Harmony
Essentials

Toon Boom Harmony Essentials V15.0

Gaming Guide

Toon Boom
Bring Your Stories To Life

TOON BOOM ANIMATION INC.
4200 Saint-Laurent, 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

Trademarks

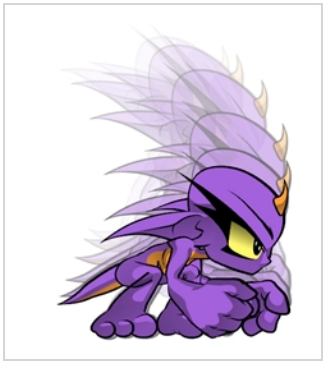
Publication Date

01-15-2020

Table of Contents

Chapter 1: Gaming Guide

Harmony can be used to create and animate characters that can be imported in Unity. The procedure is very similar to rigging and animating characters for a regular animated production, and requires no programming skills. However, you must follow some guidelines when creating and animating your character to make sure it is properly imported in Unity.

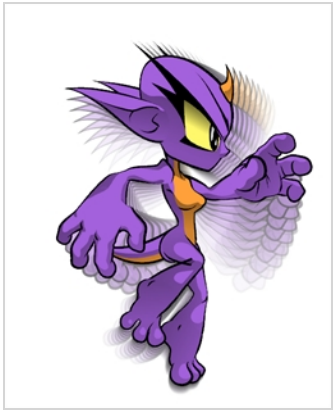


The procedure for creating game assets in Harmony can be broken down into the following steps.

- Rig and animate your characters in Harmony, while following the guidelines for rigging and animating game characters in this guide.
- Separate your character's different animation clips.
- Create anchors and props for your character, if needed.
- Export the character from Harmony into a sprite sheet format that can be imported in Unity.
- Import the exported data into Unity.

Rigging Guidelines

The following section explains how to create game assets in Harmony. You should go over this section before you start creating your character rigs, as it not only contains useful guidelines and tips for rigging game characters in Harmony, it also contains important information on what you can and cannot do when using Harmony to rig and animate game characters.



Exporting Rigged Sprite Sheets or Rendered Sprite Sheets

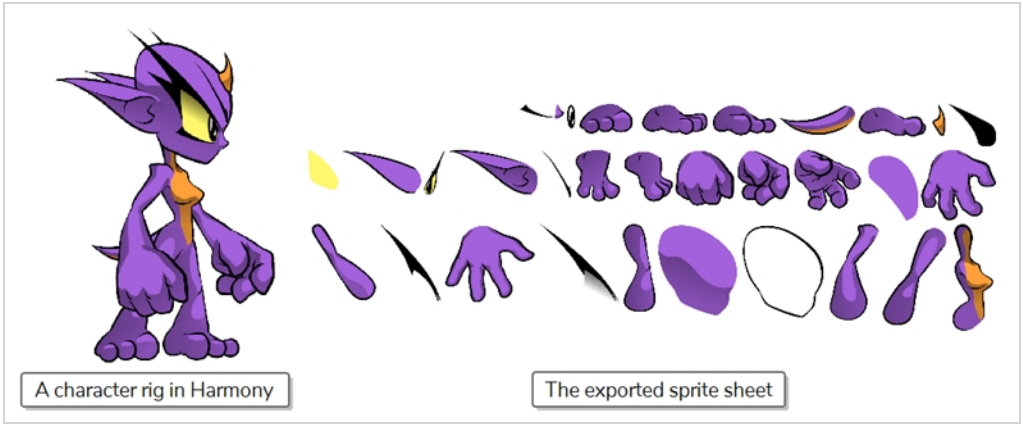
When creating game assets in Harmony, the first decision you should take is which type of export you intend to make:

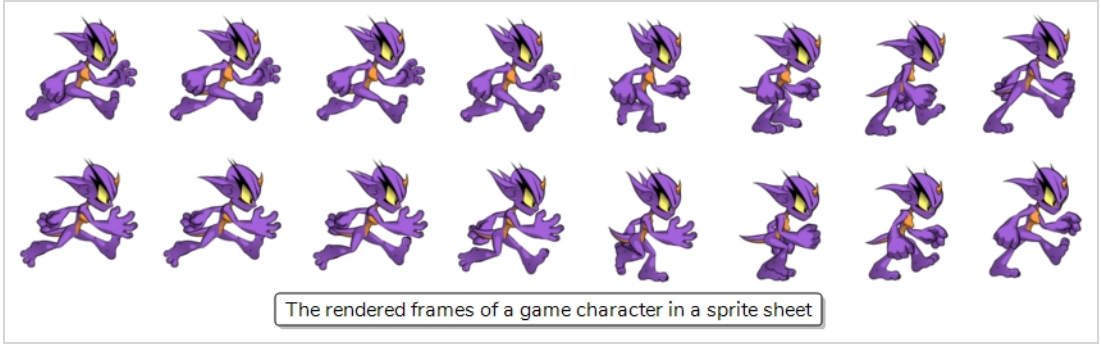
- **Rigged Sprite Sheets** is where you export a sprite sheet with one sprite for each layer in your character along with metadata allowing Unity to build and animate your character using those sprites. This is the traditional way of exporting sprite sheets, and most other Harmony features related to gaming are optimized for it.

  > **NOTE**
  >
  > In Harmony, this is simply referred to as exporting sprite sheets.

- **Rendered Sprite Sheets** is when you export a sprite sheet with one sprite representing your entire character, rendered and flattened, for each animation frame in your scene. This means that Unity does not have to reproduce the way Harmony renders your character, as it is already rendered. This method creates significantly heavier packages, does not work with some other gaming features in Harmony and is harder to import in Unity, but it ensures that what you see in Harmony is also what you will see in your game.

  > **NOTE**
  >
  > In Harmony, this is referred to as exporting to Easel JS.



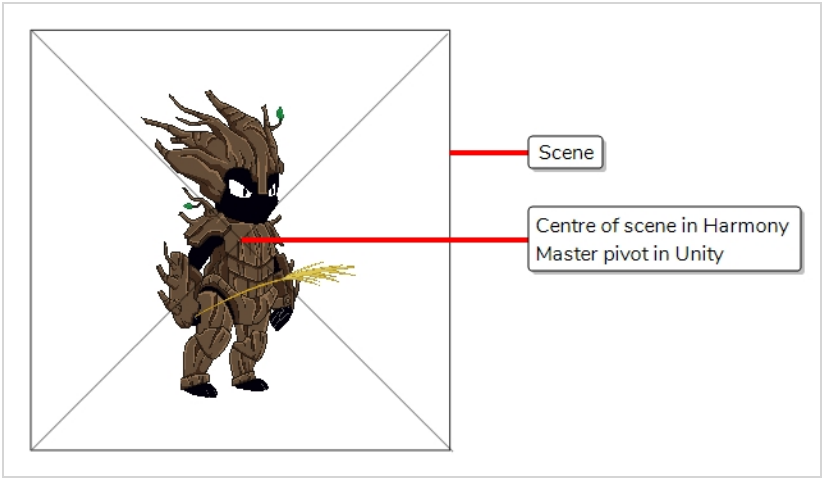


The following table offers a detailed comparison of all the advantages and limitations of between the traditional method of exporting sprite sheets versus exporting to Easel JS:

	Advantages	**Disadvantages**
Rigged Sprite Sheets	• Significantly lighter packages. • Allows you to easily import the assets in Unity using the Harmony Game SDK package. • Allows you to create props and anchors in Harmony.	• Only supports *Game Bone*-type deformations. • Does not support any effect except for the Cutter. • Using cutters has several limitations. • The matte port of drawing layers cannot be used.
Rendered Sprite Sheets (Easel JS)	• All methods of animating, including morphing, curve deformations and envelope deformations are supported. • Effects are supported. • What you see is what you get.	• Harder to import in Unity. This format cannot be imported using the Harmony Game SDK. Developers must write their own scripts to import the exported sprite sheets. • Exported sprite sheets are significantly heavier in file size, which may make it unsuitable for mobile games due to file-size and bandwidth limitations. • Does not allow you to create prop and anchors in Harmony.

Scene Set-Up Guidelines

The following are guidelines you must follow when creating and setting up scenes in which you intend to create game assets for Unity.

Scenes	You must create one scene for each character. A scene can contain all of a character's different animations and props, but should not contain several characters that need to act independently from each other.
Resolution	Your scene's resolution should be square, meaning its width and height should be equal. For example, 1024 x 1024 is a proper scene resolution for a game object. It is highly recommended to use a power of two for the dimensions of your scene, such as 64, 128, 256, 512, 1024, etc. This is because graphics cards naturally handle textures with such dimensions. While recent video cards will adjust to non-standard texture sizes, which might cause some overhead and slow down your game's performance, older video cards might not be able to display these textures at all.
Main Pivot	When imported in Unity, the master pivot of your GameObject will be exactly where the centre of your scene is. Hence, if you want the pivot to be at a specific position on your character in Unity, you should position your character accordingly in your Harmony scene. For example, if you want the pivot point of a bipedal character to be between its two feet, at the centre of its drop shadow, you should position this character so that the point in the middle of its feet is in the centre of the scene. 
Naming	**Do not use non-alphanumerical characters in scene, layer, drawing and group names. Only use letters, numbers, dashes and underscores. Unity does not support Unicode characters in names.**

Guidelines for Rigging Game Characters in Harmony

The following are guidelines to follow when creating a character rig in Harmony if you plan to export it for use in Unity.

NOTE

You do not need to follow these guidelines if you intend to export a rendered sprite sheet—see Exporting Rigged Sprite Sheets or Rendered Sprite Sheets on page 7.

Resources Usage

When rigging and animating game sprites, you need to keep in mind the size of the exported sprite sheets as well as the resources required to play out their animations.

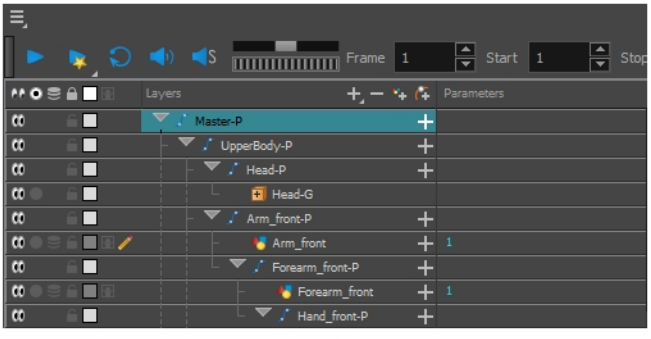
If you are working on a console or PC game, you have a lot of leeway for both, as they often have high performance video cards and a lot of storage space.

If you are working on a mobile game, you might need to take measures to ensure your animation and sprite sheets are lightweight, as mobile devices have slower rendering hardware and more limited storage space. Also, some users may want to download your game over a mobile data network with costly data charges.

Hierarchy

When creating your character rig's structure, you should use a simple peg-layer hierarchy, which means:

- Each body part of the character that you want to animate independently is on its own drawing layer.
- Each drawing layer has its own parent peg, which should be used for animating the layer.
- Pegs can be rigged in a hierarchy to make body parts attached to each other move together. For example, a hand's peg can be rigged under the forearm's peg, and the forearm's peg under the arm's peg, etc.
- You can create groups to keep your rig structure tidy and organized.



Body parts that are meant to move together with other body parts (i.e.: the head with the neck, the forearm with the arm, etc.) can be rigged together in either one of these ways:

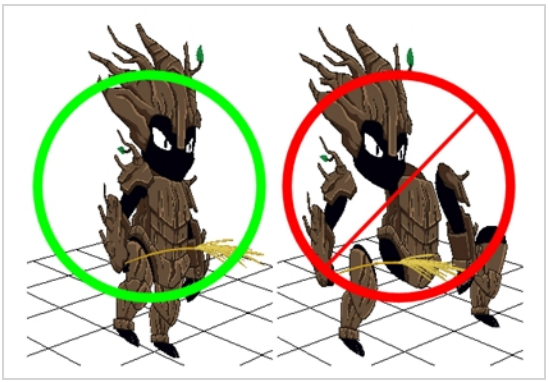
- By rigging the peg of the child part under the peg of the parent part. For example, you could rig the forearm's parent peg to be the child of the bicep's parent peg. Moving the bicep's peg

would automatically move the forearm.

- By rigging both parts under a common peg. For example, you could rig the forearm and the bicep's parents pegs under a dedicated peg for the whole arm. You could hence move the bicep's peg independently, or move the arm peg to move both the bicep and the forearm together.

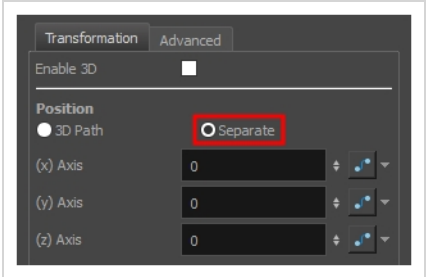
z-Axis Positioning

Nudging layers on the z-Axis to change the order in which they appear is supported and its effect will be visible in Unity. However, you should not attempt to put layers at significant distances from each other on the z-Axis in Harmony, as they will not be rendered in perspective in Unity.



Animation functions

By default, pegs use 3D Path functions to record their position. It is recommended, but not required, to configure your pegs to use Separate position functions instead, as 3D Path functions are a bit heavier on performance. You can make a peg use separate functions by opening its Layer properties, then selecting **Separate** under the Position section.



NOTE

Converting a peg or layer's function type to Separate will discard any existing animation on that peg or layer.

How to make a peg's position functions separate

1. Do one of the following to access the peg's Layer Properties:
   - Open the Layer Properties view, then select the peg layer in the Timeline view.
   - In the Timeline view, double-click on the peg layer, but not inside its name, to open its Layer Properties dialog.
2. In the **Transformation** tab of the peg's Layer Properties, under **Position**, select **Separate**.

Effects

Aside from the Cutter, you should not use any effect, as no other effect supported by Harmony can be exported to Unity.

NOTES

- This includes Colour-Override and other simple colour effects. No such effect will be exported to Unity.
- The Cutter effect also has limitations when rigging a character for gaming—see *Guidelines for using the Cutter Effect for Game Assets in Harmony* on page 13.

Bitmap

The use of bitmap image files (i.e.: **`.png`** or **`.tga`** files) in your rig is not supported.

Symbols

The use of symbols is not supported.

Guidelines for using the Cutter Effect for Game Assets in Harmony

If you intend to use cutters in a character rig that you plan to import in Unity, you must follow the following guidelines to ensure your character displays as expected in your game.

NOTE

You do not need to follow these guidelines if you intend to export a rendered sprite sheet—see Exporting Rigged Sprite Sheets or Rendered Sprite Sheets on page 7.

No Consecutive Cutters

You can only connect a single cutter under any drawing. If you connect several cutters in a chain, only one of them will take effect in Unity.

No Using groups as Mattes

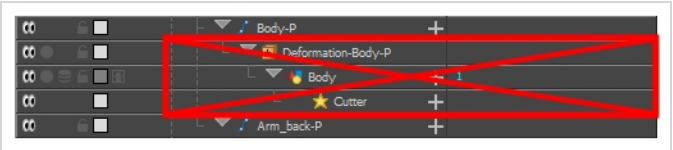
You can only connect individual drawings to the matte connection of a Cutter. You cannot use a group or effect as the matte of a Cutter.

No Using the Drawing Layer's Matte connection

Although drawing nodes have their own matte connection, you should not use them. You must use Cutter effects if you want to mask parts of certain layers.

No Cutting a Deformed Drawing

You cannot connect a cutter under a drawing that is under a deformation.



Guidelines and Tips for Animating Game Assets in Harmony

The following are guidelines to follow when animating a character in Harmony if you intend to export it for use in Unity.

NOTE

You do not need to follow these guidelines if you intend to export a rendered sprite sheet—see Exporting Rigged Sprite Sheets or Rendered Sprite Sheets on page 7.

Animation on Pegs	When animating game sprites in Harmony, you should animate the body parts on their parent pegs or their deformations, and not on the drawing layers directly, to keep the drawing information and the animation information separate.  **TIP** To facilitate this workflow, you can enable the Peg Selecton Mode option in the Tool Properties of the Transform tool. This will ensure that, when you click on a drawing layer in the Camera view, its parent peg is selected instead.
Animation Techniques	It is recommended to use geometrical transformations—done with the Transform tool or the Advanced Animation tools—to animate the body parts of your rig as much as possible, as those are easier for Unity and for graphics cards to render. You should only use deformations, drawings swaps or frame by frame animation when necessary.
Deformations	If you want to use deformations for your animation, you must use Game Bone deformations. Game Bone deformations are similar to regular Bone deformations, but have less options and are designed to render the way Unity would render them—see About Game Bone Deformations on page 62.
Exported Assets	Drawings that are not used by your character or its props will not be exported with the sprite sheet. Hence, you do not need to delete unused drawings from your scene to optimize the exported assets.

Guidelines for Adding Audio to your Game Assets

The Harmony Game SDK supports exporting sound clip from a Harmony scene into a Unity GameObject. However, there are important limitations to consider if you plan to do this.

No Clipping	If you clip your sound effects in Harmony, they will still be imported as whole in Unity. Hence, you should edit your sound effects in a sound editing software before importing them in Harmony. Likewise, if a sound effect extends past the end of the scene, it will not be cut off in Unity. It will play until the end.
No Volume Adjustments	If you adjust the volume envelope of a sound effect in Harmony, these adjustments will not be exported to Unity.
Importing into Unity	Sound effects exported with your Harmony assets will not automatically play with your animations in Unity. The developer must themselves program the logic for playing sound clips.

Because of these limitations, it is highly recommended to only import audio in your Harmony scene and export them with your sprite sheets when necessary, such as when you need to lip-sync your character's mouth with their dialogue. Otherwise, it is recommended to add your sound clips in Unity instead.

Creating Multiple Clips for a Game Object in Harmony

A game sprite typically has more than one animation. With Harmony, you can create all of your character's animation clips in the same scene, provided that you separate them properly.

To separate your clips, you must create one scene version for each clip you want to make. To accomplish this, you must proceed as follows:

1. First, you create a scene and, in this scene, create the full rig for your character, without animating it.
2. Then, you save a new version of your scene for the first animation clip you want to create. To do this, select **File > Save As New Version** from the top menu. Give this scene version the name you want your clip to have. For example, if you want to create your character's idle animation, you could name this scene version *Idle*.
3. In this scene version, create your character's animation. Make sure the animation starts as soon as the scene begins and that the scene ends as soon as the animation is over.
4. To create another clip, repeat steps 2 and 3 until you have all the clips you need for your character.

In the end, your scene and its versions may have something like this:

Scene	**Version**
SpaceDuck	• Idle • Run • Jump • Shoot

About Anchors and Props

Anchors and props are a feature of Unity allowing you to attach a prop, such as an item or a weapon, to a body part in your character rig. This is done by creating an anchor on your rig, which is a point that can be moved and rotated, and to which the prop can be bound. For example, you could make the hand of your character an anchor, and the weapon the character wields with that hand a prop.

A prop can be enabled and disabled in Unity. Also, a single prop can have several different drawings in it, which can be switched in Unity at any time. Hence, creating props for your character allows them to carry or wear different objects and equipment without rendering a sprite sheet of your character for each object.

NOTE

To be able to export your character with anchors and props, you must export your character as a rigged sprite sheet—see *Exporting Rigged Sprite Sheets or Rendered Sprite Sheets* on page 7.

Creating an Anchor for a Game Character

An anchor is simply a layer that has been marked as an anchor. Creating anchors is required to make props follow the animation of the layer they are rigged to.

In Unity, an anchor is a point, whereas a layer in Harmony is more like a canvas. Hence, the exact position of the anchor is based on the position of the layer's pivot point in Harmony. Because of this, it is recommended to create a layer specifically for the anchor in Harmony, instead of using one of your character's existing layers, so that you can position its pivot point without affecting your character's animation.

For example, if the character's hand is meant to be an anchor for a weapon prop, the hand's pivot point would usually be on the wrist to optimize animation, but you'd want the anchor for this hand to be positioned in the palm. Hence, you could create an empty drawing layer, rigged under the same peg as the hand, make this layer the anchor, and position its pivot point in the hand's palm.

NOTE

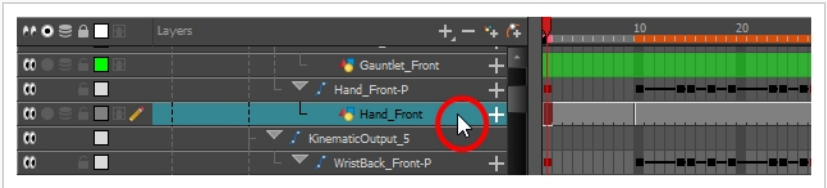
For a layer to work as an anchor, it must have a drawing, because drawings store pivot points. Hence, if you want an empty layer to be used as an anchor, you must put an empty drawing in it so that you can set its pivot point.

How to create an anchor

1. If the Game toolbar is not already in your workspace, do one of the following to add it:
   - In the top menu, select **Windows > Toolbars > Game**.
   - Right-click on any existing toolbar and select **Game**.

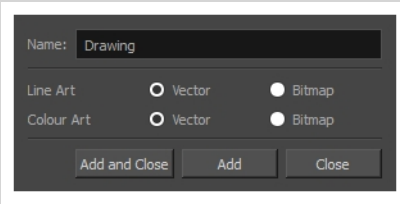
   The Game toolbar is added to your workspace.

2. In the Timeline view, scroll to and select the layer for the body part which you want to use as an anchor.



3. Create a new drawing layer by doing one of the following:

    - In the toolbar over the layers list, click on Add Drawing Layer
    - Right-click on the layer and select **Insert > Drawing Layer**.
    - Press Ctrl + R

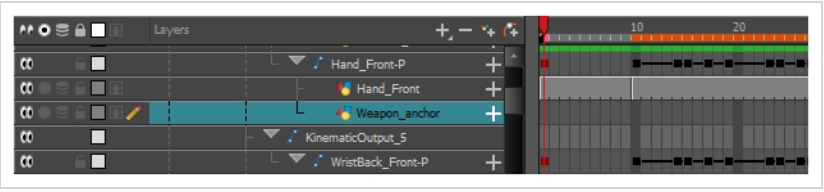


4. In the Add Drawing Layer dialog, type in a name for your anchor.

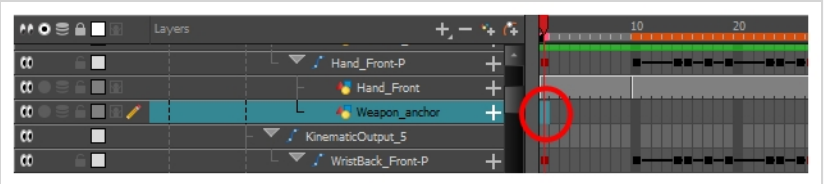
> **TIP**
>
> To make it easier to associate your anchor and its props in Unity, it is recommended to give your layer a name that indicates which prop it will be attached to. For example, if you are creating an anchor for a prop layer named **`Weapon`**, you could name it **`Weapon_anchor`**.

5. Click on **Add and Close**.

    The anchor layer will be created as a sibling to the body part you previously selected. If you animated this body part's parent peg, the anchor layer will move at the same time as that body part.

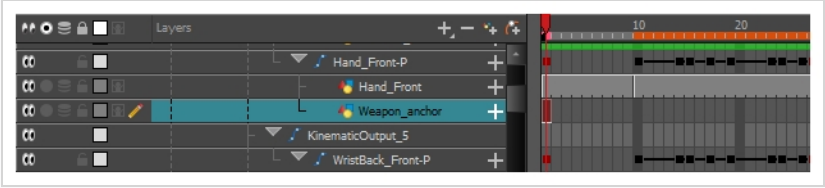


6. In the Timeline view, select the first frame in the anchor layer.



7. In the anchor layer, create an empty drawing by doing one of the following:

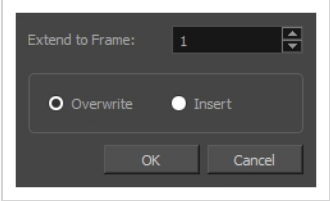
    - In the Timeline view, right-click on the first frame of the layer and select **Drawings > Create Empty Drawing**.
    - In the top menu, select **Drawing > Create Empty Drawing**.
    - Press Alt + Shift + R.



8. Extend the exposure of the empty drawing until the end of your scene by doing one of the following:

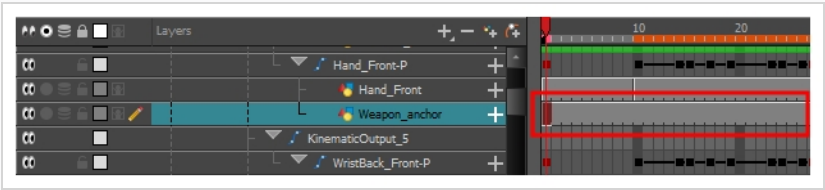
    - In the top menu, select **Animation > Cell > Extend Exposure**.
    - Press F5.

    The Extend Exposure dialog appears.

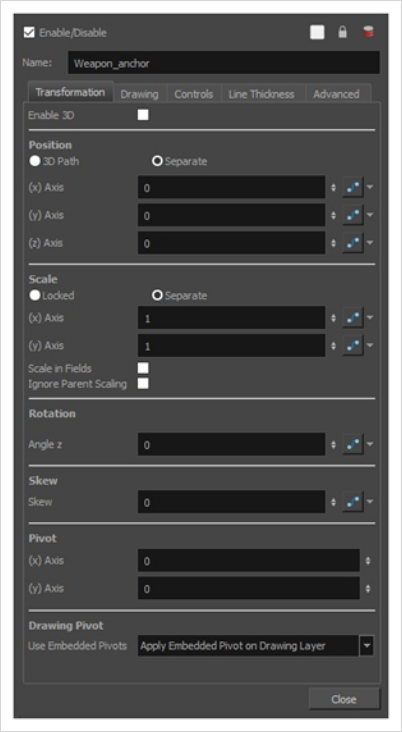


9. In the **Extend to Frame** field of the Extend Exposure dialog, type in the amount of frames in your scene.
10. Click on **OK**.

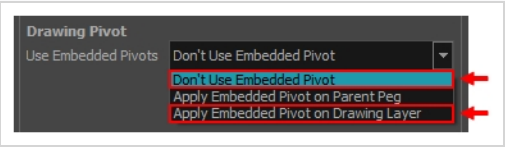
    The exposure of the anchor drawing is extended until the end of the scene.



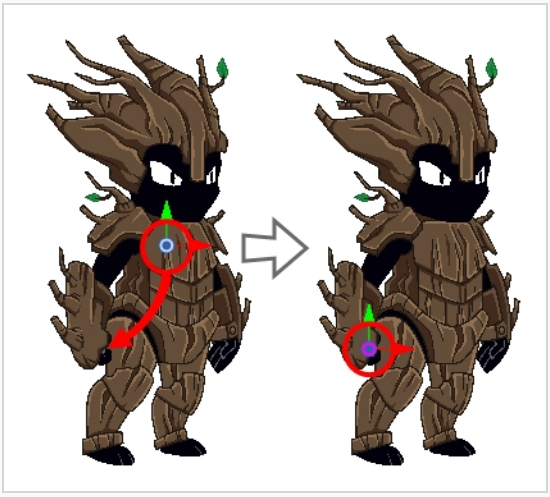
11. In the Timeline view, double-click on the anchor layer (but not on its name) to open its Layer Properties dialog.



12. In the **Transformation** tab, under the **Drawing Pivot** section, open the **Use Embedded Pivots** drop-down and select either **Don't Use Embedded Pivot** or **Apply Embedded Pivot**.

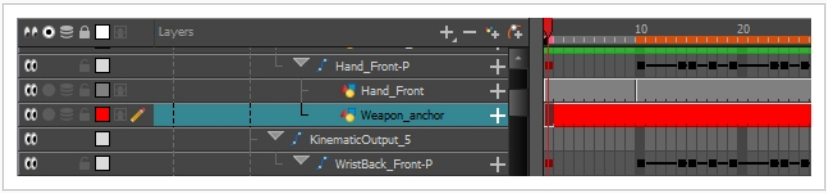


13. Close the Layer Properties dialog.
14. In the Advanced Animation toolbar, select either the **Translate**, **Rotate**, **Scale** or **Skew** tool.
15. In the Camera view, click and drag on the pivot point and position it where you want the anchor to be positioned.



16. In the Game toolbar, click on **Toggle Anchor**.

    The layer will now be an anchor. Its colour in the Timeline view will be changed to red.



    In the **Metadata Editor** view, its **Node Metadata** should contain the following entry:

    `game.isAnchor`

    You now have an anchor layer for which you can create props.

Creating Props for Game Characters

A prop is simply a drawing layer that has been marked as a prop. To make sure it follows its anchor's animation, it must be either rigged as a child or a sibling of the anchor in the rig hierarchy.

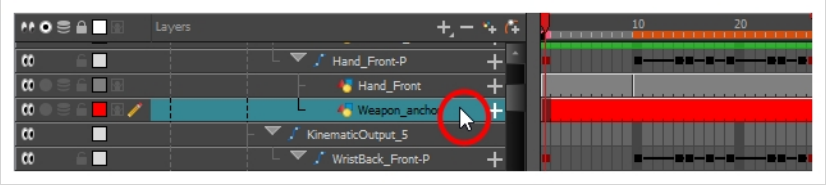
A prop can have one or several drawings. All drawings in a prop will be exported with your character, and it will be possible to switch between the different drawings for your prop in Unity.

How to create a prop

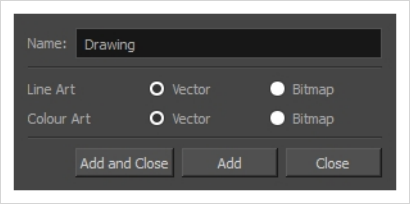
1. If the Game toolbar is not already in your workspace, do one of the following to add it:
   - In the top menu, select **Windows > Toolbars > Game**.
   - Right-click on any existing toolbar and select **Game**.

   The Game toolbar is added to your workspace.

2. Select the anchor layer to which you want your prop to be attached.

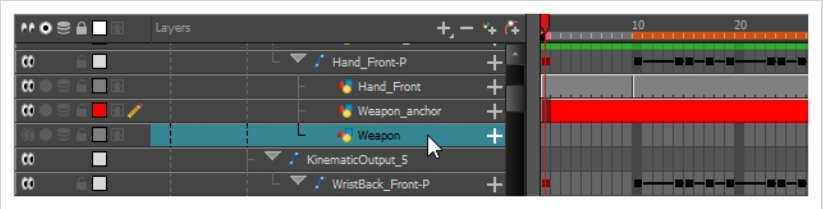


3. Create a new drawing layer by doing one of the following:
   - In the toolbar over the layers list, click on Add Drawing Layer
   - Right-click on the layer and select **Insert > Drawing Layer**.
   - Press Ctrl + R



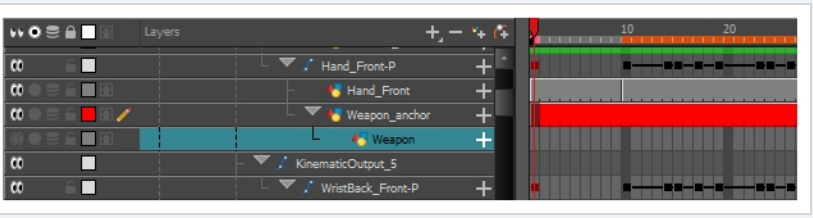
4. In the Add Drawing Layer dialog, type in a name for your prop layer.
5. Click on **Add and Close**.

The prop layer will be created as a sibling to the anchor layer you previously selected. If you animate the anchor layer's parent peg, the prop will also move along with it.

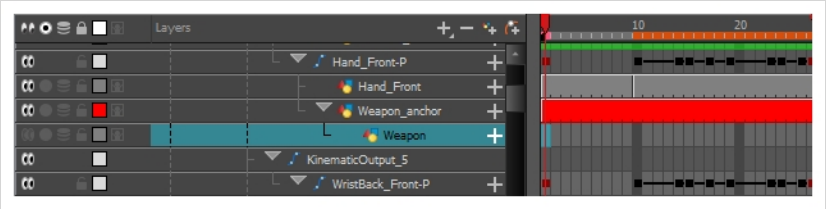


TIP

You can also rig the prop to be a child of the anchor layer, like so:

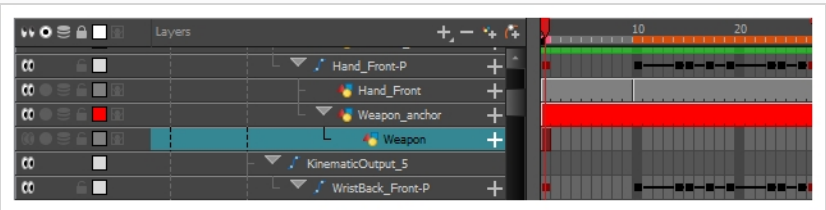


6. In the Timeline view, select the first frame of the prop layer.



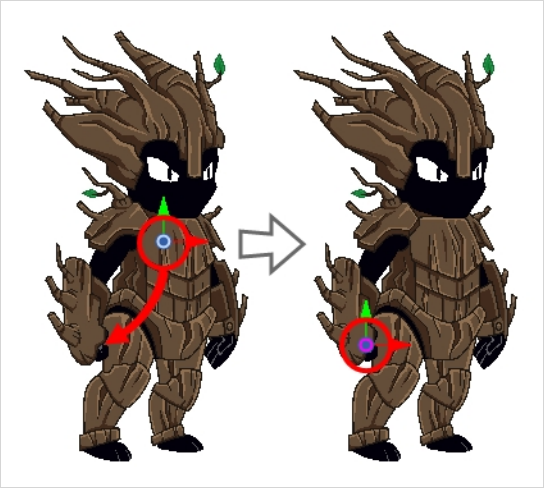
7. Create an empty drawing in the prop layer by doing one of the following:

   - In the Timeline view, right-click on the first frame of the layer and select **Drawings > Create Empty Drawing**.
   - In the top menu, select **Drawing > Create Empty Drawing**.
   - Press Alt + Shift + R.



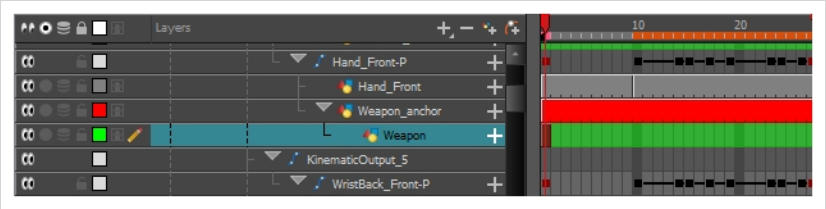
8. In the Advanced Animation toolbar, select either the **Translate**, **Rotate**, **Scale** or **Skew** tool.

9. In the Camera view, click and drag on the pivot point and position it where you want the anchor to be positioned.



10. In the Game toolbar, click on **Toggle Prop**.

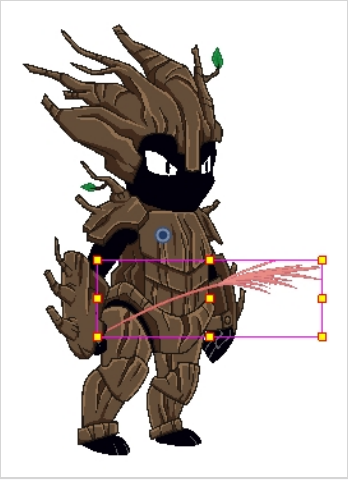
The layer will now be a prop layer. Its colour in the Timeline view will be changed to **green**.



In the Metadata Editor view, its Node Metadata will contain the following entry:

```
game.isProp
```

You can now start drawing your props in the prop layer.



TIP

You can create several drawings in a single prop layer. You will get to toggle between those drawings after attaching the prop to the anchor in Unity.

Adding Metadata Notes to Game Props

You can add metadata notes to props, which will have a name and value. This metadata will be accessible in Unity. It will be visible in the Inspector view when the prop is selected in the Hierarchy view

You can use this mechanic to add properties to your props in Harmony, then use them in Unity. For example, if a character has a weapon for a prop, you could add the attack power of that weapon as a metadata note for the prop.

How to add metadata to a prop

1. If the Metadata Editor view is not in your workspace, do one of the following:
   - 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**.
2. In the Timeline view, select the prop layer to which you want to add metadata.

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

3. In the Node Metadata section, click on the + **Add Node Metadata Entry** button to create a new metadata entry.

   The Add Metadata dialog box appears.

4. In the Add Metadata dialog box, type in **`game.`** followed by the name of the metadata variable you want to add.

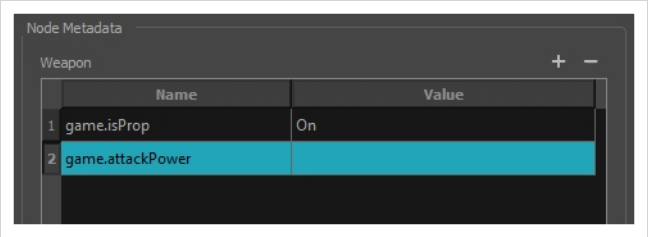
   NOTE

   For the prop's metadata to be imported in Unity, its name must begin with **`game`** followed by a period.

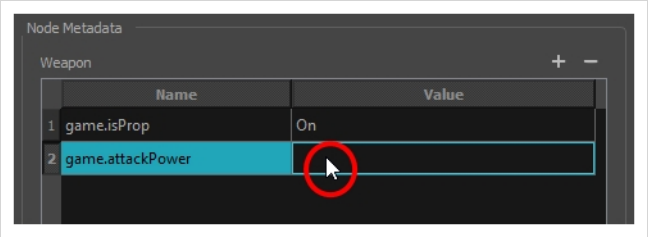
5. Do one of the following:

   - Click on the **Add** button if you want to add more metadata entries.
   - Click on the **Add and Close** button to add this metadata entry and close the Add Metadata dialog.

   The metadata entries you created as added to the Metadata Editor view, in the Node Metadata section.

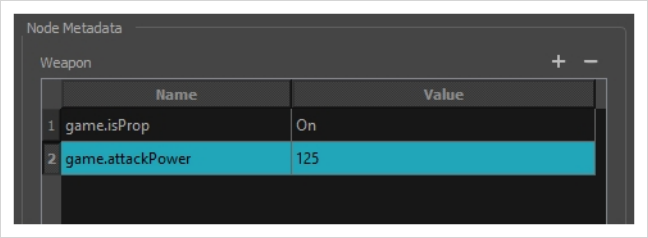


6. In the Metadata Editor view, double-click on the cell right of the name of the new metadata entry, in the Value column, to edit the entry's value.



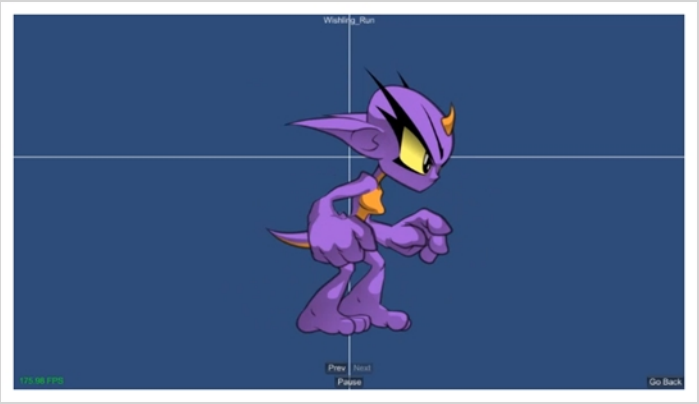
7. Type in the value for this metadata entry and press ENTER.

   You now created a metadata entry with a name and value for your prop.



Using the Harmony Game Previewer

If you are not a developer and would like to see if your animations and rigs will work once imported in Unity, you can use the Harmony Previewer application. You can visualize your characters in the Unity game engine with no programming required.



To get the latest version of the Harmony Game Previewer, visit our Gaming landing page:

https://www.toonboom.com/games

How to use the previewer on Windows

1. Download the Harmony Game Previewer for Windows:
    - https://docs.toonboom.com/go/download/GamePreviewer-12-2-windows
2. Extract the following to your hard drive:
    - GamePreviewer-[version].exe
    - GamePreviewer-[version]_Data folder
3. Copy your export from Harmony to this folder:
    - GamePreviewer-[version]_Data\StreamingAssets\HarmonyResources\
4. Start the Game Previewer application.
5. Select the export you want to preview.

How to use the previewer on macOS

1. Download the Harmony Game Previewer for macOS:
    - https://docs.toonboom.com/go/download/GamePreviewer-12-2-mac

2. Extract the GamePreviewer-[version].app.
3. Right-click on the GamePreviewer-[version].app and select **Show Package Contents**.
4. Copy your export from Harmony to this folder:
   Contents/Data/StreamingAssets/HarmonyResources/
5. Start the Game Previewer application.
6. Choose the export you want to preview

About Exporting to Unity

Using the **Export to Sprite Sheet** and **Export to Easel JS** buttons in the Game toolbar, you can export your character into a package containing sprite sheets and metadata that can be imported into a game engine.

Specifically, if you use the Export to Sprite Sheet option, your character will be exported as a rigged, animated skeleton with sprites, in a format that can easily be imported using the Harmony Game SDK package for Unity, without any scripting required.

If you use the Export to Easel JS option, your character will be exported with each animation frame rendered as a full body sprite, along with metadata that can be used to import it in Unity. However, you will need to program the logic for importing the spritesheet into Unity yourself.

TIP

Since both the Export to Sprite Sheet and Export to Easel JS dialogs are scripts, you can modify them in Harmony's Script Editor view, to make them export in a format that is compatible with your game engine. Those scripts are named **`TB_ExportToSpriteSheets.js`** and **`TB_ExportToEaselJS.js`**, respectively.

Before exporting your character as a spritesheet, ensure that:

- Your scene is saved. Harmony creates the sprites using the drawing files in your scene folder. Hence, any unsaved changes will not appear in the exported sprite sheets.
- The dimensions of your scene are set to a number that is a power of two, such as 512, 1024, 2048, etc. Graphic cards work with textures the dimensions of which are a power of two.
- Your character rig is positioned so that the centre of the stage is where you want its pivot point to be.
- You respected all the guidelines for rigging and animating game assets—see *Rigging Guidelines* on page 6.

Installing the Scripts for Exporting Game Sprite Sheets

The scripts that are required for exporting game assets have been significantly updated in Harmony 15.0.5.

Because scripts packaged with Harmony are copied to your user preferences when you first launch Harmony, but do not get replaced by Harmony when you launch it again, if you have used any version of Harmony between 15.0.0 and 15.0.4 inclusively, your preferences have the older versions of these scripts, and the updated scripts will not be copied to your preferences so as to avoid overwriting these older scripts.

Hence, you must update the scripts in your preferences manually in order to be able to export your game assets from Harmony. To do this, you must copy them from the Harmony installation directory to your preferences.

How to copy the new gaming scripts to your preferences on Windows

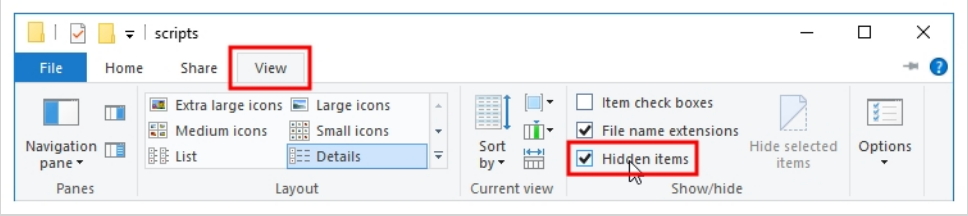
1. Open an Explorer window.
2. Navigate to the following location:

   `C:\Program Files (x86)\Toon Boom Animation\Toon Boom Harmony 15.0 Essentials\resources\scripts`

3. Select the following file:

   `TB_ExportToSpriteSheets.js`

4. Right-click on the selected files and select **Copy**.
5. In the top menu, select **View** and check the **Hidden items** option in the **Show/hide** section.



6. Navigate to the folder of your user's scripts:
   - If you are using Harmony Stand Alone:

     `C:\Users\<your username>\AppData\Toon Boom Animation\Toon Boom Harmony Essentials\1500-scripts`

   - If you are using Harmony Server:

     `C:\USA_DB\users\<username>\Harmony Essentials\1500-scripts`

7. Right-click inside the folder and select **Paste**.
8. When prompted, confirm that you want to replace the existing files.

How to copy the new gaming scripts to your preferences on macOS

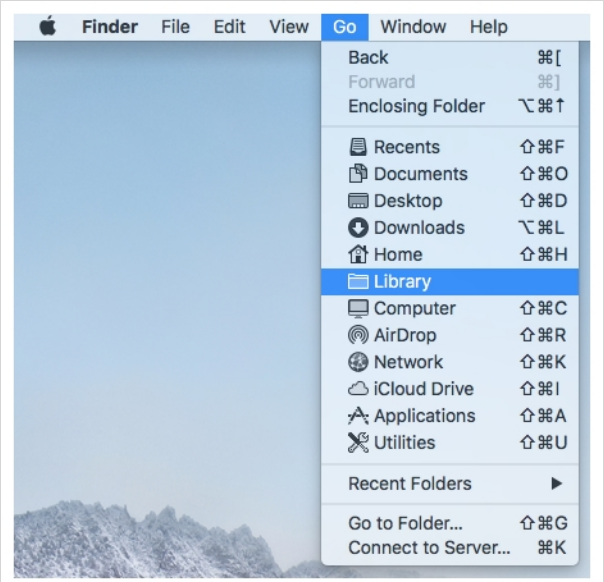
1. Open a Finder window.
2. Navigate to the following location:

   `/Applications/Toon Boom Harmony 15.0 Essentials/tba/resources/scripts`

3. Se,lect the following file:

   `TB_ExportToSpriteSheets.js`

4. Right-click on the selected file and select **Copy "TB_ExportToSpriteSheets.js"**.
5. Navigate to the folder of your user's scripts:
   - If you are using Harmony Stand Alone:
     a. While holding the Alt key, in the top menu, select **Go > Library**.



     b. From there, go to **`Preferences/Toon Boom Animation/Toon Boom Harmony Essentials/1500-scripts`**.

   - If you are using Harmony Server:
     a. In the top menu, select **Go > Computer**.
     b. From there, double-click on the name of your computer's main partition (usually **Macintosh HD**), then go to **USA_DB/users/<username>/Harmony Essentials/1500-scripts**.

6. Right-click inside the folder and select **Paste**.
7. When prompted, confirm that you want to replace the existing files.

Exporting a Game Character as a Rigged Sprite Sheets

The Export to Sprite Sheets dialog allows you to export your rig into a sprite sheet and its animation into animation metadata, in a package that can be directly imported into Unity using the Harmony package for Unity.

To export all the clips for your character, you will need to start by exporting the sprite sheet from your base scene, then open each scene version and export their animation clip one by one.

How to export sprite sheets for a character

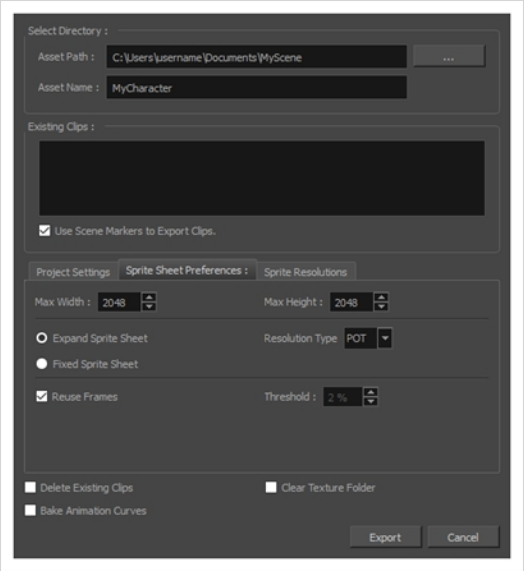
1. If the Game toolbar is not already in your workspace, do one of the following to add it:
   - In the top menu, select **Windows > Toolbars > Game**.
   - Right-click on any existing toolbar and select **Game**.

   The Game toolbar is added to your workspace.

2. Open the base scene version for your character, that is: the scene version in which you built your character rig.
3. **Make sure every drawing used by every scene version of your character is exposed at least once in this scene version.** To do this:
   a. Add as many frames to your scene as there are drawings in the layer with the most drawings.
   b. Leave the first frame with the drawings for your character's normal poses.
   c. In subsequent frames, expose the other drawings in each layer until every drawing used for all of your character's animation clips is exposed for at least one frame.
4. In the Game toolbar, click on the **Export to Sprite Sheets** button.

   The Export to Sprite Sheets dialog appears.



5. Right of the **Asset Path** field, click on the **...** button.

   A folder select dialog opens.

6. Browse to and select the directory in which you want to create your asset directory.

   > **NOTE**
   >
   > You do not need to create a directory specifically for your exported assets. One will be created with the name in the Asset Name field.

7. In the **Asset Name** field, type in the name of the directory in which to export your assets. This will also be the name of the GameObject in Unity.

8. Under **Existing Clips**, make sure the **Use Scene Markers to Export Clips** option is unchecked.

   > **NOTE**
   >
   > Scene markers are not supported in Harmony Essentials. They are supported in Harmony Advanced and Harmony Premium.

9. Select the **Sprite Resolutions** tab.

   This tab contains a list of all the resolutions in which your sprite sheets will be exported.

10. Do the following to export in the resolutions you want your game support:

    - To add a resolution, click on the **+ Add Resolution** button. A row will be added to the list. Double-click on each field in the new row to type in its name, its width and its height, in that

order from left to right.

- To remove a resolution, select it in the list, then click on the **— Remove Resolution** button.

11. For most usages, the other default parameters in the dialog should be fine. To learn more about the parameters in the Export to Sprite Sheet dialog, refer to *Export to Sprite Sheet Dialog Box* on page 68.
12. Click on **Export**.

    Harmony will export the sprite sheet for your base rig.
13. Open the scene version for the first animation clip for your character.
14. Repeat steps **5** to **14** to export the animation for this scene version. Make sure to export in the exact same directory in which you exported the sprite sheet for your base rig.
15. Repeat for each scene version containing an animation clip for your character.

Exporting Game Assets to Easel JS

The Export to Easel JS dialog allows you to export your game character as a sprite sheet, in which each animation frame in your scene is rendered as an individual sprite.

Since the Export to Easel JS dialog renders each frame in your scene, there will not be any visible difference between the way your character is rendered by Harmony and how it is rendered in your game engine.

However, the Export to Easel JS dialog has the following important limitations:

- Contrary to the Export to Sprite Sheet dialog, the Export to Easel JS only outputs a sprite sheet and an XML document delimiting each sprite in the sheet. Hence, they cannot be imported into Unity using the Harmony Game SDK package. You must program the logic to import those sprite sheets into your game engine yourself.
- The Export to Easel JS dialog does not export anchors and props.
- Fully rendered sprite sheets are much heavier in size than rigged sprite sheets, which is what the Export to Sprite Sheet dialog exports.

How to export a game character to Easel JS

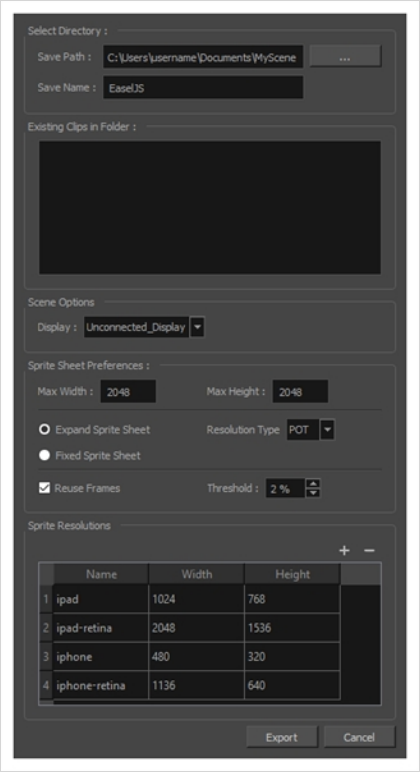
1. Do the following to add the Game toolbar to your workspace:
    - In the top menu, select **Windows > Toolbars > Game**.
    - Right-click on any existing toolbar and select **Game**.

    The Game toolbar is added to your workspace.

2. Open the base scene version for your character, that is: the scene version in which you built your character rig.
3. In the Game toolbar, click on the **Export to Easel JS** button.

    The Export to Easel JS dialog appears.



4. Right of the **Asset Path** field, click on the **...** button.

   A folder select dialog opens.

5. Browse to and select the directory in which you want to create your asset directory.

   > **NOTE**
   >
   > You do not need to create a directory specifically for your exported assets. One will be created with the name in the Asset Name field.

6. In the **Asset Name** field, type in the name of the directory in which to export your assets. This will also be the name of the GameObject in Unity.

7. In the **Sprite Resolutions** section, there is a list of all the resolutions in which your sprite sheets will be exported. Do the following to export in the resolutions you want your game support:

   - To add a resolution, click on the **+ Add Resolution** button. A row will be added to the list. Double-click on each field in the new row to type in its name, its width and its height, in that order from left to right.
   - To remove a resolution, select it in the list, then click on the **— Remove Resolution** button.

8. For most usages, the other default parameters in the dialog should be fine. To learn more about the parameters in the Export to Sprite Sheet dialog, refer to *Export to Easel JS Dialog Box* on page 72.
9. Click on **Export**.

   Harmony will export the sprite sheet for your base rig.
10. Open the scene version for the first animation clip for your character.
11. Repeat steps **3** to **10** to export the animation for this scene version. Make sure to export in the exact same directory in which you exported the sprite sheet for your base rig.
12. Repeat for each scene version containing an animation clip for your character.

Converting Exported Game Assets from XML to Binary

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 in the following directory:

HarmonySDK/Source/Utils/

This directory contains the following:

- **macosx**: Precompiled binary for macOS.
- **win32**: Precompiled binary for Windows.
- **Xml2Bin**: Xml2Bin sources.
- **Xml2Bin/proj.mac/Xml2Bin.xcodeproj**: XCode project formacOS.
- **Xml2Bin/proj.win32/Xml2Bin.sln**: Visual Studio project 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.

About Working in Unity

The following section details how to import a character which you have created in Harmony and exported using the Export to Sprite Sheet dialog into a Unity project. Once your character is imported, you will also learn how to preview the clips and the props you created in Harmony in Unity, if applicable.

NOTES

- The following section does not explain how to import a package exported using the Export to Easel JS dialog, as this dialog only exports a basic sprite sheet.
- The following section does not explain how to create a game in Unity. It only explains how to import and preview assets created in Harmony.
- No coding skills are required to follow the methods in this section

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)

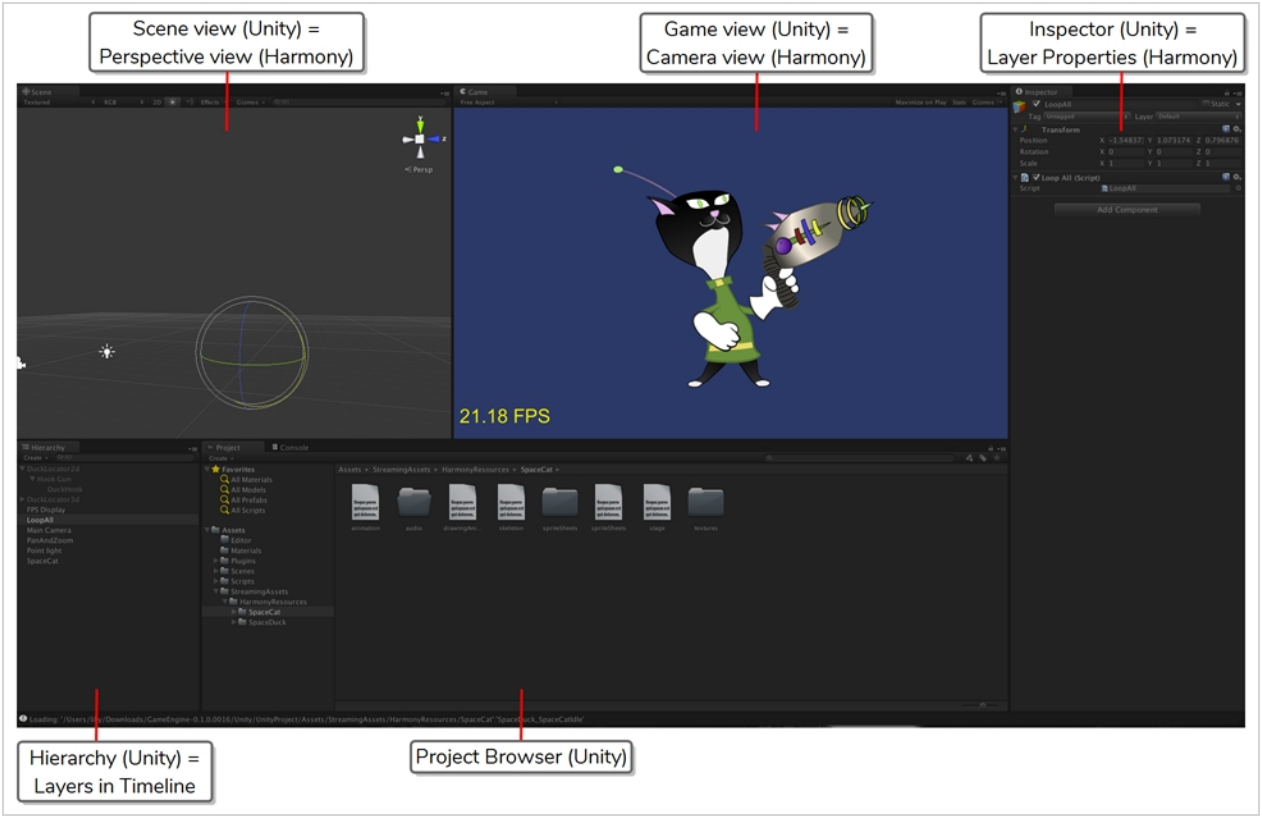
By configuring the Harmony script, you can automatically export animations into the StreamingAssets folder. This way, Unity will dynamically load the most up-to-date assets as soon as they appear in the folder. You can also manually place exported Harmony data into this location.

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.
Project Browser	---	Lets you access and manage a project's assets.

The main camera displays the scene and is located in the Hierarchy. To show a Harmony object, you must add a special component. With the main camera selected, in the Inspector, select **Add Component > Scripts > Harmony Camera**.


Scene view (Unity) =
Perspective view (Harmony)
Game view (Unity) =
Camera view (Harmony)
Inspector (Unity) =
Layer Properties (Harmony)
21.18 FPS
Hierarchy (Unity) =
Layers in Timeline
Project Browser (Unity)

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 (https://docs.toonboom.com/go/download/UnityPackage) 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.

About Game Objects

GameObjects are the fundamental objects in Unity that represent characters, props and scenery. Basically everything you create in Unity will be a GameObject. There are two main types of GameObject you can use to import Harmony data:

- **Create Empty**
- **Create Other > Plane**

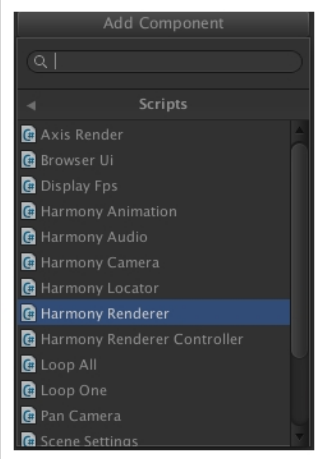
To understand the process fully, there are two examples using an empty GameObject and a plane GameObject with the SpaceCat demo file. Everything you do will be based on the GameObject you create to load your Harmony data.

Using Empty Game Objects

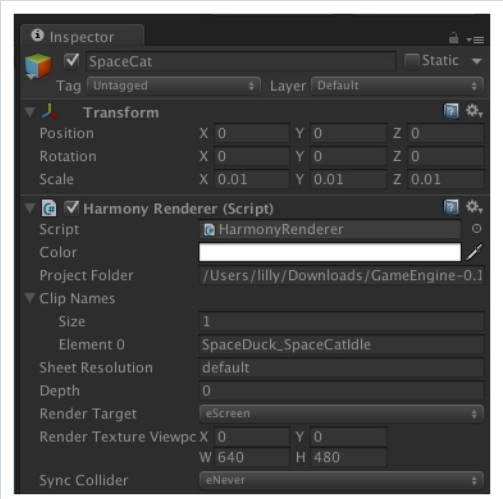
An Empty Game object is a regular GameObject in which you can load the exported Harmony data. All the individual body parts on the sprite sheet come in as separate elements, but are treated as one whole object with a dynamically created bounding box. If you put a transparency on this object, you would see the overlapping transparencies of all the individual objects. This creates a higher-quality render and should be the default if you're not animating the transparency of an object. With this option, direct rendering will read the depth information in the scene but not write it. Note that with this option, the rendering cannot be customized.

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 SpaceCat 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 SpaceCat GameObject.
    - In the Inspector, go to **Add Component > Scripts > Harmony Renderer**.



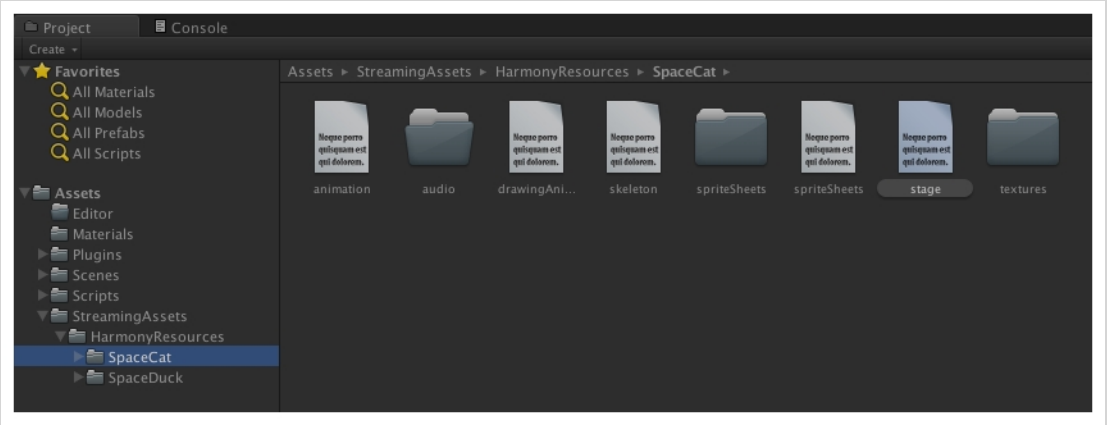
A new section appears in the Inspector called Harmony Renderer (Script).



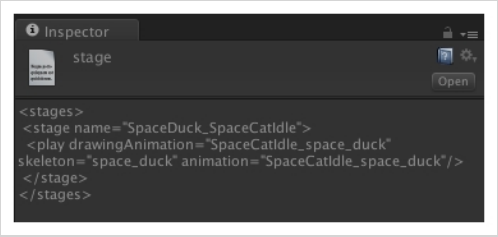
This will let you load an animation file. You must put the path of the asset relative to the HarmonyResources folder. Before this, you must export from Harmony directly to this folder, or manually copy and paste the data here.

4. You exported a scene called SpaceCat. In the Project Folder, place **HarmonyResources/SpaceCat** under the Transform information at the top, under Scale. Set the values to 0.01 for X, Y, and Z because the default Harmony project is for an HD screen.
5. You won't see anything until you define a few parameters:
    - Set the Clip Name Size to 1. A second field will appear, Element 0.
    - Enter the clip name for the animation you want to load.
6. Double-check on the right Clip Name to put in Element 0; you can check the output you exported from Harmony.
7. Double-check the **`stage.xml`** file, which was exported to the folder you specified when exporting from Harmony. Our default export path is to StreamingAssets, so it will automatically load in the game engine.
8. Look in the Project tab, under **Assets > StreamingAssets > HarmonyResources > [ExportName].**

    You'll find the data that was exported. In this case, we're looking for SpaceCat. Select the file marked **`stage`** to load the data you want to double-check.



The text data appears in the Inspector.



The name to use for loading the object is the stage name. In this case, **`SpaceDuck_SpaceCatIdle`**.

9. If your GameObject has animation in it, and you want it to load the animation data from the Harmony export, add one more component by doing the following:
    - Select the GameObject.
    - In the Inspector, select **Add Component** and select **Scripts > Harmony Animation**.
10. To loop the animation, you must set a GameObject to define it:
    - Select **GameObject > Create Empty** and rename it: **LoopAll**.
    - Click the **LoopAll** GameObject to display its properties in the Inspector.
    - Click **Add Component > Scripts > Loop All**.

    When you play the animation, it will loop as it plays back. The Loop All command simply iterates on all GameObjects on runtime and asks them to loop, which is useful when testing the animation. When programming the actual game, you should write additional scripts to control the different elements separately.

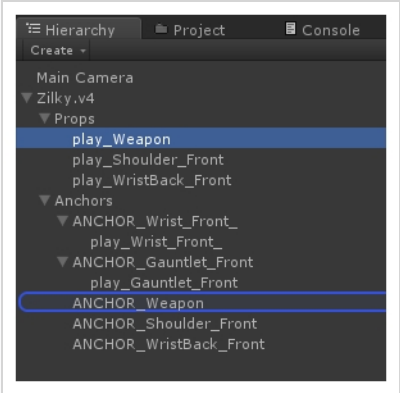
Attaching Props to Anchors in Unity

An anchor is an attachment point you can use to bind one or several props to a specific location that is predefined by the anchor.

A prop is a game item that will be attached to an anchor. It could be a stick held within the character's hand, a helmet attached to the character's head...the possibilities are endless. Props are easy to turn on or off, and easily swappable from within Harmony or Unity.

How to attach the prop to an anchor

1. Open up your character in the Hierarchy view to have access to both the props and the anchors.
2. Click on the prop you wish to attach, and drag it onto the appropriate anchor.



3. Repeat these steps for each prop and anchor in your project.
4. In the Inspector view, under Frame, you can select which frame you wish to show. If you only had one drawing in your prop, then you will not have multiple options.

5. If part of your character is cut off by the plane of the character, now that the props are showing, you will need to regenerate the mesh. To do so, select your GameObject. After this is done, go to **Inspector View > Render > Generate Mesh**.

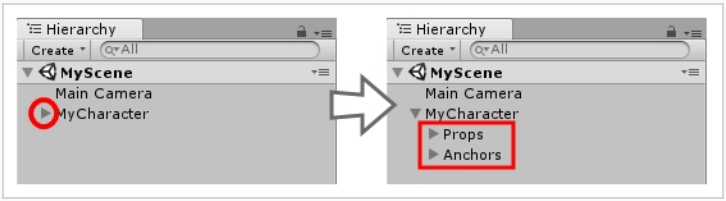
Attaching Props to Anchors in Unity

An anchor is an attachment point you can use to bind one or several props to a specific location that is predefined by the anchor.

A prop is a game item that will be attached to an anchor. It could be a stick held within the character's hand, a helmet attached to the character's head...the possibilities are endless. Props are easy to turn on or off, and easily swappable from within Harmony or Unity.

How to attach the prop to an anchor

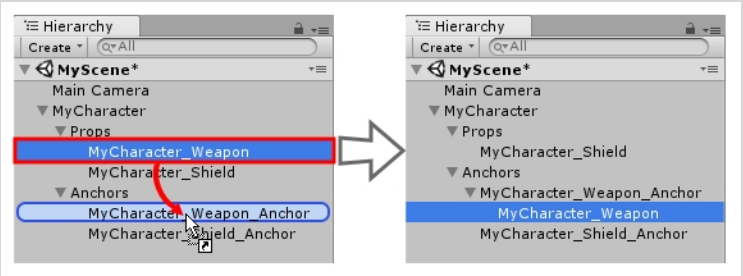
1. In the **Hierarchy** view, click on the right-pointing arrow left of your GameObject to expand its content and see its Props and Anchors hierarchies.



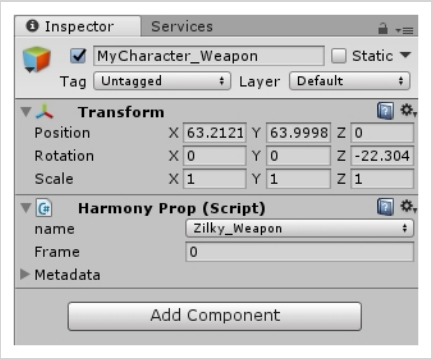
2. Expand both the Props and Anchors hierarchies to view your character's list of props and anchors.

3. Select a prop, then drag and drop it over the anchor you want to attach it to. It should end up being

a child of the anchor.

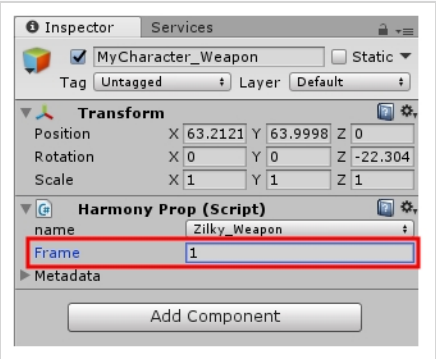


4. Select the prop in the **Hierarchy** view to view its properties in the **Inspector** view.

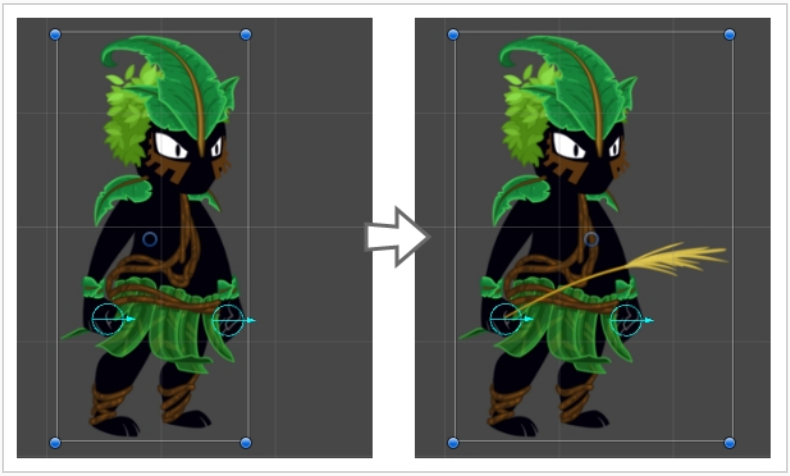


Each prop has a Frame property. The prop has one frame for each drawing that was in the prop's drawing layer in Harmony. Its default value, 0, makes the prop invisible.

5. Select the **Frame** field.
6. Type in the index of the drawing you want to display in the prop object. For example, if your prop only had one drawing in Harmony, type **1** to display it. If your prop had 5 drawings, type a number between **1** and **5** to display one of those drawings in the prop object.



The selected prop frame will display in the Scene view, attached to its anchor.



Troubleshooting

Unity Questions

Q. My Harmony asset does not appear in the Unity Editor.

A. There are several possible reasons for this. Check the following:

- Has the HarmonyRenderer plugin been copied to the Assets/Plugins directory? The plugin is named **`HarmonyRenderer.bundle`** on macOS and **`HarmonyRenderer.dll`** on Windows.
- Is the clip name valid ? Make sure the clip name exists in the stage.xml file.
- Are the binary files up to date? If you updated the Toon Boom Game Engine SDK to a newer version, make sure to regenerate the binary files if you use them so they're compatible with the SDK.

Q. My Harmony asset does not appear in iOS or Android when exported from Unity.

A. There are several possible reasons for this. Check the following:

- Make sure the Harmony assets are copied in the **`StreamingAssets`** directory. Since the HarmonyRenderer plugin cannot use Unity's asset database, the Harmony assets were packaged as plain files so they are accessible to the plugin.
- Is the plugin for iOS the same version as the one used for the editor? The plugin for iOS is in the **`Assets/Plugins/iOS`** directory.
- Is the clip name spelled correctly? Windows and macOS aren't usually case sensitive file systems, but iOS and Android are. This implies that an asset that is visible on the editor will not appear on the mobile platform because it cannot be found.
- Make sure the size of your sprite sheets does not exceed the hardware limit of the device you're working on. Otherwise, the texture might not appear at all.
- Is the path to the project folder an absolute path? If it is, it might not work on the mobile device as that path will not exist on the platform. Make sure to use relative paths to the **`StreamingAssets`** directory.

Q. My Harmony asset does not composite properly with other 2D sprites in Unity. The 2D sprites always appear behind the Harmony assets even if I change the order index in the GameObjects.

A. The 2D sprites from Unity and the Harmony assets cannot currently composite together. There are, however, several solutions for working around this problem:

Use 2D textures instead of 2D sprites. The 2D textures, when attached to a geometry in Unity, will composite like the other 3D elements in the scene and you can change the compositing order by incrementing or decrementing the Z value of the transform.

Use multiple cameras to render your scene in parts. Here's how:

Harmony Camera

1. Create a new camera.
2. Make sure you set the Harmony GameObjects to a specific layer (top-right of the game object inspector).
3. In the camera, set the culling mask to this layer (and the other layers you want rendered in the background).
4. Add the HarmonyCamera component to this camera so the Harmony GameObjects render properly.

Sprite Camera

1. Create a new camera.
2. Make sure to set the 2D sprite objects to a specific layer.
3. In the camera, set the culling mask to this layer (and the other layers you want rendered in front of the 2D sprites).
4. Make sure to set the clear flags to **Depth Only** or **Don't Clear**.
5. Set the depth to be higher than the Harmony camera.
6. Afterwards, if you move the camera, make sure to move both cameras simultaneously if you want to keep Game Ojects in sync.

Q. How can you specify the frame rate of the animation? My animation does not play at the same frame rate in Harmony and in the Game Previewer in the Unity Editor.

A. The Game Previewer previews the animation at a default of 24 frames per second. In your game, using the HarmonyAnimation component, you can schedule animations at the speed you want.

Example

```
HarmonyAnimation animation = GetComponent<HarmonyAnimation>();
if (animation != null)
{
animation.PlayAnimation( frameRate, clipName );
}
```

Q. I cannot see my character in the Scene view in Unity. Is there any way to preview the Harmony asset?

A. Since the rendering of Harmony assets is implemented in a plugin, it does not update any Renderer component in the GameObject, and so cannot be shown directly in the Scene view.

However, you should still be able to preview your assets in the Game view (even when not playing). To help find and position assets, we also added bounding box shapes rendered in the editor and the Game view.

Q. How is memory management handled in Unity for Harmony assets?

A. Most of the memory extensive algorithms are executed in the C++ plugin. The data structure is maintained alive as long as a GameObject refers to it. This means you can instantiate as many clones of a character without

reloading the data. Upon destruction, the GameObject will issue a clean-up call to the plugin and delete its associated rendering object. The sprite sheet and animation data are kept alive as long as a single GameObject uses them and a cleanup call is issued once they are destroyed.

Q. The Harmony assets memory does not appear in the Unity profiler. Can we profile the assets memory?

A. The Unity profiler does not pick up Harmony assets in its traces. Since the Harmony GameObject refers to a C++ plugin for its asset management and rendering, its memory cannot be recorded in the profiler. Profilers such as the one provided with XCode or Visual Studio should allow you to track your application's memory including the Harmony assets and should also provide a more accurate result.

General Questions

Q. How can you specify the resolution used for the drawings in the sprite sheet?

A. The drawing resolution is not fixed in Harmony. Since you're exporting the scene to multiple sprite sheets in multiple resolutions, every drawing will have a different resolution in the end. You can use the Drawing view to see the resolution of a drawing compared to the rest of the scene. By enabling the light table in the Drawing view, you will see all of the scene's drawings untransformed in the same proportion as they will appear in the sprite sheet. More specifically, the resolution used in the sprite sheet for a drawing is the size it appears untransformed in the Camera view, scaled to the sprite sheet resolution ("Sprite Resolutions" in the Export Sprite Sheet dialog box).

Once you know that, you can easily adjust the size of a single drawing. Use the Select tool to scale down a drawing and use the Transform tool to scale it back up so it does not change visually in the Camera view. This should allow you to alter the size of drawings in the sprite sheet.

Q. The assets take time to load on mobile devices. Is there any way to speed up the process?

A. Harmony exports an interchange XML format that is intended to be compatible with multiple game engines. However, reading data from the XML files is slow and this is mostly intended to test the assets in your project. At some point, they should be converted to a binary compact data structure that can be read into memory a lot faster than the XML.

You can use the Xml2Bin tool provided with the game engine SDK to convert your XML files to binary. This tool is in the **`Utils/Xml2Bin`** folder and a precompiled version for macOS is available in the **`Utils/macosx`** folder.

Q. There are problems with the character I exported in the game engine. Pieces are not exported or are not showing in the proper order. What can I do ?

A. To integrate with the game engine SDK, all animations in Harmony must follow certain guidelines. Here are some of the key points for building a character that is compatible with the rendering in the game engine SDK:

- Build your rig in the Timeline view. The game engine SDK data structure is similar to what is displayed in the Timeline view and an advanced node system might not always export properly.
- Because every top level node in your scene will render separately in the game engine SDK, they may not intersect. A top level node is also isolated in its separate sprite sheet to allow the artist better memory

segmentation of the character. If you created a character with top level nodes and this was not the intended result, create a top level group or peg to group them all together.

- Use micro-Z ordering if necessary. With no Z ordering, a child layer will appear behind its parent (as it does in Harmony). To change the ordering, add some Z offset to the child layer.
- The export script does not handle plain bitmap images. Toon Boom vector drawings and Toon Boom bitmap drawings are supported though.
- The export script does not handle symbols.

Chapter 2: Related Topics

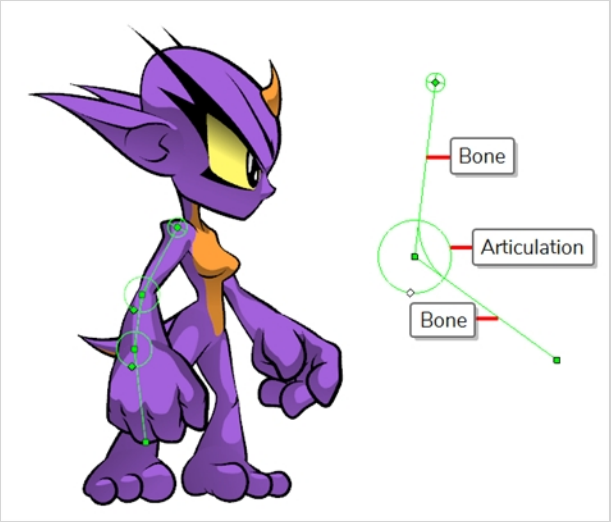
The following section contains topics from the Harmony user guide that are useful for creating gaming rigs in Harmony.

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 difference between the Bone and Game Bone deformations is that Game Bone deformations do not have a Bias property. The articulation folds also look slightly more rounded.



Selecting Elements for Deformation

Deformation groups are added to your scene's hierarchy in different places, depending on whether a drawing or peg element is selected.

How to select the element on which to create a deformer

A deformer affects all the layers under its hierarchy. When you create a new deformer using the rigging tool, it is automatically created as a parent of the selected layer, and will affect the selected layer and all its children. Hence, it's important to select the right layer before attempting to create a deformer.

For example, if you wish to create a deformer for an arm, and the arm is broken into several layers, you could rig the arm in a hierarchy where the upper arm is the root, the forearm is the upper arm's child and the hand is the forearm's child. Then, if you create your deformer on the upper arm, it will be created as a parent of the upper arm, and will hence affect the whole arm.

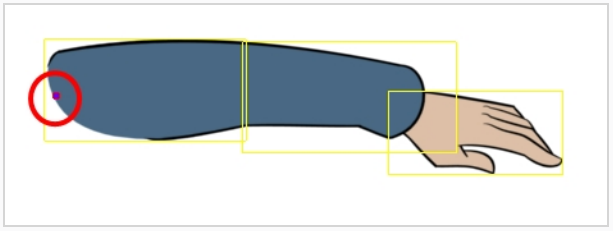
You can also group several layers under a peg, and create your deformer with this peg selected. When you create a deformer on a peg, the deformer is created as a child of the peg, but as a parent of all of this peg's children. This is because a peg is likely to be more useful over a deformer than under. If you move layers that are under a deformer, they will exit the intended deformation zone, and may appear severely warped and distorted. Hence, it's better to create deformations under pegs as much as possible, and pegs under deformers should only be animated if the deformer is left untouched. Since the deformer will be created as a parent of all the peg's children, it will affect all the layers under the peg.

Creating Game Bone Deformation Chains

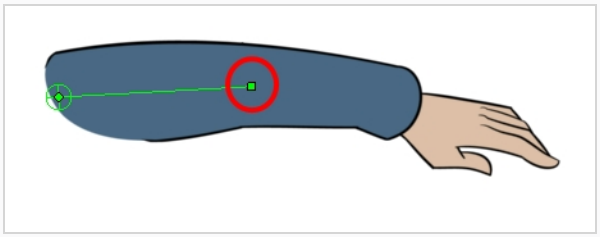
If you are working on a character that is meant to be exported as a game asset for use in Unity, and you want to use deformations to animate your character, you must use Game Bone deformations. These deformations replicate the type of deformation supported by Unity and can be exported to Unity as they are, without having to bake the deformations into drawings first.

How to create a Game Bone deformation chain

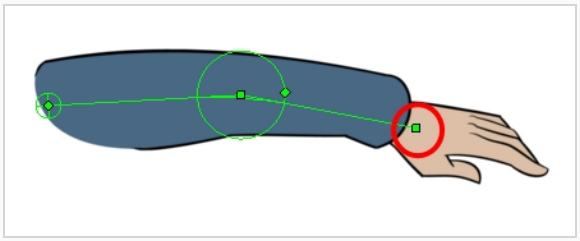
1. Make sure the right element for the creation of your deformation chain is selected—see *Selecting Elements for Deformation* on page 63.
2. Once your element is selected, select the **Rigging** tool in the Deformation toolbar
3. In the **Tool Properties** view, click on **Game Bone** to enable Game Bone mode.
4. Place the cursor at the root of your drawing or drawing hierarchy, click once and release. For example, if you're creating a deformer for a whole arm, click on the shoulder joint.



5. Move your cursor at the location where you want your first bone to finish and your second bone to start, and click again. An articulation control point will automatically be inserted between each bone you create.

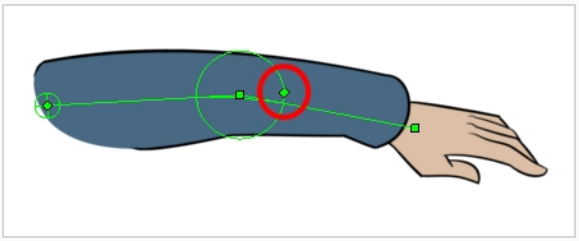


6. Move your cursor where you want the next articulation to be and click to create the next point.



Observe that, as you create a new point, the previous point now has a circle around it. This is the radius of the articulation, which allows you to determine how much of the drawing should be part of the articulation. Every point in a bone deformer, except for the first and the last one, has a radius setting.

7. Click and hold the previous articulation's radius manipulator and adjust the articulation's radius so that it covers the limb.



8. Repeat this until you are finished creating the Bone chain. Make sure you build each articulation in the right order going from the root to the extremity.

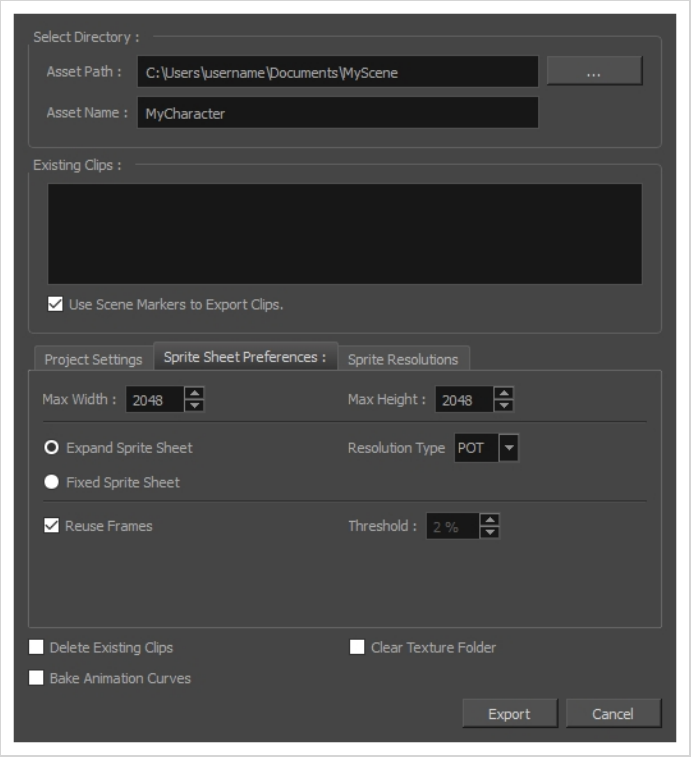
Chapter 3: User Interface Reference

The following section contains pages from the Harmony Reference guide that are useful for creating gaming rigs in Harmony.

Export to Sprite Sheet Dialog Box

The Export to Sprite Sheets dialog allows you to export your rig into a sprite sheet and its animation into animation metadata, in a package that can be directly imported into Unity using the Harmony package for Unity.

To export all the clips for your character, you will need to start by exporting the sprite sheet from your base scene, then open each scene version and export their animation clip one by one.



How to access the Export to Sprite Sheets window

1. Do one of the following:
    - In the top menu, select **Windows > Toolbars > Game**.
    - Right-click on any existing toolbar and select **Game**.

2. Click the **Export to Sprite Sheets** button.

Parameter	Description
Select Directory	
Asset Path	The target directory in which to export your sprite sheets.
Asset Name	The name of the folder in which the spritesheet and its metadata will be saved. This folder will be located inside the Save Path folder. If it does not exist, it will be created. This will also be the name of your asset in Unity.
Existing Clips	
Existing Clips List	If you already exported clips for this character, this will display the list of clips that already exist in the target directory.
Use Scene Markers to Export Clips	Uses scene markers to define the export range, instead of exporting the entire scene. **NOTE** Scene markers are not supported in Harmony Essentials. They are supported in Harmony Advanced and Harmony Premium.
Project Settings	
Unit Scale	Lets you change the scale when exporting to Unity to accommodate the size of the export without it affecting the Harmony scene. This helps you resize assets properly for Unity without having to resize them in Harmony. The basic scale is one Animation Field for one Unity unit.
Preset	Lets you select the unit conversion from Harmony to Unity, by selecting one of 4 presets. Note: a field is a unit of measure in traditional animation grid.
Sprite Sheet Preferences	
Max Width	The maximum width, in pixels, of the exported sprite sheets.
Max Height	The maximum height, in pixels, of the exported sprite sheets.
Expand Sprite Sheet	Creates sprite sheets in the smallest possible size to contain all the sprites, up

Parameter	Description
	until it reaches the maximum width and height.
Fixed Sprite Sheet	Creates a texture of the specified size (Max Width and Max Height) even if it does not fill it up completely by all the drawings in your scene.
Resolution Type	Allows you to choose between one of these two resolution types: • **POT**: Exports to sprite sheets with sizes that are a power of 2. For example: 1024 x 1024. This is optimized for many graphics cards, but consumes more memory. • **NPOT**: Some game engines are optimized specifically to render to non powers of two, so that it will avoid those numbers. Example: 1000 x 1000.
Reuse Frames	This option works in tandem with the Threshold option. The export will compare the drawings in your project to reuse a maximum of similar drawings and reduce the amount of information found in the sprite sheet, making it lighter. The export will omit the creation of new drawings if the difference is less than the threshold percentage.
Threshold	Calculates the differences between multiple drawings. A 2% threshold will prevent the creation of a new drawing if the drawing is too similar to an existing drawing. For instance, with a 2% threshold, and my drawing is 100 pixels big, only 2 of those pixels need to be different from my other drawing in order to create a new one. The higher the threshold, the fewer similar drawings you will have.
Sprite Resolutions	
Add Resolution	Allows you to add a resolution in which to export your sprite sheets.
Remove Resolution	Removes the selected resolution from the resolutions list.
Resolutions List	The list of resolutions in which the sprite sheet will be exported. The whole sprite sheet will be rendered once for each resolution in the list.
Delete Existing Clips	Select this option to delete clips that you already exported in the export directory.
Bake Animation Curves	Instead of saving your character's animation as interpolations, this bakes them into frame-by-frame geometric transformations. This can help increase performance in your game, as your game engine will not have to interpolate your

Parameter	Description
	character's animations on its own. It will however create a heavier package.
Clear Texture Folder	When creating the sprite sheet, Harmony creates temporary files in a folder named texture inside the scene directory. Check this option to automatically delete this folder after exporting your sprite sheets.

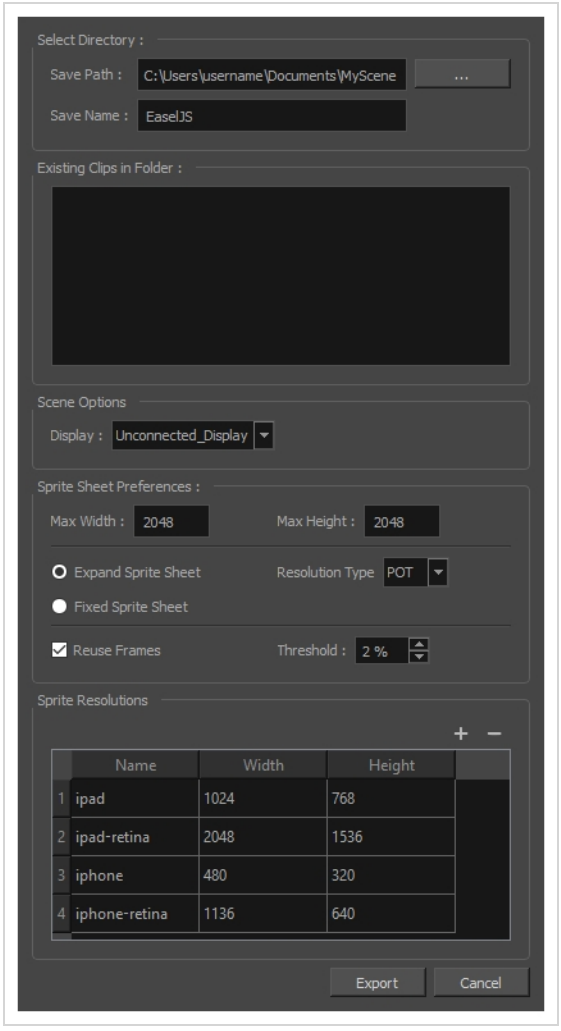
Export to Easel JS Dialog Box

The Export to Easel JS dialog allows you to export your game character as a sprite sheet, in which each animation frame in your scene is rendered as an individual sprite.

Since the Export to Easel JS dialog renders each frame in your scene, there will not be any visible difference between the way your character is rendered by Harmony and how it is rendered in your game engine.

However, the Export to Easel JS dialog has the following important limitations:

- Contrary to the Export to Sprite Sheet dialog, the Export to Easel JS only outputs a sprite sheet and an XML document delimiting each sprite in the sheet. Hence, they cannot be imported into Unity using the Harmony Game SDK package. You must program the logic to import those sprite sheets into your game engine yourself.
- The Export to Easel JS dialog does not export anchors and props.
- Fully rendered sprite sheets are much heavier in size than rigged sprite sheets, which is what the Export to Sprite Sheet dialog exports.



How to access the Export to Easel JS window

1. Do one of the following:
    - In the top menu, select **Windows > Toolbars > Game**.
    - Right-click on any existing toolbar and select **Game**.

2. Click the **Export to Easel JS** button.

Parameter	Description
Select Directory	
Save Path	The target directory in which to export your sprite sheets.
Save Name	The name of the folder in which the spritesheet and its metadata will be saved. This folder will be located inside the Save Path folder. If it does not exist, it will be created. This will also be the name of your asset in Unity.
Existing Clips in Folder	
Existing Clips List	If you already exported clips for this character, this will display the list of clips that already exist in the target directory.
Scene Options	
Display	The Display from which to render your character. If the Unconnected_Display option is selected, all of the visual information in your scene will be rendered. **NOTE** You can only create Displays in Harmony Premium.
Sprite Sheet Preferences	

Max Width	The maximum width, in pixels, of the exported sprite sheets.
Max Height	The maximum height, in pixels, of the exported sprite sheets.
Expand Sprite Sheet	Creates sprite sheets in the smallest possible size to contain all the sprites, up until it reaches the maximum width and height.
Fixed Sprite Sheet	Creates a texture of the specified size (Max Width and Max Height) even if it does not fill it up completely by all the drawings in your scene.
Resolution Type	Allows you to choose between one of these two resolution types: • **POT**: Exports to sprite sheets with sizes that are a power of 2. For example: 1024 x 1024. This is optimized for many graphics cards, but consumes more memory. • **NPOT**: Some game engines are optimized specifically to render to non powers of two, so that it will avoid those numbers. Example: 1000 x 1000.
Reuse Frames	This option works in tandem with the Threshold option. The export will compare the drawings in your project to reuse a maximum of similar drawings and reduce the amount of information found in the sprite sheet, making it lighter. The export will omit the creation of new drawings if the difference is less than the threshold percentage.
Threshold	Calculates the differences between multiple drawings. A 2% threshold will prevent the creation of a new drawing if the drawing is too similar to an existing drawing. For instance, with a 2% threshold, and my drawing is 100 pixels big, only 2 of those pixels need to be different from my other drawing in order to create a new one. The higher the threshold, the fewer similar drawings you will have.
Sprite Resolutions	
Add Resolution	Allows you to add a resolution in which to export your sprite sheets.
Remove Resolution	Removes the selected resolution from the resolutions list.
Resolutions List	The list of resolutions in which the sprite sheet will be exported. The whole sprite sheet will be rendered once for each resolution in the list.

Game Toolbar

The Game toolbar contains tools for setting anchors and props, as well as exporting to sprite sheets and Easel JS.

NOTE

For tasks related to this toolbar, see Gaming Guide on page 5 and About Anchors and Props on page 17,

How to access the Game toolbar

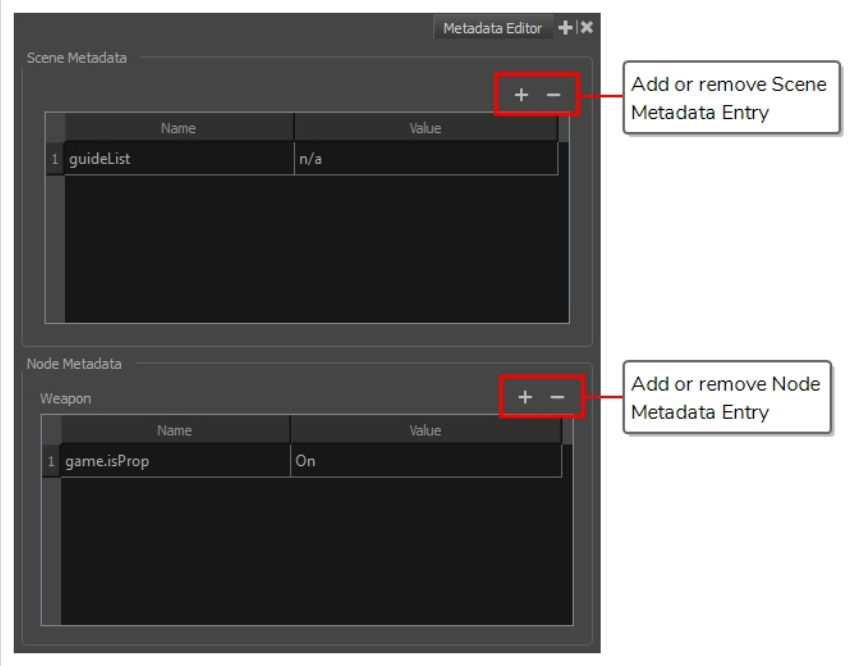
1. Do one of the following:
    - In the top menu, select **Windows > Toolbars > Game**.
    - Right-click on any existing toolbar and select **Game**.

Icon	Tool Name	Description
	Toggle Anchor	Adds or remove the Anchor property for the selected layer. When a layer is an anchor, its pivot point is exported as an anchor in Unity. A prop can be attached to the anchor to follow the anchor's animation. Since the anchor is the layer's pivot, it will move and rotate with the anchor layer's movement as well as its parent layers' movement. **TIP** You can see whether a layer has the Anchor property in the Metadata Editor view—see Metadata Editor View on page 77
	Toggle Prop	Adds or remove the Prop property for the selected layer. When a layer is a prop, it is exported as a prop to Unity. A prop can be attached to an anchor on your character, which makes it follow the anchor's animation.

Icon	Tool Name	Description
		TIP You can see whether a layer has the Prop property in the Metadata Editor view—see *Metadata Editor View* on page 77
	Export to Sprite Sheets	Opens the Export to Sprite Sheets dialog, from which you can export your scene as a rigged sprite sheet that can be imported in Unity and which can have props and anchors—see *Export to Sprite Sheet Dialog Box* on page 68.
	Export to Easel JS	Opens the Export to Easel JS dialog, from which you can export your scene as a rendered sprite sheet that can be imported in Unity—see *Export to Easel JS Dialog Box* on page 72.

Metadata Editor View

The Metadata Editor view allows you to view and modify the information for props and anchors when creating a game character or asset in Harmony. It displays your scene's metadata as well as the metadata for the currently selected layer and allows you to add, remove or change the value of metadata nodes.



Usually, you do not need to manually add or modify metadata for a game character or asset using the Metadata Editor. The commands available in the Game toolbar can be used to define props and anchors, which will automatically fill your scene and layers with the proper metadata. The Metadata Editor is however useful for visualizing the way Harmony stores this information, to tweak it as necessary and to debug any issue that could occur when exporting a game asset to Unity.

IMPORTANT

You should avoid using any character but letters, numbers, dashes and underscores in the Metadata Editor.

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**.

Parameter	Description

<table>
<tr>
<td>Scene Metadata</td>
<td>
Displays the metadata for the Harmony scene. Each entry has a name and a value.

Typically, metadata for a scene includes the following entry:

<table>
<tr><th>Metadata</th><th>Description</th></tr>
<tr><td><code>guideList</code></td><td>This is a node in which the guides created in the Guide View are stored. This can be ignored.</td></tr>
</table>
</td>
</tr>
<tr>
<td>Node Metadata</td>
<td>
Displays the metadata for the currently selected layer. Each metadata entry has a name and a value.

Typically, metadata for a node includes the following entries:

<table>
<tr><th>Metadata</th><th>Description</th></tr>
<tr><td><code>game.isAnchor</code></td><td>When set to On, this sets the selected layer as an anchor. This means that, in Unity, the layer's pivot will be an anchor to which props can be attached. A prop will follow its anchor's movement.</td></tr>
<tr><td><code>game.isProp</code></td><td>When set to On, this sets the selected layer as a prop. In Unity, a prop can be attached to an anchor to follow its movement.</td></tr>
<tr><td><code>game.<metadata name></code></td><td>When a layer has the <code>game.isProp</code> property, you can add any metadata node to it provided that the name of the metadata node starts with <code>game</code> followed by a period (<code>.</code>). This metadata will be exported to the list of metadata values for the prop, which can be used in Unity.</td></tr>
</table>
</td>
</tr>
</table>