

Harmony22

Toon Boom Harmony22 Gaming Guide



TOON BOOM ANIMATION INC.

4200 St.Laurent Blvd, Suite 1020
Montreal, Quebec, Canada
H2W 2R2

+1 514 278 8666

contact@toonboom.com
toonboom.com

Legal Notices

Toon Boom Animation Inc.
4200 Saint-Laurent, Suite 1020
Montreal, Quebec, Canada
H2W 2R2

Tel: +1 514 278 8666

Fax: +1 514 278 2666

toonboom.com

Disclaimer

The content of this document is the property of Toon Boom Animation Inc. and is copyrighted. Any reproduction in whole or in part is strictly prohibited.

The content of this document is covered by a specific limited warranty and exclusions and limit of liability under the applicable License Agreement as supplemented by the special terms and conditions for Adobe® Flash® File Format (SWF). For details, refer to the License Agreement and to those special terms and conditions.

Some icons in this document were provided with Font Awesome Free 5.6.1 by Font Awesome. These icons are provided under the CC BY 4.0 license. For more information on Font Awesome, visit <https://fontawesome.com>. For information on the license of Font Awesome Free, see <https://fontawesome.com/license/free>.

Some icons in this document were provided with the Glyphicons Halflings font by Glyphicons. For more information on Glyphicons, visit <https://www.glyphicons.com/>.

Trademarks

Toon Boom® is a registered trademark. Harmony™ and the Toon Boom logo are trademarks of Toon Boom Animation Inc. All other trademarks of the property of their respective owners.

Toon Boom® is a registered trademark. The Toon Boom logo is a trademark of Toon Boom Animation Inc. All other trademarks of the property of their respective owners.

Publication Date

02-22-2023

Copyright © 2023 Toon Boom Animation Inc., a Corus Entertainment Inc. company. All rights reserved.

Table of Contents

Table of Contents	3
About Gaming	4
Harmony Gaming SDK 2022 Release Notes	5
Chapter 1: Exporting Harmony Data for Gaming	10
Chapter 2: About Game Asset Creation	13
Game Rigging Guidelines	14
Game Deformation Guidelines	16
About Game Bone Deformations	17
Game Cutter Guidelines	18
Creating Metadata Notes	19
Game Animation Tips	21
About the Orthographic Camera	23
Chapter 3: About Exporting to Unity	24
Setting Anchors	25
Exporting Sprite Sheets	26
Sprite Resolutions	28
Using Bake_Groups	30
Exporting to Easel JS	32
Palette Variations	34
Chapter 4: About the Harmony Unity SDK	36
Harmony Previewer	40
Asset Manipulation	42
About the Sample Unity Project	48
About the Unity Interface	49
Importing Harmony Files into Unity	51
TBG File Workflow in Unity	52
XML Folder Workflow in Unity	59
Adding a Harmony Renderer to Empty Game Objects	63
Setting Up Collisions in Unity	65

About Gaming

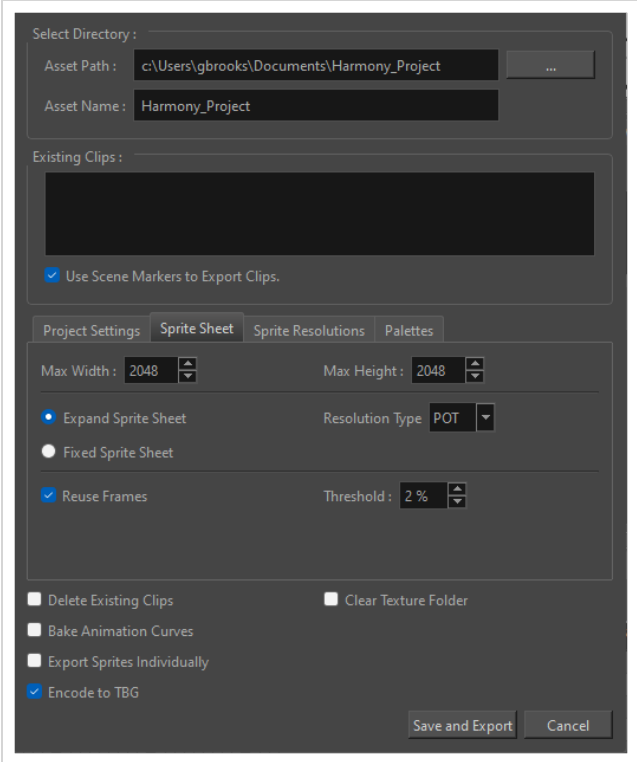
Harmony allows you to export animations created in Harmony to external game engines. This allows you to design characters with simple movements that may be used in a game engine.

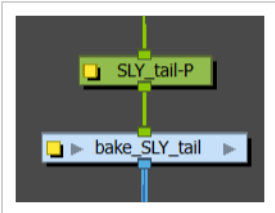
An example of a game engine that you may incorporate Harmony data with is Unity. Toon Boom has a Harmony SDK that you may download from the Unity asset store to have control over your character once imported to Unity.

This guide will demonstrate how to export animations from Harmony. In addition, the Harmony Game SDK for Harmony integration with Unity will also be detailed.

Harmony Gaming SDK 2022 Release Notes

Exporting From Harmony

Feature	Description
<p>New Exporter Options</p>	<p>To facilitate better support within Game Editors, the Export to Sprite Sheet interface has been given some additional features to pack more information into the exported file.</p>  <p>New Options</p> <ul style="list-style-type: none"> • Palette selection tab - Exclude certain palettes that you don't want included in your exported file, this will cover all possible variations of palettes that will result in different colors showing on the character • Encode to TBG - zips the XML files to the TBG custom file format so that they may be imported to Unity with Unity custom importers. When the files are imported, they will be treated like any asset in Unity and will contain sub-assets for textures, sprites, and animations. This improves the ergonomics of creating and moving character data and checking changes into source control. • Improved performance on "Save and Export" - Exporting process generally takes 1/3 of the time compared to Harmony 21, for both XML folders and .tbg files

Feature	Description
"bake_" Groups	<p>There are many nodes that are not possible to render in the Game SDK currently. As a means of bringing in the value of those nodes to your export, there is a new feature to group drawings and the nodes that affect them into a new group with the "bake_" prefix (eg. bake_hips).</p>  <p>Any group named with the "bake_" prefix will be analysed for unique frames of animation for effects, movements, and drawing substitutions. When exported to a spritesheet, each unique frame from this group will export as a sprite. If many frames of the timeline have nothing changed within that group, it will all be considered the same exported sprite in the spritesheet</p>

Importing TBG files in Unity

Feature	Description
Automatic Importing Process	<p>Unity will automatically detect changes to TBG files and re-import them using a custom importer. The file can then be dragged into the scene as a prefab, much like you'd expect from an FBX file.</p> <p>Textures, sprites, animations and TbgStore assets will be generated and provided as sub-assets underneath the main TBG prefab asset. These assets will not have to be managed separately in source control, and only exist in Unity's library cache. An AnimatorController and SpriteSheet asset are generated beside the character, since these are unable to operate as sub-assets.</p>
Importer Settings	<p>The Importing process can be modified from the settings in the inspector view.</p> <p>Animation Settings</p> <ul style="list-style-type: none"> • Discretization Step - The number of columns / rows that sprites will get split into if they are being deformed by bones. • Framerate - The number of frames per second that the animation runs at. • Stepped - Disable interpolation between authored frames.

Feature	Description
	<ul style="list-style-type: none"> • Create Animator Controller - Disable interpolation between authored frames. • AnimatorController - Provides a workspace to transition animations to one another given certain conditions. Can be referenced in gameplay scripts to trigger animations during gameplay. • Maintain Curves On Cloned Clips - Curves duplicated from clip sub-assets inside .tbg file (and referenced in the AnimatorController) will have their curves overridden with new data from the updated .tbg file. <p>Material Settings</p> <ul style="list-style-type: none"> • Shader - Used to render all sprites in the prefab • SRGBTexture - Map gamma color space from textures into Linear color space rendering • FilterMode - How neighboring pixels of the texture interpolate • MipmapEnabled - Generate lower resolution textures to render at further distances • CreateSpriteAtlas - If no SpriteAtlas is referenced below, a new SpriteAtlas asset will be created in the project beside the .tbg file • SpriteAtlas - All sprites from the .tbg file can be added to a SpriteAtlas, when done will be able to combine multiple novel SpriteRenderers into a single drawcall, improving rendering performance

Rendering TBG files in Unity

Feature	Description
Prefab Hierarchy	<p>The structure of the generated prefab in Unity should match the hierarchical structure set up in the Timeline in Harmony. Harmony's Node view structure of composites feeding into the output node are excluded during export.</p> <p>New GameObjects can be attached to any of these child peg transforms. This allows procedural equipment attachment.</p> <p>Also, any GameObject can be enabled or disabled at runtime, so optional visual additions on characters can be hidden until they're needed. The hierarchy could also be entirely decomposed, restructured, or duplicated at runtime. This allows the option to explode characters or shoot certain drawings as projectiles.</p>

Feature	Description
TBG Renderer Settings	The resolution, palette, material, color, and skins can be edited from the Inspector as well as the GameObject's custom scripts. The material and color for SpriteRenderers can also be set individually per drawing, so you can change the colors of a character's accessories or provide special effects to some drawings through custom shaders.
2D Animation Package Integration	Drawings deformed by GameBones in Harmony will be translated into SpriteSkin bones in Unity using Unity's "2D Animation Package". This package provides support for multi-threaded high-performance mesh deformation for SpriteRenderers as well as UI handles for moving bones in the Scene view. The bone chain can be procedurally animated using Unity's "IK Manager 2D" for blending in aiming, leaning, jiggling, waving effects, etc. after the animation Update step.
Animations	<p>All animation data is exported from Harmony per peg, so all intermediate rotations and scales are represented in the Animation window in Unity. Animations can be modified once their associated AnimationClip asset is duplicated from a TBG sub-asset to a novel asset in the project, allowing events to be added and additional curves to be provided within Unity.</p> <p>AnimationClips are selected through an AnimatorController, providing a means of transitioning between animations, either immediately or with smooth blending of peg transforms.</p>
Shader Graph Integration	Compatible shadergraph shaders are provided as part of the SDK that can be duplicated and modified, allowing for an easy interface to introduce custom game-specific visual effects on your characters.

HarmonyRenderer Improvements (XML Folders)

Feature	Description
HarmonyProject Preview	Upon selecting a character in the Unity assets folder, a preview of the character will appear in the preview window. The preview window plays a preview of the character's animations, and you will have the option to change skins without needing to drag the character into the scene view.

Fixes

The following issues have been fixed in this release of the Harmony Game SDK:

HarmonyRenderer

- Anchors previously would leak memory while updating Anchor transforms. An update to the HarmonyRenderer C++ plugin resolves this issue.
- HarmonyRenderers have improved rendering performance. Will now receive new meshes from the HarmonyRenderer C++ plugin that can be passed directly to Unity's C++ backend using newer Unity mesh generation features. This greatly reduces time taken to update animations on HarmonyRenderer characters, allowing for more characters on-screen at once.

Chapter 1: Exporting Harmony Data for Gaming

There are two main pipelines for exporting data from Harmony to your game engine:

- [Raw Game Data Export](#) on page 11
- [Frame-by-Frame Export](#) on page 12



Creating Animation in Harmony

When creating character rigs and animation for games, there are a few things to think about before starting. Artists and programmers should work together to make sure their needs are met.

- What platforms will your game be created for? Windows, Mac, Mobile, iOS, PS, XBOX, etc.
- What game engine will you be using?
- What animation style will be used to create the look of the game? Hand drawn, cut-out, with textures, etc.

These are just a few of the questions to consider before getting started. They all have an impact on how you design, build, and animate characters.

For example, if you're planning a mobile game for smartphones, then you will most likely want to keep your game under 50 MB, so it can be downloaded without having to be on Wi-Fi. In that case, your most important consideration is to create efficient characters with very tight sprite sheets and reuse a lot of the animation to keep the file sizes small. During the process, you will need to:

- Rig and animate characters in Harmony.
- Extract the Harmony data.
- Import the Harmony data into the game engine.

Keep in mind, if you're working with a custom engine, you can also process the Harmony data that's exported and use it in a custom game engine. If you need assistance with adapting data for your engine, contact store.toonboom.com/contact/support.

If you're making a game for consoles like the PS or Xbox, then you have the freedom to create larger textures. You may want to animate frame-by-frame, with a cut-out character, or both.

If you're simply going to export on a frame-by-frame sequence, then you can use all the tools in Harmony without limitations. You can then process an exported image sequence into a sprite sheet.

Raw Game Data Export



Raw game data export is appropriate when you want the file sizes to be as small as possible. Toon Boom lets you convert data directly from your Harmony scene to incorporate into a game engine. You can extract the skeleton information, drawing information, and keyframe animation data, as well as deformations (bones and articulations only), cutter, transparency nodes, and timing columns.

- **Advantage:** This is the lightest export, and will keep file sizes small which is ideal for mobile applications.
- **Disadvantage:** You are somewhat limited in the tools you can use in Harmony. You can use tools like Morphing as well as Curve and Envelope deformers, but you'll need to bake it out to drawings so they're interpreted properly in the game engine. You can use Cutter effects (masking), but you cannot cascade them, meaning you cannot have more than one in a hierarchy chain. The Game Bone deformers can be used on your rig without having to bake it to drawings.

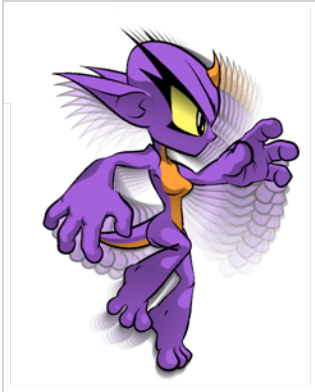
However, even with these limitations, you can create really great cut-out character animation in Harmony and extract all the compatible data. By moving, rotating, scaling, and skewing the different drawing layers, you can create advanced looking animation.

When you extract the data, you'll have sprite sheets that contain only the drawings of the body parts used in your Harmony scene file. You can also support multiple animations, such as idle, run, and jump while reusing the same skeleton and drawings.

Toon Boom has fully integrated this solution with the Unity game development rendering engine. If you are creating your game in Unity, you have a seamless pipeline without the need to re-treat the data in your game engine.

Chapter 2: About Game Asset Creation

In this section, you'll find guidelines for rigging, deformation, cutter and many useful animation tips.



Game Rigging Guidelines

The following is a list of general guidelines to keep in mind when rigging your character. As you're planning the character rig for your game, keep in mind the style of the character, and create your custom colour palette. However, there are some limitations to consider if you plan to extract the game data:

- Set your Harmony scene to be a square resolution (e.g. 1024 x 1024). You can do this in the Scene Setting dialog box. See the Reference guide
- Nudge layers in Z space if you need to reorder layers. However, significant Z offsets are not supported within a character rig.
- Make every layer in your game engine a separate scene in Harmony. If you have two characters at different depths, put them in separate scene files.
- Don't use 3D space. If you want to set things up in 3D space, you can do this when you get to your game engine.
- Set your pivot points on Peg layers using the Rotate tool to set the pivot on the entire layer. Peg pivots are recommended over drawing pivots. You should also set the pivot points on your drawing layers, even if you don't animate on them, as this will allow you to retrieve the information later on in the game engine if you need to put an anchor on a drawing layer.
- Don't use Morphing. This is not yet supported in game engines.
- Be mindful of where you put your character before exporting. The master pivot of your exported game object will be the center of your Harmony scene (0,0).
- Be sure to have a Display at the end of your hierarchy.
 - When rigging, keep in mind that some nodes are not supported by the exporter. As a result, the node structure would not be interpreted well by game engines. It is therefore recommended that you avoid using complex node structures. To do so:
 - Build your character out of Drawings attached to Pegs. Pegs can then be connected to other Pegs to form Hierarchies.
 - You should have all Pegs connected to a single Master Peg at the top of your Node view
 - Drawings should generally connect to Composites to determine the order of Drawings shown on-screen
 - If Drawings need to change order during animation, you can bump the z-depth to change which Drawing shows first
 - If you can't get your animations working well with just Drawings and Pegs, there are some other nodes you can use that are possible to export:
 - **Cutters** - Generally only cut one Drawing against one other Drawing (eg. cutting a face to the bounds of a head). You cannot 'daisy chain' one Cutter's output into another Cutter's input.
 - **Game Bone Deformations** - Generally only deform one Drawing. The bone skeleton can branch. Kinematic Outputs can be used attach Drawings to the ends of bones (eg. foot on the end of a leg)

**NOTE**

you are still able to use more complicated node structures within 'bake_groups'. See [Using Bake_Groups](#).

Keeping these tips in mind will allow you to create a tight, efficient 2D game character in Harmony while taking advantage of all the great tools.

Here are some things you should do:

- Create a simple parent-child relationship hierarchy in the Timeline view.
- Use peg layers to contain keyframe animation data, set to Separate Position.
- Use drawing layers to draw on, creating new drawings when needed.
- Use the Rotate tool to set the pivot points on the peg layers.
- Name your layers properly so if you need to fetch a specific layer's pivot point later on in the game engine, you can easily recognize the layer you need. If you have a top-level Group A, which has a child group inside it (Group B), and the drawing layer is a child of Group B, then the drawing layer is exported as A_B_DrawingLayer.
- Set your anchors where you want to have your pegs show in Unity as transforms.

You can use any of the drawing tools you want: Pencil and Brush tools, textured lines, solid areas, and gradients. Each individual drawing will be rendered out and assembled into a sprite sheet later.

**NOTE**

Because the Unity game engine does not support Unicode characters, it is recommended to avoid using it in scenes intended for games.

Game Deformation Guidelines

Harmony can export deformations to the game engine XML format. Only hierarchies made of Game Bones can be successfully exported. In addition, Kinematic Outputs were implemented to complement your game bones.

Deformations in Harmony come with a wide set of features. However, some of these features are not compatible with the game engine SDK and must not be used for gaming. Hence, Game Bone deformer do not support the following standard Bone deformer features:

- Zones of influence
- Curve and Envelope deformations
- Having more than a single pose in the deformation hierarchy
- Animating drawing pegs underneath deformation groups

The deformation in the game engine SDK does not behave exactly as the Bone deformation in Harmony.

To comply with most game engines and maintain fast calculations, the SDK implements a linear base skinning algorithm to blend the bones at articulations. You may notice some differences depending on the curvature of the articulations used. That said, compared to regular Bone deformations, GameBones most closely resemble in Harmony what you will see once imported to Unity.

**NOTE**

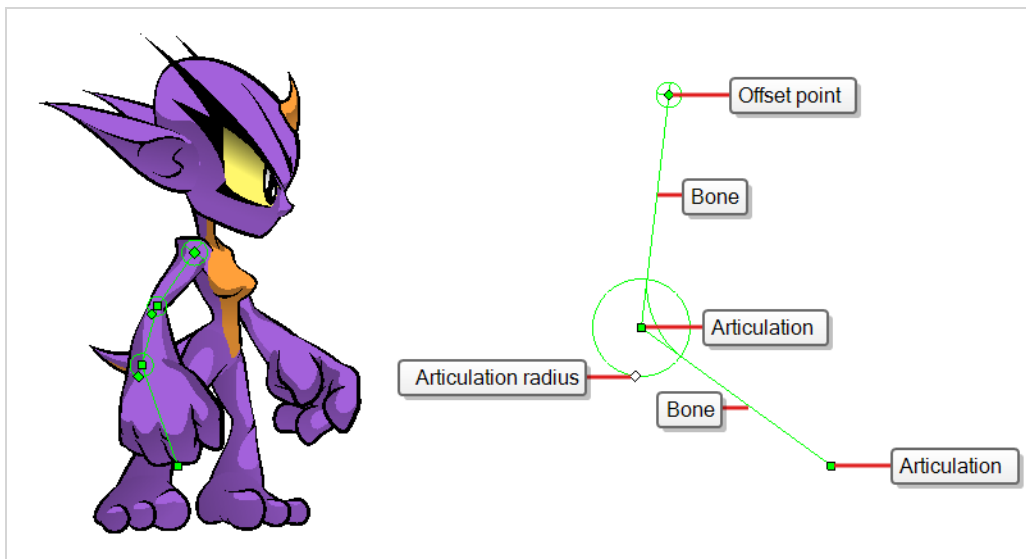
* Not currently available in the Cocos2d-x implementation of the game engine SDK.

About Game Bone Deformations

T-RIG-007-003

The Game Bone deformation is very similar to the Bone deformation. It allows you to create a bone-like structure in which each part is solid, but with articulations that are flexible. This is mostly useful for animating a character's limbs, such as the arms or legs, or other parts that can be articulated such as torsos or fingers. For example, a Game Bone deformation can be used to articulate an arm that is made of a single drawing, so that the upper arm and forearm can be moved independently, without having to draw the upper arm and the forearm on different layers. Harmony will deform the drawing to make it look articulated. The different parts of a Game Bone deformation can be rotated around their joint, extended and shortened, giving you the same capabilities as animating articulations on different layers, without having to worry about parts detaching, pivot points, or clipping outlines.

The Game Bone deformation is different from the Bone deformation in which it is optimized for game engines such as Unity. Hence, it is usually only used for game development and not in animated productions. The differences between the Bone and Game Bone deformations are that Game Bone deformations do not have Bias and Region of Influence properties. The articulation folds also look slightly more rounded.



Game Cutter Guidelines

The cutter, or mask, operation in Harmony is used for cutting off drawings with custom shapes. The game engine SDK implements both cutters and inverted cutters with the following limitations:

- For a sprite, only a single cutter drawing can be applied when it is rendered. This also applies to a composite of multiple matte drawings. The game engine SDK will only use the first matte drawing during rendering and discard the others.
- A deformed drawing cannot be cut, but a cut drawing can be deformed.

**NOTE**

Not currently available in the Cocos2d-x implementation of the game engine SDK.

Creating Metadata Notes

As you create assets for your game in Harmony, you may want to make notes about the scene or specific parts of the character or props for the programmer. These embedded notes that will be exported with your assets to Unity are known as metadata.

How to access the Metadata Editor view

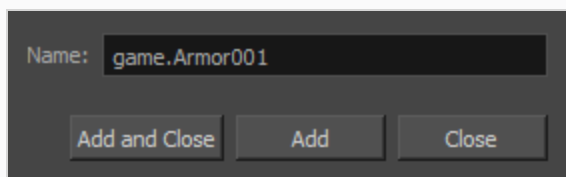
- In the top-right corner of a view, click the Add View **+** button and select **Metadata Editor**.
- In the top menu, select **Windows > Metadata Editor**.

How to create scene metadata

1. In the Metadata Editor view, in the Scene Metadata section, click on the plus **+** button to create a new metadata entry.

The Add Metadata dialog box appears.

2. In the Add Metadata dialog box, enter the name of your new scene entry.



NOTE

For any anchor or prop information created within the Metadata Editor, the naming convention must always use the prefix "game.". The term "game." is recognized automatically by Unity as a metadata information. For example: *game.Armor001*.

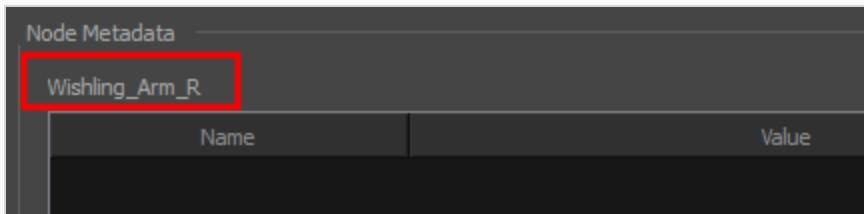
3. Click **Add and Close** if you only intend to add a single entry. Click **Add** if you intend to add multiple entries. Continue adding multiple entries, clicking **Add** after each one. Click **Close** when you are through.
4. In the Metadata Editor view, double-click on the value field for the first entry to make it editable.
5. Enter the value information for this entry.
6. Continue adding value information for all your entries.

This information will be exported with your Harmony assets. Once in Unity, scene Metadata will appear in Inspector view > Metadata when the asset is selected in the Hierarchy view.

How to create node metadata

1. In the Timeline view, click on the layer to which you would like to attach metadata.

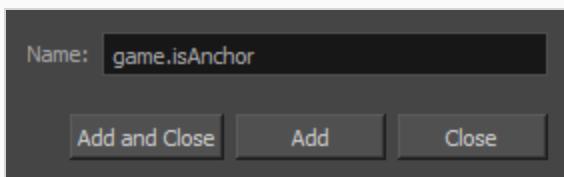
The name of the layer appears near the top of the Node Metadata section.



2. In the Node Metadata section, click on the plus **+** button to create a new metadata entry.

The Add Metadata dialog box appears.

3. In the Add Metadata dialog box, enter the name of your new node entry.



NOTE

For any anchor or prop information created within the Metadata Editor, the naming convention must always use the prefix "game.". The term "game." is recognized automatically by Unity as a metadata information. For example: *game.Armor001*.

4. Click **Add and Close** if you only intend to add a single entry. Click **Add** if you intend to add multiple entries. Continue adding multiple entries, clicking **Add** after each one. Click **Close** when you are through.
5. In the Metadata Editor view, double-click on the value field for the first entry to make it editable.
6. Enter the value information for this entry.
7. Continue adding value information for all your entries.

This information will be exported with your Harmony assets, more specifically, this information will be linked to the selected layer. Once in Unity, scene Metadata will appear in Inspector view > Metadata when the asset is selected in the Hierarchy view.

Game Animation Tips

When animating for games, depending on the type of game you're creating, you may need to limit your animation. For example, if you're creating mobile games and you want to keep the file sizes small and playback fast on all devices, then limit yourself to simple keyframe animation with as few drawing swaps as possible. If you're creating console games, you have the freedom to create more drawings and have a higher complexity. You can explore with your developers the limitations of the platforms you're exporting to, and what your game engine supports.

Here are some tips for efficient, lightweight animation:

- Use mainly transformations, such as move, rotate, scale, and skew.
- Create additional drawing swaps when needed.
- If you use Curve and Envelope deformers or Morphing, you'll need to bake out the drawings for export. Be careful when doing this, as you may want to keep the number of drawings small. Don't bake out an entire sequence, just selected drawings. You don't need to bake the Game Bone deformers.
- The bigger the drawings are in the Drawing view, the more pixels they will occupy in the texture size on the sprite sheet. When setting up your rig, make sure to not scale individual layers by using a keyframe with the Transform tool. If you want to scale things up or down, use the Select tool. This will keep things the same relative size on the sprite sheet. When you export the sprite sheets, in the script you can also set the resolution of the sprite sheet so the drawings can be scaled down for smaller devices.
- Only drawings which are exposed in the scene will be exported to the sprite sheet. For example, if you have 10 drawings in your Library view, but only two of them are showing in your scene, only those two will be exported. This keeps the sprite sheet as tight as possible.

Animating Multiple Sequences

You will most likely have multiple animations for your characters. For example, an idle sequence, a run sequence, an action sequence, and so on. You need to work in a specific structure so you can export all of these animations to a single sprite sheet.

There are two different workflows that you can use:

- Workflow 1: Separate Scenes
- Workflow 2: Separating Using Scene Markers

Workflow 1: Separate Scenes

First, create a scene file with the name of the character, such as **Space Duck**. This is the file where you can create or import your game rig. In the top menu, select **File > Save As New Version**, and give this new version the name of the animation. For example, **Idle**.

Every time you need to do a new animation using the same character, perform a Save As New Version. In the end, you may have something like this:

Scene: Space Duck

Versions:

- Idle
- Run

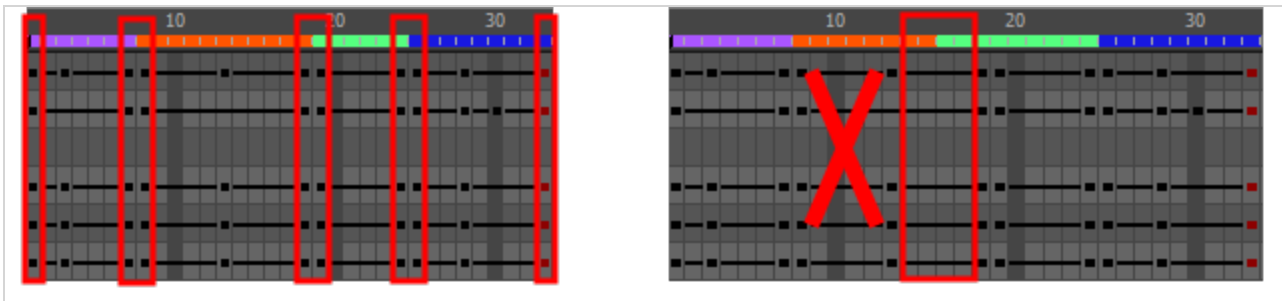
- Jump
- Shoot

When you run the export script, it will export the drawings from the current scene into the export folder. It will also let you know if there are any other scene versions that were already exported to that folder.

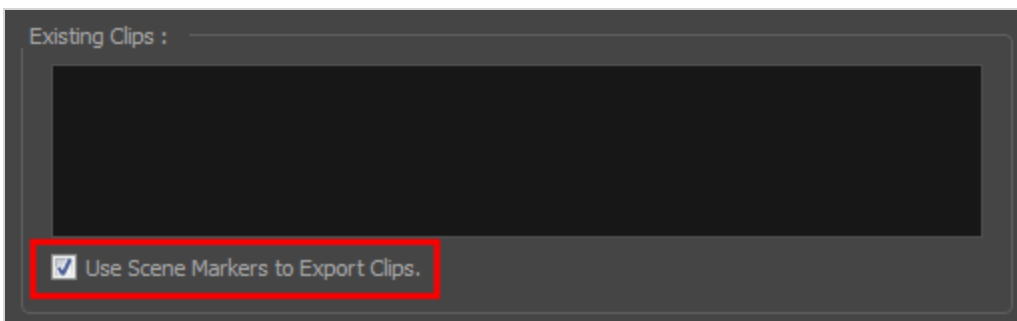
Workflow 2: Separating Using Scene Markers

You can also create all of your character animations in a single scene, one after the other, such as idle, run, jump and shoot. Then use scene markers to mark and separate the individual animations.

When you are marking individual animation frame ranges, be sure that they start and end with a keyframe. Do not create scene markers for a range of frames that starts or ends in the middle of an interpolated movement.



When exporting your sprite sheet, in the Export To Sprite Sheet dialog box, be sure to check the **Use Scene Markers to Export Clips** option. This is enabled by default.



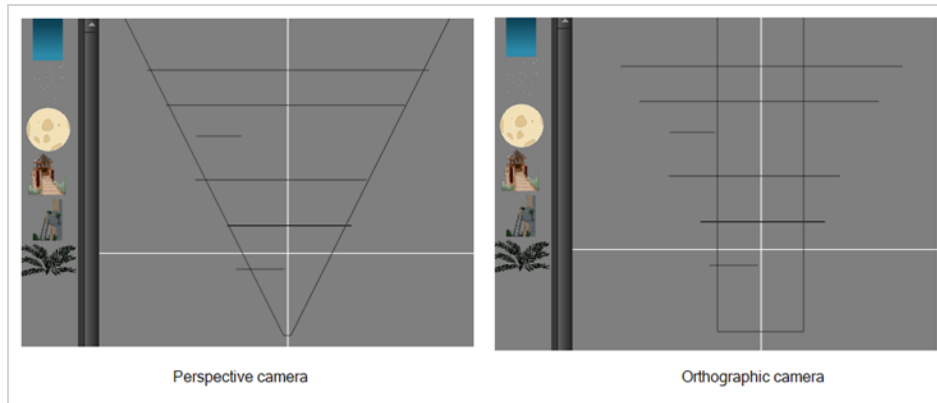
The animated clips are divided and listed in the stage.xml in the same way that they would appear if you had exported each animated sequence from separate scenes to the same file location.

About the Orthographic Camera

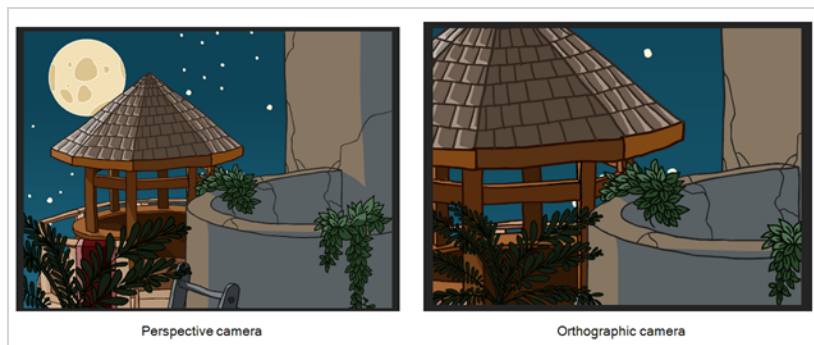
In Harmony, there are two types of cameras available:

- Perspective
- Orthographic

The orthographic camera is specific to the gaming pipeline. It changes the camera type from perspective to orthographic. It becomes a camera without vanishing points. This means there is no more perspective in the Camera view. Objects, when moved on the Z-axis, will not change in size or scale.



The orthographic camera can be set in the Scene Settings dialog box. In order to create scenes with the orthographic camera, by default, you need to create a new custom scene resolution.



Chapter 3: About Exporting to Unity

If you have a custom game engine, you can take the exported Harmony data and proceed with your usual process. Or you can modify the export script to fit your convention.



Before exporting a scene:

- Be sure to set your Display as **Display** and not Display All.
- Be certain to save your scene. Harmony makes the export based on the tvg files, any unsaved updates you have added will not be exported.
- Set your Harmony scene to be a square resolution (ex. 1024 x 1024). Select Scene > Scene Settings.

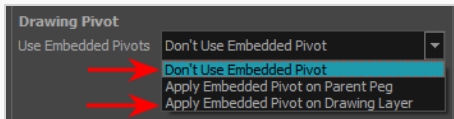
Setting Anchors


Before exporting to Unity, you should consider what to set as your anchors.

An anchor is a point on the character that you would like to reference in the game engine during the character's animation.

How to set an anchor

1. Create a new Drawing layer and rename it according to the anchor you are creating.
2. Inside the Layer Properties of your drawing layer, make sure that pivot option is set to **Don't Use Embedded Pivot** or **Apply Embedded Pivot on Drawing Layer**. The Apply Embedded Pivot on Parent Peg does not transition well to Unity and you will lose your pivot information.



3. Once this is done, you can use any of the animation tools to reposition the pivot point of your layer, such as the Rotate tool in the Advanced Animation toolbar. You should be able to see the pivot coordinates from the Layer Properties of your Drawing layer. If the pivot is not repositioned, its default place will be at the center of your Harmony scene (0,0).
4. Once this is set up, select the Drawing layer and click the Toggle Anchor  button in the Game toolbar. You can also make a multi-selection to set an anchor on multiple Drawing layers simultaneously.

The Drawing layer is highlighted in red in the Timeline and Node views. Your anchor can now be used to attach a prop.

Exporting Sprite Sheets

The Export to Sprite Sheets window exports to multiple resolutions, generating multiple .xml files and one or multiple sprites sheets depending how many sprite resolutions you defined.

This saves different animations of the same character into the same name. For example, if there's an idle, run, and jump animation, these should all share the same Asset Name. You can think of it as the overall collection of animations. Inside are the different saved scene versions whose drawings you can reuse for all the animations in that character set. Each scene version will be displayed as an item in the list.

When you export an animation, only the drawings used in that scene are exported. All the drawings are exported individually first and then atlased together into a sprite sheet.

If you saved multiple animations to the same Asset Name (i.e. SpaceDuck: run, idle), then it will re-atlas the sprite sheet to include all the drawings from all the animations in that folder, creating a new animation file, but reusing the same skeleton.



NOTE

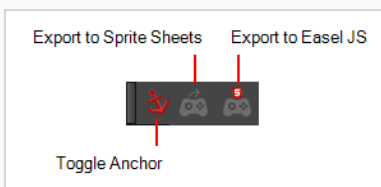
Programmers may be interested to note that the XML data exported by Harmony can be converted afterwards to a more optimized binary data structure. To convert XML to Binary format, use the Xml2Bin utility. This utility converts the XML data structure generated through the Toon Boom Harmony software to a compressed binary data structure. This utility is available in the gaming SDK under /HarmonyGameSDK/Plugins/.

- Plugins/Mac: Precompiled binary for Mac OSX.
- Plugins/Windows\x86: Precompiled binary for Windows.
- Samples/HarmonyGameSDKSource/Utils/Xml2Bin: Xml2Bin sources.
- Samples/HarmonyGameSDKSource/Utils/Xml2Bin/proj.mac.Xml2Bin.xcodeproj: XCode project for Mac OSX.
- Samples/HarmonyGameSDKSource/Utils/Xml2Bin/proj.win32/Xml2Bin.sln: Visual Studio 2010 solution for Windows.

The C++ code that handles the data structure can be reused and parsed in your own code if you want to integrate with other game engines.

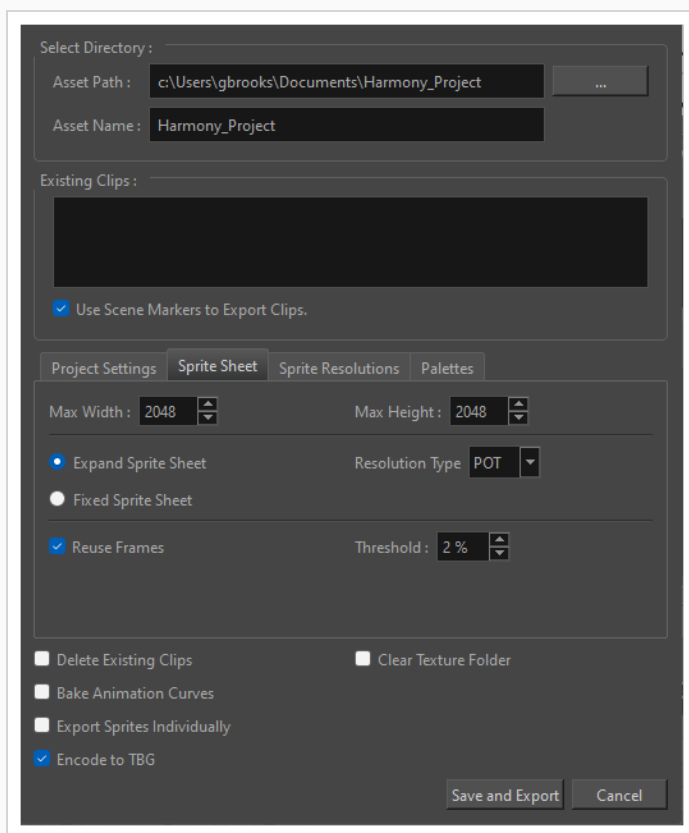
How to export sprite sheets

1. Add the Game toolbar, by selecting **Windows > Toolbars > Game**.



2. Run the script by clicking the Export to Sprite Sheets  button in the Game toolbar.

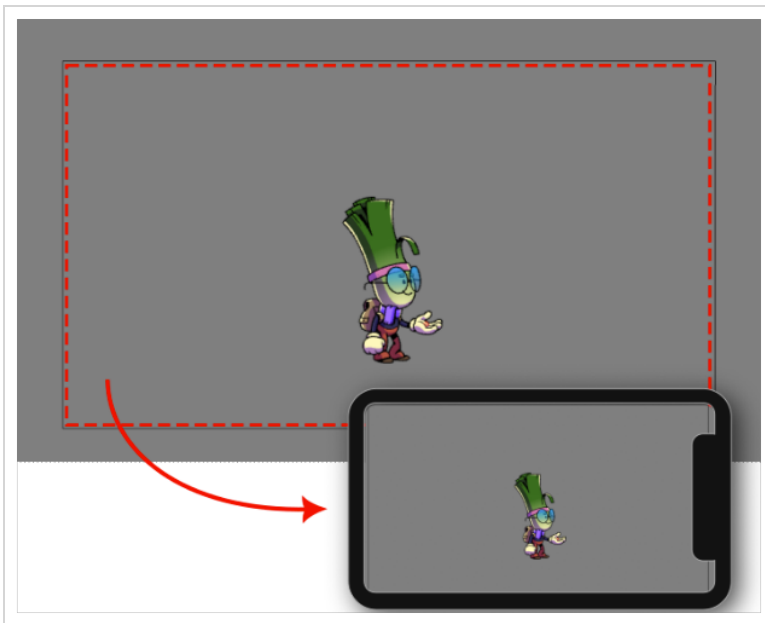
The Export to Sprite Sheets window opens.



3. Set the Save Path to the correct folder in your Unity project if you want it to update automatically. If not, you can save anywhere, then transfer the animation into your Unity project, or any other game engine you want to use.
4. Set your sprite sheet preferences.
5. Click **Export**.

Sprite Resolutions

You can set the number of pixels on a sprite depending on the targeted resolution for your game. Your target resolution may be based on the hardware you are targeting such as a television or phone screen.





A new sprite sheet will be created for each target resolution specified in the Resolutions tab in the Export to Sprite Sheet (XML) window. To access this window, you will need to enable the Games toolbar by selecting **Windows > Toolbars > Game**.



Please note that the size of the sprite will be the size of the character when it was created in Harmony.

How to setup multiple sprite resolutions

1. Enable the Game toolbar by selecting **Windows > Toolbars > Game**.
2. Select Export to Sprite Sheets . The Export to Sprite Sheet (XML) window will open.
3. In the Export to Sprite Sheet (XML) dialog, select the Sprite Resolutions tab. This tab will list all of the current target resolutions.
4. Click the Add  icon. This will create a new row for a new target resolution.

5. Enter the Name, Width, and Height of your new target resolution.

The default target resolutions are HD, FullHD, and QuadHD. The listed screen resolutions represent a target device the game could be played on.

You are free to manually add or remove resolutions from this list depending on your target resolution and hardware. Once imported into the game engine, it will be possible to select which resolution is being displayed by changing the rendered sprite sheet.



NOTES

- The sprite resolutions are affected by the scale of the drawing strokes.
- Scaling drawings or their pegs will not affect the output sprite resolution.
- If you want to increase the sprite resolution, you may scale a drawing's child layers.

Using Bake_Groups

A Bake_Group is a grouping of drawing and effect nodes that may be exported together as a single unified drawing in a sprite sheet.

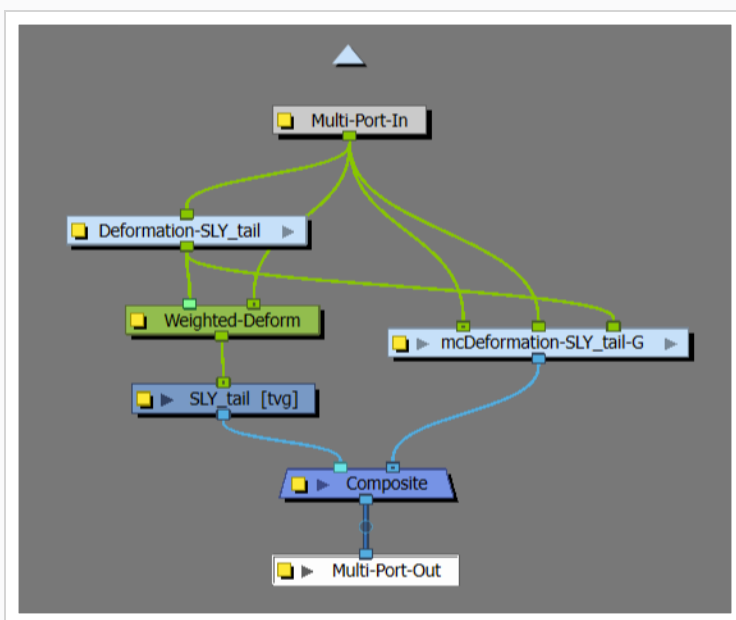
How to create a baked group

1. Group together the layers that you will bake together. It is recommended to group the layers together from the node view.



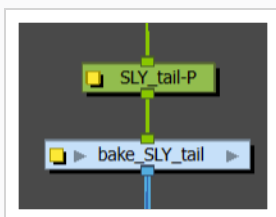
NOTE

A bake_group should only have a single input and a single output. This generally means a Composite node must be contained within the bake_group for the export to work properly. Consider also what pegs are needed to guide the drawings, as all pegs must ultimately source from a single peg outside the group.



2. In the timeline, add the "bake_" prefix to the beginning of the group name.

See the example below for how a group node with the "bake_" prefix should look. The "group" labeled "hips" should become "bake_hips" for the bake to apply.



If the “bake_” prefix is included in the group name, the group will be analyzed for frames containing unique effects, movements and drawing substitutions, and each unique frame will export as a sprite in the sprite sheet.

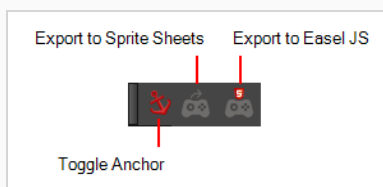
New drawing animations for these generated sprites will be created and provided to the game engine. The game engine will not have access to any of the structure inside this group, instead treating the entire group as if it were a single Drawing with drawing substitutions.


Exporting to Easel JS

The Export to Easel JS window lets you flatten an image sequence of your animation. Even if you have a fully rigged puppet or a single drawing layer with your animation sequence, the outcome will still be a flattened output of each frame, grouped together in your sprite sheet. This allows for more flexibility and freedom of work as you have access to any tools or effect modules you want to use. However, this can result in heavier files depending on the length, complexity and export size of your animation.

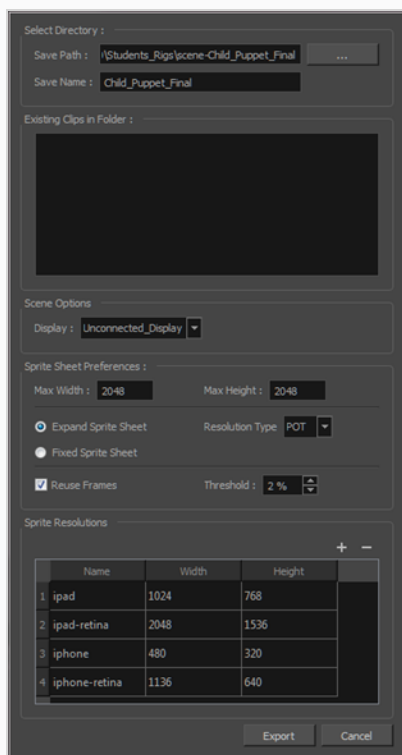
How to export to Easel JS

1. Add the Game toolbar, by selecting **Windows > Toolbars > Game**.



2. Run the script by clicking the Export to Easel JS  button in the Game toolbar.

The Export to Easel JS window opens.



3. Set the Save Path to the correct folder in your Unity project if you want it to update automatically. If not, you can save anywhere, then transfer the animation into your Unity project, or any other game engine you want to use.
4. Set your sprite sheet preferences.

5. Click **Export**.

Palette Variations

Creating versions of characters with palette differences is important when you plan to add character customization as a feature of your game. In Harmony, you may create multiple palettes for a character so that it can have a variety of different looks in your game.

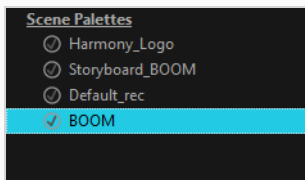
In Harmony, once you create your character and its palettes, you may export the character to sprite sheets for each palette.


For information on exporting sprite sheets.



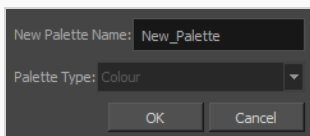
How to create palette variations for a game character

1. In the Colour view, select a palette to clone.



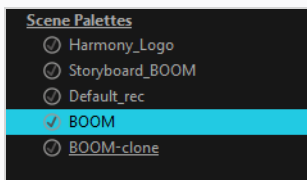
2. Do one of the following: From the Colour menu, select **Palettes > Clone** or right-click and select **Clone**.
 - Right-click on the palette and select **Clone**.
 - Open the Colour menu , then select **Palettes > Clone**.

The Clone palette dialog box opens



3. In the **New Palette Name** field, enter the name for the new palette. By default, the palette's name is based on the original palette name followed by "-clone". The palette list above this field lists the palettes that already exist in the selected location. Make sure you enter a name that is not already used by another palette.
4. Click **OK**.

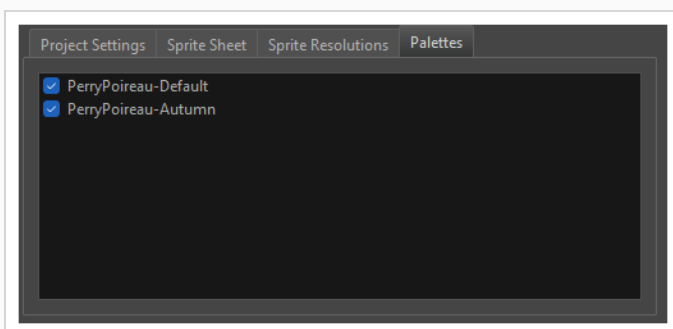
The cloned palette appears in the palette list.



All Scene Palettes will appear in the Palettes tab of the Export to Sprite Sheet window.

How to export a sprite sheet for each palette variation

1. Enable Game toolbar by doing one of the following:
 - In the top menu, select Windows > Toolbars > Game.
 - Right-click on any existing toolbar and select **Game**.
2. In the Game toolbar, select Export to Sprite Sheets 🎮.
3. Open the Palette tab. You will see all of your project's palettes and their clones.



NOTE

Only Palettes and their clones will appear. Palette duplicates will not appear.

4. Select the names of the palettes you would like to export. Each palette that is checked will export with the character as its own sprite sheet.
5. Export your Sprite Sheet.

Once the character is imported to Unity, it will be possible to select which palette is being displayed by changing the rendered sprite sheet.

Chapter 4: About the Harmony Unity SDK

Toon Boom provides a Unity package in the Unity Assets store that contains all the scripts necessary to import the data exported from Harmony. Once unpacked to a new Unity project, the following structure is available inside the Toon Boom Harmony Gaming SDK folder:

- Documentation (Developer-focused information about the Unity integration)
- Plugins (SDK libraries)
- Samples (demo scenes, exporter scripts, SDK source, and a previewer)
- Runtime (Harmony scripts that execute during gameplay)
- Editor (Harmony scripts that only run in the Editor and don't appear in builds)
- Shaders (Provides shaders that properly render Harmony-specific features)
- Materials (Prebuilt materials that utilize Harmony shaders)

These packages can be managed manually through the Package Manager interface in Unity. Developers can upgrade and downgrade packages in the event that newer Unity versions have certain package incompatibilities with the Harmony scripts.

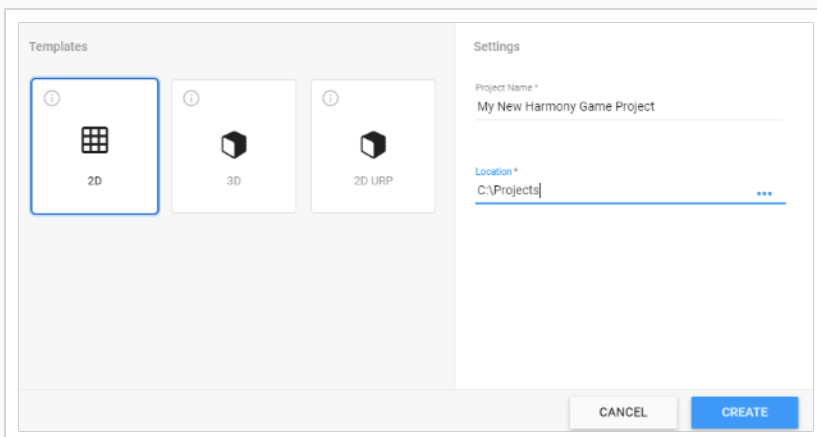
How to create a Unity SDK project

1. Create an empty Unity game project. It is recommended that you create an empty 2D Unity game project as it is the most convenient when working on 2D rendering and physics.



NOTE

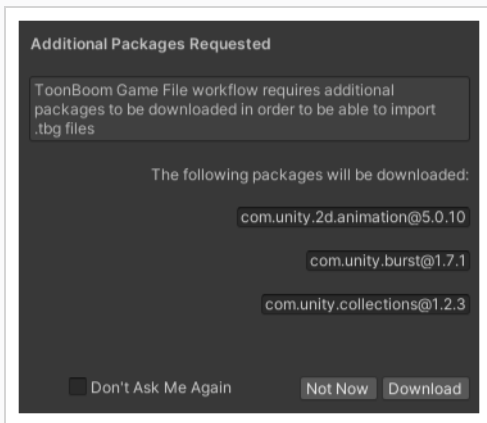
Ensure you are using Unity 2020.3.36 or later, previous versions of Unity don't have all the necessary features to support importing and rendering TBG files.



2. Install the Harmony Game SDK available in the Unity Asset Store. If your SDK was downloaded manually, you can double-click the .unitypackage file while your Unity project is open.

The Additional Packages Requested pop-up will appear to ensure that you have compatible Unity packages that facilitate optimized 2D animation at runtime.

3. Click **Download**.



Once Unity finishes downloading the required packages and compiling all the scripts, your project is ready to begin importing Harmony characters.

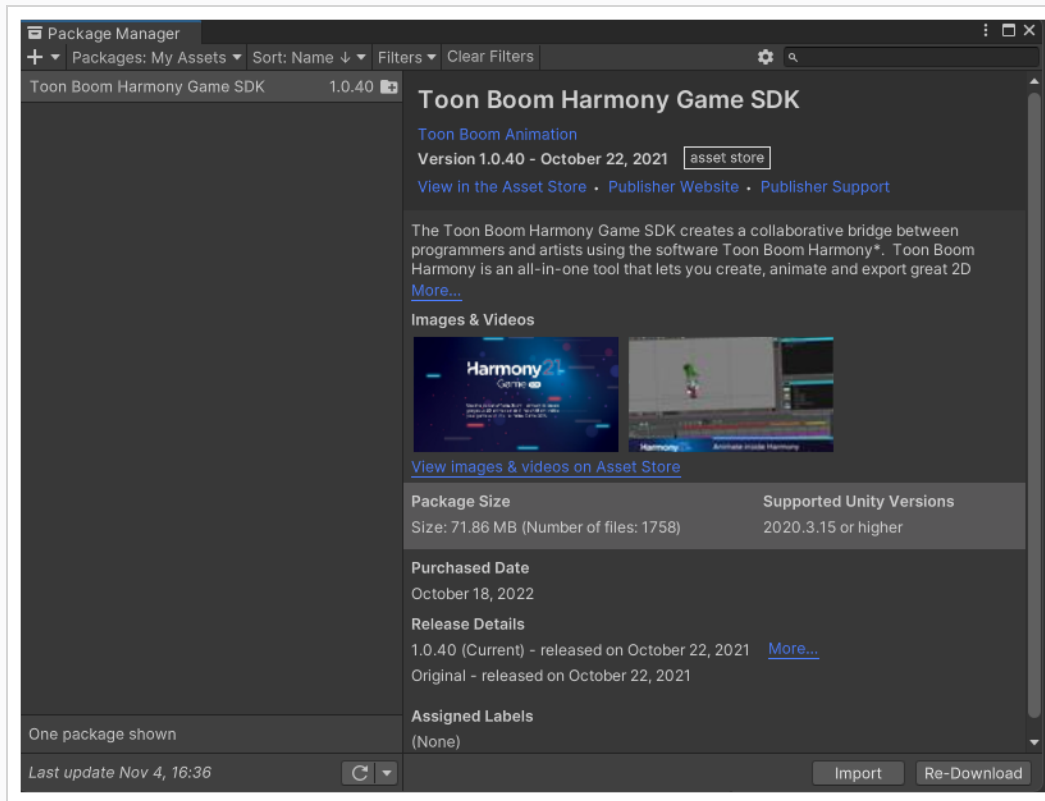


NOTES

When you have installed the Toon Boom Harmony Game SDK, the Harmony SDK package will have to be imported to every new project created in Harmony.

How to import a package to a project

1. Open the Asset Store by selecting Window > Asset Store. The Asset Store tab will open.
2. In the Asset Store, select **Open Package Manager**. The Package Manager window will open.



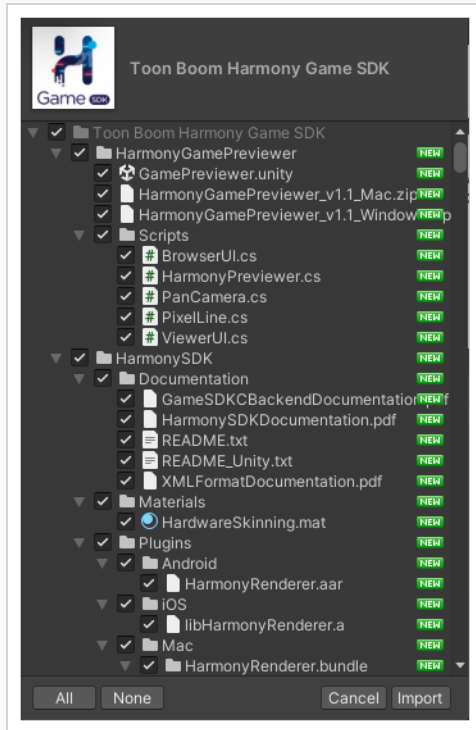
3. From the Packages dropdown at the top-left of the Package Manager, there will be four options:

- Packages: Unity Registry
- Packages: In Project
- Packages: My Assets
- Packages: Built-in

4. Select **Packages: My Assets**.

A list of your downloaded assets will display in the list below. To the right of the Package Manager tab, information about the selected asset will display.

5. When you have selected the Harmony package, select Import at the bottom of the Package Manager. The Import Unity Package window displaying the asset's files will appear.



6. Select **Import**.

Harmony Previewer

This section explains how to use the Harmony Previewer to view animations without installing Unity.

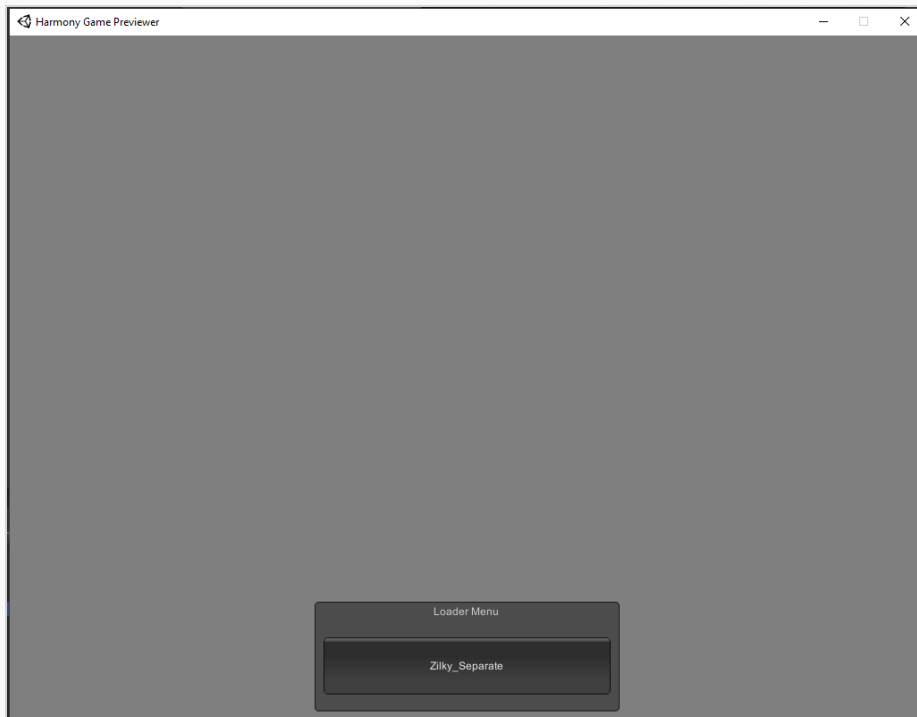
You can find a build for Windows or macOS inside the Harmony Game Previewer folder that you can just unzip.

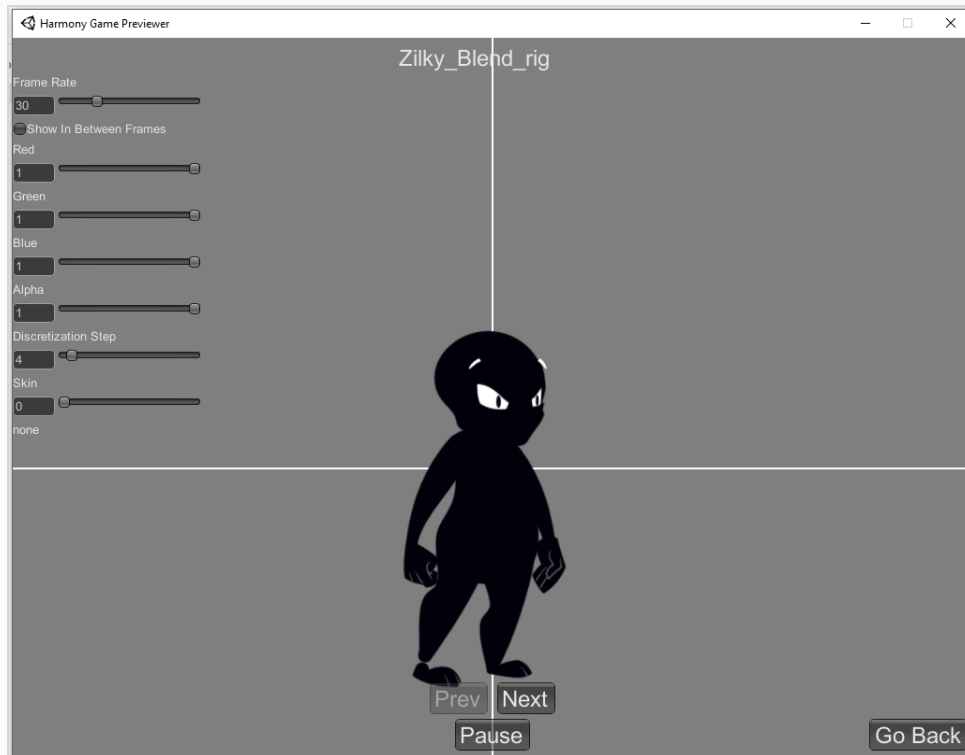


NOTE

Do not unzip the build inside your Unity project, it is advised to unzip it in its own folder.

How to use the Previewer





1. Run the *Harmony Game Previewer.exe* or *Harmony Game Previewer.app*
2. Select which project to load from the loader menu. These projects are loaded from the Harmony Resources folder in the same directory as the *Harmony Game Previewer.exe* or *Harmony Game Previewer.app*
3. Navigate through the animations using Prev / Next button.
4. Adjust the Frame Rate, Tint, Discretization Step and Skin settings with the controls at the top left.
5. Click Go back button to return to the previous menu and load a different animation.

Asset Manipulation

In this section, the components Harmony Mesh and Harmony Renderer in Unity will be described.

TBG Workflow

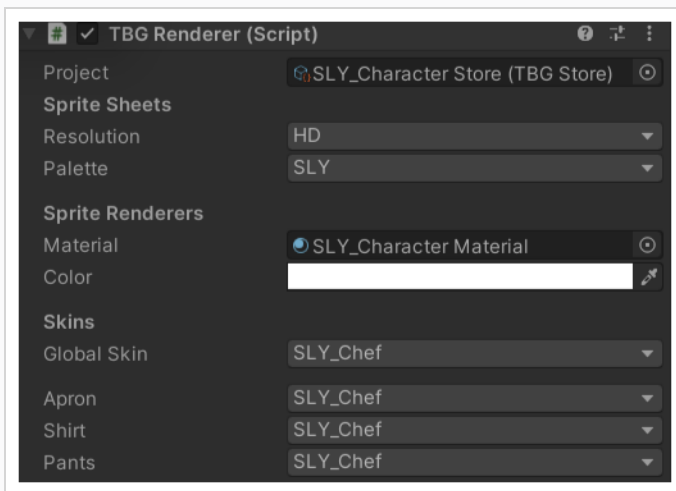
If you are importing sprites to Unity through the TBG workflow, asset manipulation options for TBG files will display in the inspector view. The asset manipulation options displayed are different from the asset manipulation options displayed in the XML workflow.

TBG Asset Renderer

Harmony Renderer for the TBG file workflow is used to control the animation and rendering of Harmony assets for TBG files as well as expose important variables you may want to change during runtime.

How to manipulate TBG Workflow rendering assets

1. Select a character in the scene. The TBG Renderer (Script) section will appear in the Inspector tab to the right.



2. Control your character's assets by setting the following:
 - **Project:** A reference to the TBGStore asset. This asset is generated by the importer and contains the TBG project data not covered by the generated AnimationClips.
 - **Resolution:** The resolution used to display this character. This matches associated sprite sheets names.
 - **Palette:** The palette used to display this character. This matches associated sprite sheet names.
 - **Material:** The material used to shade this character. Hardware Skinning is the default unlit material provided with the SDK.
 - **Color:** Option that multiply the selected color onto the asset.
 - **Skins:** Allows you to change in real-time the skin you've exported from Harmony. Each group has its own drop-down to change the skin, as well as a global dropdown to change the skin of all groups at

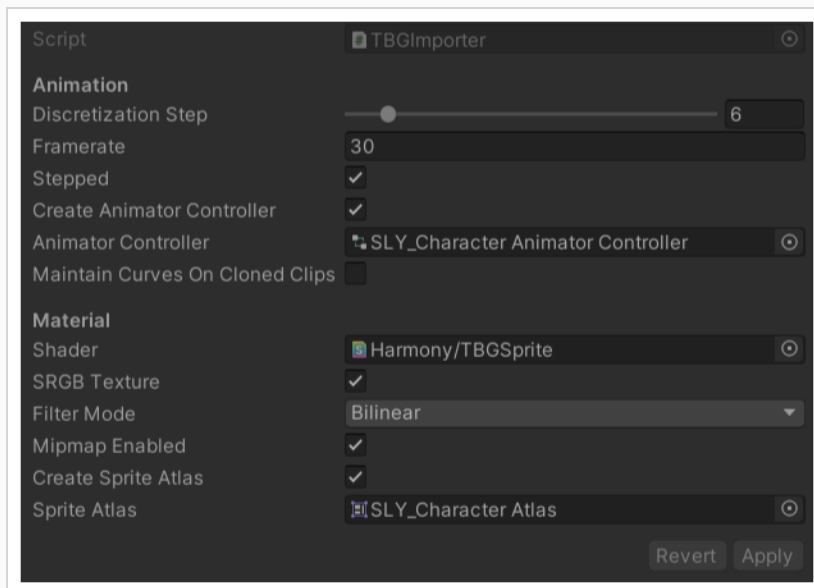
once.

TBG Importer

When TBG files are imported to Unity, the Harmony assets are converted to Unity prefabs, AnimationClips, and other assets. The TBG Importer allows you to specify how the assets and animations are created.

How to control the settings for import of Harmony TBG assets to Unity

1. Select a .tbg character file in your Project view. The inspector tab will open.
2. At the top of the Inspector view, click **Select** for the Model. The TBG Importer view will open and display the TBG Importer options.



3. Control how animations will import by setting the following:

- **Animation Settings**

- **Discretization Steps:** A slider representing the number of columns / rows that sprites will get split into if they are being deformed by bones
- **Frame Rate:** The animation frame rate used by Unity's animation system to playback the Harmony frames. This should match your frame rate in Harmony.
- **Stepped:** If checked, the animations will pass between integer frames and thus not interpolate smoothly. If unchecked, the animation will be played back between integer frames, for a smooth animation.
- **Create Animator Controller:** If checked, this importer will automatically create a new Animator Controller containing all the generated AnimationClips and place the asset beside the prefab in the Project view.

- **Animator Controller:** When changes are made to the .tbg file, the Animator Controller referenced will be updated with the new AnimationClips.
- **Material Settings**
 - **Shader:** When the prefab is created, all SpriteRenderers will reference a generated material using this shader.
 - **SRGB Texture:** If checked this will map gamma color space from generated textures into Linear color space rendering.
 - **Filter Mode:** How neighboring pixels of the texture interpolate.
 - **Mipmap Enabled:** Generate lower resolution textures to render at further distances.
 - **Create Sprite Atlas:** If checked, this importer will automatically create a new Sprite Atlas containing all the generated Sprites and place the asset beside the prefab in the Project view.
 - **Sprite Atlas:** When changes are made to the .tbg file, the Sprite Atlas referenced will be updated with the new Sprites.

XML Workflow

If you are importing sprites to Unity through the XML workflow, asset manipulation options for XML files will display in the inspector view. The asset manipulation options displayed are different from the asset manipulation options displayed in the TBG workflow.

Harmony Mesh

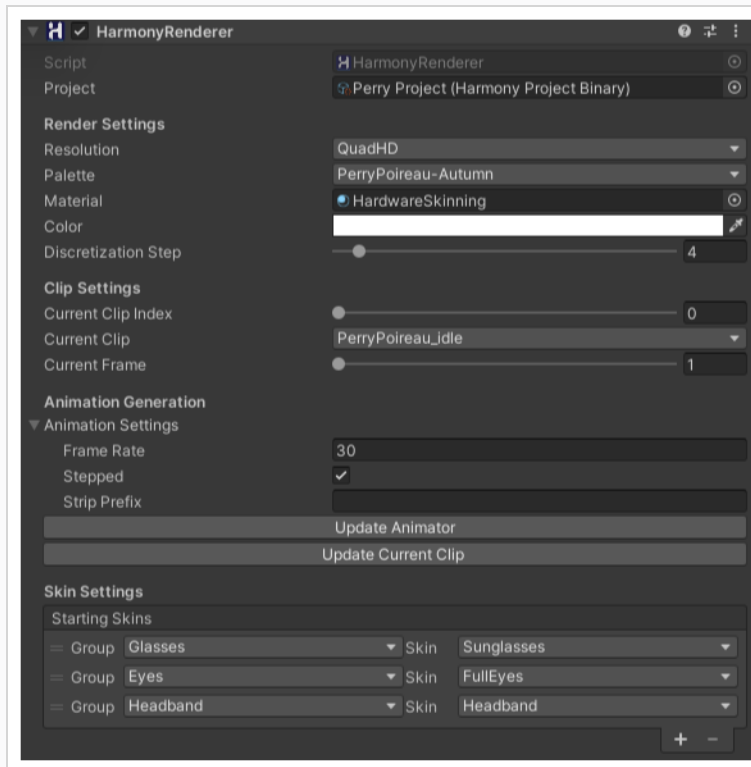
The Harmony Mesh component is used by the Harmony Renderer to create asset meshes based on the exposed drawings in your animation.

XML Asset Renderer

Harmony Renderer for the XML workflow is used to control the animation and rendering of Harmony assets for XML files as well as expose important variables you may want to change during runtime.

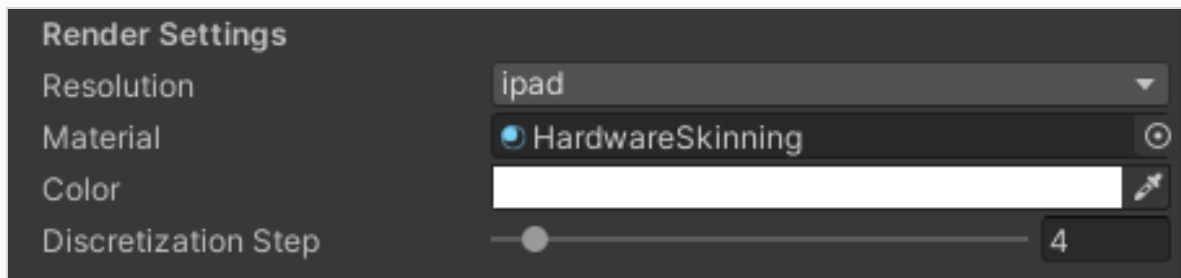
How to manipulate XML workflow rendering assets

1. Select a character in the scene
2. In the Inspector view, select Add Component at the bottom of the view. The HarmonyRenderer section will appear.



3. Control your character's assets by setting the following:

- **Project:** A reference to the Harmony Project Binary asset. This asset is generated by the importer and contains the Harmony project data in an efficient binary format.
- **Render Settings:** The render settings are used to modify your asset, either by changing the resolution or the Shader used to render or the quality of the deformation.

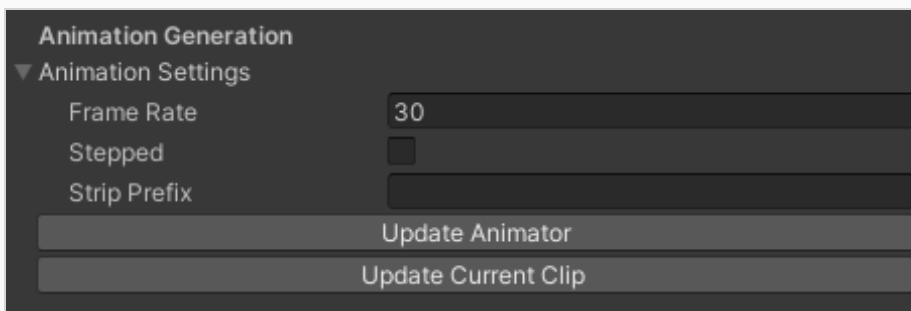


- **Resolution:** The resolution used to display this character. This matches associated sprite sheets names.
- **Palette:** The palette used to display this character. This matches associated sprite sheets names.
- **Material:** The material used to shade this character. Hardware Skinning is the default unlit material provided with the SDK.
- **Color:** Option that multiply the selected color onto the asset.
- **Discretization Steps:** Slider that defines the arrangement of all the deformed drawings.

- **Clip Settings:** The clips settings are used to see the animations without opening the Animation window. This only affects the project when in Editor mode, when you play the game, the Animator overrules the settings inside the Clip Settings.



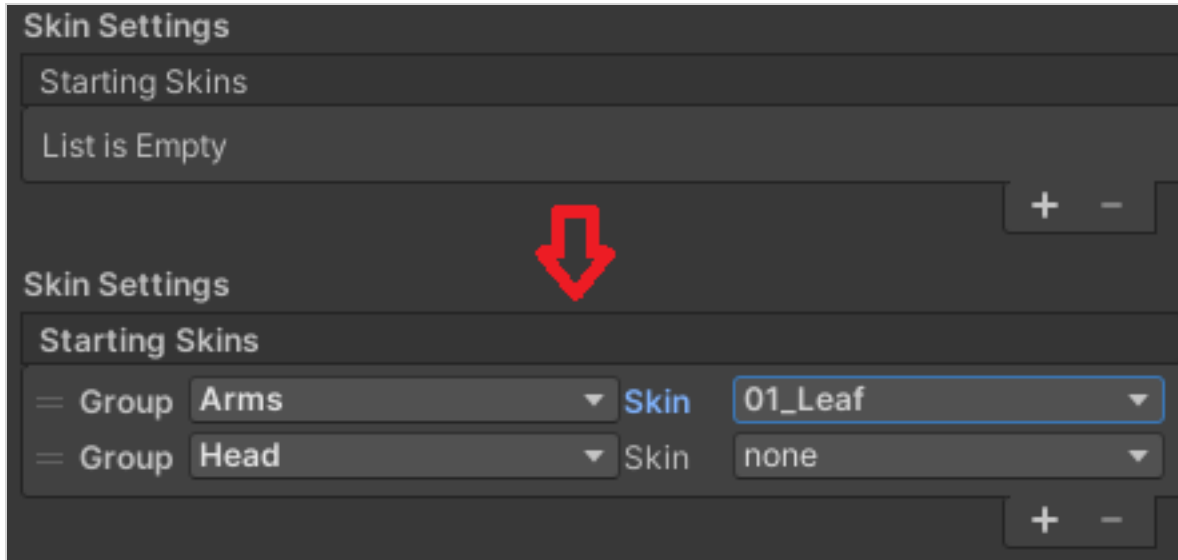
- **Current Clip Index:** A numerical clip index to display. It can be animated in the Animation window.
- **Current Clip:** A drop down for selecting which animation to display. It matches the Current Clip Index.
- **Current Frame:** The frame number currently being displayed.
- **Animation Settings:** These settings are same as importing Asset settings. It is useful while importing or re-importing asset modifications inside Harmony. Once modifications are done click Update Animator or Update Current Clip in order to save your modifications.



- **Frame Rate:** The animation frame rate used by Unity's animation system to playback the Harmony frames. This should match your frame rate in Harmony.
- **Stepped:** If checked, the animations will pass between integer frames and thus not interpolate smoothly. If unchecked, the animation will be played back between integer frames, for a smooth animation.
- **Strip Prefix:** This is used to remove a prefix on the animation name. For instance, if we want "Wishling_Idle" to become "Idle", we set the prefix to "Wishling_". This is useful where the animation name for games is same for all the characters.
- **Update Animator:** Click this button to update the entire set of animations on this character to match what is in the Harmony project. This will update existing animations with the new settings you entered and add any missing ones.
- **Update Current Clip:** This will update only the currently selected clip.

Skin Settings

The Skin Settings allow you to change in real-time the skin you've exported from Harmony. To change a skin you first need to press the + button to add a group. You need to add as many groups as you had in your Harmony export. If you had no groups, just add one and leave it on all. You can define which group you want to modify, and then which skin to assign to which group. You can then remove a group by selecting any group and pressing the - button.



How to change a skin in the XML workflow

1. Click + button to add a group/groups.
2. Modify the group, and assign the skin to group.
3. Remove a group by selecting any group and clicking – button.

About the Sample Unity Project

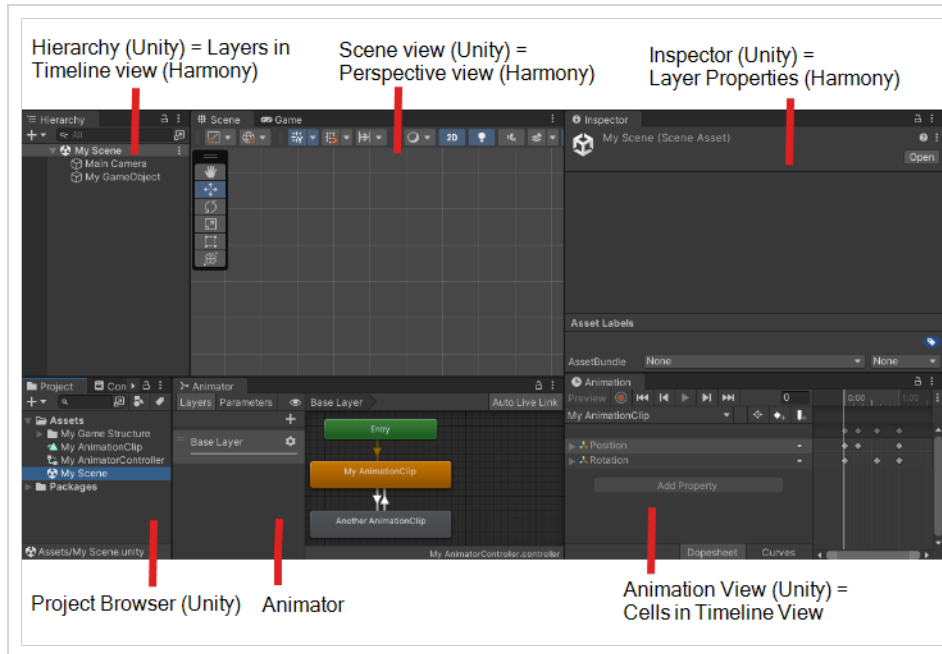
Toon Boom includes a sample Unity project that contains all the scripts necessary to import the data exported from Harmony. Inside this project is an Assets folder which contains the following folders:

- Plugins
- Scenes (demo scenes and a previewer)
- Scripts (all Harmony scripts)
- StreamingAssets (this is where all the Harmony scene files should be exported to)

By configuring the Harmony script, you can automatically export animations into the StreamingAssets folder. This way, Unity will dynamically load the most up-to-date assets as soon as they appear in the folder. You can also manually place exported Harmony data into this location.

About the Unity Interface

Here are the main components of the Unity interface and their equivalents in Harmony:



Unity	Harmony	Description
Scene view	Perspective view	This is where you set the scene, selecting and positioning environments, the player, the camera, enemies, and all other GameObjects.
Game view	Camera view	The rendered view from the camera(s) in your game. It is representative of the final, published game.
Inspector	Layer Properties	Displays detailed information about the selected GameObject, including all attached Components and their properties.
Hierarchy	Layers in Timeline view	Displays the hierarchy of elements in the scene, and lets you set up parent-child relationships for different game objects.
Animation	Cells in Timeline View	Shows keyframes of an animation per transform being animated.
Animator		Displays a graph of states and transitions that lets you set up how your characters will respond to game events and player actions
Project Browser	---	Lets you access and manage a project's assets.

Here are some data types that Unity interacts with, and their equivalents in Harmony:

Unity	Harmony	Description
Sprite	Drawing	A single image that can be shown on a SpriteRenderer.
SpriteRenderer	Layer	Displays one sprite at a time. Sprites can be swapped during an animation to construe motion of a character's body part.
Transform	Peg	Used to form the hierarchy of a character's body. It can be scaled, rotated or translated to construe motion of a character's body part. Skew has to be derived from a hierarchy of scales and rotations.
GameObject		Always paired with a Transform. GameObjects host Components that can display and simulate gameplay. The SDK provides custom Behaviors that run on GameObjects to display Harmony-specific visuals.
Asset		The core data in a Unity project that is used for visuals, physics, gameplay and sound-effects. Generally these are files, but can be hosted as sub-assets inside of larger files. Textures, AnimationClips, Sprites, AnimatorControllers, AudioClips and Prefabs are all Assets.

Importing Harmony Files into Unity

Once you've finished creating your artwork and cyclable animated character movements in Harmony, it's time to import them into Unity for game integration.

How to import Harmony files directly into Unity

1. Create a Unity 2D project.
2. The SDK Harmony project needs to be imported to your project in order for the assets to be brought in. Do one of the following:
 - Go to the Unity Asset Store (assetstore.unity3d.com/en/#!/content/31211) and save the Harmony Game SDK file on your computer. Then import it in your project from the following: **Top Menu > Asset > Import Package > Custom Package**.
 - Search for the Harmony Game SDK in the Unity Asset Store, then download and import the package. You can access the Asset Store from the following: **Top Menu > Window > Asset Store**.



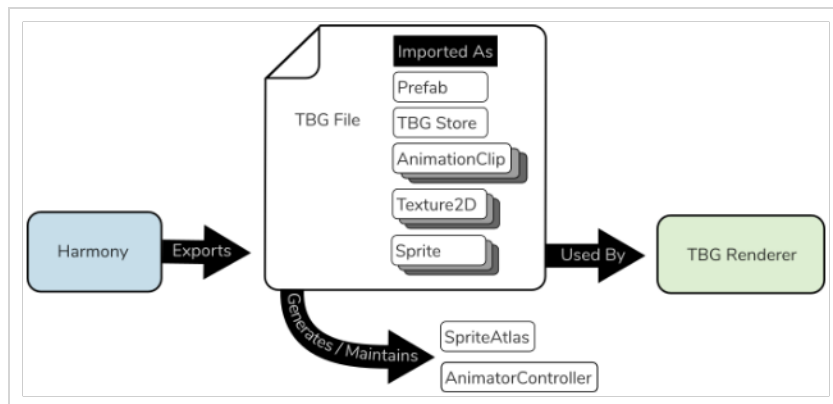
NOTE

For each new Unity project, you must repeat step 2 as it contains the necessary files and folders to make the Harmony import process work.

3. Add the exported game object to the scene by going to Top Menu > GameObject > Harmony
 - Harmony Object: Lets you browse for the exported Harmony project folder. If you've already saved this to your StreamingAssets folder, you can browse to it through there. Then it adds the Harmony scene to your Unity file, and sets up the rendering, audio, and animation scripts you need to get going.
 - Harmony Texture Object: Lets you browse for the exported Harmony project folder similarly to Harmony Object, but creates a Unity plane to render your animation on it.

TBG File Workflow in Unity

In the TBG (Toon Boom Gaming) file workflow, Unity will immediately interpret a TBG file exported from Harmony as a new asset that it can reference to derive new Textures, Sprites, project data, and a Prefab from the file. The resulting Prefab can be dragged into the Unity scene to be immediately viewed and animated. When the TBG file is overwritten from a new export in Harmony, the TBG Importer can automatically perform the re-import process to update the existing prefab, sprites and textures. This allows artists to rapidly iterate on their characters and see their results immediately in-game.




The resulting TBG Renderer prefab contains a full hierarchy of Transforms and SpriteRenderers to display the character in a Unity-friendly rig. Additionally, a TBG Renderer script is included that ensures Sprites are swapped during an animation's progress and cutters properly apply to relevant SpriteRenderers.



NOTE

The XML format used in TBG files is a newer version, and may not be compatible with the XML Folder workflow for more complicated characters. It's a good idea to keep your original Harmony files and re-export them with different settings later.

How to import a TBG file into Unity

1. In Harmony, find the Export to Sprite Sheet  button in the Game toolbar.
2. In the Export to Sprite Sheet window, check **Encode as TBG**.
3. For the Asset Path, enter the file path where you would like to save your character.
4. Select **Save and Export**. A loading bar will appear that will complete the export process and generate a new TBG file.
5. Open your Unity project. Unity should automatically load the TBG as a Prefab.

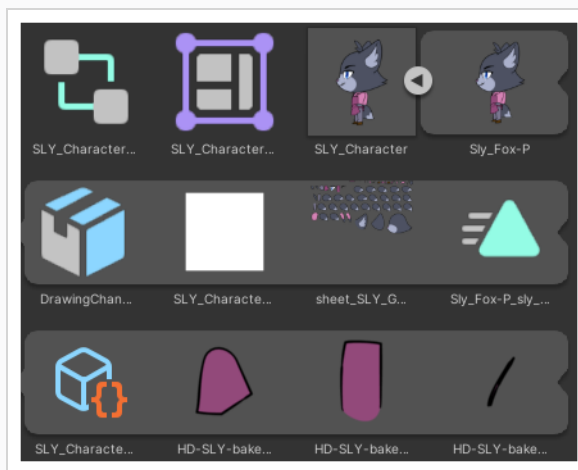
**NOTE**

If your character is not appearing as a Prefab, please ensure:

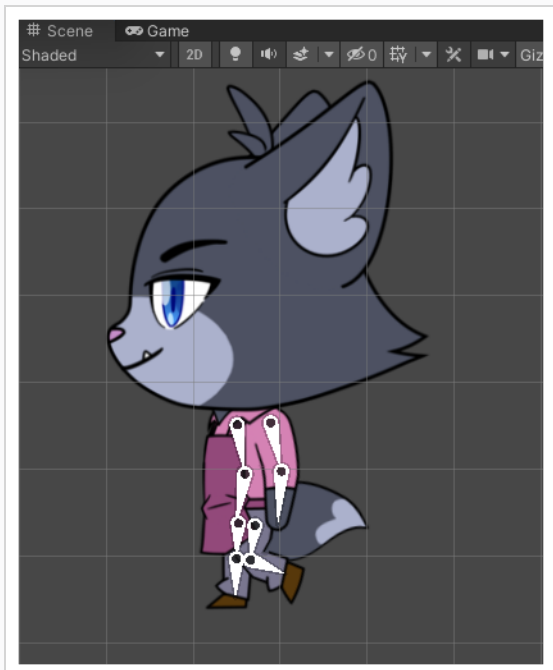
- You have the HarmonySDK installed for your Unity project. It must be installed for every new project created.
- You do not have a scene marker extending beyond the exposure of a character.

In your Unity project, there should be:

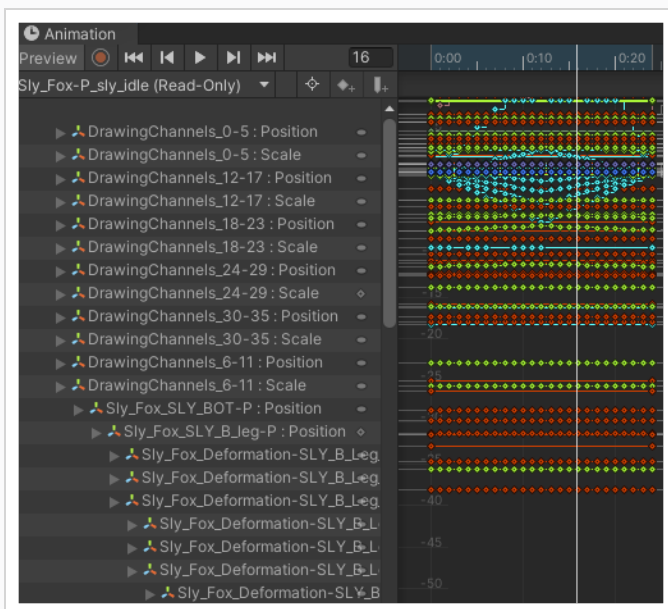
- Sub-assets for Textures, Sprites, TBG Store data, and AnimationClips.
- Two new assets for the file's SpriteAtlas and AnimatorController beside the TBG Prefab.



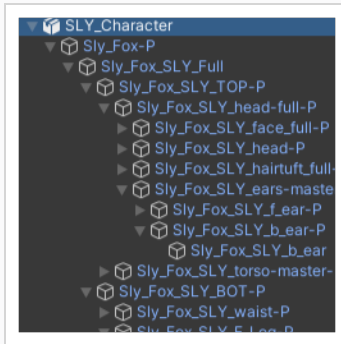
The TBG Prefab can now be dragged into the Hierarchy, where it will be displayed and interactable.



By selecting the Prefab in the Hierarchy, you will be able to view the character's animations from the Animation view.

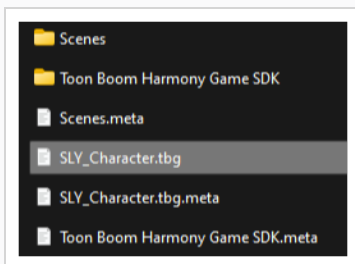


Opening up the Prefab in the Hierarchy view by clicking the small arrow, you will be able to explore the character's hierarchy very similarly to what is shown in Harmony's Timeline.

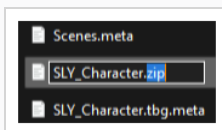


How to manually inspect the contents of a TBG file

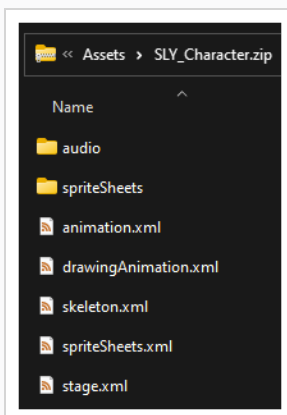
1. Search for the TBG file in your computer's file explorer



2. If you use Windows or Mac, rename the file extension as the .tbq file extension is not recognized by Windows and Mac computers.



Once renamed, Windows and Mac will allow you to explore the contents of the .zip file, so you can ensure your SpriteSheets are reasonably sized or that your data files have the contents you expect.



**NOTE**

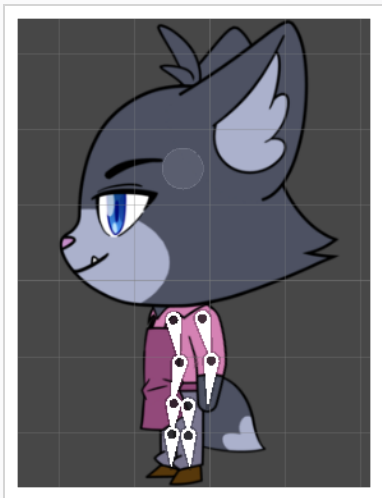
To ensure Unity doesn't have any trouble, you can change the file extension back to .tbg before resuming work with Unity. Generally this inspection process need only be done when you're on a computer that doesn't have Unity installed.

Adding Inverse Kinematics to a TBG Renderer Prefab

You can utilize Unity's own 2D Inverse Kinematics feature to add procedural targets to your characters limbs. You can use SLY_Character from the TBG Workflow example scene for this process.

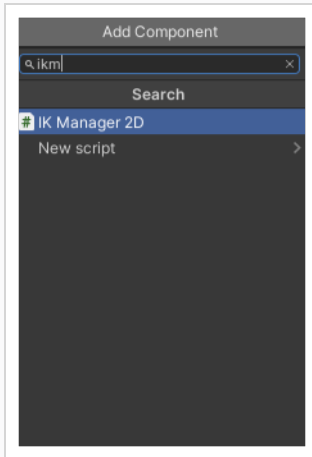
How to add Inverse Kinematics to a TBG Renderer Prefab

1. Find and drag SLY_Character from the Project view to your Scene view, and press F to frame the character in your view.

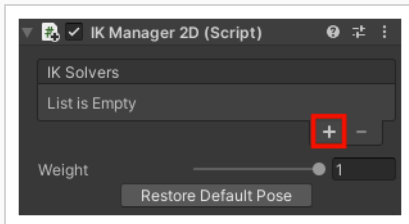
**NOTE**

Ensure the SLY_Character is selected in the hierarchy.

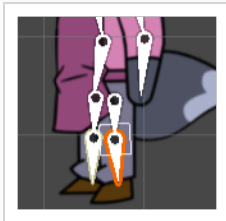
2. In the Inspector view, add an IK Manager 2D component.



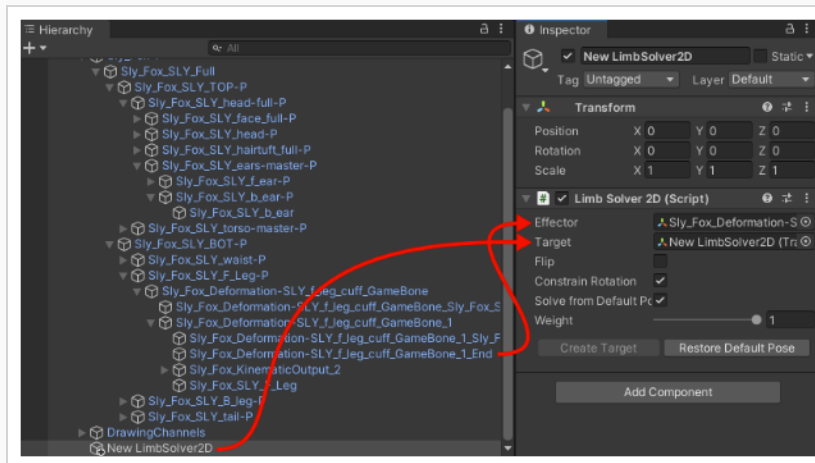
3. In the IK Manager 2D component, click the + button to add a new IK solver, and select **Limb**.



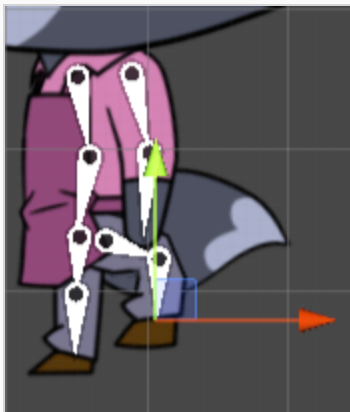
4. Click the bottom-right bone of Sly's legs to expose its place in the hierarchy. We will be using its sibling GameObject that has "..._GameBone_1_end" in the name.



5. With the "New LimbSolver2D" Game Object selected, drag "New LimbSolver2D" into the Target slot of the LimbSolver2D script and drag the "_leg_cuff_GameBone_1_end" GameObject into the Effector slot.



Once this is done, you can move around the New LimbSolver2D transform and see the limb bend to match the target.



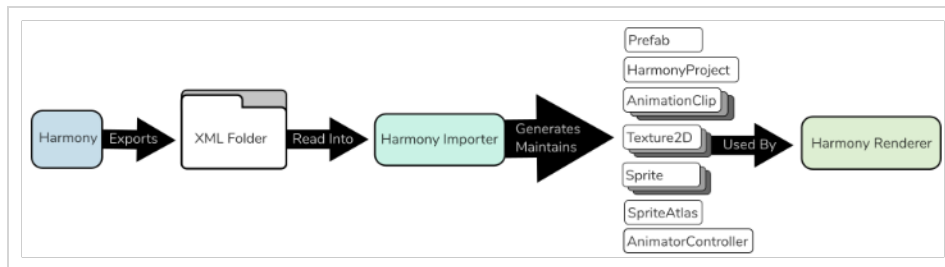
To switch the direction the knee bends, we can click the “Flip” checkbox in the Limb Solver 2D component.

XML Folder Workflow in Unity

In the XML folder workflow, a unique Harmony Importer asset can be created by a developer to reference an XML folder exported from Harmony. This importer can then be used to generate a Harmony Project asset and a new Harmony Renderer prefab.


For each re-exported XML file, developers need to manually set import and export paths and perform the import process from the Harmony Importer asset.

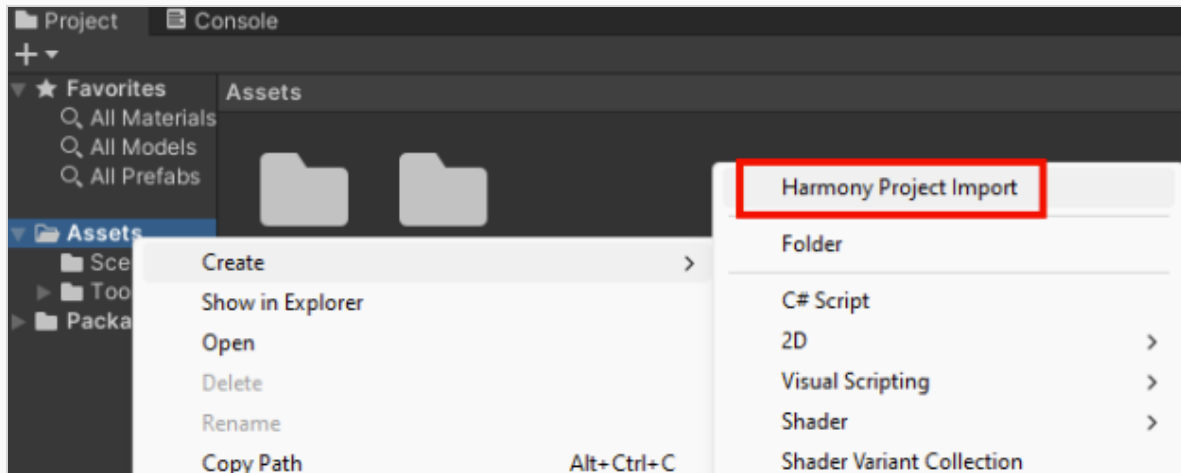
This workflow provides flexibility as to where the XML data can live. For example, it can live either inside or outside the Unity project. However, a disadvantage to this workflow is that more mistakes can be made since there are more steps involved. For example, the file path for the TBG importer will have to be manually updated when moving files around.



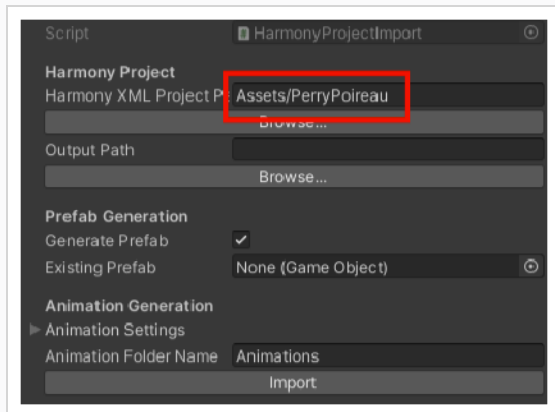
The resulting Harmony Renderer prefab allows high-performance character rendering by generating a mesh within a C++ plugin. This is preferable for crowds of characters that don't require complicated custom rendering features.

How to import an XML folder into Unity

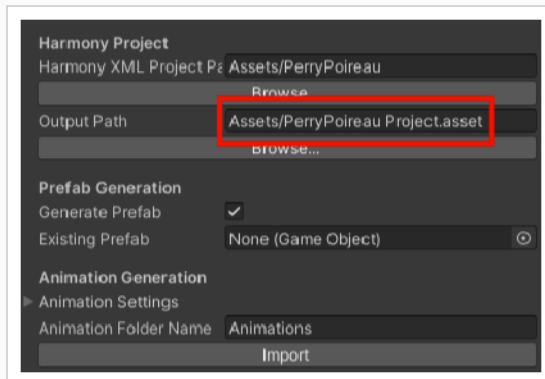
1. Within Harmony, find the Export to Sprite Sheet  button in the Game toolbar.
2. Select **Save and Export**. Ensure that the Export as TBG option is unchecked in the Export to Sprite Sheet dialog.
3. Save the XML to a folder inside of your Unity project's Assets folder.
4. In Unity, create a new Harmony Importer asset by doing one of the following:
 - In the Assets drop down in the top menu, select **Create > Harmony Project Import**.
 - Right-Click the Assets folder in the Project view and select **Create > Harmony Project Import**.



5. Name the Harmony Importer with an "Importer" suffix to avoid future confusion.
6. Copy relative path of exported folder, paste into new Harmony Importer.

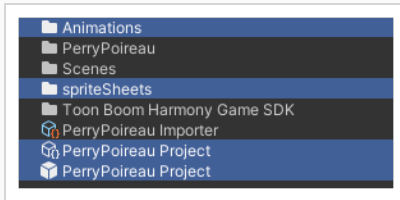


7. Copy relative path of Harmony Importer asset, paste into Harmony Importer, replace "Importer" with "Project" in the path.

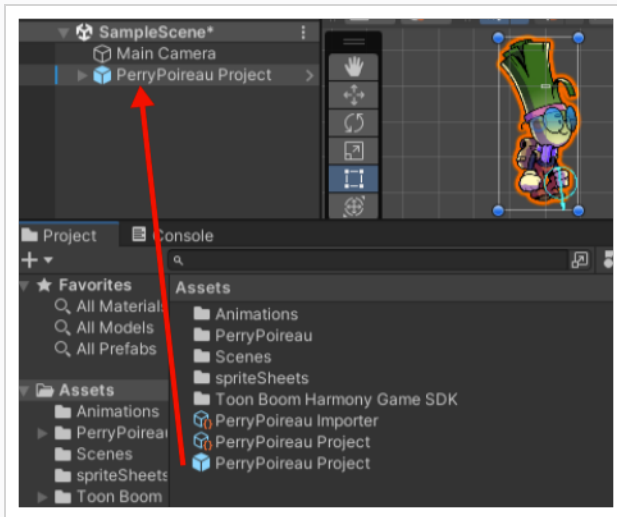


8. Click **Import**.

A new HarmonyProject asset will be created as well as a new Prefab with Harmony Renderer that references HarmonyProject.



The Prefab asset can be dragged from the Project view into the Hierarchy view and Scene view to show your character in your game

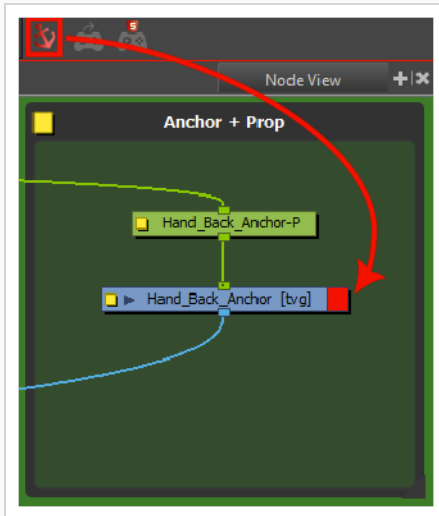


Importing Anchors

In Harmony, Anchors provide a way to indicate which pegs and drawings are positions of interest for gameplay scripts in Unity. Any drawing or peg can be assigned an Anchor.

How to import an anchor to Unity

1. Select a layer in the Timeline view or node in the Node view and select the Toggle Anchor button in the Game toolbar.



2. Import the character to Unity.

The next time the XML import process is performed, the new Anchor will appear in Unity under the character's imported Prefab as a new HarmonyAnchor.



This new GameObject will follow the position, rotation and scale of the current animation playing on the character. This allows you to dynamically attach gameplay objects as children to the anchors to carry, equip and use.



NOTE

Moving anchor positions will not affect the Harmony Renderer visuals. To view and affect the entire structure of the character in Unity and not just the anchors, you may want to consider using the TBG workflow

Adding Audio Sources

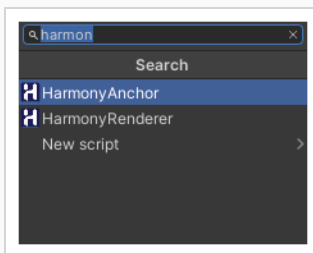
When you create scene files in Harmony with audio, the audio is exported into the project folder. The Harmony Importer will automatically create an AudioSource and add it to the generated Prefab. The Harmony Renderer can then trigger playback of the AudioClip files at points during animation playback, as was authored within Harmony.

Adding a Harmony Renderer to Empty Game Objects

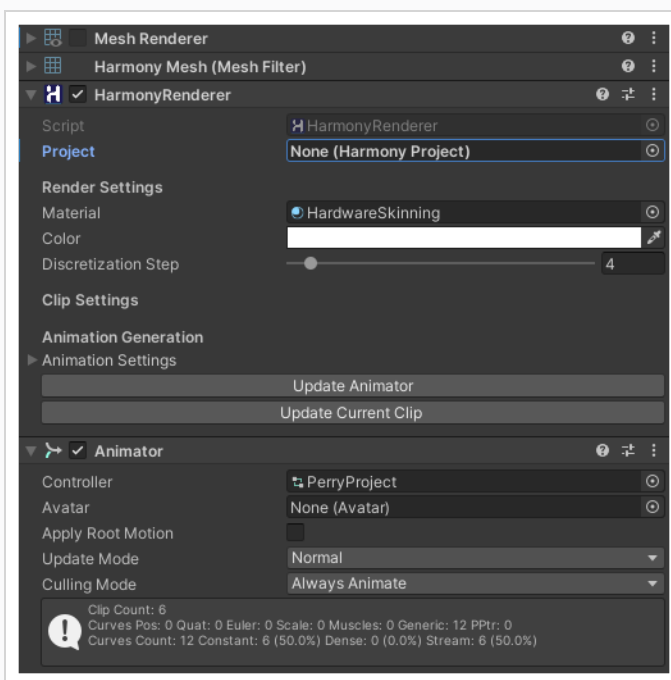
If you don't have the prefab from your Harmony Importer, it's possible to still create a new Harmony Renderer component on an empty Game Object to achieve the same end result.

How to use an empty GameObject

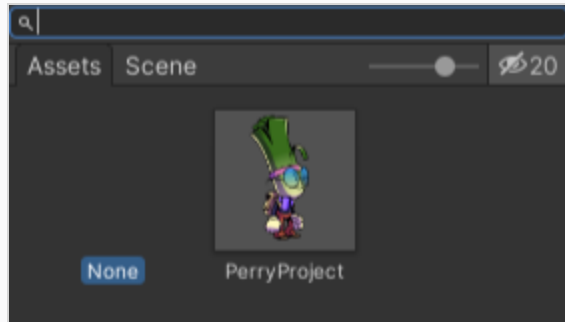
1. Select **GameObject > Create Empty**.
2. Rename the empty GameObject so it's clear in the Hierarchy. Since we're using the PerryPoireau demo file, rename the GameObject to **SpaceCat** by doing one of the following:
 - Double-click on the name and rename it in the Hierarchy.
 - Select the GameObject and rename it in the Inspector.
3. At this point, it's an empty object. Accessing Harmony data is done through scripts.
 - Select the PerryPoireau GameObject.
 - In the Inspector, go to **Add Component > Scripts > Harmony Renderer**.



Four new sections appear in the Inspector called Mesh Renderer, Harmony Mesh (Mesh Filter), and Harmony Renderer.



From here you need to attach a Harmony Project to this newly created Harmony Renderer. By clicking on the circle at the right of the empty (Harmony Project) slot, you can select the project you have already imported using the steps to create a Harmony Project using the XML Folder workflow.



Setting Up Collisions in Unity

Extracting the bounding box information is useful when you want to make something collide with your character. When you put a 2D character into a 3D scene, you may want to use Physics to make the 2D character collide with the 3D plane.

Colliders were improved to provide several different types of colliders for use in Unity. You can select polygon colliders and box colliders for a more accurate bounding box.

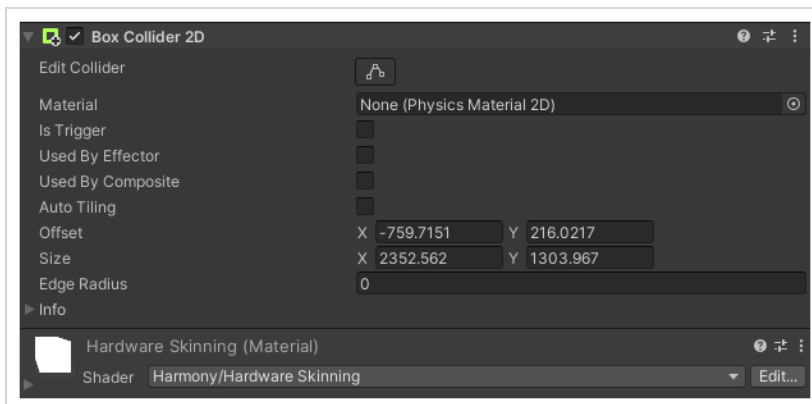
How to set up collisions

1. Select your imported character.
2. In the inspector, select **Add Component > Physics 2D > Box Collider 2D**.
3. To enable the Physics on the character, go to **Add Component > Physics 2D > Rigid Body 2D**.



NOTE

You can also use 3D physics on the imported files.



You can now adjust the variables and see the results.