time-line of the new animation, and then press Cntrl+V.

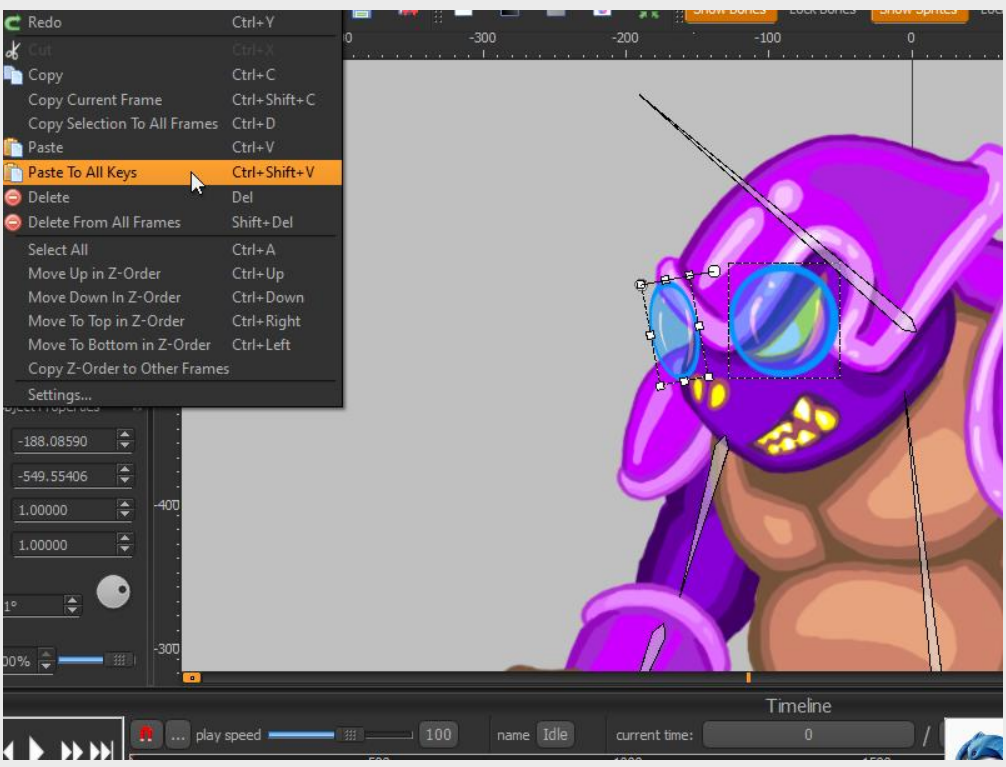


Adding Additional Sprites to a Finished Animation

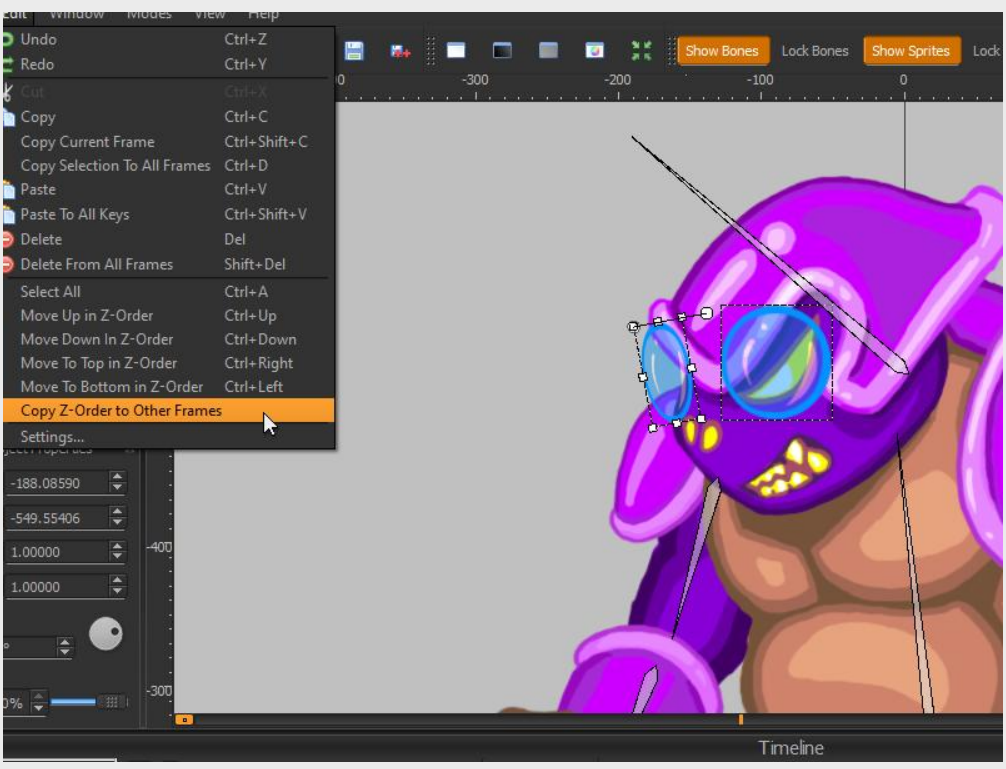
What if you've already created an entire animation, with lots of key-frames, but then decide you should add something to the character...like sunglasses for example? If you've animated your character with bones then Spriter makes the solution simple.



Simply go the the very first key-frame at time 0, add your sunglasses image to the frame and perfect its position, scale, rotation etc to fit perfectly on the characters face. Then assign that new sunglasses sprite to the head bone by selecting the head bone, holding B and then left-clicking on the sunglasses sprite. You can test that the sunglasses are now firmly attached to the characters face by rotating the head bone, and then pressing Cntrl+Z to undo the movement.



Now that the sunglasses are perfectly placed and a child of the head bone, just select them, press Cntrl+C, then in the menu, choose "edit/paste to all keys", or by pressing Cntrl+Shift+V. After perhaps a few seconds of processing, you can scroll through your time-line or play your animation to see that the sunglasses are now properly positioned and attached to the head on all frames.



If the process causes issues with the z-order of some frames, just find or edit a ket-frame to have proper z-order and then choose: "Edit/Copy Z-Order to Other Frames" from Spriter's menu and that will copy the proper z-order to the rest of the animation.

Key All Versus Key Selected

There are many ways to create a key-frame for a single object, all selected objects, or everything on the canvas (in the currently location on the time-line) with Spriter.

By default, making any kind of change to a sprite or bone will create a key-frame for that object wherever you happen to be in the time-line.

There can be time's however, where you would like to "key" one or more objects at a point in the time-line without altering it. This can be useful to "protect" the position, angle width, height and opacity of an object at that specific point in the time-line, so that when you alter a later point in the time-line, the objects you've keyed stay the way you wanted them to in the previous spot in the time-line.

There are other times where you'd like to do the same for the entire frame. (everything in the canvas at that point in the time-line).

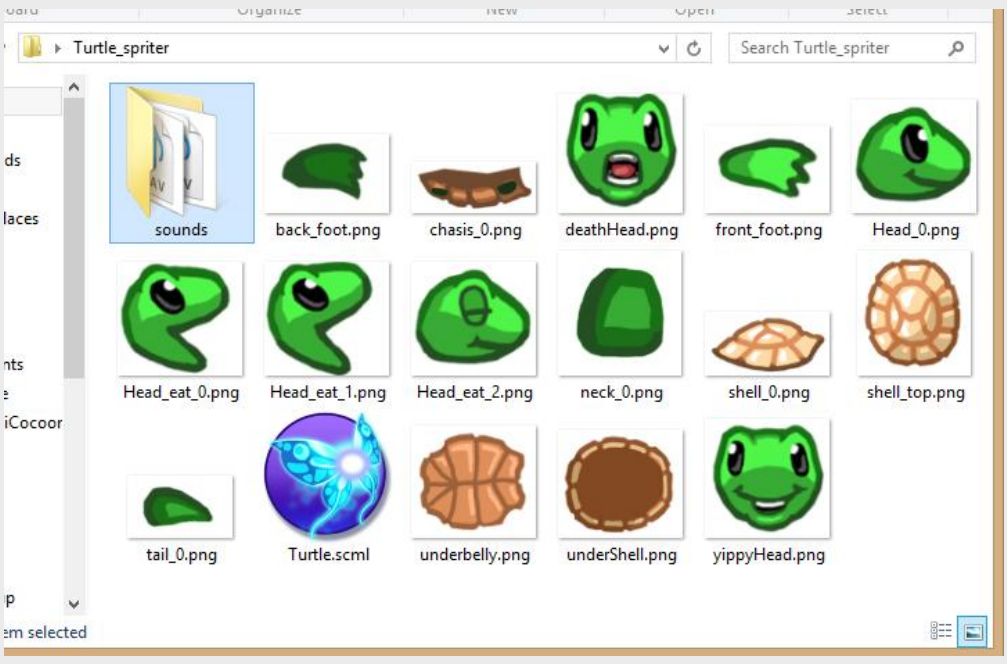
For these sorts of situations there are two useful buttons toward the top-right of the time-line window.

The "Key selected" button will create a key for any objects you currently have selected in the canvas at the current point in the time-line.

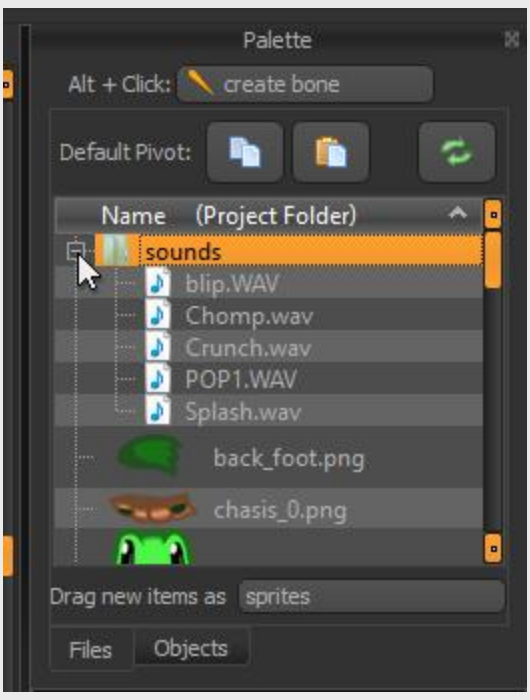
The "Key all" button will create a key for all objects that exist in the canvas at that point in the time-line whether or not they are selected.

Adding Sounds to Your Animation

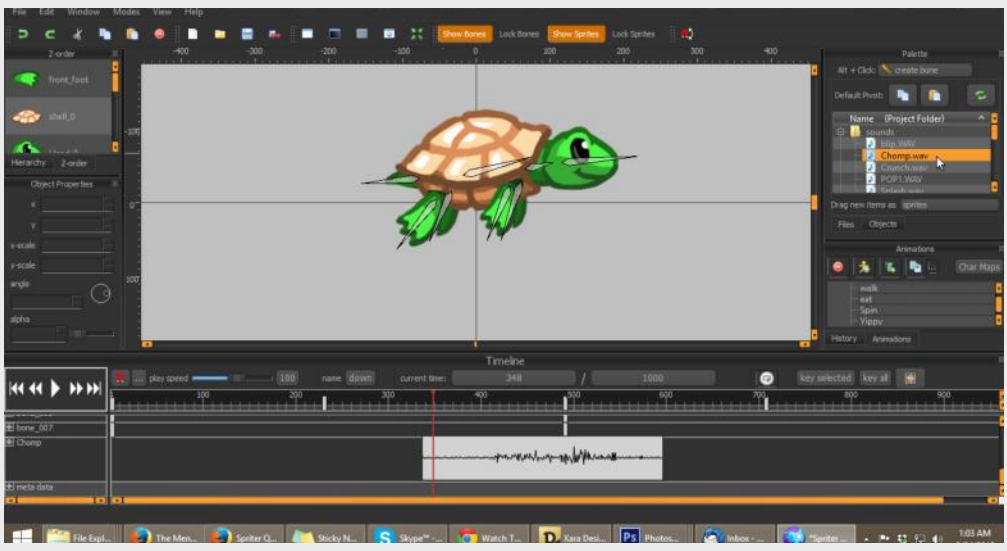
Once your animation is looking good and the timing is to your liking you might want to add sound effects to further bring it to life. Spriter allows you to trigger the playback of as many WAV files as you'd like at any point along the timeline.



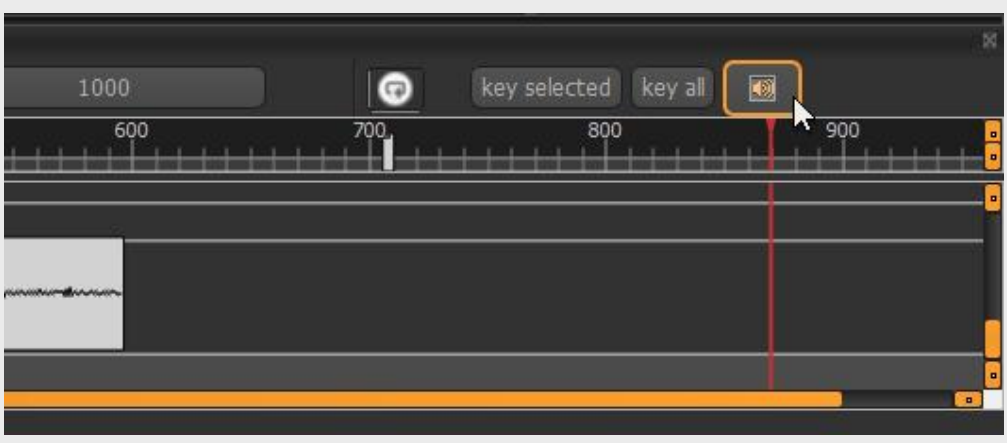
The first thing to do is gather or create your sound effects. (make sure they are WAV format) and put them in your Spriter project folder. We suggest you create a subfolder for them so they are easy to find.



When you load up your scml or scon file in Spriter, you should see the foulder containing your sound effects in the "files palette" on the right hand side of your screen. If you left click on the little + Icon to the left of the folder it will open up so you can see all of your sound files. You can left click on any of the sound files in the list to preview (hear) the sound.



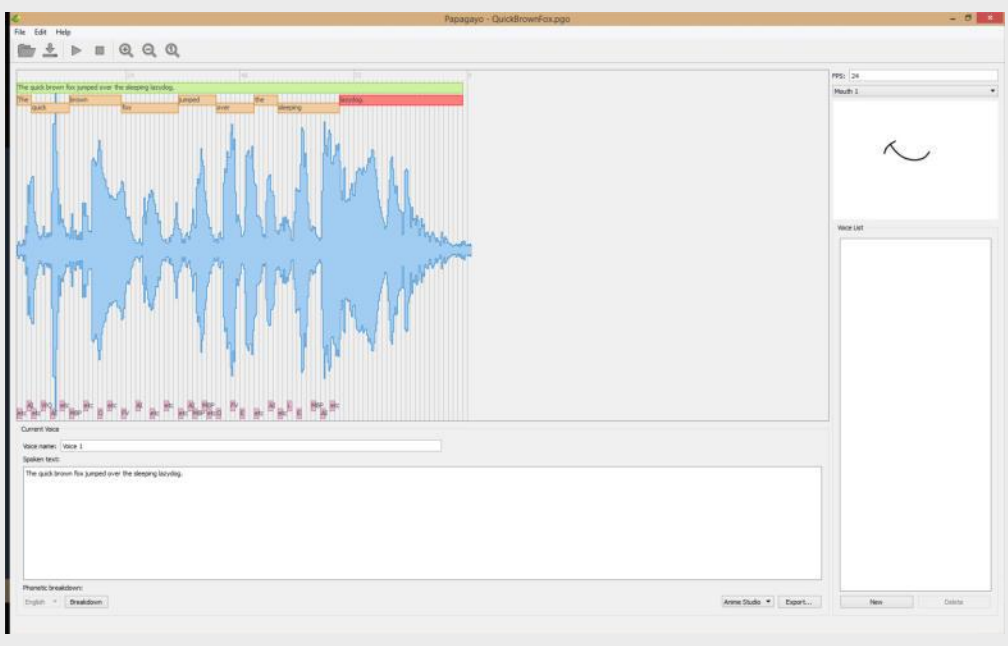
Once you've picked the sound effect you need, just scroll through to anywhere you'd like in the animations timeline to puck the spot at which the sound effect should appear and right click on the sound files name in the files palette. This will place the sound effect in the timeline at that spot. Be sure to expand your timeline view by dragging the top of the timeling bar upward, and scroll down near the bottom to see your added sound represented as the graphic of the sound file.



Notice the small speaker icon near the top right of the timeline palette. Left clicking this icon will toggle between 3 diferent sound playback/preview options. The two obvious ones are mute and standard playback, the third option lets yets you hear the sound in real-time as you scrub through the time-line. This option is ideal for perfectly synchronizing visual aspects of the animation with key parts of the sound effect.

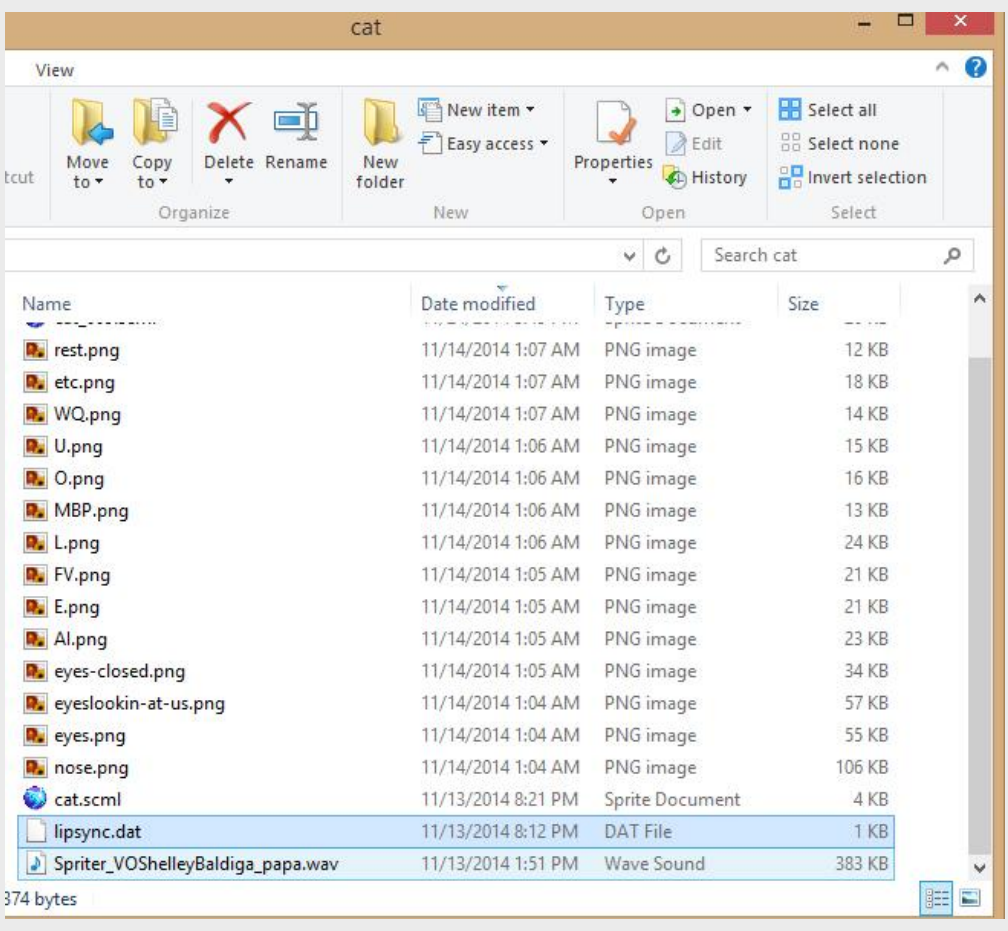
Papagayo™ Lip Sync Support in Spriter Pro

Papagayo is a great (and free) program made by Mike Clifton and is distributed by LostMarble.com.Papagayo is designed to help you line up phonemes (mouth shapes) with the actual recorded sound of actors speaking. Papagayo makes it easy to lip sync animated characters by making the process very simple - just type in the words being spoken (or copy/paste them from the animation's script), then drag the words on top of the sound's waveform until they line up with the proper sounds.

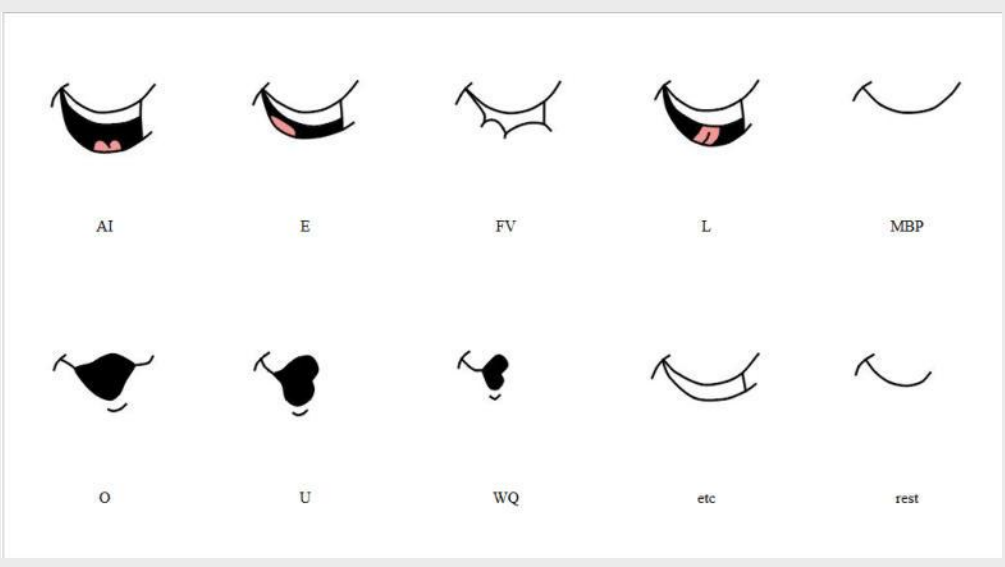


Papagayo can be downloaded from http://www.lostmarble.com/papagayo/ for either Windows or Mac.To add lip syncing to your Spriter animation you must follow the following steps after installing Papagayo:

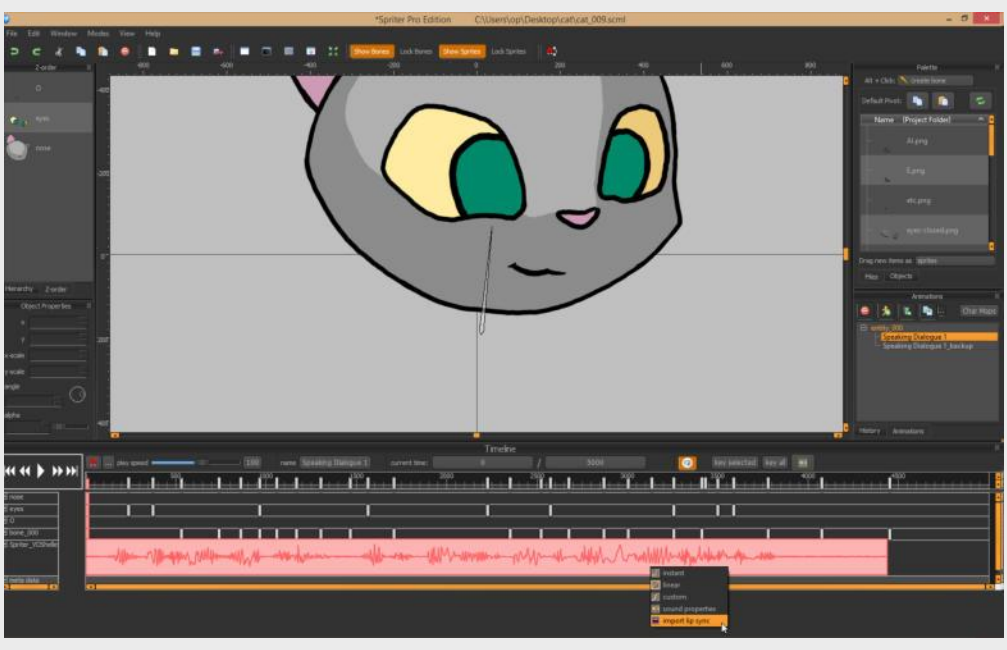
1) Start Papagayo and load up your WAV file containing the dialogue into Papagayo by dropping the file into the designated spot in Papagayo's window. Then in the "spoken text" window, type all of the words spoken in the sound file in their propper order. once you do so you'll see orange blocks representing all the individual words spread out across the graphic representation of the sound file. Now left click and drag on the beginning block or the ending block of each word to properly make each word start and end properly according to the actual sounds.



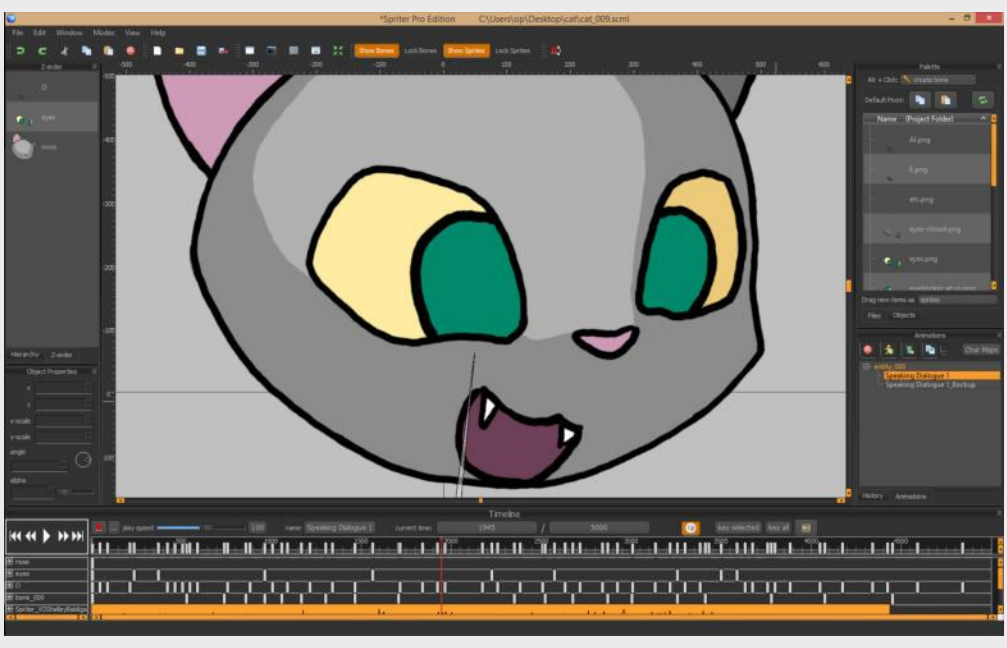
2) Once all the words are properly aligned, export the file (DAT) from Papagayo making sure to have the export mode set to Anime Studio. (it is by default). Then make sure the sound file. (WAV) and the DAT file from Papagayo are in your Spriter projects folder, either directly or in a sub folder.



3) In order for lip syncing to work, you need to have created a series of swappable mouth images, each carefully designed to represent the pronunciation of specific sounds that people make while speaking. Each file must be carefully named exactly like the reference chart above shows. Create a folder in your Spriter project folder for each of the angles the mouth will be seen from...for example a folder called "mouth_sideview" and one called "mouth_frontview" and put all the appropriate mouth images into their folders. The animation you'll be applying lip sync to should already have one of these mouth images present throughout the entire timeline, the "rest" image for example. If your head changes angles within the animation, you can image swap the mouth image at that point in the timeline from the mouth image "rest" in one folder to the "rest" image in the folder for the different angle...such as changing from front view to side view. (you can find other mouth shape references here: http://www.brashmonkeygames.com/spriter/Papagayo/PapagayoMouthShapes.html)



4) In Spriter, once the animation is otherwise finished, add the sound file at the proper point in the animations time line. **IMPORTANT: Because the next step in the process will permanently add a very large amount of key frames to the animation, which would make editing the animation after the fact very difficult, we highly recommend you first create a clone of the animation as a back-up BEFORE you proceed to step 4!**

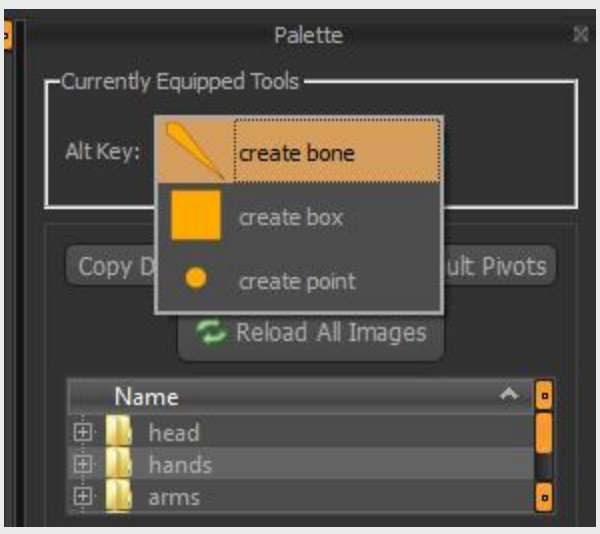


5) Assuming you've made a backup copy of your otherwise finished animation as described above, now just find and right click on the graphical representation of the sound file in the timeline, and choose "import lip sync". Then a small dialogue box will appear asking you to choose the lip Sinc File (This is the DAT file exported from Papagayo) and the "Mouth Timeline" (this is asking for the name of the object timeline, aka the sprite which Spriter will be image swapping to represent all the mouth shapes for the lip syncing. Once these options are set properly, click OK and the lyp syncing process should add all the necessary keyframes to the animation in order to change the mouth image at the appropriate times to represent the sound being made at that moment in time. Remember, if you need to change the animation after importing the lip sync, you should once again copy the back up of the animation that was created BEFORE the import of lip sync, edit the copy, and then re-import the lip sync. This process will ensure you always have a clean, lip sync free version of the animation to change and edit as needed

Adding Collision Rectangles to Frames

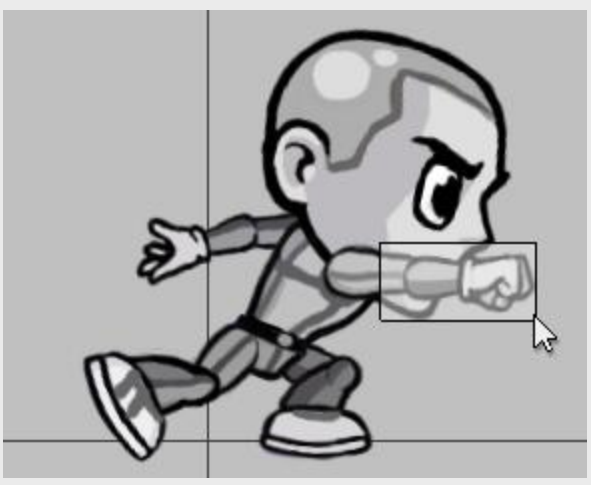
By default when you hold the Alt key and left click and drag in the canvas, you'll create a bone. There are actually two other types of objects you can create via this method which aren't used for animation, but instead are for game-play related information. One of these two objects is collision rectangles. Collision rectangles can be used by a game engine to designate the actual areas the game will detect collisions with. For example, during a punch animation, you can create a collision rectangle to designate the area of the animation (near the fist) which the game would use to see if the punch makes contact with enemies in the game. You could also use collision rectangles to designate regions of the body which would take different amounts of damage from attacks, or would trigger different reactions if collided with.

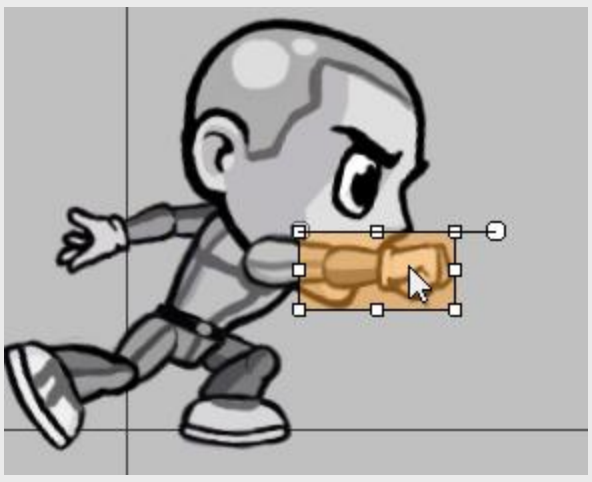
To create Collision rectangles, do the following:



1) Left click on the Currently Equipped Tool type at the top of the Palette located near the top right of your screen, and select "create box"

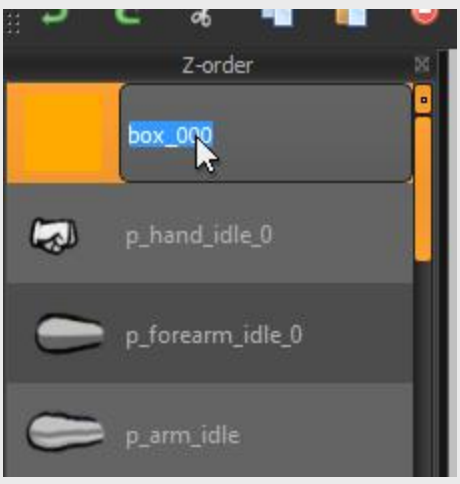
2) Make sure you're at the point in the timeline (or keyframe) you want the collision rectangle to exist, then hold the Alt key and left click and drag on the canvas to create your collision rectangle. You can then copy and paste it to other keyframes, or use the paste to all keyframes option to propagate the Collision rectangles across the entire animation.





3) Once in place, if left click, transform gizmo's appear so the rectangle can be stretched, rotated, and have its pivot point changed just like images(sprites) can. These differences will be tweened between keyframes which include the same collision rectangle!

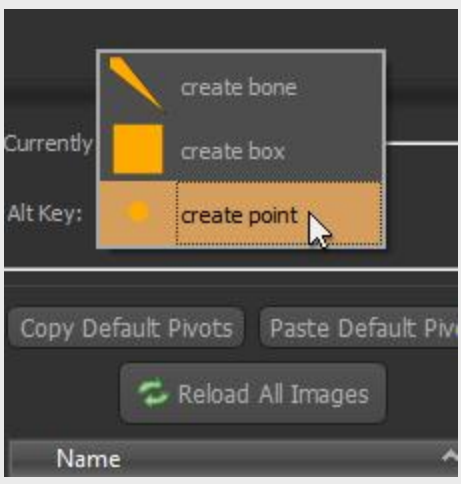
4) By double-clicking on the name of the rectangle in the Z-order palette, you can give it any name you'd like for easy recognition of the rectangles purpose.



Adding Action Points to Frames

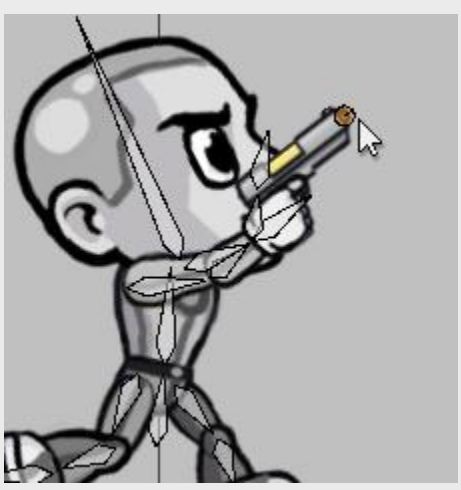
Action Points can be placed anywhere in your animation and can be used by game engines to determined from which coordinate objects will be spawned (for example where should a bullet come from), and at which angle. Just as with Collision Rectangles, you can place as many individual Action Points per key-frame as you'd like, and give them all distinct names to easily keep track of them within Spriter and within your game engine.

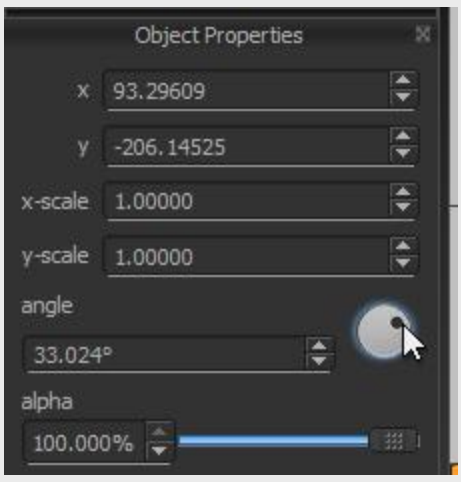
To create Action Points, do the following.



1) Left click on the Currently Equipped Tool type at the top of the Palette located near the top right of your screen, and select "create point"

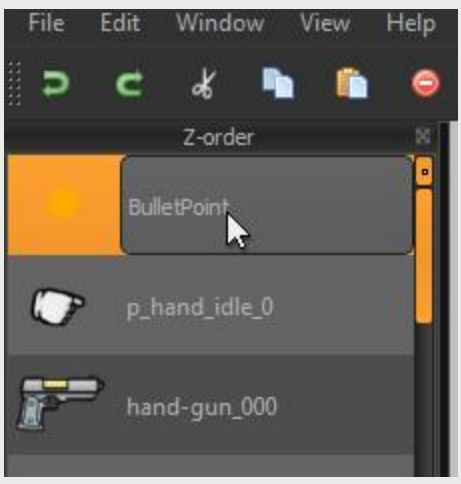
2) Make sure you're at the position in the timeline (or keyframe) you want the Action Point to exist, then hold the Alt key and left click and drag on the canvas to create your Action point and simultaniously set its angle. Just a quick click with no dragging would create the point with it's angle set to zero (aiming perfectly rightward). You can then copy and paste it to other keyframes, or use the paste to all keyframes option to propagate the Action Point across the entire animation if you so desire.





3) Once in place, you can left click and drag it to change its position, or use the Object Properties dialogue to tweak its angle.

4) By double-clicking on the name of the rectangle in the Z-order palette, you can give it any name you'd like for easy recognition of the rectangles purpose.



What Are Character Maps

Character Maps represent one of the great benefits of using Spriter's modular animation method. Character Maps allow you to quickly and easily create variations of a character (or object), or entirely new characters (or objects) by taking the animations you've already created and swapping out some or all of images with new ones.

Imagine you're making a game where the hero character can acquire new weapons, armor etc. With Spriter and Character Maps, you can animate your character once, and simply create and combine Character Maps to instantly create and preview any combination of the alternate attire and equipment. These visual variations on your character can of course be exported out as sequential images for use in game engines without direct support for Spriter animation data, however the real benefits are realized when you use the actual Spriter animation data and Character Maps within your game engine, giving you silky smooth tweened animation for a potentially massive collection of characters and variations of characters using a tiny fraction of the time, file-space, and ram that non-modular animation methods would require.

You can not only swap images with other images, you can also designate images to be hidden (or not drawn on screen) in any given Character Map. Picture a game character which starts out with no cape, but can later acquire one.... You'd animate your character with the cape, then create the starting Character Map to hide all the cape images, then use a new character map,(or in this case, just turn off the no-cape Character Map) to reveal the cape. The possibilities are endless...Sunglasses, hats, helmets, knee-pads, wings, scorpion tails, you name it!

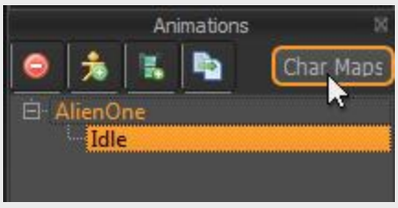
Before you Begin making Character Maps:

There are several important things to keep in mind while animating and creating your initial character which you'll want to use with Character Maps:
1) Organizing the part images for your initial character into separate folders based on groups of images you'll want to replace in Character Maps will save you lots of time. For example, having a single folder specifically for each character variations head images will make it much easier to find and designate the corresponding images. In fact, Spriter can actually automate the association of images with replacement images if you stay organized. More on this is a moment.
2) Things will be quicker and easier still of you give all alternate images the same exact name and image size as the original images use to create your animations (just in a new folder). For example, notice how in this simple demo project, there are two image folders, one called “red” which contains the handful of images used to create the animated character, and the folder called “blue” which contain the corresponding images required to change the default red character into the very different looking blue character. Notice how the corresponding images in each folder have the exact same name and images size.

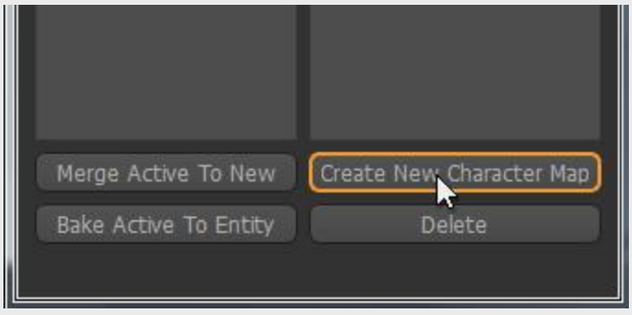
IMPORTANT: These are suggestions which can keep your project well organized and save you time if you plan on creating many character variations which swap many images with alternate ones, but these are NOT requirements! If you only plan of replacing a few images with Character Maps, or simply don't want to give your folder and images structures this much forethought, it is by no means a necessity.

Creating Character Maps

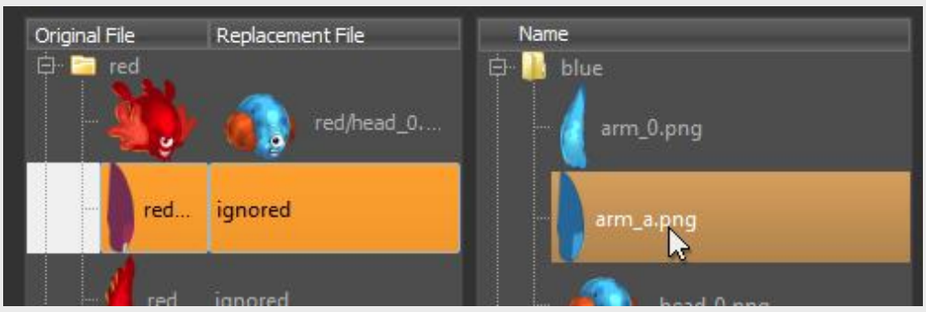
1) Create The Character Map. Click the "CharMaps" button at the top-right of the Animations palette. This will bring up the Character Maps Palette, which we'll come back to shortly.



Now click the "Create New Character Map" button toward the bottom right of the Character Maps Palette. This will bring up the actual dialogue for creating Character Maps.

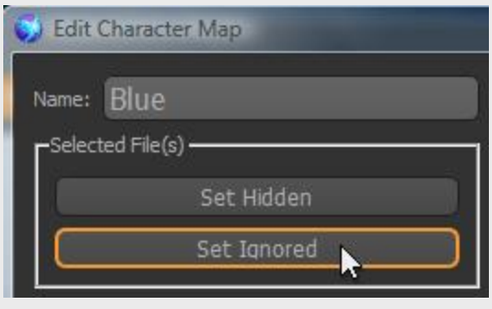


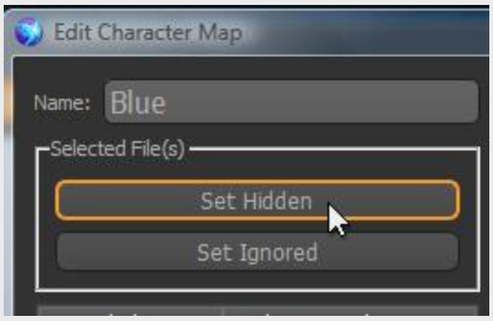
2) Assigning Images to be replaced by a Character Map. Now use the left column of the "Edit Character Map" Palette to navigate to the images you want to effect with this Character Map. Left click on an image you want to effect, and then use the right column of the "Edit Character Map" palette to find the image you would like to replace the currently selected image with. When you've found the replacement image, right click it to assign it. It will then appear in the left column by the side of the image it will be replacing. Repeat this process for all images you want replaced by this Character Map.



In this example shown in these images, because there are two folders with sets of images with exactly the same names, you can actually assign entire folders at a time in your Character Map. To do so, simply left click the entire folder in the left column of the "Edit Character Map" Palette, and then right-clicking on the folder containing the replacement images in the right column of the "Edit Character Map" Palette.

3) Setting Images to be ignored (not changed) by a Character Map. If you accidentally assign a new image to an image you do not want to be replaced, simply select that image once more in the left column and then click the "set ignored" button toward the top left of the "Edit Character Map" palette.

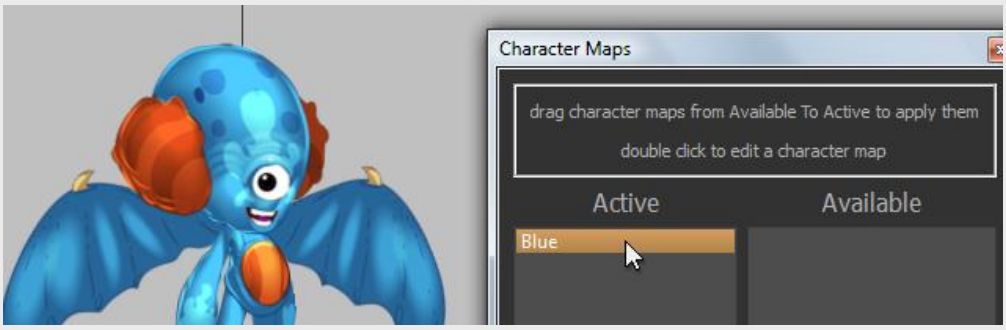




4) Setting Images to be hidden by a Character Map instead of replaced. If there are images you'd like to be hidden instead of replaced by a Character map, simply select the image in the left column of the "Edit Character Map" palette and then click the "Set Hidden" button toward the top-left of the "Edit Character Map" palette.

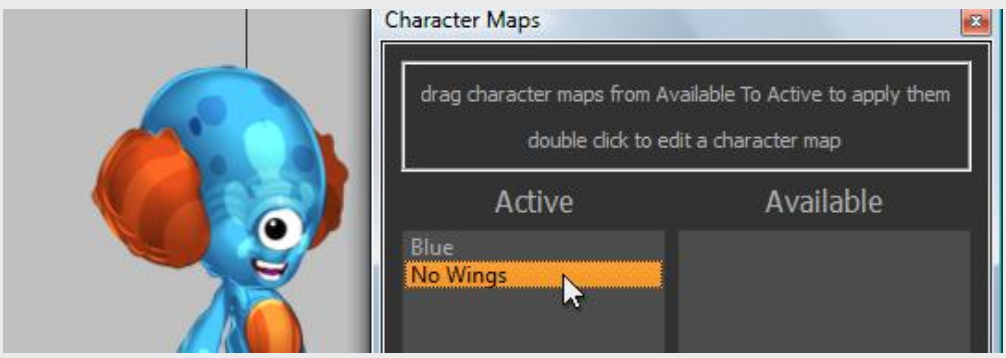
NOTE: All of these options can be applied to multiple images at once. You can hold the Cntrl key while left clicking images to multi-select them.

Activating and Stacking Character Maps



5) Setting your finished Character map to active. Now that you've created your Character Map, click the OK button at the bottom right of the "Edit Character Map" palette. You should now see the name of your new Character map in the right column (available) of the "Character Maps" palette. To see your Character Map in action, simply left click and drag the name of your Character map from the right column to the left column (active). You should see the character map have immediate effect in the Canvas if you have an animation selected which uses images which you've "effected" with your Character Map.

Note: This new look for your character will even be reflected if you export animations as PNG images while Character Maps are active.



6) Stacking Character Maps for advanced uses. You can create as many Character Maps as you'd like, and make each one handle specific changes to your object or character. For example, you could have one set of Character Maps called "blue pants", "red pants", "Bermuda shorts" ect which all swap out the original pants images with alternate images to change your character's pants. You can then have a similar set of Character Maps to handle changing the characters shirts. Finally you can have other Character maps that reveal or hide the character's sunglasses, baseball cap etc.

Once all these character maps are finished you can "stack them", meaning make more then one of them active at a time to simultaneously change the entire wardrobe/equipment set of a character to any of countless possible combinations! Spriter plug-in's for most authoring systems will support this functionality at run-time, so you'll be able to make games which can combine Char-Maps on the fly based on player decisions and actual game situations.

See how in this example, I'm using one Character Map to change from the red character to the blue, and a second character map to hide the wings.

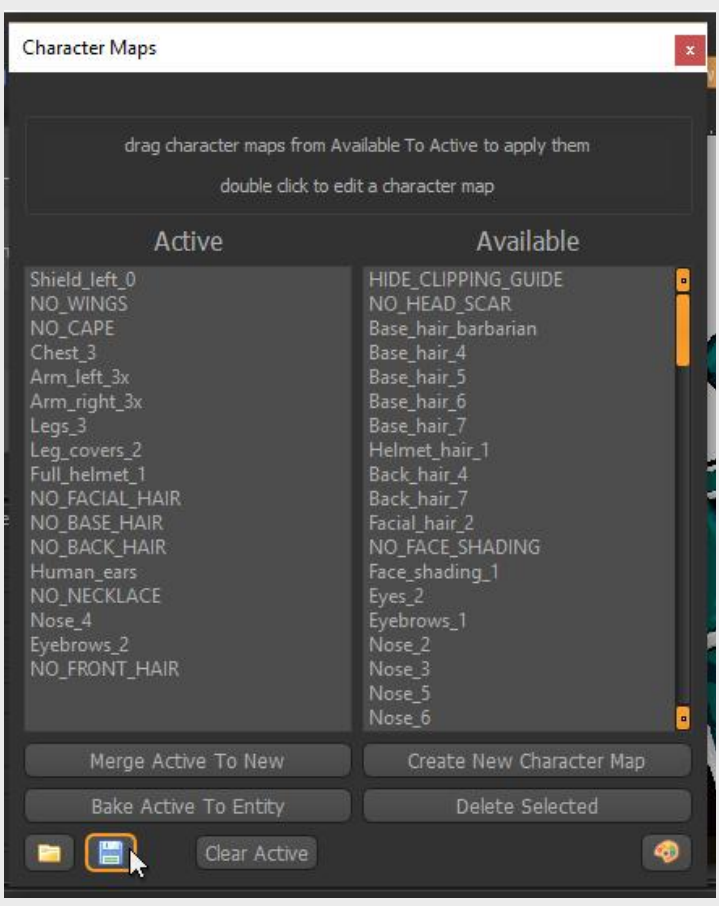
Saving Character Map Sets as Character Files

As mentioned in the previous section, you can apply multiple character maps at the same time, with, for example, one controlling the pants style, one controlling shirt, etc etc. This provides a very effective way of letting you mix and match possible combinations to create or allow for a massive amount of possible combinations for characters, props etc. Our "RPG Heroes Art Pack" uses this exact methodology to let you create custom RPG character portraits and Sprites. We'll be using it in this section as a reference.

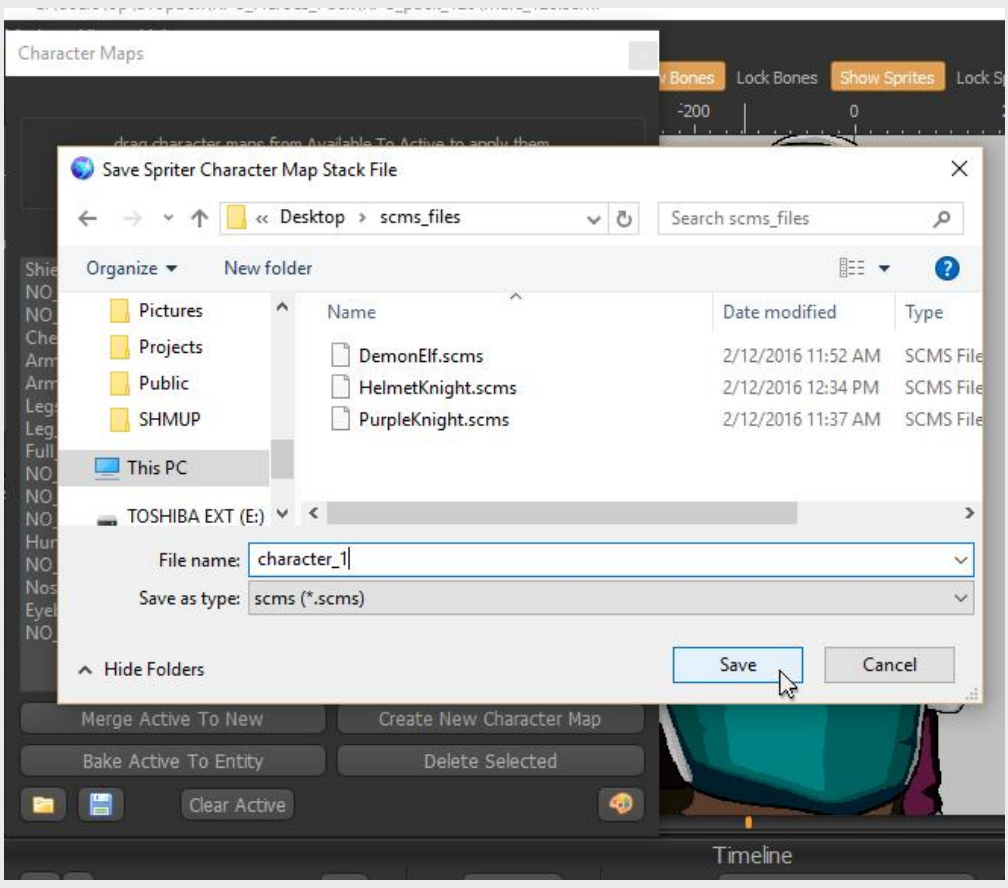
Once you've taken the time to create a custom character by activating several character maps from a large selection, you wouldn't want to have to redo this every time you load the art pack into Spriter, to export new animations, or make tweaks to the character like change the hat they are wearing etc. For this reason, Spriter Pro lets you actually save out the currently active character maps, (along with any custom color palette selection you've done, which is explained here), so you can easily reload them whenever you need to get back to that specific character set-up.

Here's how:

1) First drag any character maps you'd like to the "Active" column on the left side of the "Character Maps" palette. If you change your mind on any character map, just drag it back to the "Available" column. Once you are happy with your new custom character, Click the small, blue computer disk icon near the lower left of the "Character Maps" palette.



2) Choose a destination folder (it doesn't matter where, so long as you remember where you saved it and remember which Spriter Project it works with… we recommend making a folder specifically for character files for each specific Spriter project you work on that requires character files)



3) Choose a name for your character file and save. That's it!

You can now reload this character (set of activated character maps) whenever you'd like by first loading up the specific Spriter (scml or scon) file that it pertains to, then opening the "Character Maps" dialogue, clicking the yellow folder "load file" icon at the lower left of the "Character Maps" palette, then directing Spriter to your desired character file.

What's really cool is, even among different and separate Spriter file or projects, so long as you name all of the character maps exactly the same (and make them apply the same sorts of visual changes, for example "blue pants, red shirt etc") then you can make the separate Spriter projects compatible!

We did just that when we created the RPG Heroes Art Pack! Because of this, the user can create a customized character in the high res version, save out the character, t hen load up that same character as either a 48x48 or 32x32 pixel art version in those respective Spriter files!

TexturePacker Support

IMPORTANT: If you'll be using your actual Spriter data files (scml or scon) in your game engine, be sure whatever Spriter implementation you'll be using supports this or any other of Spriter's more advanced features before using them.

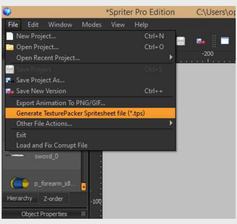
TexturePacker _by CodeAndWeb is pretty much the industry standard tool for creating optimized sprite sheets (aka "texture atlases"). You can go to (https://www.codeandweb.com/texturepacker) to learn more about its feature set and the benefits it offers.

Many Spriter users requested TexturePacker support for Spriter, so we added two different ways you can benefit from (and use) TexturePacker created sprite sheets in Spriter Pro. Many thanks to TexturePacker Developer Andreas Lowe for working with us to add Spriter specific support to Texture Packer.

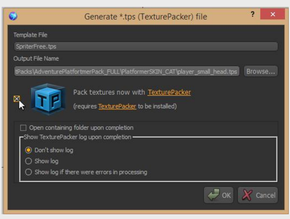
There are two very different ways you can use Spriter with TexturePacker. **Both require TexturePacker to be installed and require the full version of Texture packer to allow the most flexibility and so that none of the images in the sprite sheet will be watermarked.** You can get the free version from https://www.codeandweb.com/texturepacker/download Here are the two options:

1) Using TexturePacker to merge all of the image files that your Spriter project is using into sprite sheets once the project is finished.

In this instance, you simply use the standard Spriter work flow of creating individual image files for each sprite image (body parts etc.), organized in sub folders in your project folder, then, once finished, you choose an option to have TexturePacker automatically merge all the used images into optimized sprite sheets for you.



To do this, all you have to do (assuming you have TexturePacker installed and are otherwise finished with your Spriter project) is choose File/Generate TexturePacker Spritesheet file (*.tps)

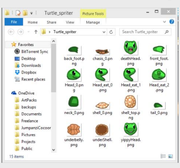


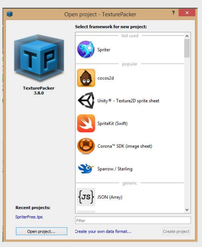
Then check the "Pack textures now with TexturePacker" check-box and adjust any of the sub-settings as desired. Spriter will then have TexturePacker generate the spritesheet images and data files and add them to your Spriter project folder.

2) Using a previously created TexturePacker sprite sheet as though its a sub-folder of separate image files.

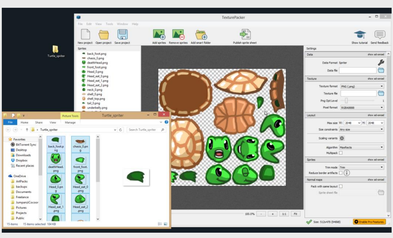
In this instance, Spriter can actually load in sprite sheets which had been previously created by TexturePacker and let you use them as though they are typical sub folders with separate images in them!

To use this option, the first thing you'll need to do is create a sprite sheet using the full version of TexturePacker. To do so, follow the following steps:



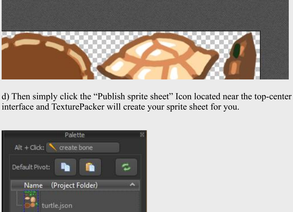


a) Have your individual image files ready and organized in folders so you'll easily be able to tell TexturePacker which images should be merged into sprite sheets. Then load up Texture Packer and choose "Spriter" as the data format type.

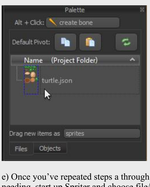


b) Now select all your image files and drag them onto the canvas in TexturePacker as prompted. This will organize the spritesheet for you. Advanced users can then play with TexturePacker's additional settings as desired or needed.

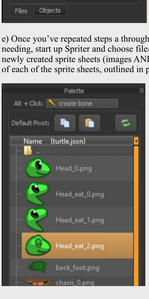
c) Now set the desired name and location of the data file which will be created by left clicking on the folder icon to the right of the text field labelled "Data file" and then use the dialogue that will appear to select the location and type the desired name. once you've done this it will automatically set the name of the image file that TexturePacker will create to match.



d) Then simply click the "Publish sprite sheet" Icon located near the top-center of TexturePacker's interface and TexturePacker will create your sprite sheet for you.



e) Once you've repeated steps a through d as much as needed to create all the sprite sheets you'll be needing, start up Spriter and choose file/new project and then select the folder which has all of your newly created sprite sheets (images AND data files). Once you've done this you should see a thumbnail of each of the sprite sheets, outlined in purple.

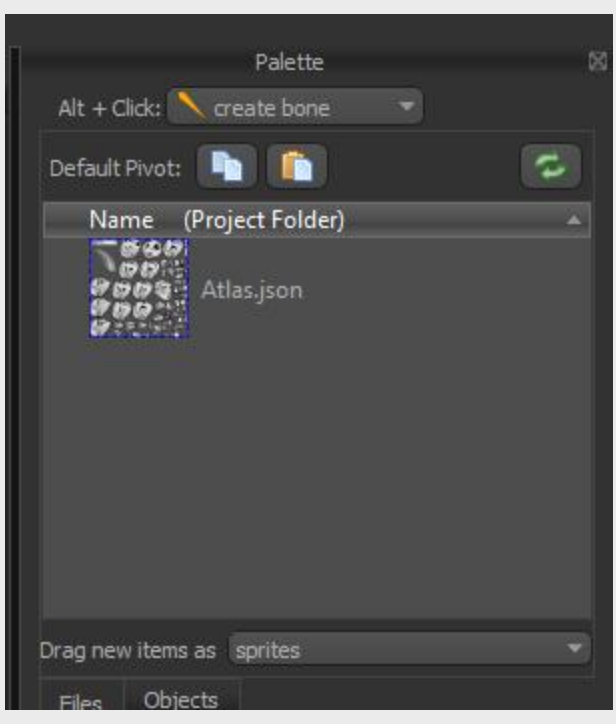


f) Now try double-clicking on any of the sprite sheet thumbnails and you should see it open up as though its a sub folder of separate images! Now you can use Spriter just as you are used to doing with separate images, treating each sprite sheet as though its a sub-folder of separate images.

AGAIN: Be very careful if you plan on using your Spriter files with any particular authoring system. Make sure that the Spriter implementation you'll be using can support these TexturePacker related features before using them.

Creating a Texture Atlas using Clone of Your Project (Pro Only)

While Spriter allows you the convenience of using separate image files and effortlessly adding new ones needed to your project as needed, an actual game engine benefits from having the source images for animations consolidated into texture atlases. A texture atlas is a single large image which contains multiple images, along with a text files (in this case .json format) which tells your game engine the position, size, and name of each individual image in the texture atlas image.

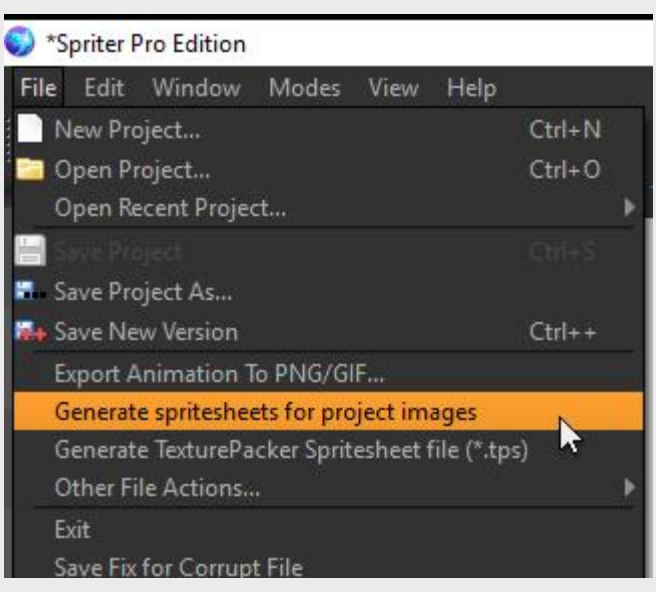


While some authoring systems like Unity and Construct 2 have automatic methods to handle the individual images and consolidate them into texture atlases, other languages or authoring systems might not have such built in functionality. Luckily, Spriter Pro has a built-in feature which will allow you to take your Spriter project you've made with separate images, and create a clone of it which will use optimized texture atlases.

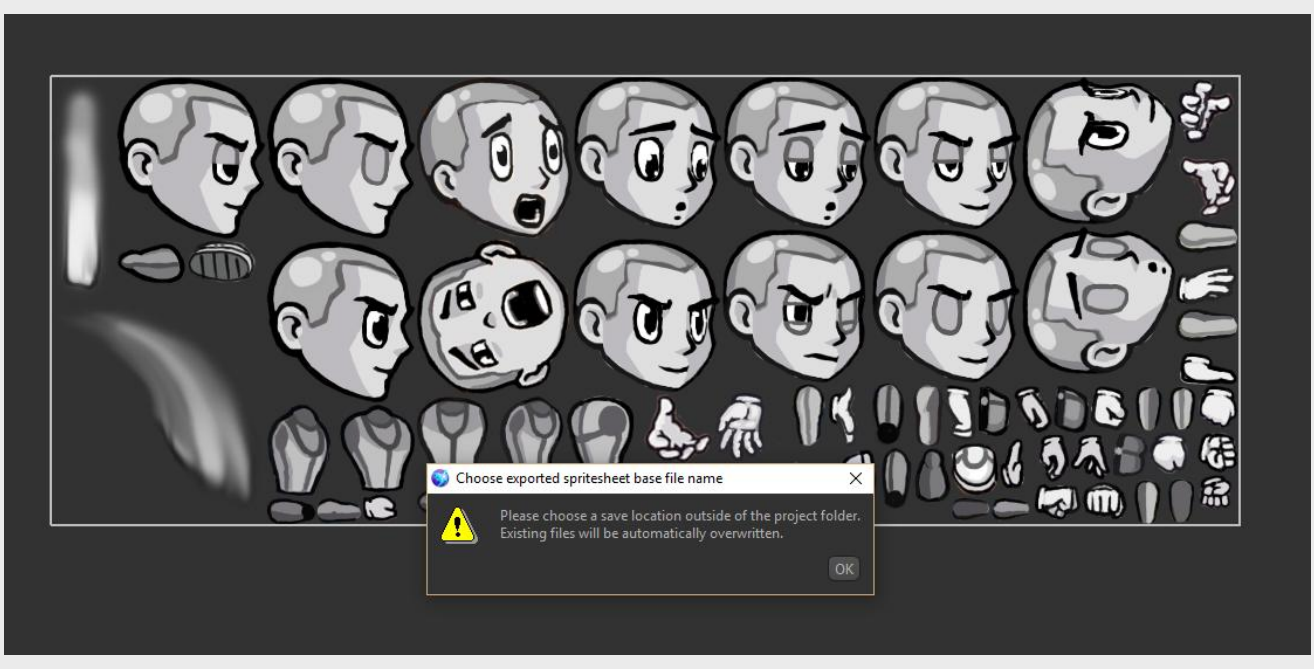
IMPORTANT: At the time this manual was created, due to this being a new feature, none of the official Spriter implementations support playback of Spriter files which use these texture-atlases. While we'll do our best to make sure our official reference implementations include support for this feature in the near future, we can't guarantee how soon, and have no control over if or when support will be ported to 3rd party Spriter API's. For this reason, before using this feature, be sure to make sure whichever authoring system you will be using has support for Spriter and texture atlases.

Here's how to do it.

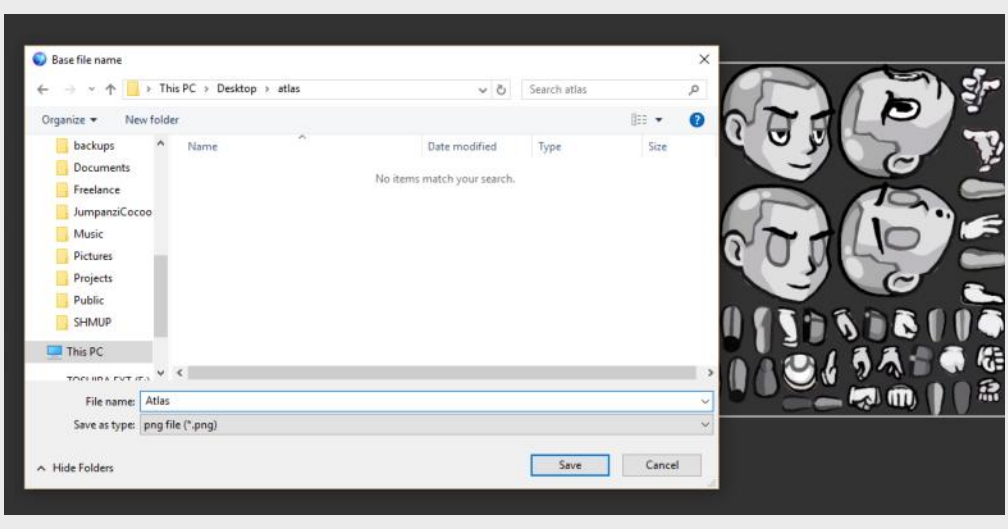
1) Once your Spriter project is finished, choose File/Generate spritesheets for project images from Spriter's menu.



2) The "pack images"dialogue and a small pop-up will appear. The pop up is telling you to select a target folder for the texture atlases and clone project which will be created. Click OK and then a file selector dialogue will appear. Pick your destination folder and type the name prefix you'd like the generated files to have. Then click OK.

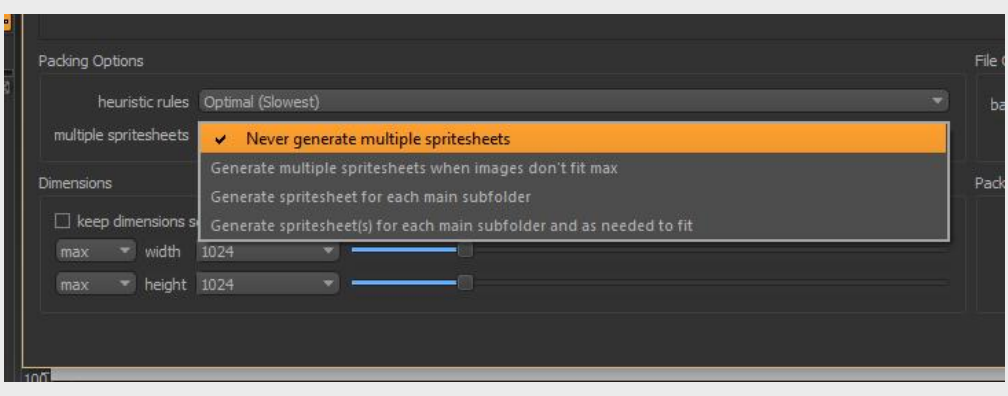


3) Now you have access to all the settings in the "pack images" dialogue. The first thing to do is decide if you just want to create texture atlases or if you want texture atlases AND a clone of your Spriter project which will use them instead of the separate images. If the later is the case, then make sure the "save spritesheeted project" checkbox is checked.

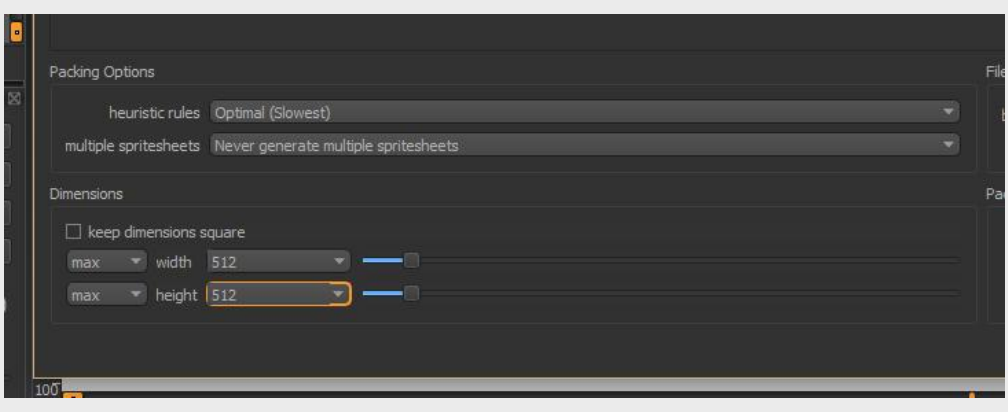


4) The next thing you'll want to do is decide how Spriter will organize the images when creating the atlases. In the "multiple spritesheets", select from one of the following options… whichever best suits your needs:

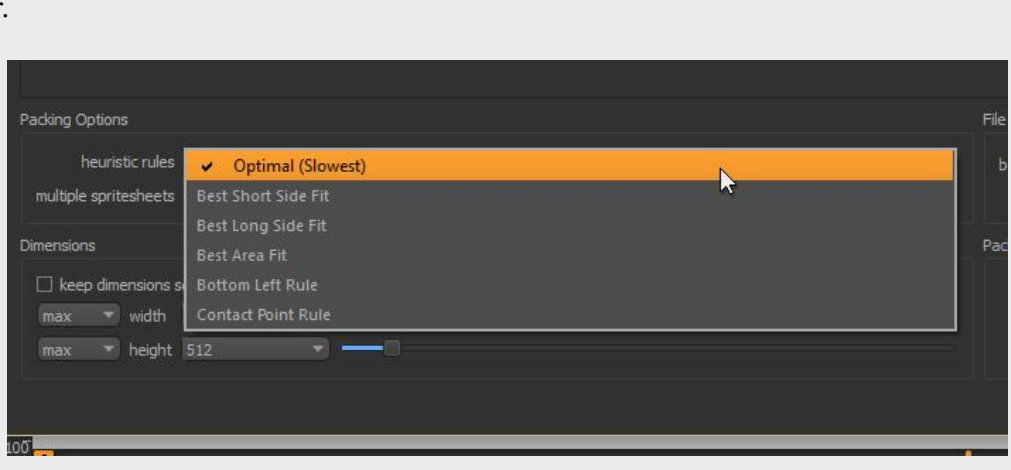
- "Never generate multiple spritesheets." (if you use this option, be sure you set the texture atlas maximum size large enough to fit all the images your project uses.)
- "Generate multiple sritesheets when images don't fit maximum." (This option lets Spriter create as many texture atlases as needed to fit all the images.
- "Generate spritesheets for each main folder." (this option will make a texture atlas for each image folder in the main folder. Any images sub folders of those folders will be included in the single texture atlas for each main folder in the project.)
- "Generate spritesheets for each main folder and as needed to fit." (this option covers all your bases, starting with an atlas per folder, but also making additional atlases per folder if required to fit all the images for each folder into the initial atlas for that folder.)



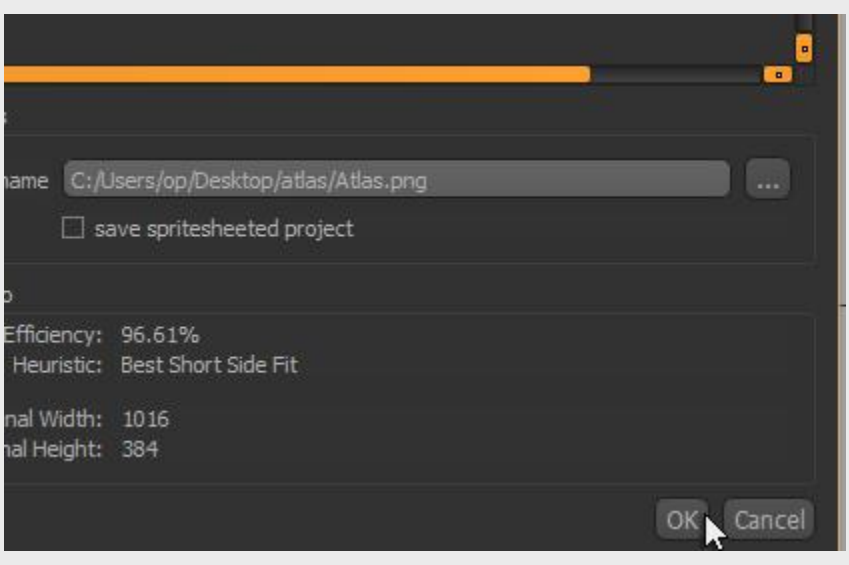
5) Set the maximum size for any texture atlas image. The default is 1024, but you can set this to suit your needs, HOWEVER, make sure this maximum size, coupled with the setting you chose in step 4 allow for all images to fit into the texture atlas(es). An easy way to tell is the OK button at the bottom-right of the "pack images" dialogue. It will be greyed out and not clickable if your settings can not accommodate all of your projects images.



6) If you're dealing with an especially large project it might be possible that Spriter might take some time to find the most optimized texture atlases (though in our testing even large projects per packed in a matter of seconds). If for some reason the default "heuristics" setting is taking too long, you can use the drop down list in the "heuristic rules" setting to pick an option that might be slightly less optimized, but will process faster.



7) Once all settings are to your liking, just click the "OK" button near the bottom-right of the "pack images"dialogue and your texture atlases (and Spriter file which uses them) will be created in the target folder you had selected!



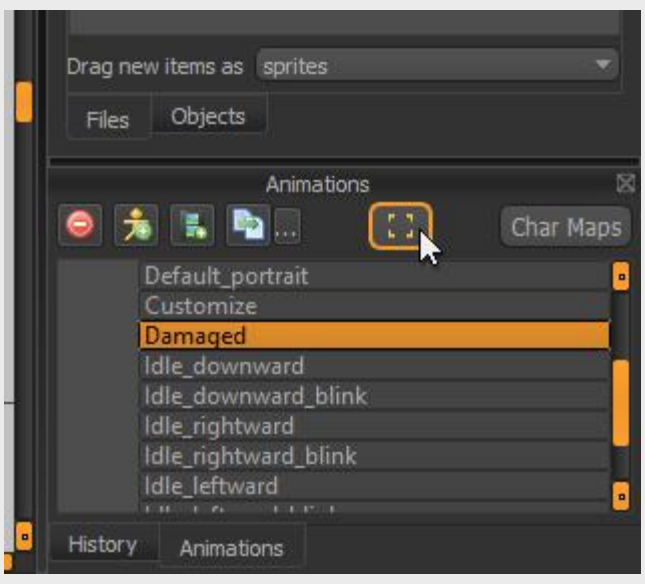
Creating Custom Trimming Settings For Each Animation

For anyone who will be using Spriter to create animations to be used in game engines that require sequential image frames or sprite-sheets, one important requirement is often to have every frame (possibly of all animations) cropped to a very specific pixel dimension.

Spriter offers a great way to visually set the cropping of any animation in a very visual way. The cropping settings for each animation will be saved the next time you save your Spriter file, so if you ever need to tweak then re-export your animations you won't need to set the cropping a second time.

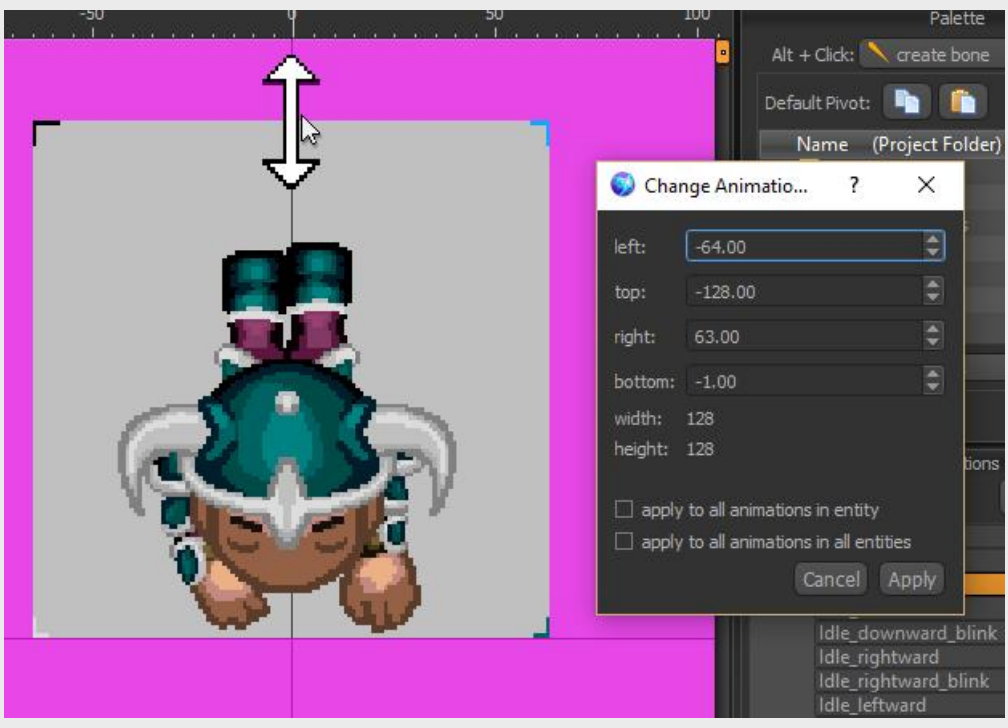
Here's how:

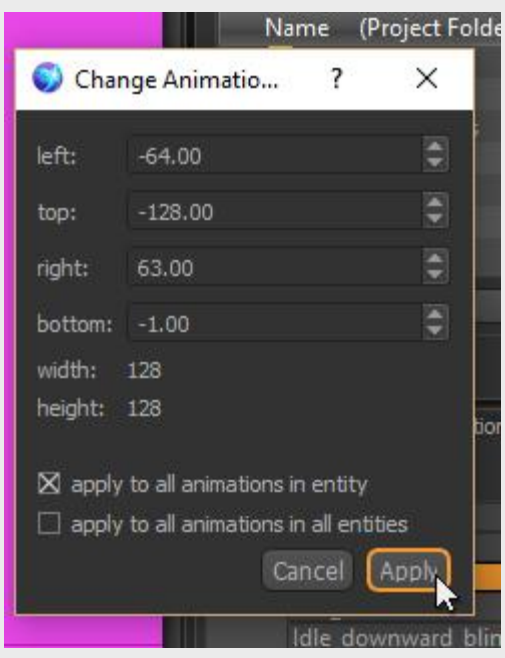
1) Select the animation you'd like to create or edit the custom trimming settings for, then click the small orange cropping square icon ("Change Animation Export Box Size") to bring up the special cropping dialogue. The canvas area should change in appearance, with the cropped out area of the canvas being a magenta color and a grey rectangle representing the portion of the canvas which would be visible in the resulting exported animation.



2) If you hover your mouse pointer near any edge of the grey rectangle, you'll see arrows appear which sow that if you left click and drag you'll be able to adjust the position of that edge to resize and reposition the clipping rectangle. The "Change Animation Export Size Preset"dialogue box will update on the fly to let you know the exact pixel dimensions of the resulting animation frames.

IMPORTANT: You can use the 1 and 2 keys to change which key-frame you're viewing in the canvas or lift click and slide along the timeline to look at any point in the animation to make sure you're not going to accidentally crop out part of your art-work.



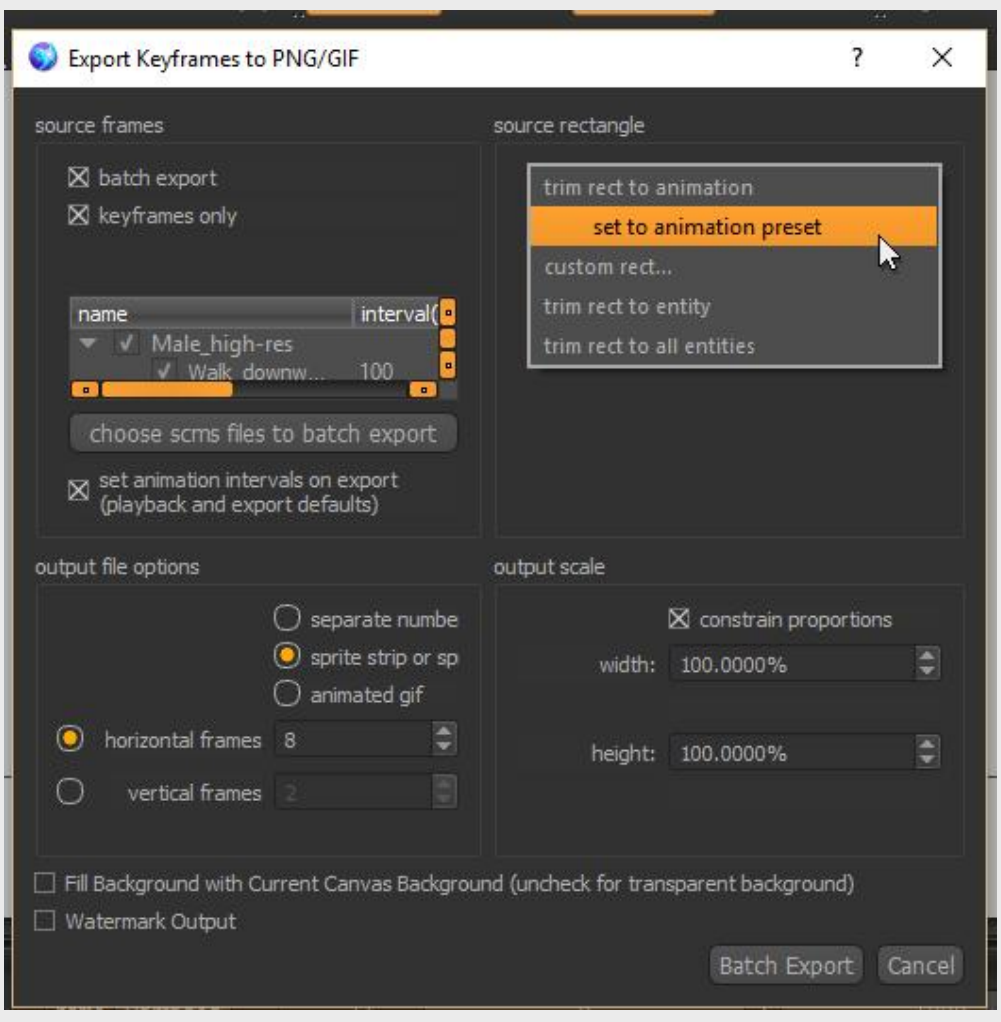


3) You can also type directly into any of the numerical settings in the dialogue box to adjust the cropping numerically.

Once you've perfected your cropping settings you must decide if this should be applied only to this animation, to all animations in the Entity, or to all animations across all entities, and to check our un-check the appropriate check-boxes at the bottom of the dialogue box.

4) Once you are sure of your settings, click "Apply" at the lower right of the dialogue box and your settings are ready to use. Don't forget to save your Spriter file so you don't lose your custom cropping settings!

5) Once you're ready to export your animation, be sure to change the "source rectangle" option in the "Export Keyframes to PNG/GIF" dialogue to "set to animation preset" for your custom cropping settings to be used.



Using Spriter's Custom Color Features

As you may have read by now with Spriter's "Character Maps" feature, you can set up your Spriter project to allow for real-time swapping or hiding of any of the used images in order to create large varieties of possible visual combinations, for alternate characters, outfits, etc. It's possible to greatly expand on this level of customization by adding the ability to exchange and combine color combinations as well.

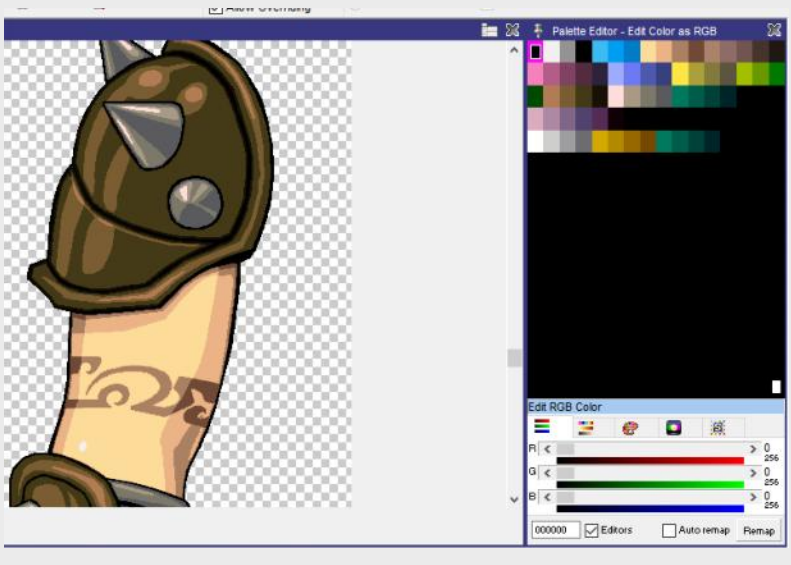
Our RPG Heroes Art Pack uses this combination of Character Maps and Custom Color Palette manipulation to offer a massive assortment of possible character combinations to the user. While this is feature does offer some great benefits, it's much more complicated to use than Character Maps, and puts much stronger technical demands on the artist creating the images.

Another thing that is very important to keep in mind is that at the time this manual is being written, no Spriter run-time (plug-in) for any language or authoring system has support for using these Color Palette manipulations at run time. This means unless you confirm that the runtime you'll be using for your game authoring system of choice actually supports the Color Palette features, then be sure you are using it only for creating exported full frame images or sprite sheets, or, to "bake out" the final Color Palette combination to all images permanently before using them in your game engine.

Here's how it works:

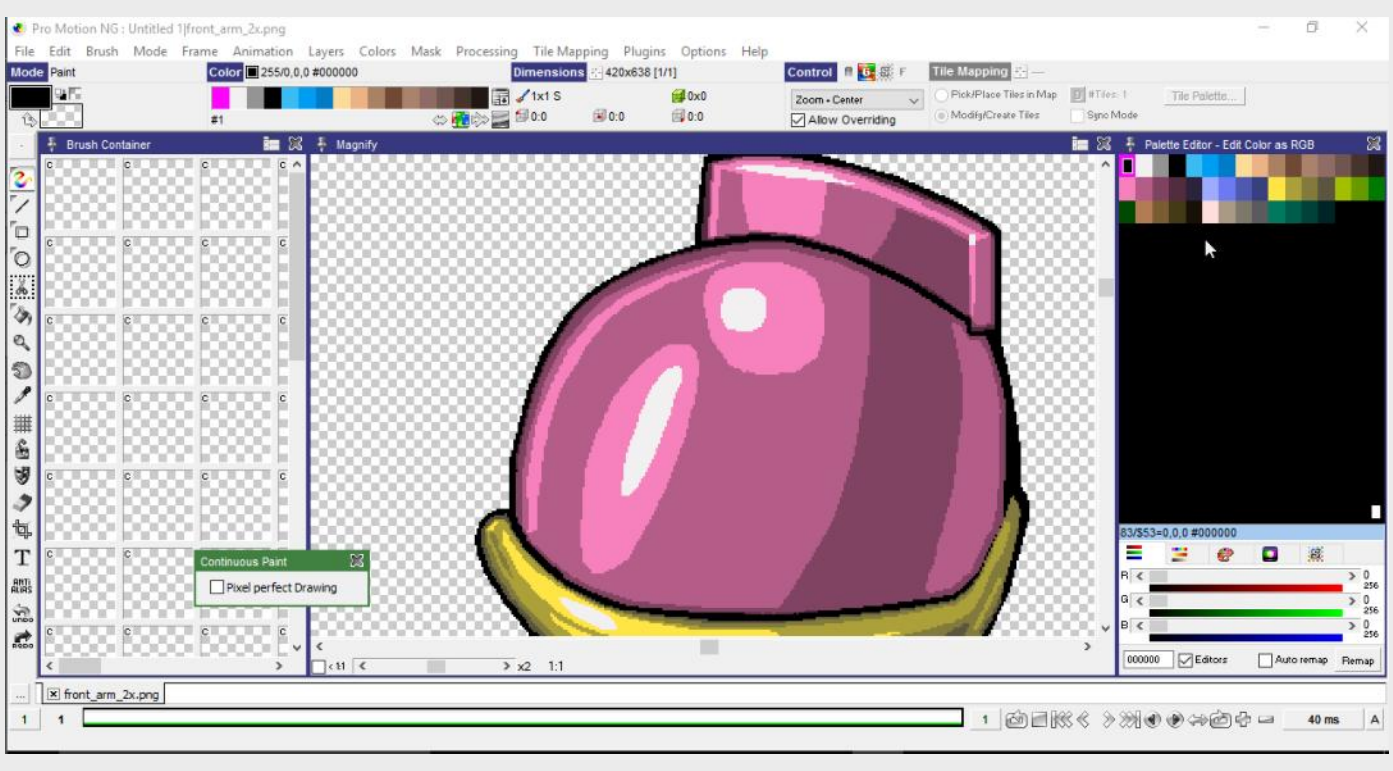
The most important thing to know if you want to use this feature is the difference between "full color" images and "indexed color" images":

Full color images contain the full color information for every pixel of the image, as well as the level of opacity for each pixel. This makes the file size much larger, but allows for absolute freedom for the artist at the time of creating the artwork. This is what's typically used in modern games and web graphics.

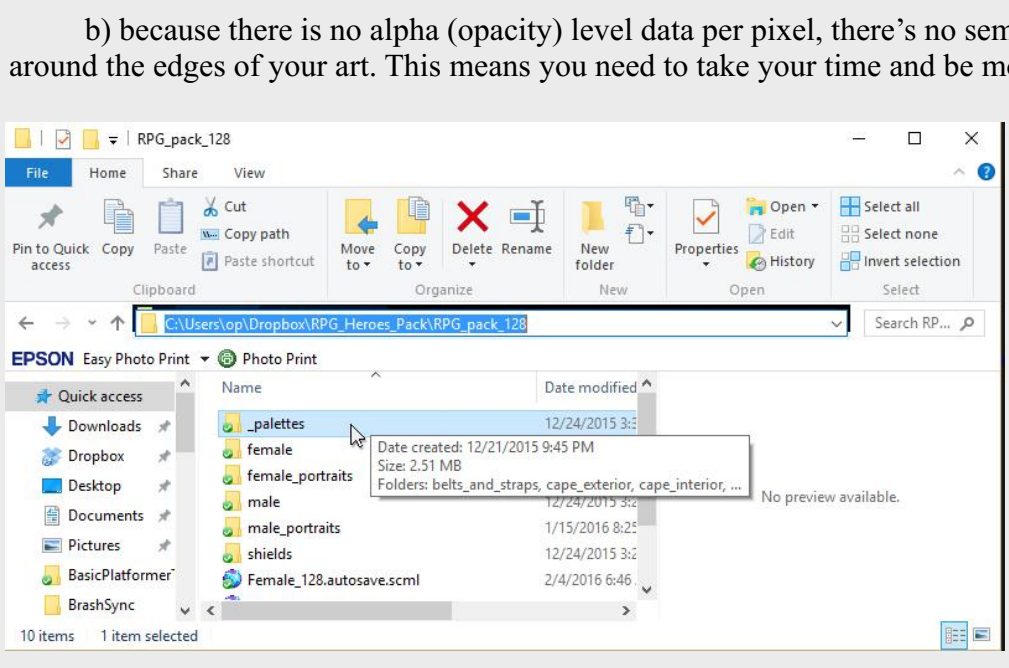


Indexed color images allow for a limited number of colors that all pixels must share from. This limited set of colors is often referred to as the color "palette", and each color in the palette has an index (a number) color index zero would mean the first color, index 255 would be the last color (256 colors total). This type of images takes up less file space because instead of having full color data for every last pixel, it simply has the color index from the palette that any given pixel should use. The other benefit of indexed color images is, that with programming tricks, tools like Spriter can allow users to customize these "color palettes" on the fly, so suddenly, the user is not just choosing from 5 different style of shirts for example, but can now also mix and match from an nearly unlimited number of color options for each shirt style as well! On top of the limited number of colors however, there are other trade-offs when using indexed color mode.

a) In most or all modern digital painting tools, the vast majority of features will not work in indexed color mode! For this reason, if you'd like to use the color features, we recommend you find and learn a pixel-art program, specialized for working in indexed color mode. The one I'll be using in all the tutorials will be Pro Motion NG by www.cosmigo.com



b) because there is no alpha (opacity) level data per pixel, there's no semi-transparent anti-aliasing around the edges of your art. This means you need to take your time and be more precise with your art-work and make sure the outlines are as clean as possible to ensure high visual quality of the final animations in game.



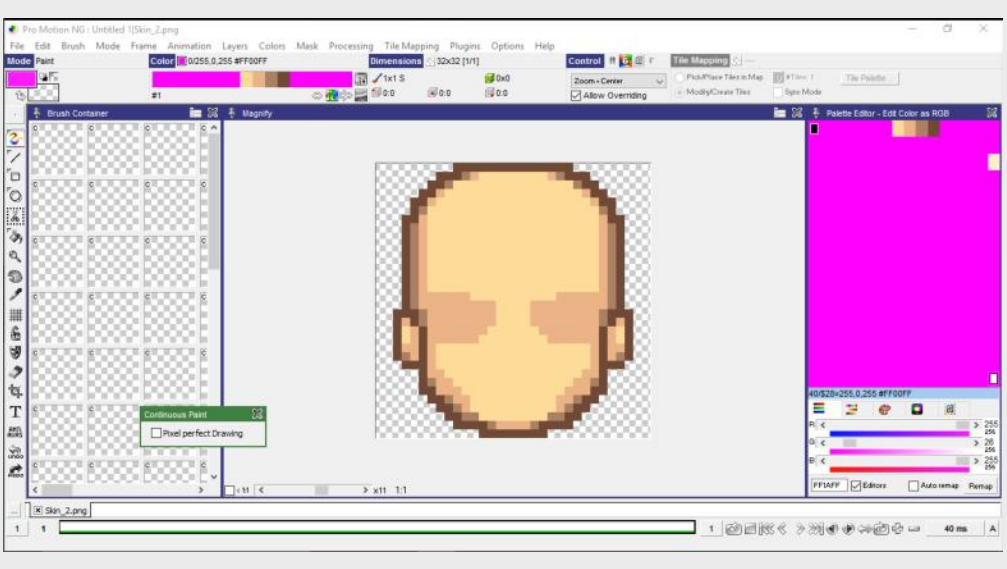
HERE'S HOW

1) Use a pixel art program like Pro Motion, create your art, making sure all aspects/images for the animations you'll be creating are using the same, organized color palette… for example, make sure all color ranges are from light to dark, make color indexes 1 through 8 the colors used for skin (color zero typically reserved for the transparent background color) Be sure to save your images out with color zero (the background color) set to transparent.

IMPORTANT: Be sure to make sure you're saving your images in indexed color mode and that you do not have the program set to "save in lowest bit depth" or "save with fewest colors" as these settings would destroy the order of the color palette.

2) Now that your Spriter project has images to use and you've started to assemble your animations, you may be ready to explore color palette customization options. In order to do this, the first thing you'll need to do is create a folder called "_palettes" in the root folder of your Spriter Project. It must have that exact name. You can then set up sub folders (named whatever you'd like) within the "_palettes" folder.

3) Now you have to create special palette image files (in your "_palettes" folder) which will give Spriter the color data needed for the colors you want this palette image to change. Let's get back to the idea of skin tones. In this example, lets say you had used the first several indexes (after index zero, which is transparent) for skin tones in all your images. To create a palette image to allow for applying a new skin tone, you'd need to edit those specific color indexes in an images to have the new skin colors.

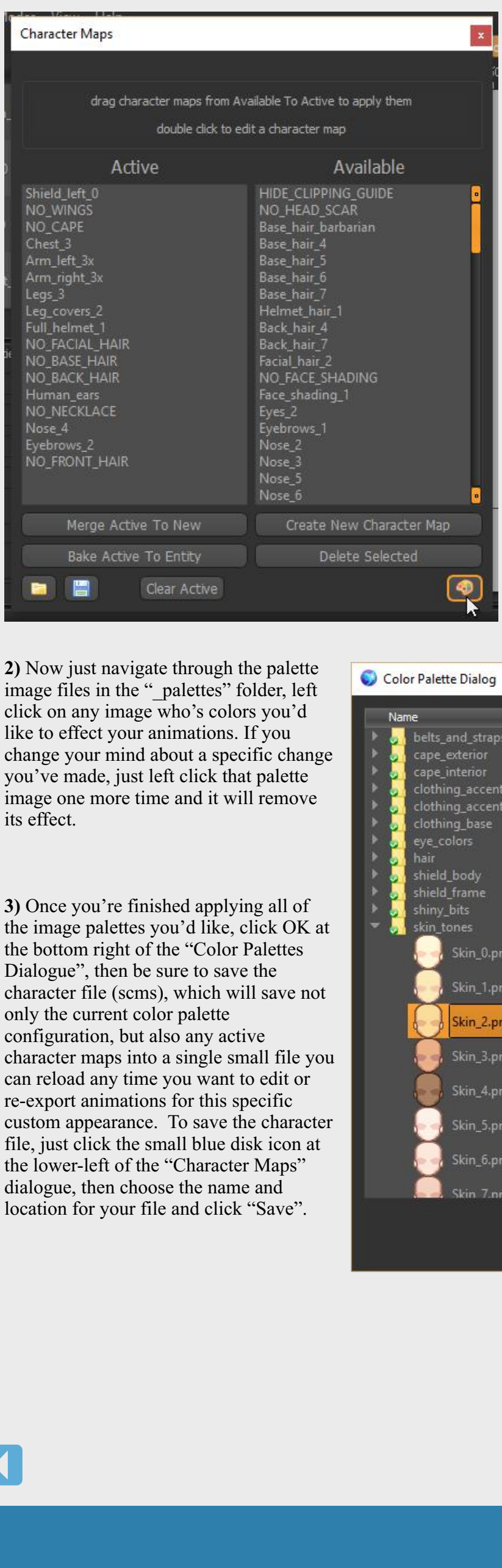


I highly recommend you create a custom 32x32 pixel image representing what the palette image will be effecting, and which uses those specific color indexes. Look how for the RPG Heroes Pack, I made the skin-tone palette images look like a blank face. This allows the user to instantly recognize both what the image will effect, AND what the new skin-tone will look like.

IMPORTANT: Aside from making sure all images use the same palette arrangement (same indexes being used for the same things, such as skin tones), the most important thing you need to remember is that ALL color indexes other than the colors you want your palette image to effect MUST be set to what is called "programmer pink". This is RGB value 255,0,255 or hex(web) value: ff00ff . This is how Spriter knows which colors to effect when the user selects that particular palette image.

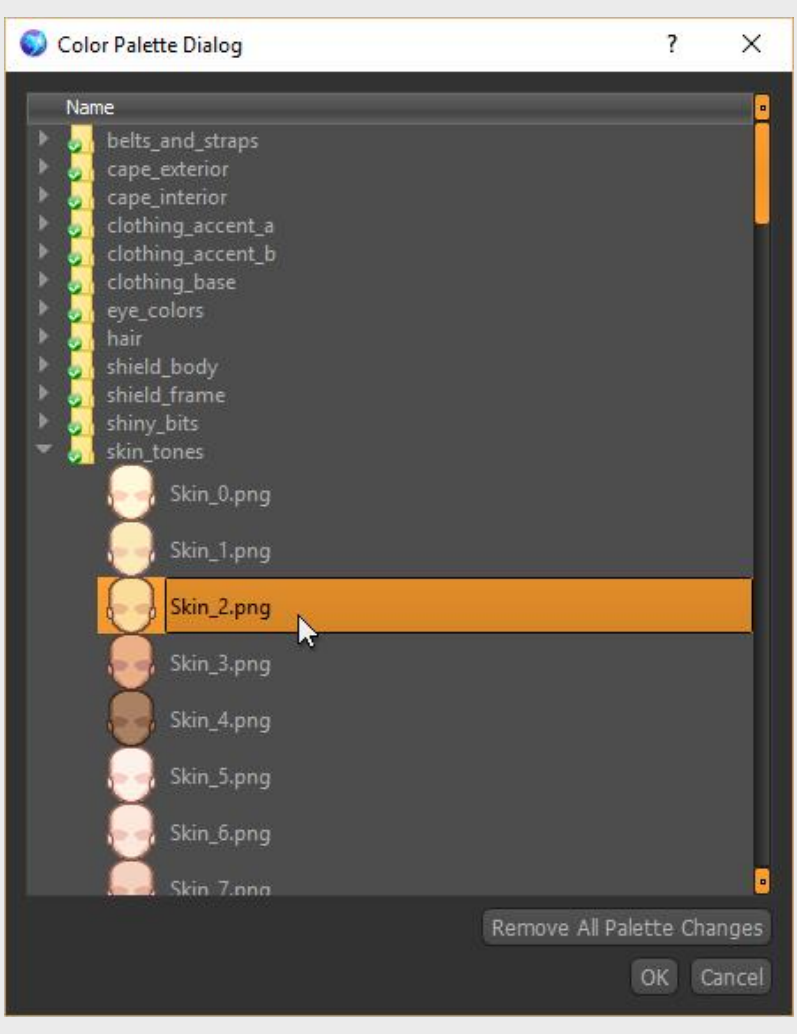
Once your Spriter project has animations or frames made of indexed images, and palette files ready to effect them, you're ready to start applying the palette files to the current character map "stack" so that you can create custom visual variations for your animations.

To do so, you'd do the following:



1) Click the "Char Maps" button at the top right of the animations palette to bring up the Character Maps dialogue. Then click the small painting palette looking icon at the bottom right of the Character Maps dialogue. This will bring up the "Color Palette Dialogue"

2) Now just navigate through the palette image files in the "_palettes" folder, left click on any image who's colors you'd like to effect your animations. If you change your mind about a specific change you've made, just left click that palette image one more time and it will remove its effect.



3) Once you're finished applying all of the image palettes you'd like, click OK at the bottom right of the "Color Palettes Dialogue", then be sure to save the character file (scms), which will save not only the current color palette configuration, but also any active character maps into a single small file you can reload any time you want to edit or re-export animations for this specific custom appearance. To save the character file, just click the small blue disk icon at the lower-left of the "Character Maps" dialogue, then choose the name and location for your file and click "Save".

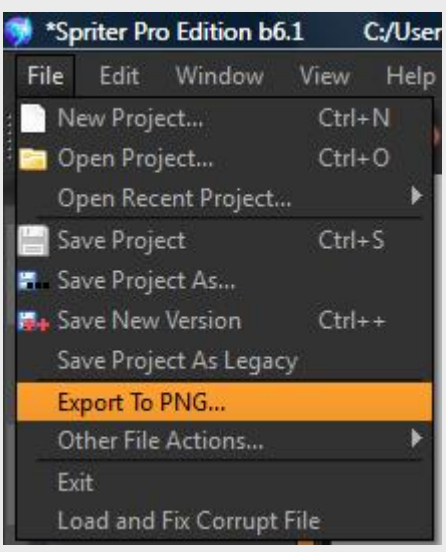
Exporting Animations as Sequential Images or GIFs

Ideally you'll be able to use your tweened and optimized Spriter animations directly in the game authoring system of your choice, but obviously this can't always be the case. Sometimes you'll need to create animations for a game engine that can't support Spriter files directly.

Similarly, sometimes you might create an animation for things like explosions which end up using dozens of tweened, rotating and scaling images at once, but plastering your game-screen with lots of these explosions might adversly effect your frame-rate. Therefore it would be better for your game if the explosion could be converted to just a sequence of full frame images (to reduce the need for all the seperate draws and tweening calculations).

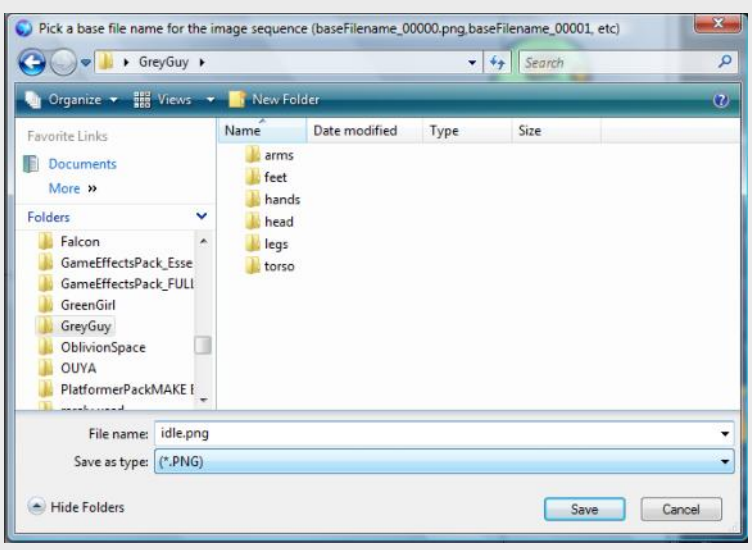
For these types of circumstances, Spriter lets you export finished animations as sequential images at any frame rate (number of frame images per second) you'd like.

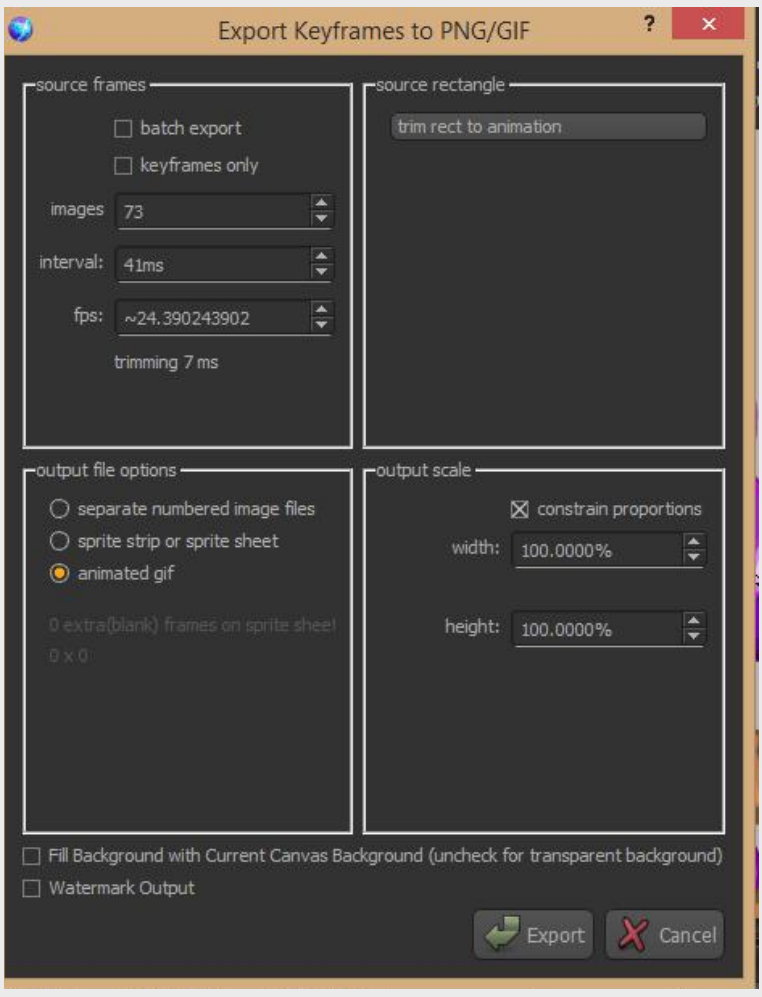
Here's how:



1) Once your animation is finished, make sure that animation is selected in the Animations Palette and then from Spriter's menu choose File/Export To PNG.

2) A file selector will appear, allowing you to designate the location and name for the images to be saved. Navigate to or create the folder you'd like the images to be saved in and then type the name for the images and press enter or click on the save button.





3) The Export Keyframes To PNG Dialogue box will appear, giving you a plethora of options to choose from while exporting your animation as sequential images. First, decide if you want to only export the keyframes themselves, if you'd like to export frames based on a specific number of frames per second, or if you'd lust like to tell Spriter how many frames total the export should create for the entire animation. You can also just tell Spriter to export an image per however many miliseconds you'd like. These are the first 3 options in the palette.

4) Using the second part of the export dialogue you can decide how your animation will be trimmed. Most of the time, You'll want to leave it at the default settings, as its sure to not accidentall clip (remove) parts of any of the frame. But, if you want more specific controll, or specifically want to crop the animation, then you can use the settings in this part of the palette to specify exact cropping coordinates. Just click the grey rectangle that says "trim rect to animation" and the other options will appear. Choosing any of them will reveal additional settings specific to that cropping option.

5) You can use the bottom section of the palette to decide at what scale you'd like to export your animation. By default it will be set to 100 percent. (actual size). Just use these settings to determine what size you'd like the animation exported to. Once you've decide and you've made sure all the settings from the previous steps are correct, just click OK and the exporting will commence into the folder you had selected.

6) Notice the option checkbox called "batch export" at the top left of the export dialogue window. If you check this option, you'll be able to choose to export all or any number of specific animations all at once, all with identical settings. This can save you tons of time and assure all animations are cropped (and trimmed) to the same exact dimensions etc.

Adding Variables to an Animation (Pro Only)

Spriter Pro offers the ability to add and several gameplay related types of data to your animations, with the ability to change their values or settings at any time throughout the animation. These values, if numerical can even tween smoothly from one keyframe to the next. If this sounds confusing so far, don't worry, We'll go over the different data types and an example use one by one.

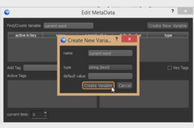
VARIABLES: Variables can be string (text), floats (numbers with decimal points), or integers (whole numbers). You can give variables any name you'd like, and even give them a default value to start off with. Variables are a way to have your Spriter animations tell your game engine all kinds of information that can change aspects of gameplay, visual que's for the user etc.

Text Variables: Sometimes text is more useful or easy to understand than numbers, and can either communicate a string to your game engine, or directly to the player.

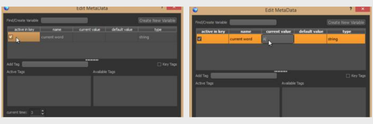
Example: Imagine you were making a game in which there were animated devices on screen that would display specific letters or words, and cycle through, changing the word or letter they display over time, on a loop. The player can click on the device while its displaying that specific letter or word in order to input that specific letter or word to the game, perhaps to solve a puzzle or answer a question.

Of course you could just check the specific times along the timeline in Spriter where each letter or word is displayed and hard-code those time intervals into your game engine to represent each character...but not only is this tedious and unintuitive, it would mean if you changed the timing of your animation you'd need to check and replace all your hard-coded time numbers in the game engine which represent each letter or word. Luckily, With Spriter pro you can work in a much more intuitive and flexible manner. Here's how:

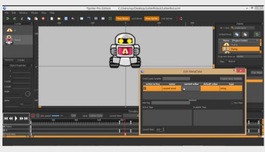
1) Expand your timeline panel upward to reveal the timelines for each specific object. If you scroll to the very bottom of all timelines you'll see a timeline titled "meta data" Double click in the meta data timeline exactly where the first letter or word is visible and should now be clickable to the player. This will bring up a dialogue which you will use to create and edit your "current word" variable. Type in the name you'd like for your variable in the find/create variable box at the top-left of the dialogue, then click the "create variable" box.



2) This will bring up a second dialogue box which will allow you to set the type (we want text), and the default value. If the animation starts on a specific word or letter you can set that as the default if you'd like. In this example we'll assume the animated device is displaying the letter "A" by default. Once you've entered your starting word or letter, (if there is no starting word or letter, just leave the default blank.) click "create variable".



3) You'll now see your variable in the variable list in the "Edit MetaData" dialogue box. Click on the "active in key" checkbox and this will create a keyframe in the metadata timeline for this new variable. Now you can doubleclick on the "current value" box and type in the word or letter that the device animation is currently displaying. Once you've finished typing the letter or word, press enter and then close the dialogue box via the red X close icon at the top right of the window. Don't worry, your changes will be automatically saved.



4) Now you should double click later on in the timeline, when the animation is either showing no letter or word, or has changed to a new letter or word. This will bring the "Edit MetaData" dialogue back up. Now reselect the variable you've already made by left clicking on it's name, and then check the "active in key" checkbox again. Now you can double click in the "current value" box again and enther the new text value for the new letter or word that's being displayed by the animation.

5) Once you've repeated step 4 for the entire animation so that all visible letters have a corresponding text value set in the metadata for that portion of the timeline, then save your Spriter project.

In the game engine itself, once this Spriter file is being read and displayed, the logic of the game engine would be something like:

Is the player clicking on the Spriter object displaying the device animation?
If yes, then set the text value of the letter or word to the current value of the "current word" variable from the metadata.

Then you'd have the game react accordingly depending on whether or not the word or letter the player clicked on was the appropriate answer.

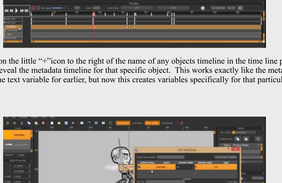
INTEGER AND FLOAT VARIABLES work exactly the same, except they are useful when you need to communicate changing numerical data to your game engine, depending on the current time of the currently playing animation. One cool bonus to numerical variables is that they automatically tween between each of their keyframes in the metadata timeline if the numerical value is different on each of the key frames. Don't forget, with things that tween, you can right click and hold on their specific key frames in the timeline and choose the type of tweening you'd like to use, including no tweening at all. (instant)

PER OBJECT METADATA: In the previous example we created a variable in the metadata for the entire animation. This is useful if the data you wish to create is relative to the timing of the entire animation, but there's a better option for when you want to create and communicate value which should related directly to specific objects within the animation.

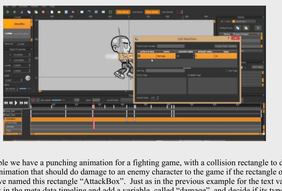
We'll use collision rectangles as an example:

Assigning Numeric Data to a collision rectangle: As you might already know, you can place as many distinct collision rectangles as you'd like at any given point along the time line of an animation. Their size, shape, and angle can tween between their key frames. Each Collision rectangle can have a different name. This alone offers some pretty awesome control of game play related data right within Spriter, but Spriter takes it much further than that....

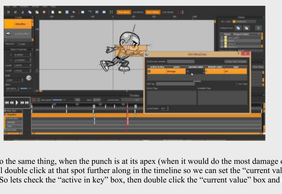
EXAMPLE: In this example we'll be adding a variable to a collision rectangle to control the amount of damage it would inflict when it collides with enemies within a game.



1) If you click on the little "+"icon to the right of the name of any objects timeline in the time line palette it will expand to reveal the metadata timeline for that specific object. This works exactly like the metadata we've created the text variable for earlier, but now this creates variables specifically for that particular object!



2) In our example we have a punching animation for a fighting game, with a collision rectangle to designate the part of the animation that should do damage to an enemy character to the game if the rectangle overlaps the enemy. We've named this rectangle "AttackBox". Just as in the previous example for the text variable lets double click in the meta data timeline and add a variable, called "damage", and decide if its type os going to be float or integer. For most game types integer would be fine. Lets leave its default value blank, and set its current value to 10. Don't forget to make sure its set to "active in key".



3) Now we'll do the same thing, when the punch is at its apex (when it would do the most damage on impact) and well double click at that spot further along in the timeline so we can set the "current value" to a higher number. So lets check the "active in key" box, then double click the "current value" box and then set it to 20.

In your game's engine, the logic would be something like this:

Upon overlap of player Damage Collision box and an enemy.
AND the enemy is in a state where it can take damage.
THEN make the enemy play his taking damage animation.
AND temporarily set the enemy's state so it can no longer take damage. AND subtract the current value of the Variable :Damage" from the Collision Rectangle called "Damage" from the hitpoints of that specific enemy.

As you can see, Spriter makes creating, editing and tweaking game play related data, and synchronizing it perfectly with each animation very easy and intuitive. In this example, not only have we included the hit strength of a punch right into its collision rectangle, but we even made the number tween and change depending on how much damage that punch in motion would inflict if it collided with an enemy at that specific millisecond!

Be sure to read the next section: "Adding "Tags" to an Animation", because Tags are the other, super powerful and flexible half of the example logic I provided above for the collision box delivering the damage ot the enemy. Specifically the portion:"AND the enemy is in a state where it can take damage."

Tags are Spriters easy and intuitive way to designate what "state" your entity or character is in at any moment along the time line of each animation.

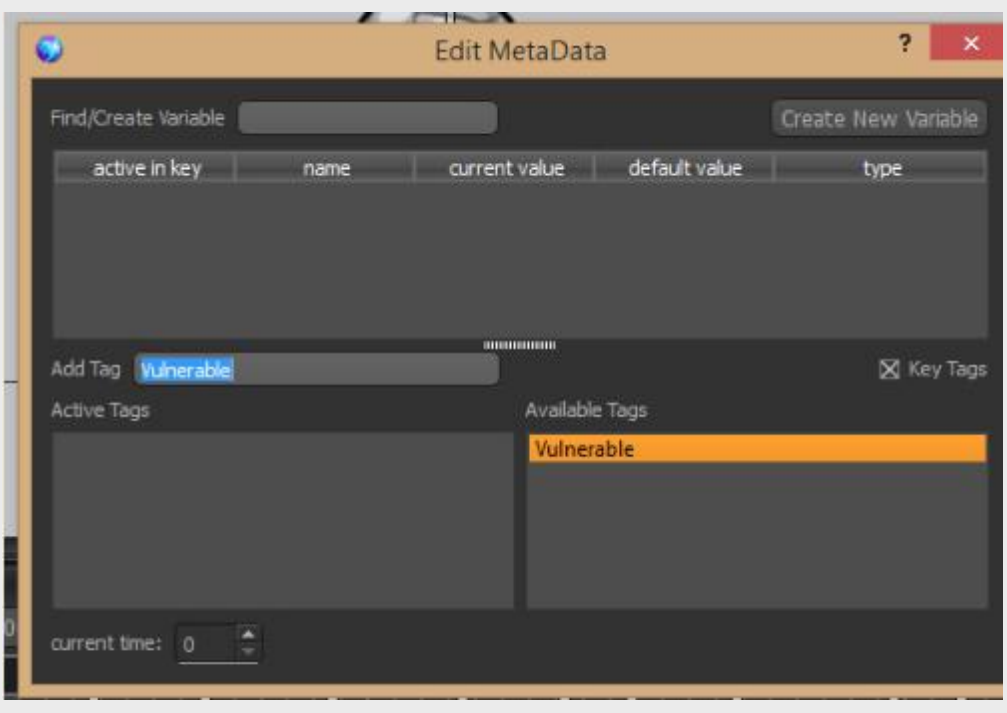
Adding Tags to an Animation (Pro Only)

One of the most powerful and easy tools people use to control what can happen any any given time in their game is called a "state machine". In a nutshell, a state machine is the part of a game engine that follows the games logic and decides what can happen in any given situation depending on the "state" of the objects involved. In the previous chapter "Adding Variables to an Animation" I used the example of a player character's punching animation.

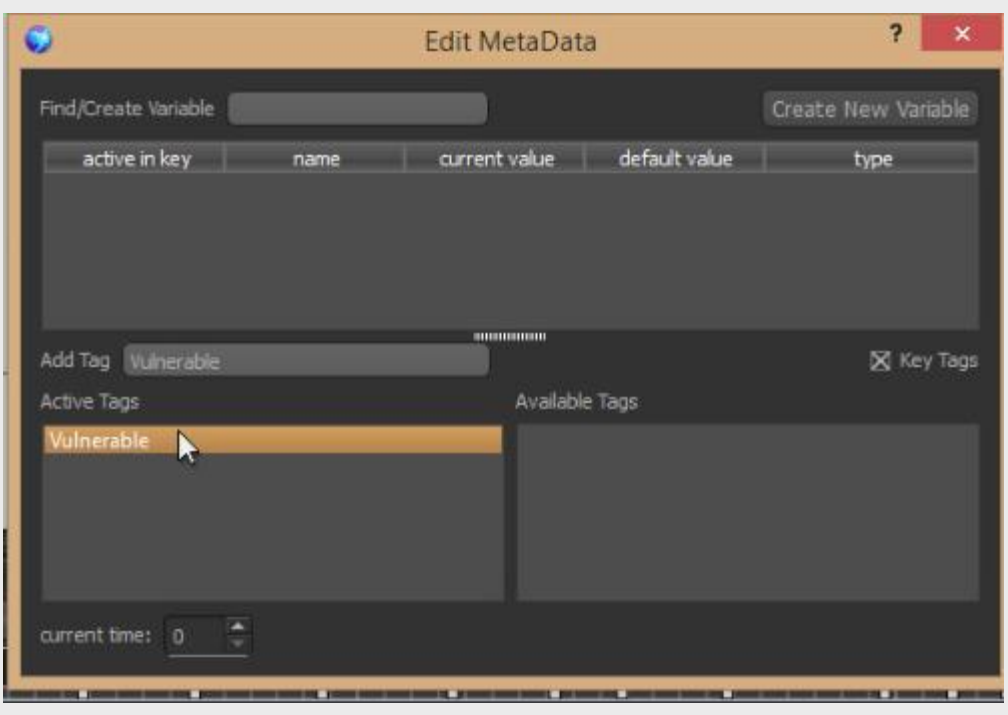
Ideally when a game engine checks the collisions of things, such as the player with an enemy, it simultaneously checks to see if the enemy is currently in a state where it should be able to take damage etc. The tricky part is, depending on the design of any given game, a character/entity might be able to be in several states at once. Spriter Pro can help you perfectly designate, organize and "stack" any number of state possibilities per entity, per animation, per millisecond.Adding Tags to an Animation Spriter Pro User's Manual version 1.0 Quickstart Sprites Bones Animating Character Maps Index

EXAMPLE: In this example we have an enemy character with three animations: idle, getting hit, and blocking (guarding). The first animation we'll add a tag to is "idle".

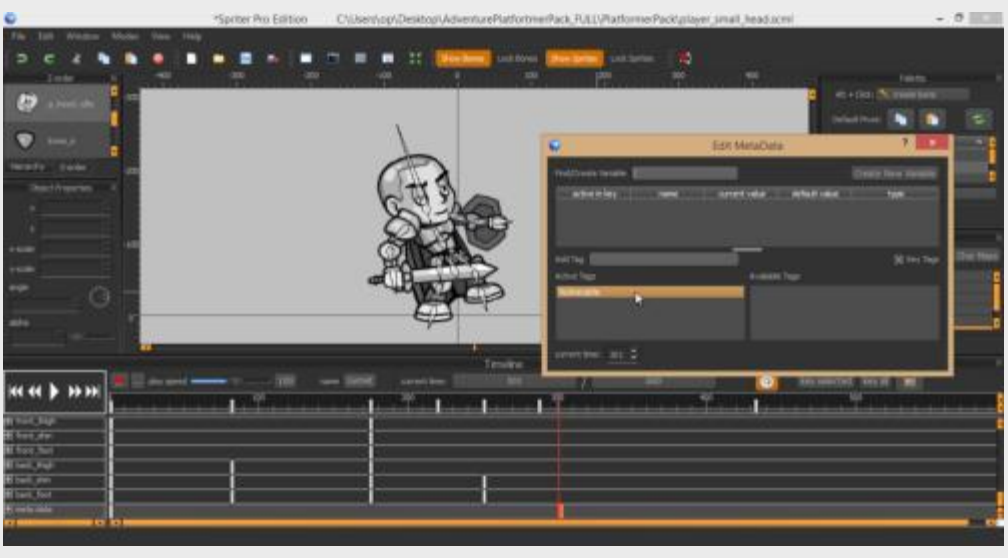
1) Choose the Idle animation, expand the time line palette upward to reveal the individual object timelines, and then scroll to the very bottom so we can edit the metadata timeline for the entire animation. Doubleclick on the metadate timeline right at the very beginning (zero milliseconds).



2) This will bring up the "Edit MetaData" dialogue box. This time, instead of creating a variable, type in a name in the "Add Tag" box and press enter. In this example I'm creating a Tag called "Vulnerable". Once you press enter, you should see the tag you just created in the "Available Tags" column.



3) Now check the "key Tags" checkbox toward the middle right of the dialogue and then drag the name o the Tag you just created from the "Available Tags" Collumn to the "Active Tags" column. Not only have you created the Tag, but you've also told Spriter that starting at the beginning of the animation, this Tag is active...in other words, from the start of this animation, one of the states of this entity is "vulnerable"!...meaning, it can get hit and take damage from the player.



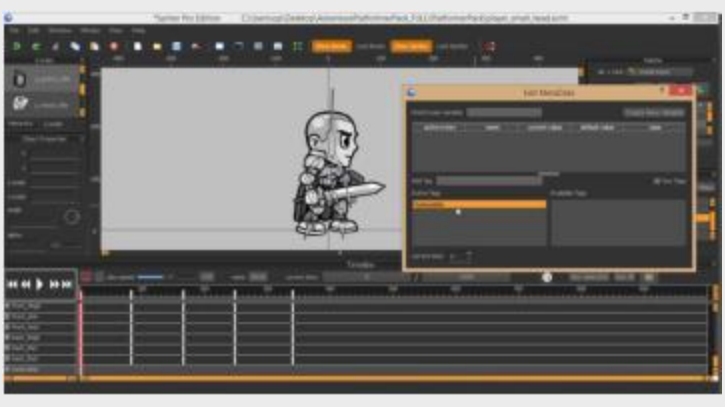
4) Now lets go to the "getting hit" animation.
The tricky part of a characters getting hit animation in a game is (depending on the game) the enemy getting hit should no longer be vulnerable to new damage until after the getting hit animation is finished playing, or, should not be vulnerable to additional damage until at least a certain amount of the getting hit animation has played. This allows for instances like the ability to repeatedly hit enemies with a sequence of attacks in a fighting game etc.

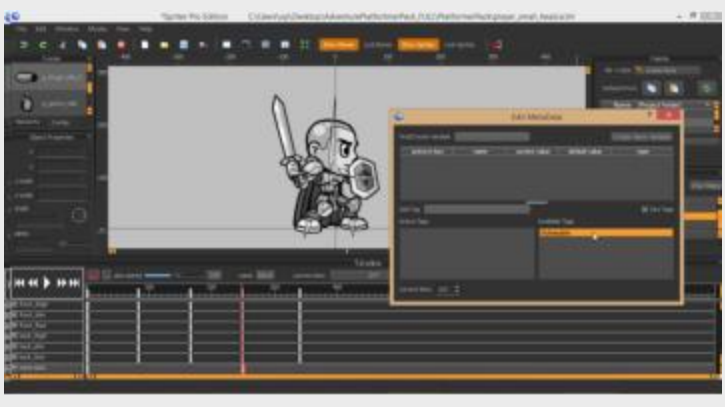
For this example, we're going to make the enemy impervious to other damage for the first half second of his getting hit animation, and then once again vulnerable to attacks after that initial half second.

Go to the mid-way point of the Meta Data timeline for the entire animation (the very bottom of the timeline palette) and double click to bring up the "Edit MetaData" dialogue. You should see the "Vulnerable" Tag already in the "Available Tags" column. No need to recreate it. Tags are automatically universal throughout the Entity you create them for.

Drag the "Vulnerable" Tag from the "Available Tags" column to the "Active Tags" column, making sure the "key tags" checkbox is checked (on).

Finally, use the "current time" box at the very bottom-left of the "Edit MetaData" dialogue in order to navigate back to the very beginning of the timeline (zero) and then make sure the "Vulnerable" Tag is in the "Available Tags" and not the "Active Tags" column. Now the enemy's "Vulnerable" Tag (or state in this case) should be invulnerable for the first half of the getting hit animation and vulnerable for the second half of the animation.

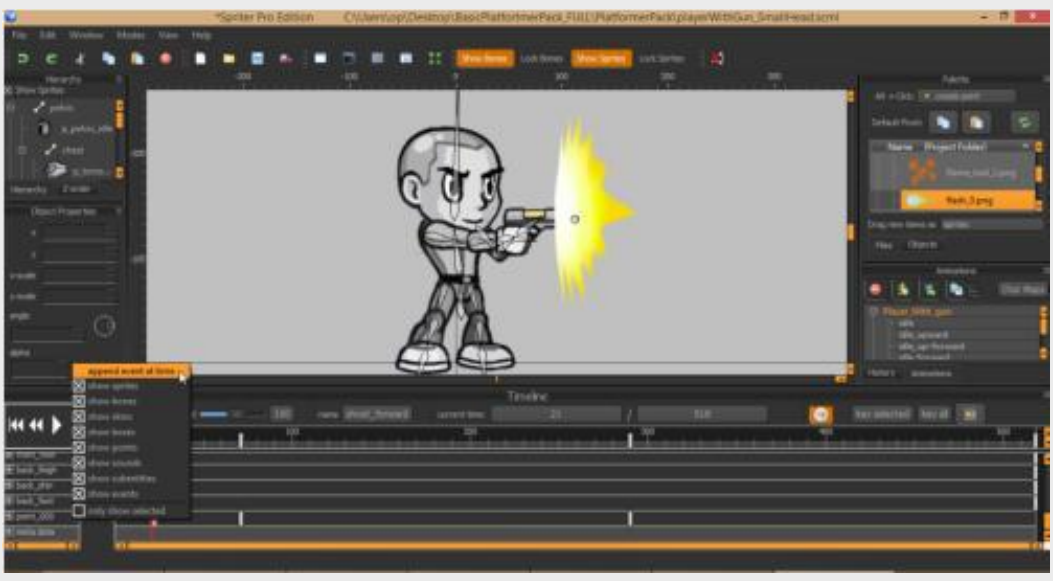




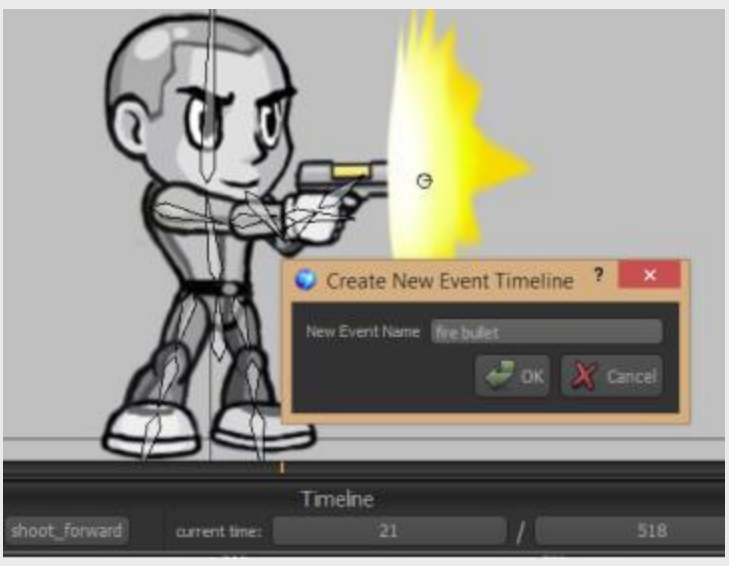
5) For the Blocking animation we would follow the same steps as above... in this case, for the first fraction of a second, the enemy character does not yet have his guard up and should therefore have the "Vulnerable" Tag active, and then from that point on in the animation should not be vulnerable...in other words, make sure from that point on in the metadata timeline that the "Vulnerable" Tag is in the "Available Tags" column and not in the "Active Tags" column.

Adding Event Triggers to an Animation (Pro Only)

There are many instances in a video game where you want an animation to trigger some kind of action within the game engine. In Spriter we call these "Events". A common event you might want to trigger at key points of an animation would be the creation and firing of a bullet sprite at the exact moment of a muzzle flash in a gun firing animation. Of course, it can get much more sophisticated than that. You might want to trigger several events at once, or several over the course of an animation. This is very easy with Spriter Pro. For the sake of keeping these simple we'll use the shooting animation example.



1) Scrub through the timeline to find the exact moment in the shooting animation at which you'd like the game engine to spawn a bullet from the barrel of the gun. (This would be the perfect place to use a "Action Point" in the animation, the let the game engine know the precise position to spawn the bullet from and at what angle to send it flying!) Now that you are at the right Spot in the timeline, right click on the actual left hand part of the expanded timeline palette that displays the names of each time-line and choose the top option that appears, called "append event at time".

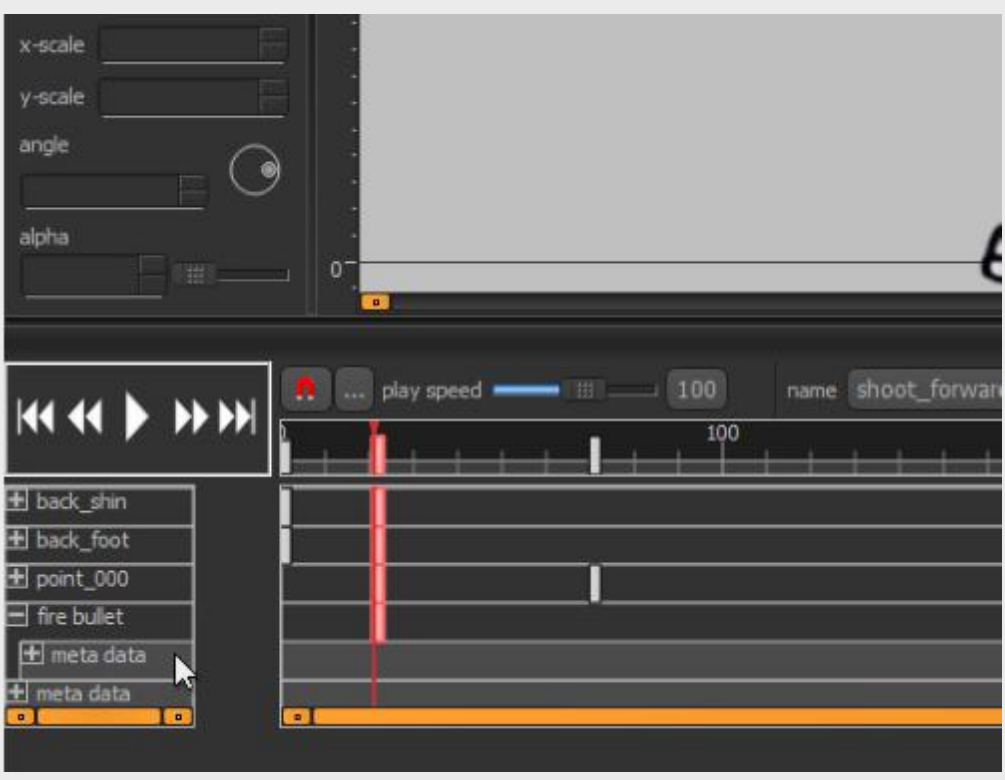


2) A small dialogue will appear which allows you name your Event. Just type in the name you want this event to have and click "OK" or press enter. This places your new event trigger at that spot in the timeline.

The logic in your game engine should look something like this:

If the event called "fire bullet" is triggered.
Then create a bullet sprite at the position of the Action point called "gunbarrell"
And set the angle of the bullet sprite to the angle designated by the Action point.
Set bullet sprite to whatever speed is appropriate.

A more sophisticated example of the use of "Events" would be a long animation of a wizard character casting a spell. Imagine an early event to trigger the start of some lighting and weather effects as the wizard begins chanting his spell. Imagine shortly thereafter another event triggers the beginning of a screen shake...subtle at first, but growing more powerful as the incantation nears it completion. And finally an event timed perfectly when the wizard finishes the spell by slamming the end of his staff to the ground for emphasis, which could trigger a much more violent camera shake, particle effects, universal damage to on-screen enemies, etc etc. The possibilities are endless.



We're not done yet though, if you pay close attention you'll see that the new Event you created has room for it's own MetaData in its timeline! This means the events themselves can carry with them additional information, giving you perfect control of the variables and settings involved with whatever it is you want triggered in the game engine. Please see "adding variables to an animation" and "adding Tags to an animation" to learn more.

Creating a Scaled Clone of a Spriter Project (source images and all)

Unless your needs are quite high resolution to begin with, it's often a good idea to create your Spriter Project at a larger scale than the final result needed for your game. For example, if you are animating a character for your game who will stand at roughly 128 pixels tall in the final game, you might want to make your Spriter project and art so that your character stands at 256 pixels tall inside Spriter. (Note: This is not the case if you're specifically making a retro style pixel art game, in which case, you should work at exactly the same scale as the finished game will need.)

Here's why:

Working at a larger size means you can work faster and sloppier when creating your artwork in the first place, because when you eventually scale it to 50 percent of the full size, it will look much cleaner.

Working at a larger scale also means if you are creating a game where the camera can zoom in or out, then when the camera zooms in your character wont become pixelated or blurry. (until or unless the camera zoom exceeds the size of the actual character art.)

If you work at the larger size you'll have it to use for marketing art or a future, high resolution sequel should the need ever arise.

What to be careful of:

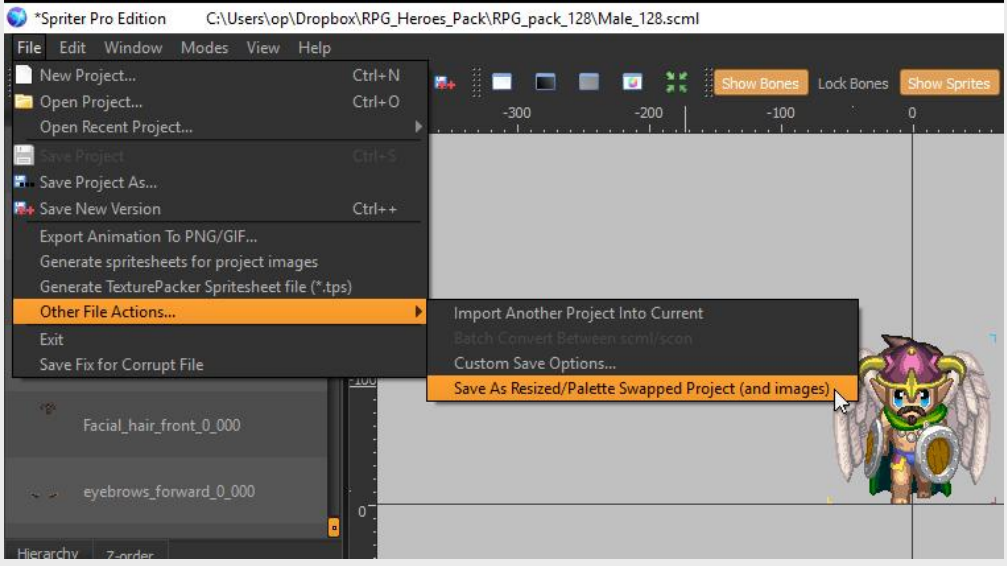
If your game is already going to need large, high resolution art, made of many images in the hundreds or thousand plus pixels in height or width, You should avoid working at a larger scale as you'll likely run into performance issues or even crashes if the number and size of the images being used exceeds the available graphics memory.

Whether you are in need to enlarge or reduce your Spriter project, Spriter Pro makes this easy.

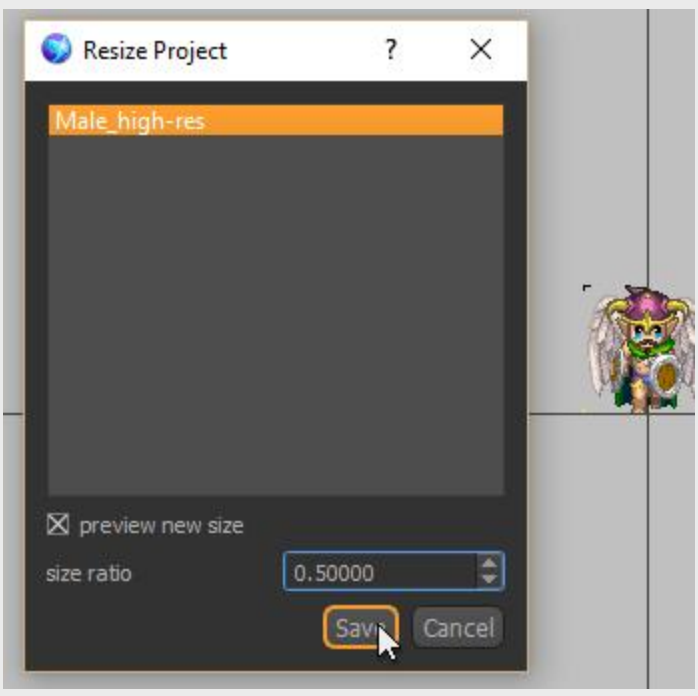
Here's how:

1) BACK UP YOUR ENTIRE SPRITER PROJECT! This is true for any creative person using any software to create data they want to keep. We highly recommend you use a back-up system with version control. This way you can always revert back to any older version of your project in case of an emergency.

2) Choose File|Other File Actions|Save as Resized|Palette Swapped Project (and images) from Spriter's menu. Note: If your project uses indexed color images this process will result in the images in the cloned copy losing their indexed color mode and they will no longer work with color palette customizations, so be sure to keep the original Spriter project backed up so you can always retrieve the original images whenever needed.



3) The "Resize Project" Dialogue will appear. Change the size ratio as you'd like (1.0 is full size, 0.5 is half size, 2.0 is full size etc.) Then click "Save"



4) Be sure to guide the file dialogue to a new folder of choice where the scaled project will be saved (create one if need be) and then choose save. Spriter will do its thing, and the next time you check in your designated folder there will be a complete scaled clone of your Spriter project.

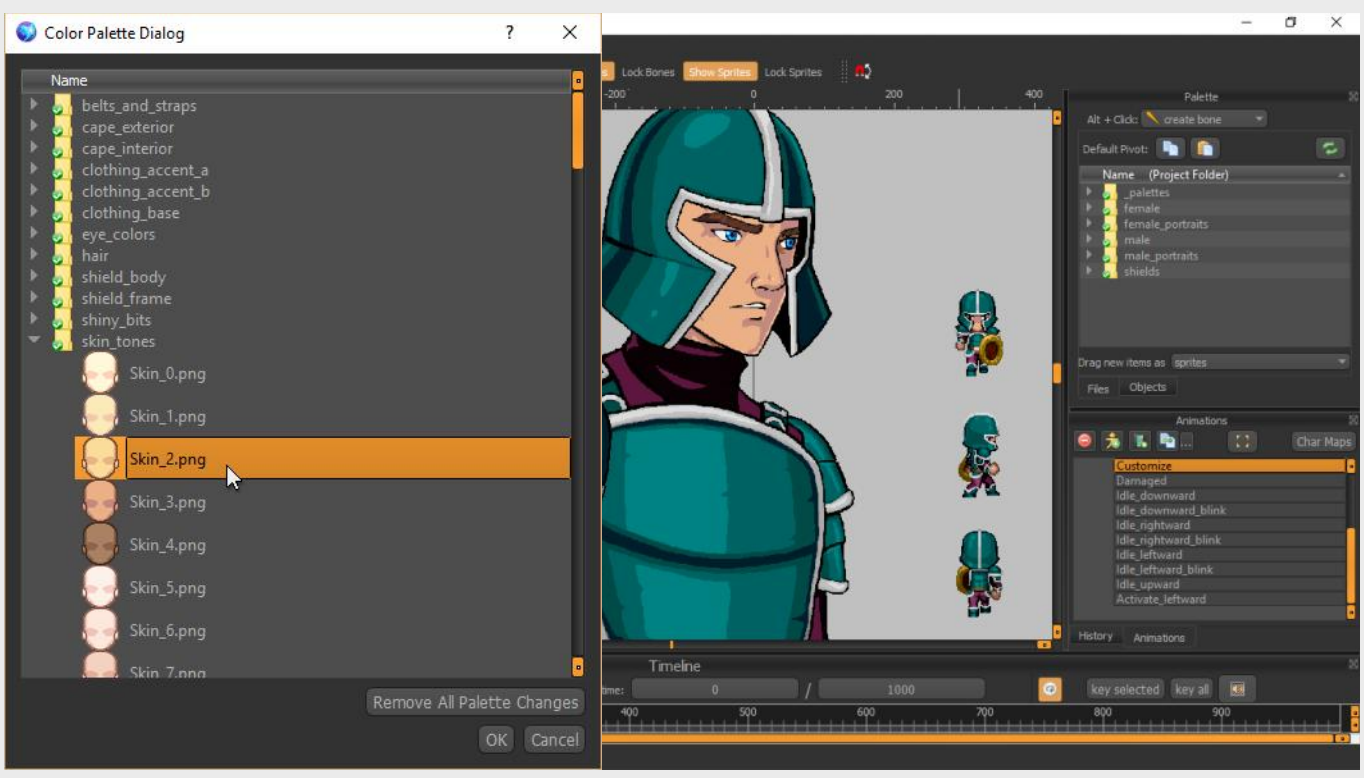
Creating Color Customized Clones Of your Project

This is a highly specialized feature for a very specific set of needs. It is only usable with Spriter projects which use indexed color images and the color palette customization feature covered in this section.

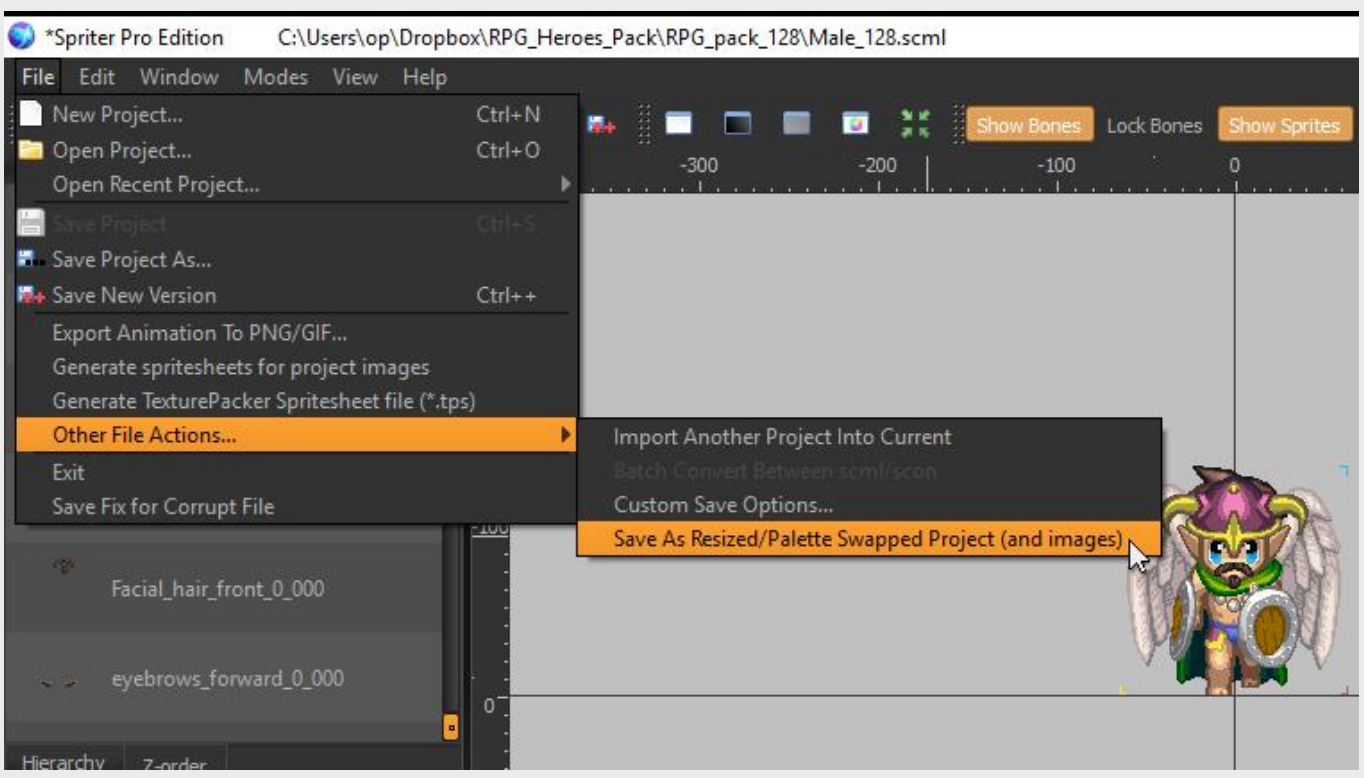
Let's say you are using an Art Pack that allows color customization like our RPG Heroes Art Pack, or you've created an art pack of your own with the same palette manipulation features and now you want to export the character you've just created as a Spriter project that has the custom colors "baked in" to the images used to make the character. You would need to do this is if you want to use the Spriter version of the custom character you just created and not just exported sprite-sheets in the actual game. This is because (at least at the time this manual was written) No Spriter run-time supports custom palette manipulation at run-time.

Here's how you'd do it:

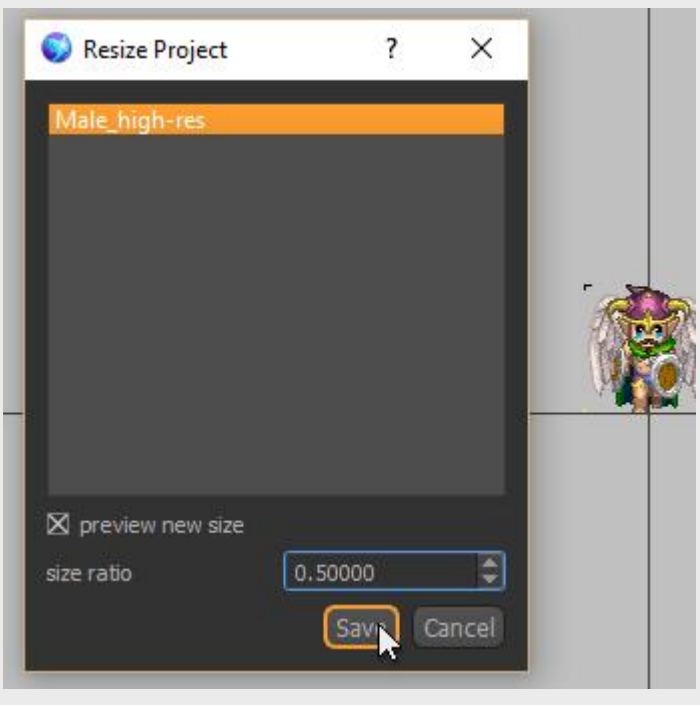
1) Finish setting up your custom color palette to arrive at the desired look. Or load a character file which includes custom color settings so that it visually updates your project in Spriter to the colors you'd like the cloned project to use.



2) Choose "File|Other File Options|Save as Resized|Palette Swapped Project (and images)"
This will bring up the "Resize Project" Dialogue.



3) If you also want to change the size of your animation, enter a new scale, or leave it at 1.0 if you do not want to change the size of the animation. Then click "Save" and direct the file dialogue that appears to the desired folder and filename for your new Spriter Project.

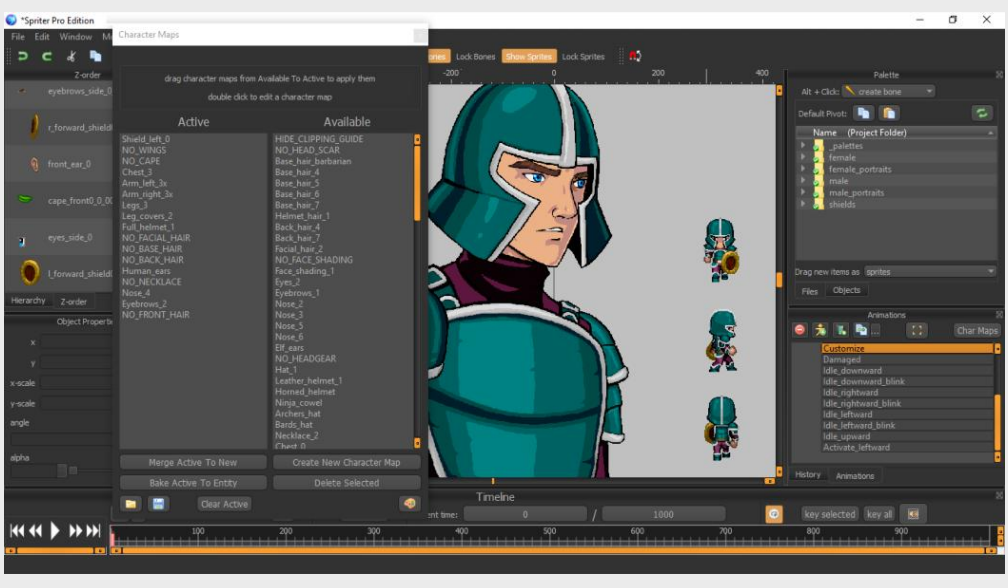


When you check inside the target folder you should see your new project clone, complete with images which now use the custom color settings you had created.

IMPORTANT! The new images, while looking the same, but with your new colors, will be non-indexed images, and would no longer work with the custom palette system, so ALWAYS keep the original project with the indexed color images backed up in case you ever need it to try new color combinations.

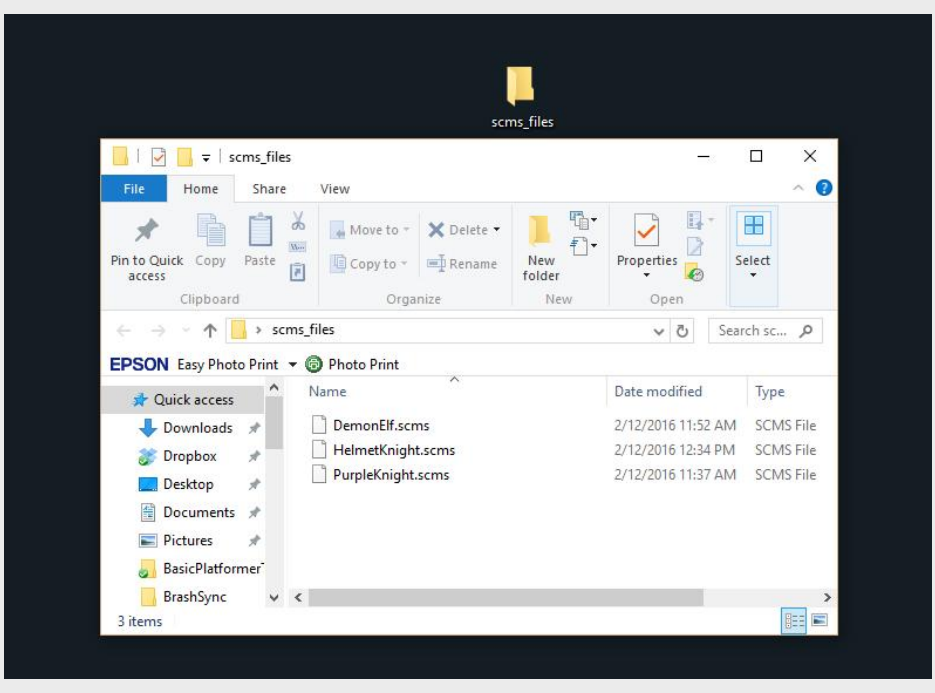
Batch Exporting Animations From Character Files

If you've created a variety of visual variations of your animations via character maps and/or custom palette settings and then saved each character variant as a character file (scms), and you want to export sequential images, GIF animations or sprite sheets of your animations for each of these, there's an automated method to have Spriter do this for you.

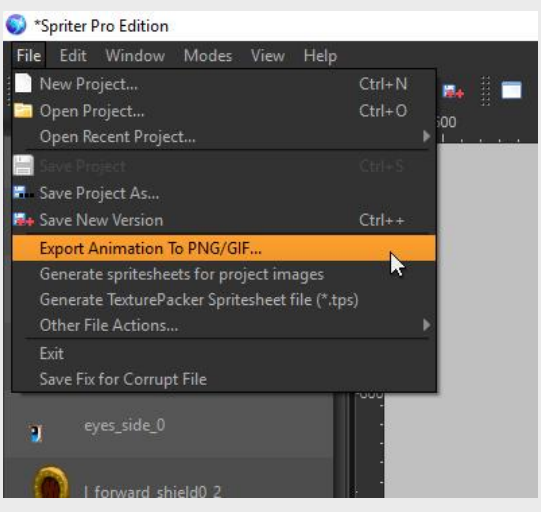


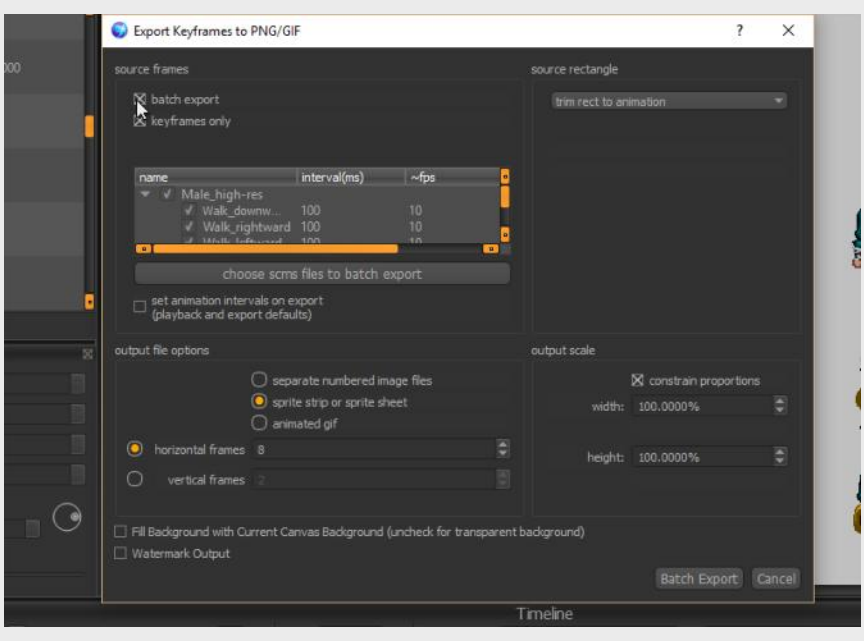
Here's how to do it:

1) Make sure all of your character (.scms) files are in one folder.



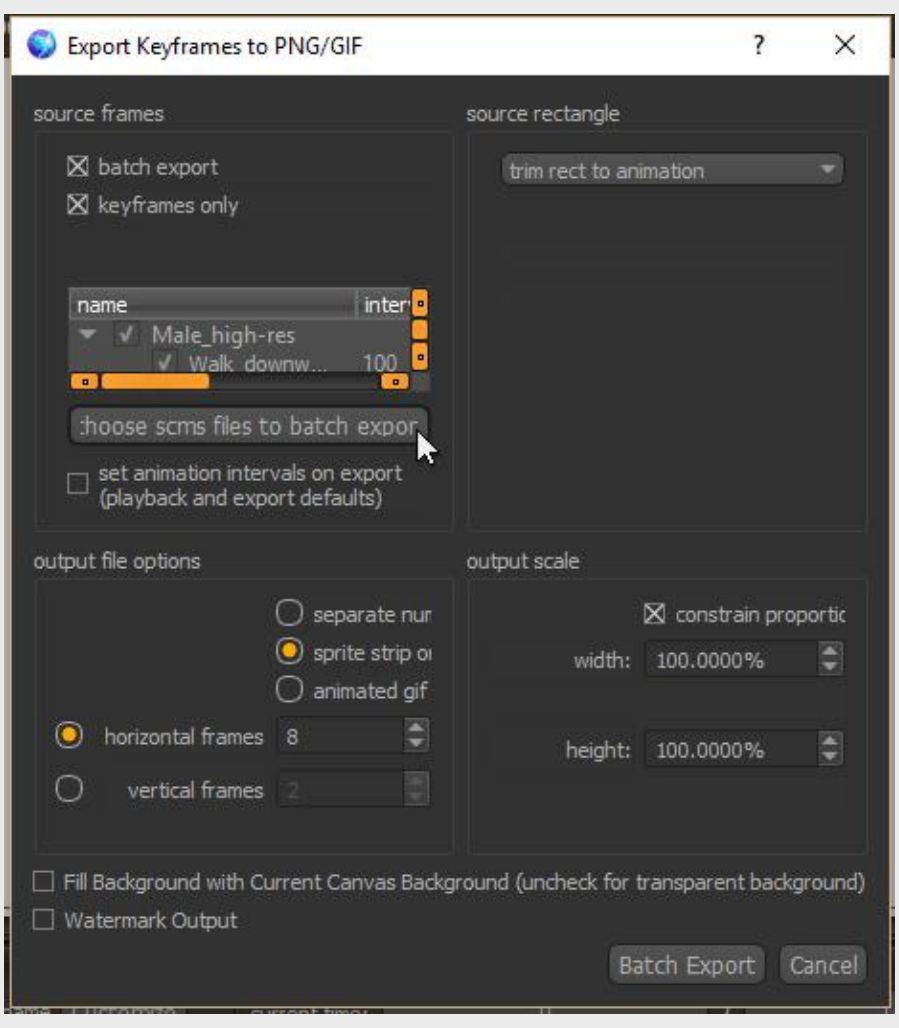
2) Choose "File/Export Animation to PNG/GIF. This will bring up the "Export Keyframes to PNG/GIF" dialogue. Check the batch export checkbox at the top left of the dialogue in the "source frames" section.



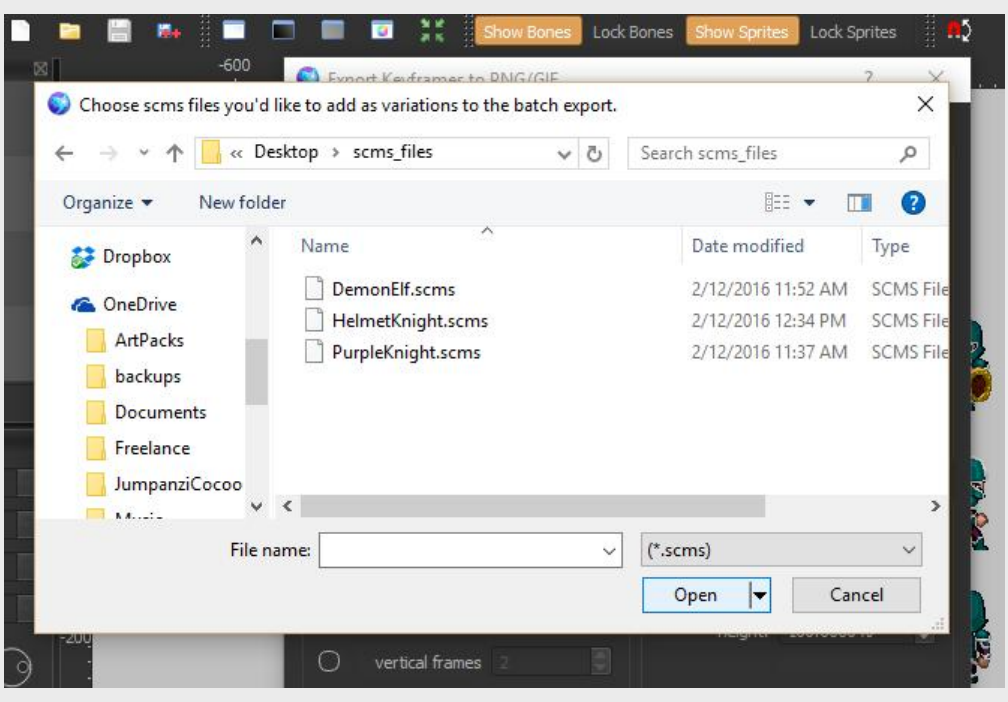


3) Set all options as needed, to designate the required cropping, FPS, scale, file format etc. as covered in our "Exporting Animations as Images or GIF's" section.

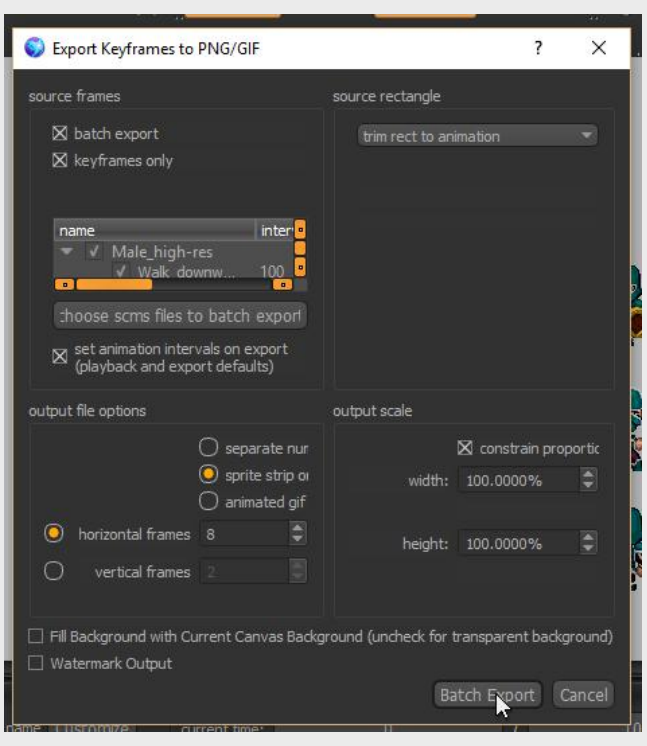
4) Now click the "Choose scms files to batch export" button toward the bottom of the "source frames' section of the "Export Keyframes to PNG/GIF" dialogue.

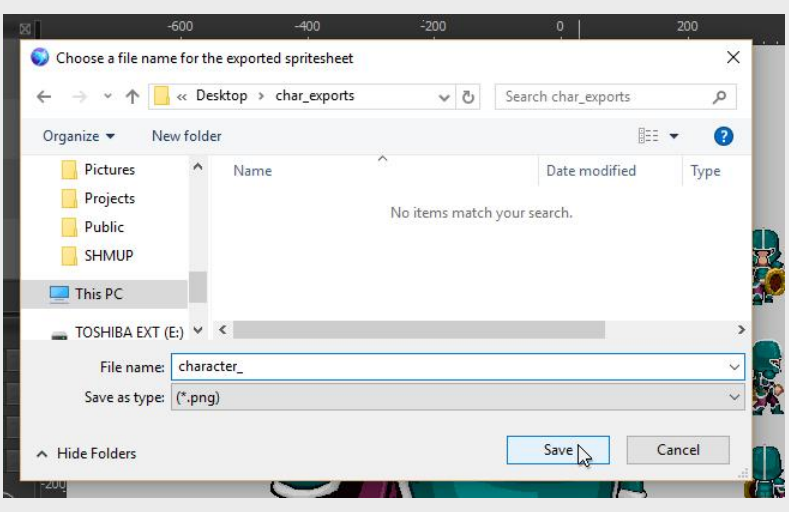


5) Use the File selector that appears to multi-select any of the scms files you'd like to have exported according to the settings you'd made in step 3. Then click "Open"

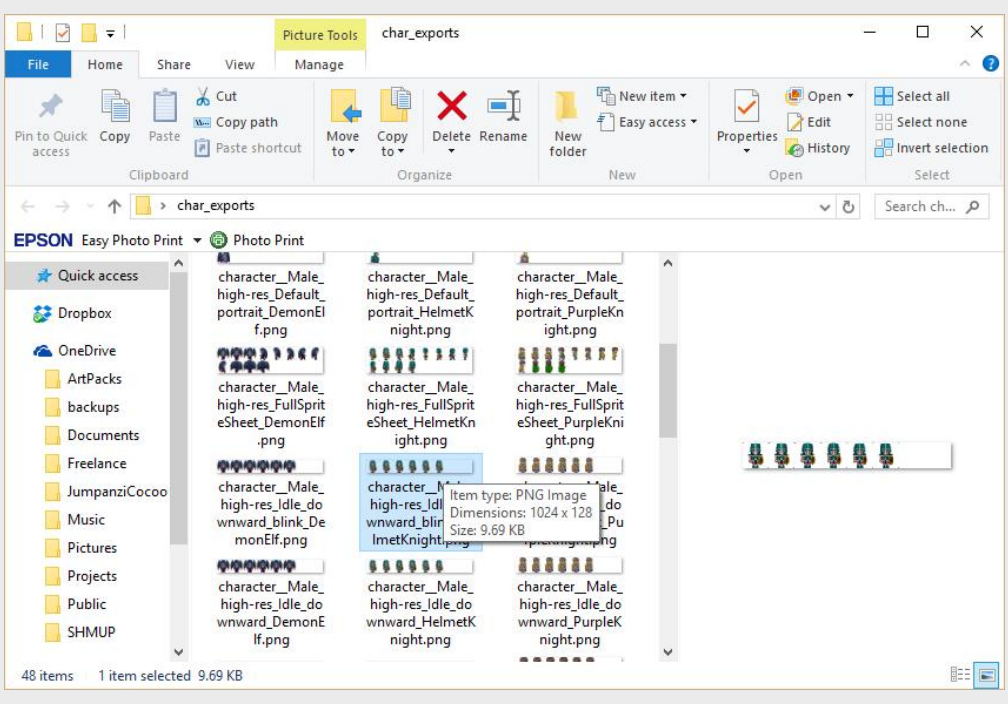


6) Next, click the "Batch Export" button. This will bring up another file selector. Choose the folder and base file-name you'd like the exported PNG's, GIF's or Sprite sheets to be saved to. Then click Save.





7) When you check the folder you designated you should find all your exported animations in whatever format you had set.



Importing One Spriter Project Into Another (Pro Only)

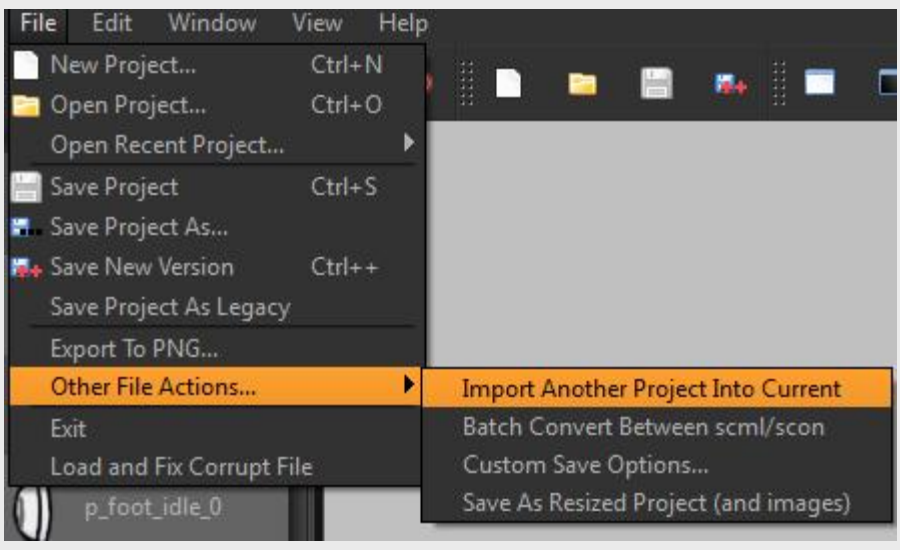
For large projects multiple artists might be creating animations simultaneously. In cases like this, trying to coordinate the sharing and working on only one Spriter Project would prove inconvenient to say the least.

For this reason, Spriter has a feature which allows for each artist to create their animations in their own independent Spriter Projects which can then be merged into a single Spriter project.

WARNING: This is an advanced feature which causes permanent changes to scml or scon files as well as copying image files and folders into the main folder of the target Spriter Project. For this reason you should ALWAYS make backup copies of all Spriter Projects before using this feature!

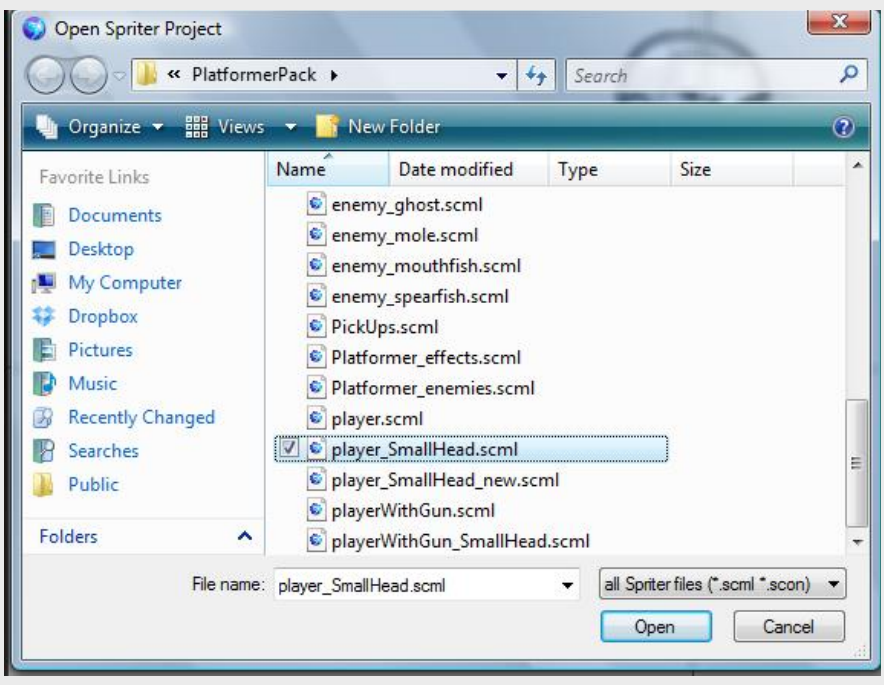
Also, while in the process of backing up its important to note that for Spriter to merge two projects, they must start completely separately...meaning in separate main folders. You can not merge Spriter files (scml or scon) which are already in the same main project folder....so If you need to merge two such Spriter files, then you'll first need to make 2 copies of the entire folder which includes both Spriter files.

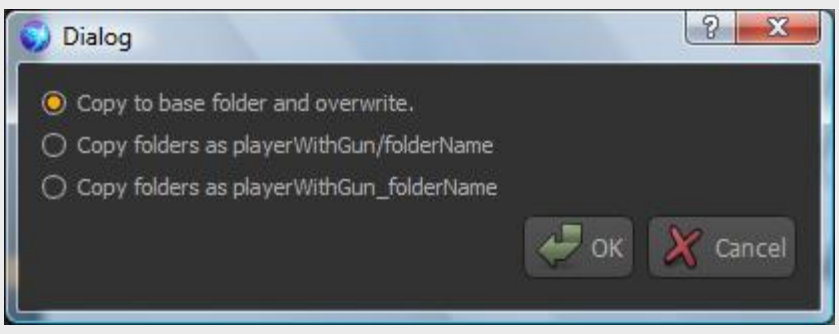
Now that you've backed up the Spriter projects, here are the steps to merge one Spriter project into another:



1) Start up Spriter and load in the Spriter file you'd like to merge the second file into. Then from Spriter's menu, choose: File/Other File Actions/Import Another Project Into Current.

2) The File selector dialogue will appear. Use it to navigate to and select the Spriter file (scml or scon) which you'd like to merge with the currently opened one.





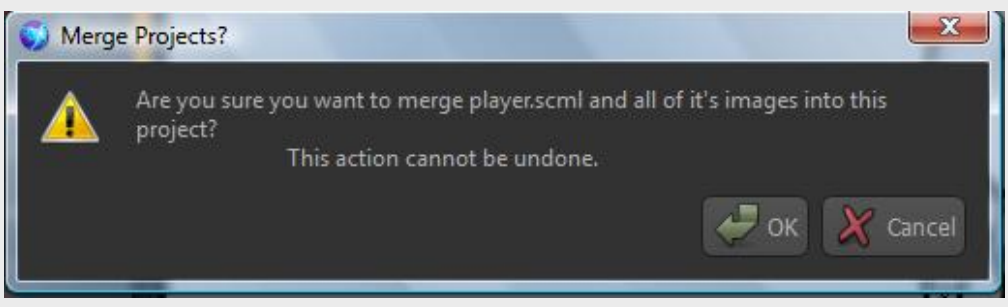
3) A new dialogue box will appear letting you choose one of three different merging options. These pertain to how images and their folders will be dealt with during the merge.

Copy to base and overwrite) The first option will copy everything over to the main project folder of the currently opened file without renaming any image files or folders. This option is most useful when multiple artists might be using some of the same images from the same image folders to make animations...such as using some of the same effects images, or each actually creating different animations for the same character. **Be very careful** when working like this...to make sure no one will lose their own work, its best practice to adopt a file naming convention where each image and animation that an artist creates will start or end with the initials of that artist. This would insure that if multiple artists end up creating images with would otherwise have been given the same name, that one won't overwrite the other when being merged...because for example one image would be named MP_head_0 and the other EM_head_0.

Copy folders as *name of file being imported/foldername) This option will safely create a single folder in the main folder of the Spriter file currently opened and copy all of the image folders and files of the Spriter project being imported into this new folder...thus ensuring that none of its image folders or files will overwrite any belonging by the currently opened file.

Copy folders as *name of file being imported_foldername) This option will automatically change the name of all image folders being imported with the name of the file being imported as the prefix...this would insure that no images from the currently opened file could possibly be overwritten. This option is best if you're merging Spriter files which don't share any images in the first place.

NOTE: While the second two merging options are very safe, they can be wasteful if each of the projects being merged share identical images..because once merged using the last two options, they won't be sharing the same image files, they'll be separately using identical looking but independent files. Safe, but wasteful... so if all artists are animating using many of the same images, its recommended you use the first option, BUT to have named your image files and folders carefully so no one will lose any of their work.



4) Once you've selected your merging method, a final dialogue will appear giving you the option to continue or cancel. If you're ready to merge, click OK and the file you selected will merge into the currently opened one. When the process is finished you should see all entities from the file you selected appear at the bottom of the entities list in the currently opened file.

Mouse Controls and Shortcut Keys

Alt + (Drag) Left Mouse Button = Create & Transform Bones (or Action points or collision rectangles)

(Hold) B (While Bone is Selected) = Select Toggle Bone Children

(Hold) M + (Drag) Left Mouse Button = Move 0,0 coordinate (frame offset) of entire animation.

Left click an already selected Bone= View Resize Controls for Bones

(Drag) Middle Mouse Button or (Hold) Spacebar + (Drag) Left Mouse Button = Pan View in Canvas

(Hold) Ctrl + Mouse Wheel (While in Work Area) = Quick Zoom in Canvas

Mouse Wheel (without holding Ctrl while in work area) = Normal Zoom in Canvas

(Hold) Ctrl + Mouse Wheel (While in Timeline/Hierarchy/Z-order/File Palette/Object Palette) = Zoom

Ctrl + Space (In) & Ctrl + Alt + Space (Out) = Zoom (for when using a Stylus, Trackball, or Mouse w/o Wheel)

Ctrl + Alt +Shift+ Space = Restore Zoom to 100%

Ctrl + Up/Down (While a Single Sprite Selected) = Move Selected Sprite Up/Down One Position in the Z-order

Ctrl + Left/Right (While a Single Sprite Selected) = Move Selected Sprite to the Bottom/Top of the Z-order

Control + C = Copy selected item or currently selected data

Control + Shift + C = Copy an entire frame (Even if tweened between key frames)

Control + Shift + V = Paste to all frames (If objects are parented on the copied frame, they will paste to the parents on the other frames)

Control + D = copy selected object and paste it to all key frames.

Shift + Delete key = Delete selected objects from all frames

(Hold) Ctrl (While you Drag, Rotate, Resize, or Apply IK to an Object or Group of Objects) = Clone (Clones will Retain Relative Parenting & Z-order to one another & will be Parented to the Original Parents if they were not Cloned)

Z = Select all Descendent Sprites and Bones of the Current Selection (i.e. Select Shoulder Bone, Tap Z, it will Select the Upper-arm Bone, and Forearm Bone, and All Sprites for All 3 Bones)

Shift + Z = Select all Ancestor Sprites and Bones of the Current Selection (i.e. Select hand Bone, Press Shift+Z, it will Select the Forearm Bone, Upper-arm Bone, and All Sprites for All 3 Bones)

Arrow Keys (While Object is Selected) = Nudge Object 1 Pixel in Pressed Direction

1 = Go to Previous key frame

Shift + 1 = Go to the previous key frame which contains a key for the currently selected object.

2 = Go to next key frame

Shift + 2 = Go to the next key frame which contains a key for the currently selected object.

Control + 1 = go back in the timeline by 1 millisecond (If timeline snapping is NOT active)

Control + 2= go forward in the timeline by 1 millisecond (If timeline snapping is NOT active)

Control + 1 = go back in the timeline to the previous snap interval (If timeline snapping is active)

Control + 2= go forward in the time to the next snap interval (If timeline snapping is active)

3 = Rewind to Start

4 = Play

5 = Fast Forward to End

7 = Toggle Show Bones

8 = Toggle Lock Bones

9 = Toggle Show Sprites

0 = Toggle Lock Sprites

Acknowledgements

BrashMonkey would like to extend its most humble appreciation to everyone who has supported Spriter and the following people who helped make Spriter a reality by backing our Kickstarter campaign:

Andrew Wooldridge
Simón Ochsenius
Knut J Håland
Gavan Woolery
Leandro Martins
Tadej Gregorcic
newt
T.L. McCabe
Seth Brown
Thiago Escudeiro Craveiro
Dan Fessler
Ian "LobsterSundew" Kragh
Rahul Gupta
Treid
Mutant Sparrow
Tom Gullen
Sam Crisp
Brendan Frye
Kristian Bauer
Mark Brown
Jonny Ree
Nathan Williford
James Green
Jake Vander Ende
Noble Kale
Tyriq Plummer
Frederic Dreuilhe
Tina Ng
Manuel Vögele
Dan Dias
Joel Nyström
Jeroen Ruigrok van der Werven
Marc Robert / Trebor777
Ted -Pragma- Barlas
Robert Aguilar
@MarkedOne
marzsman
Ryan Maloney
Krin
Adam Pliska
Jeff Slutter
Bryan Winters
Ashley Gullen
Patrick Hogan
Kaneda Sajfar
Jason P. Kaplan
Daniel Kaplan
Julian Spillane
Andrew Carvalho
Kevin Fanning
Zachary G. Wright
David "Jellybit" Freeman
Alex
Matthew Hewitson
Miguel Sternberg-Spooky Squid Games
Bijan Beglari
Craig Perko
Hadley Daniels
David Amador
whoTHEdog
Kevin Munn
Simon jensen
Nicolás Viegas Palermo
Eric Miron
Ricky Lai
David Jones
John Nesky
Rob Storm
Ray Wenderlich
Finn Spencer
Anchel Labena
Cody Church
junkboy
Scott Robert Lawrence
Nathan Cole
Josh Loe
Rafael Gaino
Wesley Noble (Wes1180)
Geek & Dad™
c++
James Hofmann
Benjamin Bateman
Shane Langnes
mr_fab
Daniel Orellana
JScott
twoFly
Jesse Chounard
Matt Rix
Scott Franks

George S. Blott
Juan Fornos
Tomasz "Fanotherpg" Kaczmarek
zac-interactive
Mike Parent
Kelly Weaver
Henry Kropf
Guilherme Töws
Elliot Trinidad
Michael Stockwell
John Butkus
Jerry Mickle
Michael Indyk
Damian Sinclair
Alec Thomson
Gregory Gonzalez
Jonathan Younger
Austin Roush
Miko Kuta
Alex Bolinder
abitofcode
Alex Sink
PandaChuZero
Félix Martín Arrieta
Jim Crawford
Matt H.
Slycrel
Peter McAtominey
Martin E. Labanic
Thomas Hajcak
Jessica Gehrer
Robert Haddon
John Akerson
Kazekai
Nibiru Studios
Matthew Sherman
Kitwana Akil
Emma & Fredrik
Toolism
Kellen Lutx
Kevin Lee
Nicholas Bernardi
Pod Cubed
Benedict Apuna
David Hooks
Christian L
Ben Morris
William C Crawford
Thomas "c-Row" Touzimsky
Michael Serens
Matt Perrin
Julio Iglesias
Bob
Nick Pattison
Joseph Neuman
Bradley Robinson
Brian Nicolucci
TenTonToon
Tim Scheiman
The Todd!
Michael Foy
Gabriel
Chris Millin
Heath Marks
Sorin Stefan Nicolin
Sean Nicholls
Benny Balmaceda
Christian Fisch
Jetro Lauha
Brennan Sarich
Tracy Valleau
Derek Elder
Louai Abu-Osba
Sergi Marcet
Arthur Ward Jr
Chris Zamanillo
John W. Marsden
Blake Maloof
Bowling Pin
SketchDeluxe Rekinkiev
sebasong
Kyle Y
John Norman
Bo Li
Matt Laurig
Charles M.
Stefan Hekele
Daniel Korner
Nascent Games
Nadine Schaeffer

"none"
@netsrak
Corey H Philips
SPX
Torrence Davis
Studio Kontrabida
co_Opernicus
Philippe Chabot
Jonathan Rowe
RobCob
Gauss
David Boucher
Joel Marine
Alister McConnell
Chad Joan
Eric McQuiggan
The Construct 2 plugin
Paul Tarr
Vheela
Corey Nolan
Stew Shearer
Sara G. - TwoBitArt
jarda128
raist.cz
Richard@TotalMonkery
Steve Wetherill
Zaidin '0z' Amiot
Martins Zeme
Tomas Motosfacebook.com/PepilloGrito
Luis@newgrounds
Kai Korpi
Cyrilku
Carlos Leituga
Hexxagonal
tjhei
Hathway
John Descheneau
Andrew S. Stamps
Eric Roberts
Matthew Scott
Janita Puska
Stefan Schäfers
Glenn Corpes
Logan
Hector G. Robles
tsernobyl
dragonchasers.com
Gordon Cranford
Alpha 404
George Davison
Pablo Iglesias
Daniel Baumartz
Shane Neville
PixolPalette
Maxime GUY
Dylan 'BSRaven' Denwood
mrkoala
Nathan "SoreThumb" B
Space Captain Steve
Jyri "SkaiWay" Honkanen
Exato Game Studios
Adam Prack
Raph D'Amico
Junky Rhodes
Joris "Alvaron" Pyl
Jeffrey Ates
Timesuck
James
Samuel Lopez De Victoria
Ben McCormick
Laurent Curtat
Philippe Back
Julian Moschüring
Corinne Cadalin
Wesley Kerr
Marc
Jordan Browne
Nasser Elsamadisy
FJPoort
Marcin 'Vigrid' Seredynski
Peter Dijkstra
Tomi Jurvanen
Oli Coombs
Vorvek
Rene
Gareth L Stonebraker
charaldooo
Djordje Ungar
Stephen Orlando
Chad Pfister

Patrick Barrett
Chris Harback
Elliot Harris
Dr Victory
Alex "TwilightVulpine" Carvalho
Niko R
Father Octopus
James Graham
Christopher Hamilton
Christopher "DarkWolfNine" Muzatko
Maxime Parent
Robinson Taylor
bburbank
Silvio César Lizana Terra
Alex Byrom
Andrew Eiche
ironhive
Chad Armstrong
Mami Sue
Brian Crockford
Jason McGhee
Magnus Ekse
Raf Janssens
n-Space, Inc.
Sami Marsch
Zózimo Neto
Ian Beveridge
Carl Granberg
Kevin Hart
Frédérick Morin
Jon Larkin
Deren Somsanith
Andrew James Bowen
Maxime Bouchard
Jakob Marczynski
R. Cumberland
Alfred Reinold Baudisch
Billy J. Pomerleau
Marc Schefer
Sean Dayé Gubelman
Julian Lancaster
Stephen Hawkes
Sam Lamont
Jake Gilla
Andrew Sopchockchai
Karl-Johan Nilsson
Mikanuki
Montezuma
Gavin Thornton
Jeff Macalino
Mike Laurence
Colin Walsh
Justin Espedal
Luis Pabon
OcelotVarn
Justin Ma
The Dinosaurs at ArgyleBox.com
Joel Fischer
John Flynn
Claudio Morinaga
Dave Nevala
Christian Fudala
David Erosa
salmonmoose
Arturo Paracuellos
James "Firgof" Woodall
Brittany Avery
Leonardo Millan
Marty Trzpit
Arc
Dan Glastonbury
dunno
Matt Rudder
Chrsitopher Kirby
Agate Studio
Paul "Cancer" Edwards
Rick Dailey
Sami Anttila
Anderson Fabiano
Paulo Silva
James Lane
Matt Fordham
None
Jesse Klug
Phil Peron
Patrick Elliot
Melissa Darkshore
Lasse Minet Larsen
Luke Schneider
Adam Smith

Andrew Bado
Jay Frank
Jonathan Silvestre
Richard W. Boyd, II
Roman Stetsyshin
Toby Deshane
Isadora J. Tang
Cassandra Inglesby
Agustin Herrero
Chris Norman, the big nerd @ inzi.com
Mike Tallon
Matthew Stavola
Liznel Pina
Reese Mitchell
debug
Dennis
Kevin Kofler
Ryan Sheehan
Seth Sterr
Will Hellwarth
Jonathan Cheatham
Jason Pierce
John Bretana
Brian Huqueriza
Rudá Moreira
D#
L.Bignill
Karl Larsaeus
Shawn White
Nikita Nek Dudnik
thePREdiger
Daniel Jeppsson
Craig Fecteau
Thomas Giles
Andrew Macdonald
Missjudged
Cameron
goshki
Joel Barr
Wai Son Wong
@DevourerOfTime
Paul Rizik
Francesco Borg Bonaci
Erica Woolley
"NopeBye DotCom"
Peter Richards
Shadowfox1112
Hyunwoo (Mac) Ko
Mihkel Tael
Michael Kearns
J.S.Saarimaa
Rocky Wilkins
Wolfgang
headwinds studio
Porter Nielsen
Nando Guimarães
Jason Coggins
Robert Megone
Kelvin "blueporing" Marn
Julian Azevedo
Daisako Kunandra
Nick Shanafelt
Frankiesmileshow
Julia Grammer
IceXPR
Jeff Zaroyko
Pita Madgwick
Justin Brown
Brinny Langlois
Pauli 'Dids' Jokela
Flori
Brian Millett
Beth Thomas
LihimSidhe
Max McRae
Wilhansen Joseph Li
Sebastian Wärnström
Brandon Linton
Scorik Egor
Caleb S.
Chris Hudson
egon
benswinden
Markus Lundberg
Michael Gilhespy
Chad Rajski
Matthew Jaquish
Nick Stone
Luiz Marcelo Lopes Costa Junior
Chris Magson

Eric Ronning
Bruno Campagnolo de Paula
Peter J. Benton
Viktor Candolin
Joseph Barnsley
Benedict Fritz
Frogtoss Games, Inc.
Ryan Malm
James Lang
Andrew Cornett
Jesse "Kites" Rascon
Andre Kishimoto
Josh Tsui
Arrrggghhh!
Povl Eller
Eatmybiglazer
Arthur Ostapenko
Pekka Heikkinen
Filipe Silvestrim
Mikko Lähteenmäki
Voodoo Puppy Games
brsbyrk
hello-morphine
Kyle Bramlett
Sean Monahan
Sean Blevins
Johannes P.
Ville Ruusutie
Mike Merrill
Joe O'Reilly
Phil Harvey
Steven "Razlo" Bailey
Eric Freeman
Phil Loyer
Andy .E. Hamm
Alex Bergquist
Andrew 'Mastasurf' Skinner
A Asatryan
Space Time Foam
Angel Sainz
Naveen
James Poag
gil - believe.vg
Marvan Alkufai
Ian Reichert-Watts
Aaron Yee
Moritz Kobitzsch
DrunkenGrognard
Jonathan Weller
Nessy
Chris Mair
None
Pax
Miquel "Fire" Burns
Justin Woodward
Mason Merker
Eric J Lavallee
Cap. Grey Fox
JMSH
Martin "Mcfly" Tapia-Vergara
Joseph Vendlus
Alex Treppass
SVRTheBoy
Vinny D
Eric Boissard
John Chillemi
Sebastian Rothe
Axel
Pixel Polish Games
J. Lopes
Olivier Lamontagne
Zachary Murray
Open Mind Gaming - Jackson and Duncan
Leo Wichtowski
Ryan Paulitschke
Andrés de Pedro Alvarez
Joacim Eldre
Brandon Jones
Matthew Boonstra
Christopher Knowles
Matthew Edelman
Johannes Smidelöv (Coilworks)
Larry T. Hendrix
Christian Finckler
Amy Sundin
Francisco Ochoa Villalpando
Brian Carloni
Sören Schlömer
Nathan Black
Nathan Sanzone-McDowell

Katrina Pawlowski
Calum Spring
Matt Rea
Fat Cat Gameworks
XplosivBadger
Arash John Sammander
Grzesiek Gorecki
Marko Tosic
Stephen Furlani
Jordan Rance
Eric Poulton
Martin Solis
James C Beaver II
Jun Xu
Franz Michael Ressel
Stéphane Chevalier
Rannath
Geo Seven
Chaonic
Trevor Murray
Thomas Haaks
Michael Ey
George Kai Shirai
Poh Keng Jin (KJ)
Coilercard
Luke Costi
Richard Knight
Philip Mangione
Andrew Rabon
Andrew Johnson
Scott Dossey
Derek Lynch
K
Chris Locher
Scott Diggs Underwood
Nadia Cerezo
Orillian
Matthew Matkinstein
Josh Rogers
Richard W
Mike Fish
Halfbot.com
Mattie Brice
Ugly-Toys.com
Shad Gates
Westen Fry
Nicolás Hormazábal
Keith Birkett
Matthew Lussier
Scott Anderson
David Keyworth
George Sealy
Adam Green
Glenn Noy
Loel M. Phelps
MarsupialTurnip
Nan Wang
Greater Good Games
Sebastien Thibault
Richard Lackinger
Willi Schinmeyer
Kelly Robotoson
C. Yarbrough
Nils Marklund
Fredrik Andreasson
Vincent GAULT
Mark Henning
Tuan Vo
Erik Grunsten
Paul "Zolamee" Calderon
Bredon Clay
CaptainSidekick
George Koutsikos
Lantis
Ville Lehvonen
thewizisthe.biz
Anonymous
Eddy Parris
Ido Yehieli
Marcos Riffel
Ujn Hunter
Andrei Petrov
SixHeads Studios
"DurMan" Andy Jones
Ryan Austin
fidgetwidget
etoothy
Avelino Moreno M.
Jordan D Maynard
Matthew Herz

Matthew Ahrens
Shay Pierce
Alec Stamos
Justin Shimp
Marc Wakefield
Juho Kaistinen
David "Demonic_Sonic" Sheerin
Daniel A. Gerard
d a
@Buckets187
Bill Kaifer
Grugin
papercut
mac
Alastair Nottingham
Andrew Darlington
Eric Killen
Martin Grider
Octavio Arango
Rich Lockard
Kyle Romano
Jesse Thomas Steinke
David Orosz
KJ
Andrew G. Crowell (Overkill)
E.Villani
Alphonse Du
Micke
Tyler Forsythe
Robert Heard
Josh Brown
spilth.org
Eduardo Llerena Fernández
Wayne Denier
Jeehyung Lee
Rikard Peterson
Justin Wells
Philip 'MrPhil' Ludington
Timothy Heard
Jocchan
Radoslav Hodnicak
Ted Brown
Nicholas Gunther Scheurich
Viktor Sjölind
Jan-Christoph Wolf
Andrew M Stafford
Justin Loudermilk
Samuel "Zaron X" Boyd
Esteban Blanco
Andy White
Patrick Davis
Martin
Enrique Gavidia
Bayger
Teemu Kupari
Romaing
Steve Kanter
Phillip Chertok
Matt Walsh
Kyle
Jeremy Edward Bauerle
Luciano Santos
Stefan Wagner
Ryan McCabe
Nicholas Cody Johnson
Brian Allgeier
A'ly Lenzi
Erik Harris
Raymond Blocher
Nothing really
Nathan Martz
John Ahboltin
Gregory Shives
Bryan J. Adams
Dalton Roach
Nathan Jensen
Nick.M
Some guy
Mats Andrén
John Doran
Iqonik Entertainment
Michael Quandt
Leander Hasty
Dave Murphy
KiyowaraKush
Whackala Ltd.
Sander Louring
Edouard Albert
Obligator
John Bryce McCubbin

The Blake7
P.Tullmann
Jorge mikei Pereira
Frank Patrick Huhn
Petteri Halonen
Brian-Thomas Rogers
Ryan Henson Creighton
Stephen White
Lars A. Doucet
JP Stringham
acoustickitty.com
David Dion-Paquet
Tim Mensch
Jennifer Dawe (Happy UFO Studios)
Sho Kazahaya
Michelle Britton
Kieren Martin
Greg Vanderbeek
Alexander Callos
TD Ward
luiz eduardo da silva
Sean Bouchard
Andrew Maizels
David Ackbar Reynolds
www.sublimegames.com
Edward Dang
Isak Gjertsen
Bryce Van Dyk
Jon "SovanJedi" Davies
Mark Burvill
Stefan Hinterdorfer
Rob LaPoint
Erica
José Israel Figueroa Angulo
Shaun McKinnon
Scott Enders
Andy Saia
Charlie Fulton
Sam
Devon Scott-Tunkin
Phil Tibitoski, Young Horses, Inc.
James L Anderson
Gerald Kelley
Coyotesama
Michael Falkensteiner
K. Brizo
Alessandro Metta
Clement Wolfram Swennes
Sebastian Rücker
Gert-Jan Verburg
Daniel Kransberg
Martin Brabham (optedoblivion)
Kody Brown
Nicolas "Qiqo" Stephan
Todd Trann
Kurosh F.
Paul Stamp
Mark Brouwers
Dave Salinas
grinliz
Anil V Singh
Cdeadlock
Nezabyte
Johan Garcia
Jeff Matto
0
Chua Kang Ming
Omnillusion
Sean Sicher
CrushCrumble.com
Jakob Medlin
Luke Farinella
Denis Nickoleff
Lucas Lundy
Bobby Robertson
Brad Choate
Robin Wilding
Andrew Dunn
Tony Davis
Thom Hopper
Ville-Joonas Luukkonen
Brian Lee
Atmosfear1968
CW
Uros Katic
André Jacobs
M.H Speilman
Johnathan Burford
Kaiser Sicking
Herve Piton

TapSkill
Ryan Wiemeyer
DoctaJjive
Luke Stephenson
Matthew Lawrence
Tyler Owen
Alex Kerfoot
Diogo "Rishard" de Vasconcellos
Carl Tardif
Banu Adhimuka
Spaceoutking
Gabriel Tenario (Ten4 Games)
Spencer Keller
Marcus Feital
Justin Martiniak
Max Cantor
Thomas O'Connor
Bay
Devon Veldhuis
Jay Margalus
Tyler Knecht
Jessica Blank
IGDSHARE
Christopher "Rippig" Davies
Way
K
Jeremy Wilfinger
Panagiotis Peikidis
Jermzlar
Gus
Luis Del Rio IV & Dany Orm
Mark
Jacob Entzminger
Christopher Lamb
Robert Peacock
Lance Runkle
Fredrik Wangel
sqweek
Faust
Micheal Chan
Stefan Prüss
animatic vision
Dan B
Jimmy Brewster
Ren Kikuchi-Chung
Tim Hoppmann
Timothy Reed
Marcus Vinícius Sousa Leite de Carvalho
Hugo Innes
Eluruel
Jakub Koziol
Jason Church
Rodriguez Stoertebeker
Altoid
Tatham Johnson
csanyk.com
Brian Paine
Mike Knudson
Ben Dornis
Alex Popescu
Rafael Borges Ventura
Ílson Bolzan
Rich "weaponlordzero" Wiatrowski
Laszlo Uveges
Vinicius Menezes
Berkat S Tung
J Bruce
Michael Kosler
P. Dennis Waltman
Charlotte Woolley
Cory "cr0ybot" Hughart
Brian A. Hansen
Jacob Andersson
Subleak - TheLeadOrder.com
Chris Bera
Torsten Hansson
David Helmer
David Hicks
2PM Enterprises, 2pm-games.com
Retronaut42
Manuel Correia
David Rudie
M. Leclercq
Matias Toyama
Peter Davis
Wei-ju Wu
Marcos Djivelekian
Travis Sareault
Josh Stribling
Ian Eldridge

Jesus "Cattlez" Robles
Glynax
Syntactic Sugar Studio (@Bloodyaugust)
Young Wang
Woody
Raymond Paul Holmes
CuberToy
Eliot Lash
duhprey
Anthony Reyes
Gritfish
Eamon Copher
Daniel Åberg
Steven Holding
Javier de la Vega Eder
AJCopland
Lawrence Y. Louie
Johannes Lindhart Borresen
Vladimir
Michael Bacon
Iain Lobb
John Reeser
Siegmund Kruppa
Bruno Mateus
Vinz - LesJeuxVideo.com
Leonardo Broda
Matthias Panholzer
Alexander Wood
Ben Banton
Griffin 'Fushs' Idleman
@ojrac
mjau
Henrietta Rydfalk
Luc Chevarie
Michal Frystacky
Dan "ouij" G
njrmz
Dunham lockhart
Markus Rosse
Aaron Costello
Chris Cleroux
Vicente J. Urrutia Hernandez
-0
Matthew Campbell
Ken Osuna
Jacky Tang
Carlos Paulino
Zac "katori" North
Fabryz
John J. Halko
Robert S Smith
Aaron Maurice Wilson
Caitlin Ellis
Rajiv Patel
gt
Michael Hill
Bill Robinson
Woody5600
K Savich
Jasson McMorris
Kenneth Aas Hansen
Semih Energin
Adam Plumbley
Robert S.
Zebulon Pi
Alex Koti
Bjorn Nielsen
Lucas Harmon
Alexander Schilpp
Andrew Solomon
Conor & Rory Y.
Joel Mayer
Peter 'pstudio' Pedersen
Mike Lee
John Eternal
Wayne Cheng
Christiaan Kras
Elliot Hayward
Tiago Franco
Greg Laszlo
Jason!
Michael Gunn
mandos78
gerry holt
Mike Rentas
James Secunda
Jorey Dixon
Brian Pagnotti
Robert Lunghamer
Patrick Short

Phill Ryu
Ryan "breadicus" Eames
Edward R. Rochenski, Jr.
Bjarte Sebastian Hansen
None
Izzak Beck
BunsenTech
Hemi Ormsby
Michael "Kayin" O'Reilly
Clarke
Derek Pierce
Byron Wright
MaoMao
Arturo Silva
Daniel Ben
Karl Rudd
Keith Allen McDaniel
Andrew Loré
Yang Wong
Richard Jarke
Jason Kapalka
Melamber
Kevin Hamano
Christopher Bruin
Firedroid
David Pacheco
Richard Matey
Arthur Payne
Paul McClintock
John McAuley
Lucien
Jonathan Frawley
Adam Moeller
Bill Nunney
Frank Brosowski
C.Diamond
Per Viggo Bergsvik
Searge
Ashley Niland-Rowe
Tom Lister
David Eggleton
Joshua Jacobson
Jace Poage
Mike Hayes
Justin Peck
Steven Grove
John McCaffrey
Sheng-Haw Huang
wraith808
Jeff Somers
Derrick Schwabe
Alex Falkenberg
PhiForHire
Luc Allenet de Ribemont
Heitor Tashiro Sergent
Ryan Yokley
Cowly Owl
Mike Lisman
Matt Baldo
Jay Phonesavanh
artmdk
Jack Bogdan
1001 Computer
Allen
Nicholas Stockton
MrMikeMcD
Berkley Staite
faust_33
Tony Joseph (Tonytown Studio)
Radu Coc
Andre Elijah
Aruil
Matias Christensen
Aaron Gibbs
DannyeB (RuralLedge)
Bret K Ikehara
Alexei Kozlenok
BINSTO AB
Vincent Tanakas
Joseph .W. Davis
TomG
Brian Hon
Kyle Overby
David Adam Edmund Campion Donovan
Edward Davis
William Kavanagh
Tim Elzinga
Jonathan Biddle
Justin Stauffer
Boyd Trolinger

Jesse Burstyn
Damian Sommer
Michael Mullins
james fraser
Brooks Bishop
Jason Bakker
Syed Ahmad
Pavel Antokolsky aka Zigmar
Dan Phillips
Nick Salonen - ecsos.com
Orlando Fonseca Jr.
Andrew Whitaker
Tone Heighton
Flame Soulis
Dustin Firchow
Thomas Siklich
Doug Juno
Erich Hung-Che Lin
Danny Tamez
dthorpe256
Dan Silvers,Resident Game Designer@Lantana Games
Carolyn Forney
Ben O'Steen
Richard Perrin
Jason Blazkiewicz
Steve Courtney
Reymon Ortez
Lucas "Floko" Thiers
Michel Boutros
Bill Farren
Erik Zarins
Tam Toucan
Calle Englund
Pieterjan Wynendaele
Tom Ronsyn
Jonathan MacAlpine
Eugene Tan
Thomas Jackson
Michael Andryauskas
Laurens S.
Paul Merchant
Kyle Issac Dixon
CyberPixl
Ross Williams
Jonathan Goytia
Jon Ylinen
Helldrik-Fabio
Guzmán Díaz
Alchimia Studios
Bobby Noah
Denis Marion
Peter Lockhart
Guy Bensky
Enrico Mantovi
Ivan Kodjabashev
LeenUyth & Skywilly (GameSideStory.com)
Dexan Creel
Sergio Kossio
Gabriel Harrison
Trianglepants LLC
Evan 'MWK' Gale
Chris Russell
Ole Groenbaek
Juho 'Jetch' Toivonen
bobomud
Steve Barr
BlaqkSheep.com
Isaac Karth
Christiaan Moleman
Cláuvin Erlan
Guillermo Gómez
Yori
Juha Impivaara
Jordan Ault
Michael Flad
John B
Jeffrey Binder
Patrick Palmer
bcdcool
Dave Riganelli
Lance T Hildebrand
Daniel Steger
Aaron Strong
Michal Walczak
Alexander Mikhnin
Michael Barton
Richard Walsh
Ken-NZ

Kristan Alicesun
Claudio Wieland
Robert Rosborg
Madis Valdmann
Nick Harboe Sørensen
Jonathan Dearborn
Marc
Demian Walendorff
MisterKIKI
Cipherpunk
Daniel Krusenbaum
Patrick Hofsommer
Brent Osborne
Preston J Alvarado
Jack Fawcett
Cort Stratton
Israel Rodríguez Nava
James Seow
Derek Rhodehamel
Douglas Haber
Ole Korff
Bruce Markham
Kyle Staves
Luke Hoschke
EgoAnt Productions Inc.
Darren Austin
Jake Gundersen
Cristian Radu
Yannick Loriot
Zizhuang Yang
Casey Young
Singapore-MIT GAMBIT Game Lab
N/A
John Olson
Kav Latiolais
Danny Fritz
Ulrik Flænø Damm
Tam Finlay
Tim James
Phil Teschner
John E Donovan
Jay Johns
Keith "progamer_me" Hall
Peter Bickhofe
Andreas H.
Andy
James Riggall
D-n-S Design Company, LLC
Joey Lapegna
Andrea Adams
Blue Lightning Labs
Jim Sigler
Matt Mechtley
Shane Wegner
Jasmine Choinski
Kasan Wright
Joseph Catrambone
Bob Thulfram
Halley Tran
Mihaly Rendes
Matthew "The Mad Tinkerer" Mather
Jon Aaron Alcantara
Enis Bayramoglu
Nayef Al Baloushi
Stephen Rose
Matt English
Marcus St.Clair
DC O'Steen IV
Ronald Dragstra
Garrett Orth
Matthieu Castan
Sved
Jerome
Haruna Akatsuki
Ivan K. Myers Jr.
Dustin Dimpfl
Osama Dorias
John Daniels
Gary Lim (Glimy)
Fabrice
Long Le
Faide
Jeff Daze
Matthew Pekelny
Dan Tsukasa
Peter Dakessian
Dan Lovell
Michael Kwan
Joel Stewart
Ali Ragheed Alkhiro

Scott Bodenbender
Mike Branham
Ratsark
Kaisirak
Jeffrey Ellak
Sm4rt Game Studios
Quang Tran
Todd Showalter
Bob Allen
Matt Nowina
Shreeps
Lev Grinberg
Matt Graham
Bryan Rathman
Jason Trowbridge
Sean Fausett
Kyle Rudy
Pedro Thiers
Ioannis Siantos
Scot Schneebeli
Jeff Faith
Jubal Slone
Jared & Sarah
Phillip Reagan
christian evangelista
qbix
Kody R. Dillman
Jonathan Kress
Erick Tinajero Garcia
InsertDisk2
Bobjuzz
James V
Hüseyin Uslu
Ahmad Y. Doleh
Connor Daly
Mark Duval
DarkScorpion
David Pare
Nothing :)
d-enabled
Eric Prince
Chris Corbett
rune0119
Steve Boeters
Bernat Bestard
Peter To
Neil
Obinna
Pieter van der Westhuizen
Konstantinos Dimopoulos / Gnome
Jeff Ward
Zach Turner
Jake Moczydlowski
PixelHero
Casey O'Donnell
NeoModulus
Vaughan Smith
Daniel Stradwick
Colin O'Kelley
Chris "Fosterocalypse" Foster
Trevor Peterson
Grandy Peace
Osvald Neonsdotter Georgie Hunstad
RawkJS
Juaquime J. Wright
Joseph Goethals
bchan84
Seth McCauley
Vyse
Dave LeCompte
John Walker Bruce
Will Jennings
Nicholas Puleo
Howard
George Reusser
Timothy Fujimoto
Håvard Stene Skjærvik
Erik R
Daniel P. Shaefer
Matthew Kozachek
Raymund "Darth" Vidar
Ryan Worrell
Sydney Bridges
Max Drzewinski
Josh Rufer
Bobby Amp
Sean Mountcastle
Nicholas Goguen
Cory Kerr
Flavio Buccioni

Nicklas Lindgren
John Ingalls
The Haad
yutt
Brad Herman
Matthew Humphries
Disk1of5
Markus "Maverick" Seidl
rabidcow
Daniel Eichling
MikauSchekzen
Ankh
Pat Duffy
Rafael C. Pinto
Jan Willem
Herman Duyker
Gavin Bowman
Glenn Meyer
Rafael Gatti
Gutem
Kristin Carlson
Lou
Matthew Walusek
Joseph Austin
Roberto Rivera
Michele Colombo
Alex Fernandez
wasigh
Nick Reynolds
Joe Trueman
Timmux
Sadhu
Sebastien Frippiat
Malte Euler
Manuel Giesa
Andrew Burton
Michael A. Rogers
David Hewson
Andrzej Pietras
A.K.G.
Sergey Lapin
Christian Maher
Ray Jessup
Andrew Frye
John Brewer
Les Fletcher
Andrew Nielsen
Mitchell Green - TheGreenDawg
Chad Smith
Michael Fogh Kristensen
Yulay Devlet
Jimmy "Kraftwurm" Gee
Steven Diaz Smith
Andrew Sallwasser
Michael Carriere
Adam Rademacher
Schell Scivally
Kimberly Voll
Blake Schwendiman
Perceptive Pumpkin Productions, LLC
Michael Tervoort
Charles Valentine Stuard III
Ryan Roper
Link Hughes
Firas Kadhum
Fissure Game Development Club
Jose Pablo "Loko" Monge
Venetia MacGyver
Ryder Boyton
Darius Reginis
Jonathan Jacobs
Einherjar
Matthew Armarego
HAFIZ MOHD ROZLAN
Sharad Cornejo
Zachary Knight, Co-Founder Divine Knight Gaming
Joshua Buergel
dncc
Logan Stevenson
Gordon Luk
Eric Liga
Steve "Pokey" Dupuis
Josh Haycraft
Patrik Andersson
Asminthe
David Lee
A Backer of Spriter
Corv
Peter Laws

Jonathan Willis
Johnathon Timothy Aaron Cummings (JTAC) (Bluemansonic)
Justin Secunda
Christian Runeborg
Emiel Piket
Kleverpig
Simon Larouche
Jack "Segfault" Ramey
LiNSo
Paolo Munoz
markmatamoros.com
tRens
Ahingsaka Samsenesena
thinbody
Alex Parsons
Michael L. Brzezniak
Mark Ludlow
Italo Maia
Tiebreaker Studio
Alesander Simancas
Chris Tart
Steven "FatGuyLaughing" Brown
Chris Worboys
Shawn "binarycrusader" Walker
G33K.ES
Angel Jose Rivera Gonzalez
Madfam Fun, LLC
Gordon Doskas
Warpath
Dorothy P.
SimianLogic
Brandon Barney
Lee
DaCloud804
Mike Desaro
Kevin Bryant
Prasanth and Preethy
Dave Ballantine
Jack Hampton
Steven Campbell
Sacha MILLET
Paul Anguiano
Jeffrey J. Ellen
DX_Blaster
Shay Caron
Paul
Anssi Kolehmainen
Pavel "chif" Belyaev
Nathaniel Huff
Demitrius -FL
Daniel Ridgway & Ronald Flitcraft
Josh Butterworth
Lukas Kuligowski
Kelly Crittenden
Kyatric.com
Erik Rounds
Kyle Pulver
Ranoka
M. Sean Molley
Erik Schulz
Richard Kong Win Chang
Jerry Liu
Andy Keeble
happydroids.com
Ben Wilhelm
Thomas Gorence (producerism)
sebastien verbiese
Chaz Straney
Richard Barnes
Joe Chott II
Tim Borrelli
Photon Storm
IAN BIGGS
Joel Davis
Mike Kasprzak
Andrew Potozniak, founder of TozSoftware
Zachary Lewis (http://zacharylew.is)
anonymousBacker
s.flory
Luke Dixon
JAK Entertainment LLC
Pkeod
Taco Graveyard
Justin Walsh
Lavon Woods
Brett Yamamoto
Jim Vaughn
Afonso Cordeiro

Anthony Wyatt
David Wagner
Don Raúl & Tequila Works
Janet Bagchus
Jason Wehmhoener
Matt Boudreaux
Jared M.
Nicholas R. Grant
Logik State
Alexandre Campos
Abandon Hope Games
Niklas Wahrman
Mark Temple
Alex 'Ceetee' Reid
Q
David Brender
Johnny Tips
k1nave
Greg Basich
Jussi Simpanen, AdventureIslands
Haydxn
Jeremy C.
Lorika
Andrew Carroll
Jesse Langstaff
Data Realms
Ronald Dean Hallman
Greg Smith
Sebastien "deepnight" Benard
Soh Thiam Hing
Victor Thompson
Bluez
Massively Fun
Tyrathect
Drew Diamantoukos
Mark Schmelzenbach
David Silverman
Jasu
Dabou
Valentin Kraevskiy
Ingo Eisenhard
John Cutburth II
NeverNull
S.K. Studios, LLC
BrightWaveGames
Shinjan Dasgupta
Gareth Davies
Laura Richardson
Ben Reeves
Andrey Lavrov
Paul Diaz
Chris Serino
Teresa Manco
Tim Hotston
Philip Cheang / _phi
Neil 'Wex' Barrington
Halvgoeden
BeyondtheTech
Daniel Bauman
Katheryn "RaQin" Phillips
Tayari Alberga
Mike Seppy
Stank
Simon Dowling
Jason Scott
Michael McClenney
Oscar & Charlotte Allen
Josh Carberry
Shannon Daniell-Segilola
Kyle Robert Keezer
Joshua "Candescence" Braico
Panther Page
Box O Zombies
Pekka Kujansuu
Playerthree.com
John "MooseCantTalk" Moody
Jarmo Kukkola
Jared Justus
Joseph Shirley
Adrian Serrano
Guillaume Bergeron
Eric Santana
Gerred Dillon
Tryst Entangled
KarthVader
Aleksa Todorovic
Derek 'Kalakian' Stobbs
Ian Masters
Todd Rieger
Christopher Lott

Cory Bloyd
Andrew O'Neil
Mike Sandercock
Jeff Graves
Onno Scheffers
Ryan Wiley
Stephanie Germain
Moritz Voss
Benjamin E. Green
Andrea B. A.
Scott K.
Joel McCoy
Joe Castro
Bryan Andrew Butler
Magnus Esko
enfu
Caden Redpearl
Marshall Petersen
Codi Marker
Thorsten Fietzek
Casey Lent
Mobile Pie
Jason V. McMaster
Weekend Game Studio
Marcus Bengtsson
Mat Dawam Abas
Brandon Tracey
Brian Taylor
Will Q Nguyen
Cody Estes
Nathan A. Quattrini, Aaron W. Hughes
Benton Redmann
Patrick Kurts
Edward Lewis
Emmanuel A. Simon
Glenn Bacon
Andy Bower
Mike Leahy (EGR Software)
Travis Womack
Jerod M. Bennett
Kimberly Andrews
Blake
Te-Jui D. Chiu
Dean Giberson
Nicholas Beason
Jack Casey
Spencer Kendrick
Leo Lännenmäki
Frits Talbot
Aaron Little
Ron Cox
Roland Ruth
Patrick McCarron
Lorenzo Orselli
Michael Crow
Robert Troughton
Mark H
Marcos de Paula Gonçalves
Yann Granjon
Killerdog Studios
Jens Wiechers
Jason Hamilton
capmar
Harrison Perry
LordTron
Mososh, Inc (www.mososh.com)
Timothy Luke
Philip James
Adam R. "The Zotmeister" Wood
Bryan Hughes
NEBUKO
Christina Louise Warne
Claudia Doppioslash
Chris Whitman
Harida
Melvin "The Blur" Banares
NeonBlue
none
Kuowen Lo
Julien Derveeuw
Neural Echo
Alex Okita
Oded Sharon - Adventure Mob
Rob Corradi
Sven Mohrenstecher
Pen fountain
The Yogscast
Dan Raine
Jonathan Cohen
Andrew JD Neufeld

Frederico Jabulka
Joshua Skelton
Gabor Bukovics
Andrea Giorgio "Muu?" Cerioli
Adrian Browne
Pride St. Clair
Markus Nigrin
Robert McCrady (RenegadeCitizen)
Kayn
Arnold Tsang
Lance Trahan
Sean Warton
Biff Bird
Renita Orellana
Aleksei "skt" Shevchenko
Brian Bucklew
Daniel Fearon
Plato
James Holmes
zackery turner
Jeff Hangartner
JohnnySix
Chane Hollander
Martin Famiglietti
Abilio Carvalho
Nelson "LastAndroid" Beers
The Engine Company
David R
Jonathan S. Collins Leon
StuckPixel
James V. Olson
Tim Partridge
Nick Coombe @ Get Set Games
JZweigart
Nathan Rogers
Jean-Marc Nielly
Stephen Miller
D. Moonfire
Markus Eräpolku
Yang Pulse
WhileTrueFork
Ben Grue McGraw
Kimo Boissonnier
Rocco "actraiser" Di Leo
Nathan Corvino
charm-bangle
Santiago Lema - smallte.ch
esDot Studio, Inc.
Charles Randall
Yenni Brusco
Max Woerner Chase
Jarno Lehto
Josh Montoute
Dastyni
Brian Sowers
Jesse S Scott
Dr. Juris
bitmOO