ANIMA 2D

1.1.2

USER GUIDE

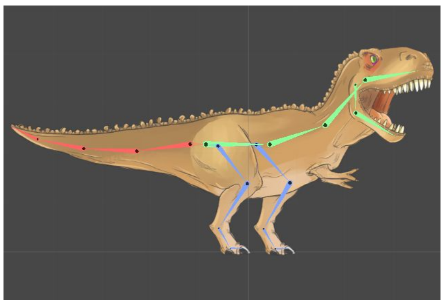
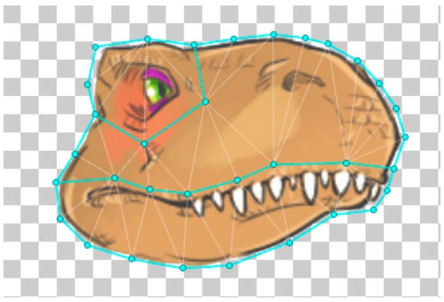
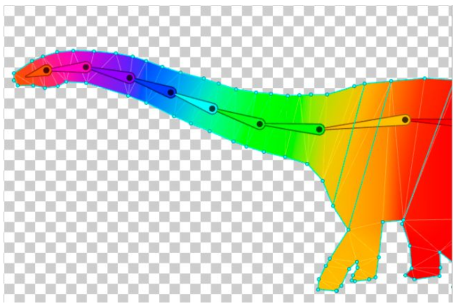
1 Table of Contents

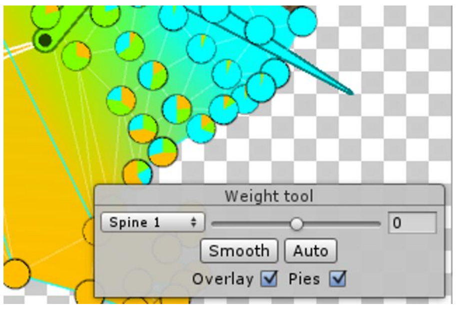
2 About Anima2D

Anima2D is the most advanced and complete **2D Skeletal Animation** solution for use with Unity 5.x. Using Anima2D you will be able to create **2D skinned characters and backgrounds** for your game right inside Unity.

Anima2D extends Unity by adding **2D Bone Hierarchies**, a powerful **SpriteMesh Editor** with full control over the resulting Mesh and **2D Inverse Kinematics.** Anima2D was created focusing in **workflow** and **seamless Unity Integration.**

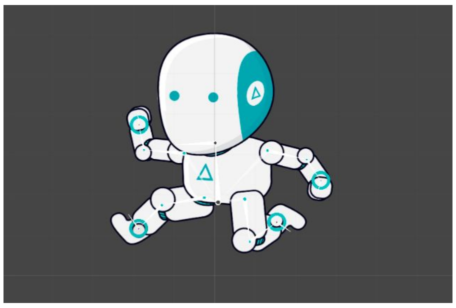
3 Features

	**Bone Hierarchies** Create Bone Hierarchies using the new **Bone2D Component**. It has never been that easy to create skeletons: just create, parent and link Anima2D's bones. Control bone's rotation just by dragging its gizmo.
	**Mesh Editor** Create complex **Meshes** from Sprites. Meshes can be fine-tuned from the **SpriteMesh Editor Window**. You can add and delete vertices, holes, edges and edge constraints to achieve the desired triangulation.
	**Automatic Weights** Anima2D calculates **Skinning Weights** for you **automatically.** We use state-of-the-art algorithms to give you the best weights possible that work out of the box in most of the situations.



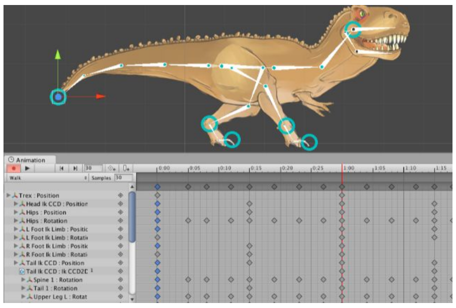
Weight tool

Edit and fine-tune your Weights using the integrated **Weight tool**. It allows single and multiple vertex editing by adding and subtracting bone's influence, smoothing between neighboring vertices and resetting to default automatic values.



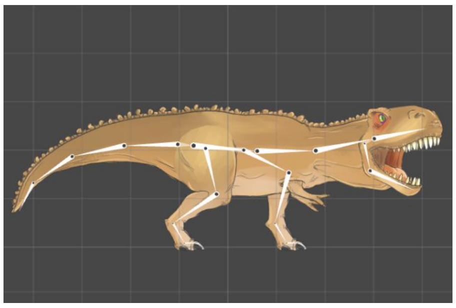
Inverse Kinematics

Anima2D includes an easy to use **2D Inverse Kinematics System** (IK) that allows to create consistent poses for limbs and bone chains of arbitrary length. IK works in the Scene View too!



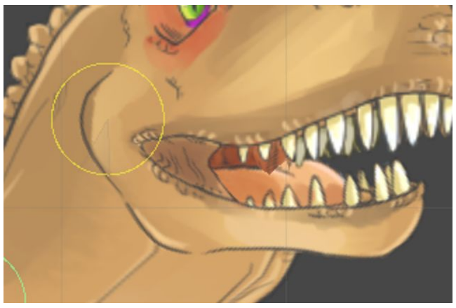
Animation Authoring

Anima2D is specially designed to create animations using Unity's Animation Window. Move, rotate bones or IK objects while in **Record Mode** and **bake** your animation to bones when you are done.



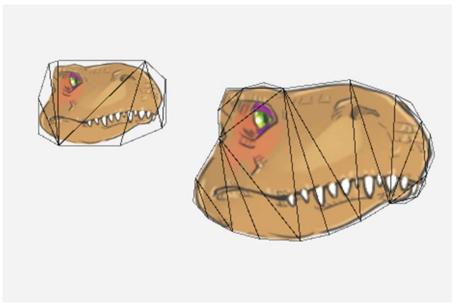
Pose Manager

Save and **Load** skeleton **poses** with a single click using the **Pose Manager** component. Poses are saved to a separate asset file and can be loaded while in record mode too!



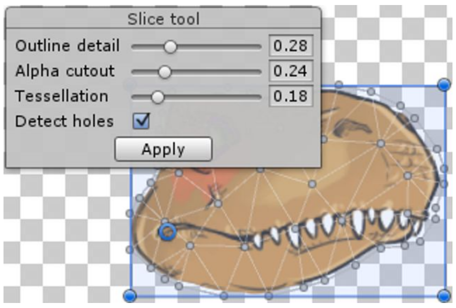
Bone Controls

Use bone controls to manipulate the **position** and **rotation** of the desired bones even if they are **hidden** or **locked** in the Scene View. Controls are useful to create animations and will also create new **Keyframes** for the affected bones.



Sprite Optimizer

Anima2D will **override** the original **Unity Sprite**'s geometry. Use that to simplify your native Sprite, **reduce overdraw**, reduce vertex count and other optimizations.

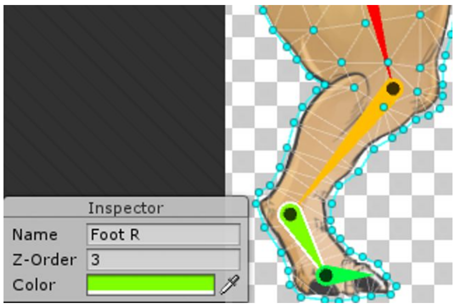


Slice tool

Outline detail	0.28
Alpha cutout	0.24
Tessellation	0.18
Detect holes	✓

Automatic Mesh Slicing

Create your new and optimized sprite geometry automatically in a few clicks. Just select the image area, set the desired parameters and Anima2D will do the rest.

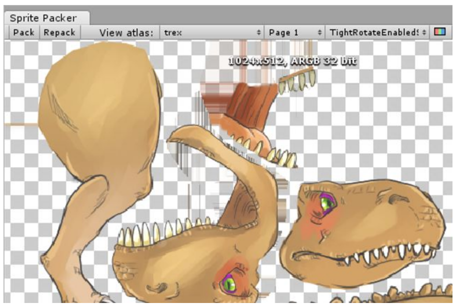


Inspector

Name	Foot R
Z-Order	3
Color	🟩

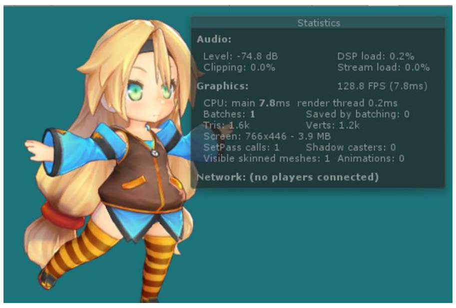
Full self-overlap control

Configure the **Z-Order** of your bind poses to define which parts of the mesh should be in front and which should be behind in case of self-overlaping.



Atlases

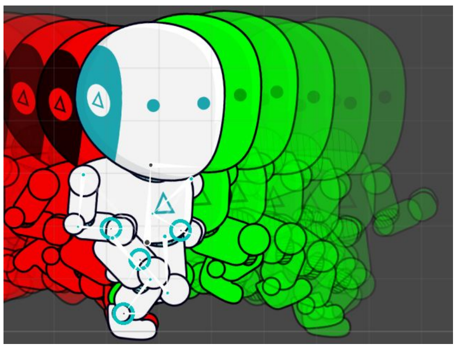
Anima2D is **fully compatible** with Unity's **native Atlases** thanks to geometry override feature. Your skinned sprites will always pick up the atlas texture if available.



```
Statistics
Audio:
  Level: -74.8 dB                DSP load: 0.2%
  Clipping: 0.0%                 Stream load: 0.0%
Graphics:                        128.8 FPS (7.8ms)
  CPU: main 7.8ms  render thread 0.2ms
  Batches: 1             Saved by batching: 0
  Tris: 1.6k             Verts: 1.2k
  Screen: 766x446 - 3.9 MB
  SetPass calls: 1       Shadow casters: 0
  Visible skinned meshes: 1  Animations: 0
Network: (no players connected)
```

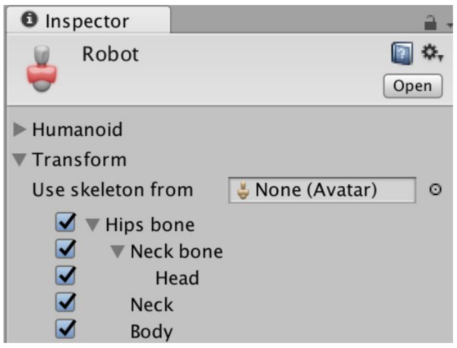
Reduce draw-calls

If all your skinned sprites use the same texture **atlas** you can **combine them in a single skinned mesh** at run-time in order to reduce draw-calls and increase your game's **performance.**



Onion Skin

Visualize following and previous frames of your animation while **creating** it.



Inspector

Robot

Open

Humanoid

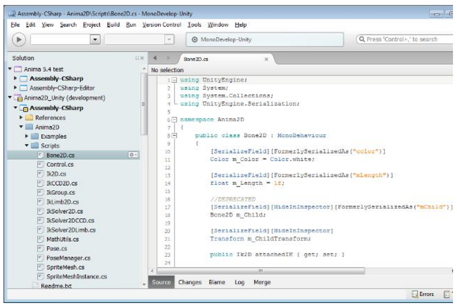
Transform

Use skeleton from: None (Avatar)

- ✓ Hips bone
  - ✓ Neck bone
    - ✓ Head
  - ✓ Neck
  - ✓ Body

Avatar Masks

Create **Avatar Masks** of your animated elements and use them in **Mecanim** layers.



```
using UnityEngine;
using System;
using System.Collections;
using UnityEngine.Serialization;

namespace Anima2D
{
    public class Bone2D : MonoBehaviour
    {
        [SerializeField][FormerlySerializedAs("color")]
        Color m_Color = Color.white;

        [SerializeField][FormerlySerializedAs("mLength")]
        float m_Length = 1f;

        //DEPRECATED
        [SerializeField][HideInInspector][FormerlySerializedAs("mChild")]
        Bone2D m_Child;

        [SerializeField][HideInInspector]
        Transform m_ChildTransform;

        public Ik2D attachedIK { get; set; }
```

Source Code Included

Anima2D comes with full source code.

Unity Integration

We care about **usability** and **integration.** We made an special effort to follow ***The Unity Way*** philosophy by iterating the tool until it became simple to use yet powerful. Full **Undo** support. Asset **previews. Drag and drop** to the Scene View. You will forget you are using a plugin!

4 First Steps

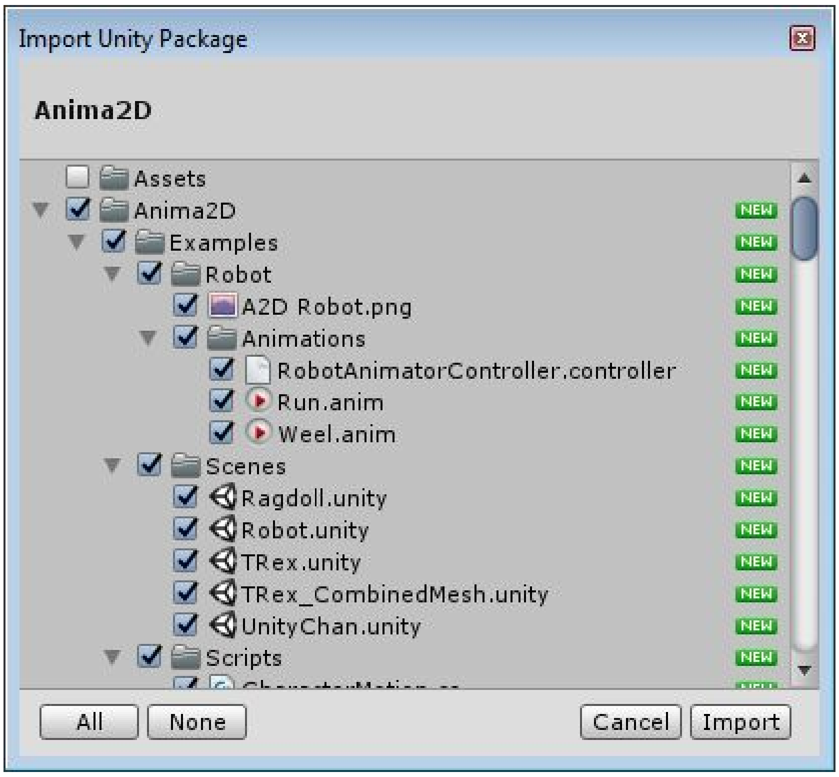
To setup Anima2D please follow the next steps:

4.1 Step 1 – Download Anima2D from Unity Asset Store

Go to the Asset Store, locate Anima2D and press the download button.

4.2 Step 2 – Import Anima2D Unity package

Once the download has finished, you will be presented with the following window. Press **Import** to complete the process.



Import Unity Package

Anima2D

```
☐ Assets
▼ ☑ Anima2D                                   NEW
  ▼ ☑ Examples                                NEW
    ▼ ☑ Robot                                 NEW
        ☑ A2D Robot.png                       NEW
      ▼ ☑ Animations                          NEW
          ☑ RobotAnimatorController.controller NEW
          ☑ Run.anim                          NEW
          ☑ Weel.anim                         NEW
    ▼ ☑ Scenes                                NEW
        ☑ Ragdoll.unity                       NEW
        ☑ Robot.unity                         NEW
        ☑ TRex.unity                          NEW
        ☑ TRex_CombinedMesh.unity             NEW
        ☑ UnityChan.unity                     NEW
    ▼ ☑ Scripts                               NEW
```

Package import dialog.

4.3 Step 3 – Start enjoying Anima2D!

You may want to check the included examples in Assets / Anima2D / Examples, the how to section for step to step guides on how to use Anima2D and the reference section on this manual for detailed information on all the components and windows. Enjoy!

5 How to…

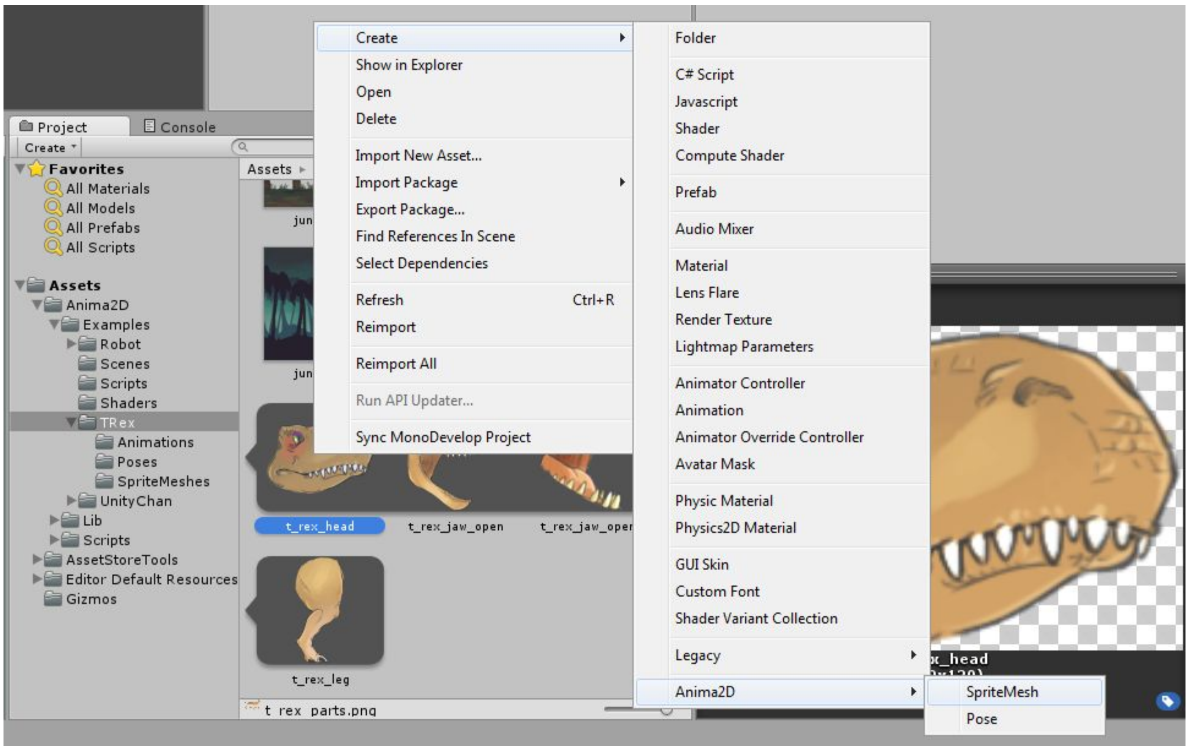
In this section we show you how to use Anima2D for skinning sprites to bones, adding inverse kinematics and creating skeletal animations:

5.1 … skin sprites to bones

1. Create a **SpriteMesh**

   First of all we need to create a **SpriteMesh** from a Unity Sprite. A SpriteMesh is a special Sprite that can be edited and skinned.

   To do that, go to your sprites folder in Project View, right-click on a Sprite and select **Create -> Anima2D -> SpriteMesh** on the context menu.

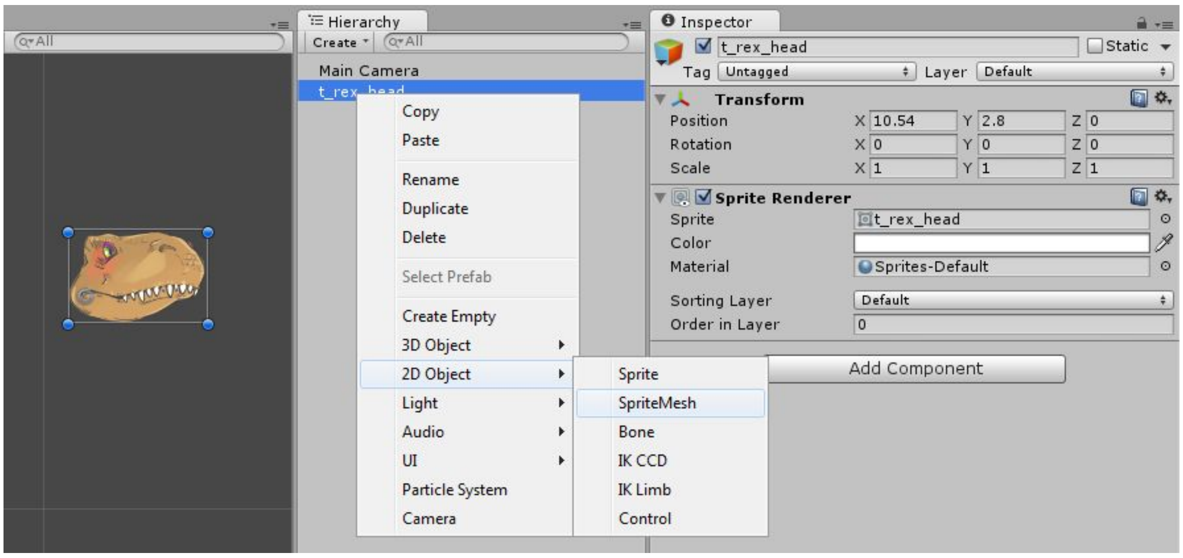


Project View context menu.

   If you want to create SpriteMeshes for all the Sprites in a texture, right-click on the texture instead.

   Important: Keep **one** SpriteMesh per Sprite only. Try not to duplicate your SpriteMesh asset files.

Another way is to right-click on a GameObject with a SpriteRenderer component and select **2D Object -> SpriteMesh.** This will replace the SpriteRenderer by a SpriteMeshInstance, generating the corresponding assets with which the plugin works.



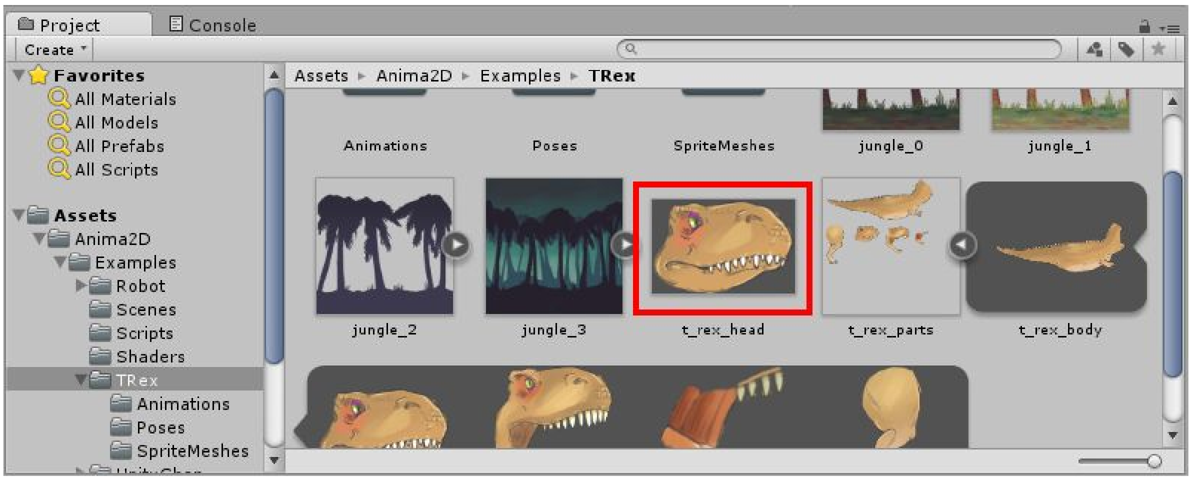
t_rex_head Static
Tag Untagged Layer Default

Transform
Position X 10.54 Y 2.8 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sprite Renderer
Sprite t_rex_head
Color
Material Sprites-Default
Sorting Layer Default
Order in Layer 0

Create SpriteMesh from Hierarchy.

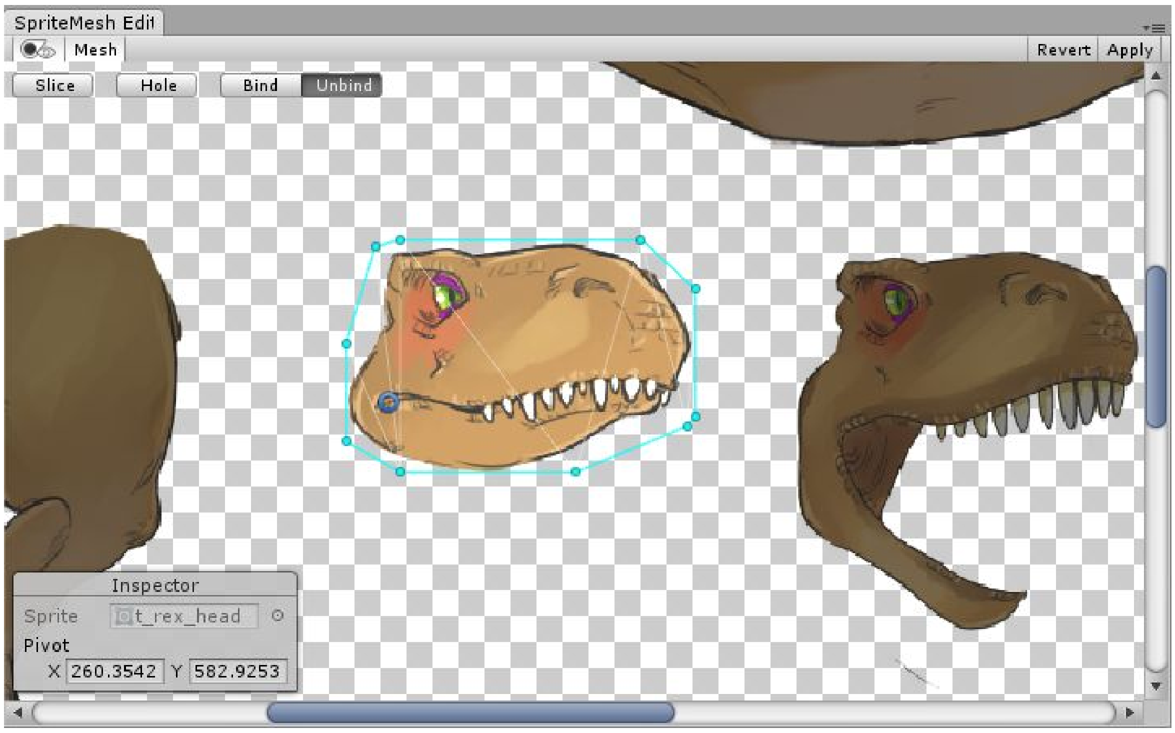
When a SpriteMesh is created, Anima2D will add a new asset similar to a Sprite.



Assets created from the original Sprite.

2. Edit the **SpriteMesh**

After creating the SpriteMesh you may want to edit the default triangulation to suit your skinning purposes. To do that, start by opening the **SpriteMesh Editor** on the menu option **Window -> Anima2D -> SpriteMesh Editor** and select the previously created asset in the Project View. The Sprite will be shown in the editor ready for some editing (read section 6 for more detailed information).



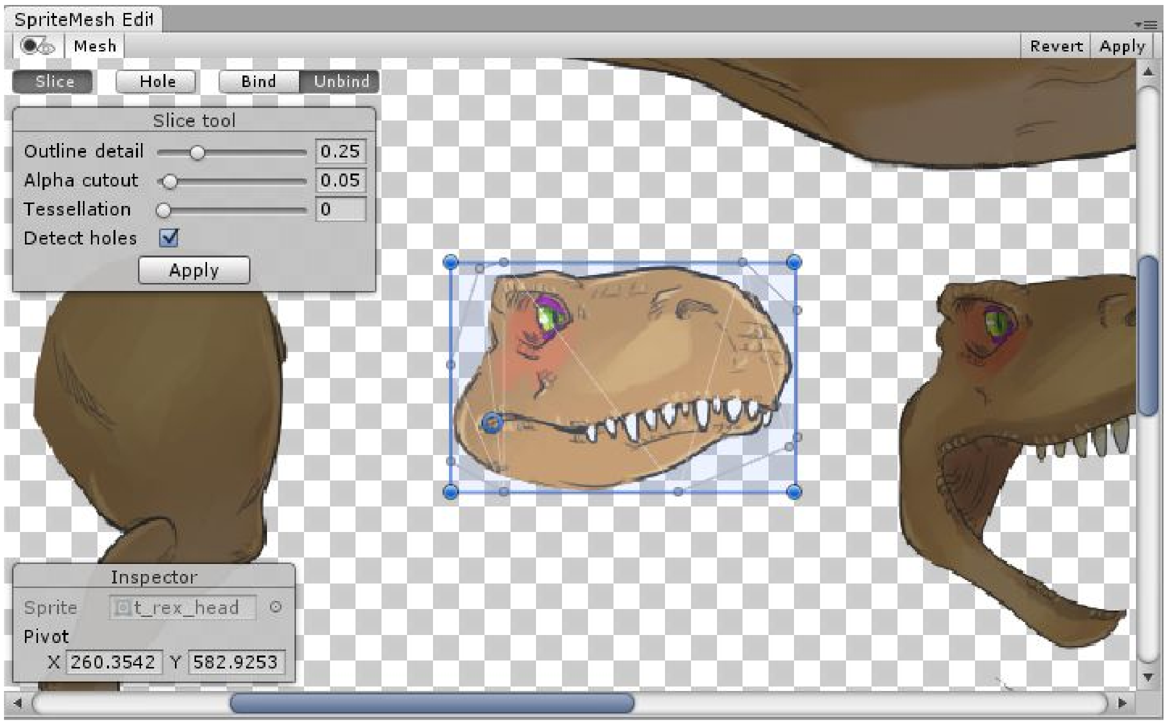
Sprite: t_rex_head
Pivot
X 260.3542 Y 582.9253

SpriteMesh Editor using the sprite's default triangulation.

In the SpriteMesh Editor you have several options to edit manually your SpriteMesh to suit your needs:

- **Move vertices**: Simply **left-click and drag** the existing vertices to the place where you want them to be.
- **Add vertices**: To add new vertices just **double-click** on the image.
- **Delete vertices**: To remove an unwanted vertex just left-click on it to **select** it and press the **Delete** / **Backspace** key on your keyboard.
- **Clear selection:** Do **right-click** anywhere.
- **Split edges**: Clear selection first. Then hold **Shift** and **left-click** to split the closest edge.
- **Add edges**: Select a single vertex first. Then hold **Shift** and **left-click** over another vertex. Otherwise a new vertex will be created.
- **Remove edges**: Select the edge you want to remove and press **Delete** / **Backspace** key on the keyboard.
- **Add holes**: Sometimes you may want to avoid showing a part of the mesh in your game. To do that you can use a hole, which you can add to an enclosed area by pressing the **Hole** button and then **double-clicking** at the place you want to add the hole to.

There is also the possibility to create the topology automatically. Click the **Slice** button to toggle the **Slice tool** (read section 6 for more detailed information).



Outline detail 0.25
Alpha cutout 0.05
Tessellation 0
Detect holes ✓

Sprite t_rex_head
Pivot
X 260.3542 Y 582.9253

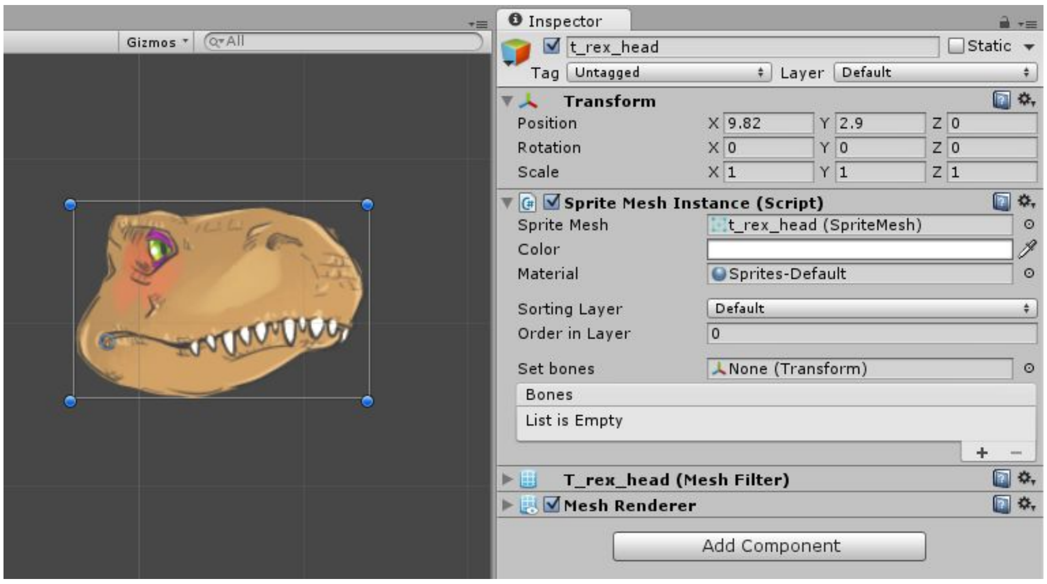
Slice tool.

Once you have your desired topology press the **Apply** button on the top right corner to store the changes. The **original Sprite** will be modified to use this edited geometry.

3. Create a **SpriteMeshInstance**

   A **SpriteMeshInstance** is very similar to a SpriteRenderer component except that it uses a SpriteMesh instead of a Sprite and it can hold a list of bones. You should use the same materials you would use with a SpriteRenderer.

   When the used SpriteMesh has no skinning information the bone list is editable.



t_rex_head Static
Tag Untagged Layer Default

Transform

Position	X 9.82	Y 2.9	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Sprite Mesh Instance (Script)

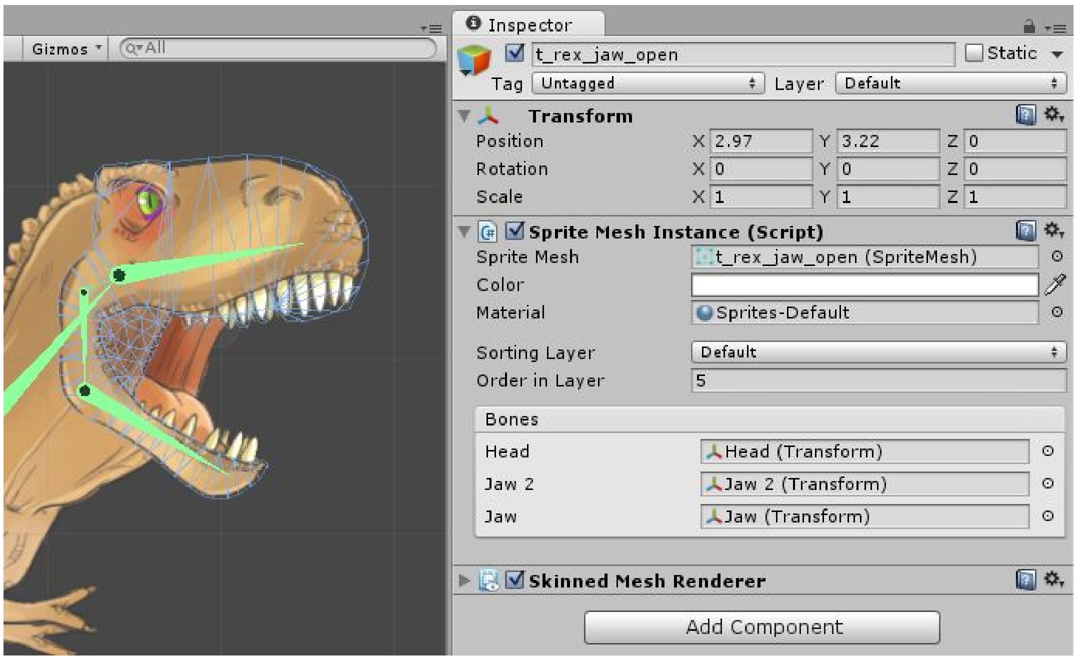
Sprite Mesh	t_rex_head (SpriteMesh)
Color	
Material	Sprites-Default
Sorting Layer	Default
Order in Layer	0
Set bones	None (Transform)

Bones
List is Empty

T_rex_head (Mesh Filter)
Mesh Renderer

SpriteMeshInstance with no skinning information.

Otherwise the list will be fixed and will tell you which bones are needed to be able to enable skinning.



t_rex_jaw_open Static
Tag Untagged Layer Default

Transform

Position	X 2.97	Y 3.22	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Sprite Mesh Instance (Script)

Sprite Mesh	t_rex_jaw_open (SpriteMesh)
Color	
Material	Sprites-Default
Sorting Layer	Default
Order in Layer	5

Bones

Head	Head (Transform)
Jaw 2	Jaw 2 (Transform)
Jaw	Jaw (Transform)

Skinned Mesh Renderer

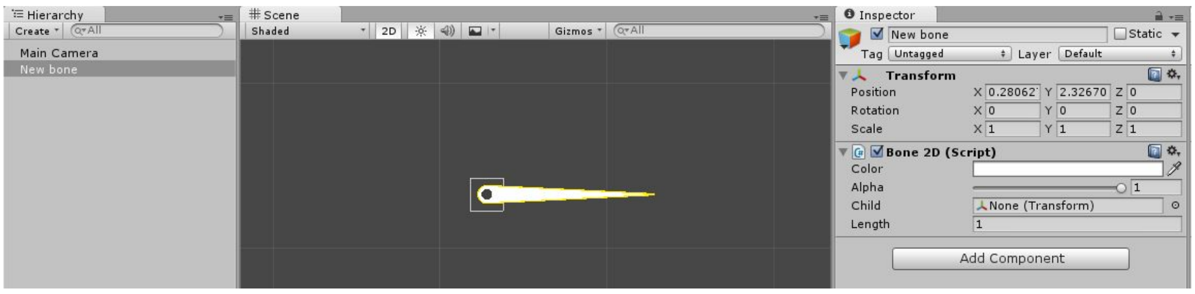
SpriteMeshInstance with skinning information.

To create a SpriteMeshInstance just **drag-and-drop** a SpriteMesh from the Project View to the scene.

4. Create a **Bone hierarchy**

   A **Bone2D** is an object that have a length and can be linked to a child creating a chain. Simply parent, unparent, duplicate, rename them, etc. in order to create a skeleton.

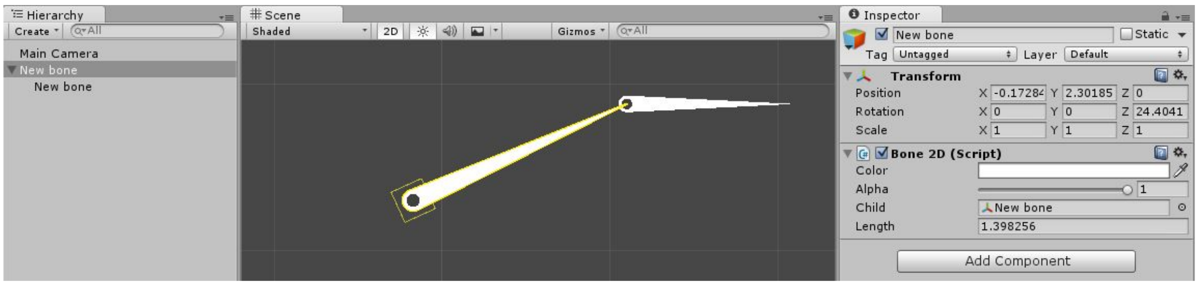
   To create a bone **right-click** in the hierarchy view and select **2D Object -> Bone** from the popup menu, or alternatively use the shortcut **Alt + Shift + B**. If you create a bone while another one is selected it will chain, whenever possible.



New bone
Static
Tag: Untagged
Layer: Default
Transform
Position X 0.28062 Y 2.32670 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Bone 2D (Script)
Color
Alpha 1
Child: None (Transform)
Length 1

New bone.

To create a **bone chain** you need to specify a bone in the **Child** field. A bone with a child will orientate to its child and adjust its length to match its end-point to the child's position.

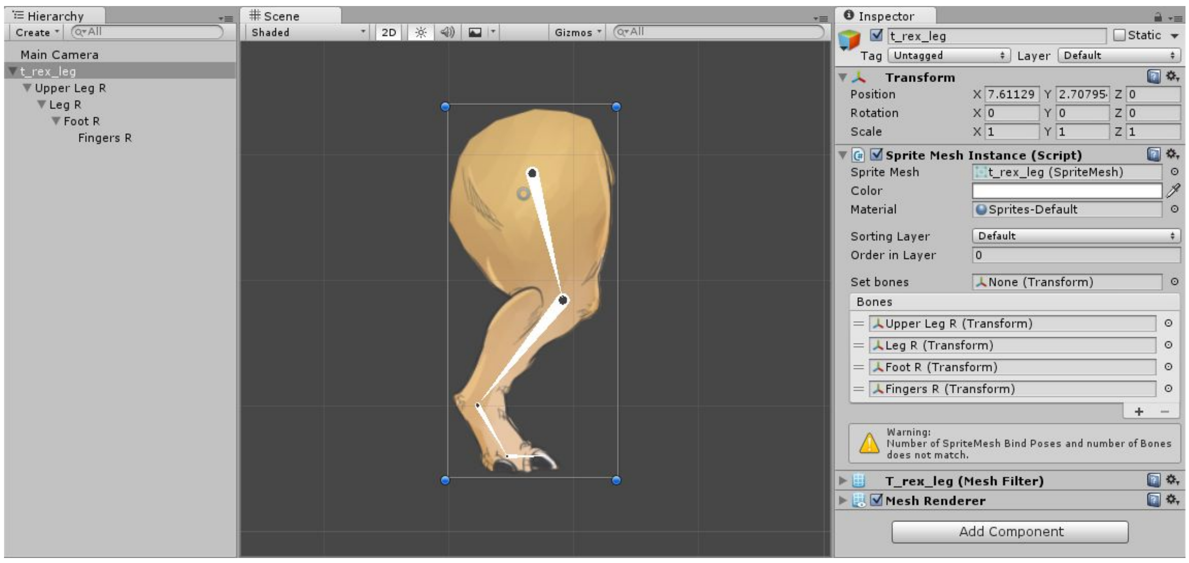


New bone
Static
Tag: Untagged
Layer: Default
Transform
Position X -0.17284 Y 2.30185 Z 0
Rotation X 0 Y 0 Z 24.4041
Scale X 1 Y 1 Z 1
Bone 2D (Script)
Color
Alpha 1
Child: New bone
Length 1.398256

Bone with a linked child bone.

5. **Set bones** to a SpriteMeshInstance

   Once you have both a SpriteMeshInstance and a bone hierarchy aligned in your scene, select the SpriteMeshInstance and drag the bone hierarchy to the **Set bones** field. Then you will be able to see a list of bones that will be used by the instance.



Main Camera
t_rex_leg
Upper Leg R
Leg R
Foot R
Fingers R

t_rex_leg Static
Tag Untagged Layer Default

Transform
Position X 7.61129 Y 2.70795 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sprite Mesh Instance (Script)
Sprite Mesh t_rex_leg (SpriteMesh)
Color
Material Sprites-Default
Sorting Layer Default
Order in Layer 0
Set bones None (Transform)
Bones
Upper Leg R (Transform)
Leg R (Transform)
Foot R (Transform)
Fingers R (Transform)

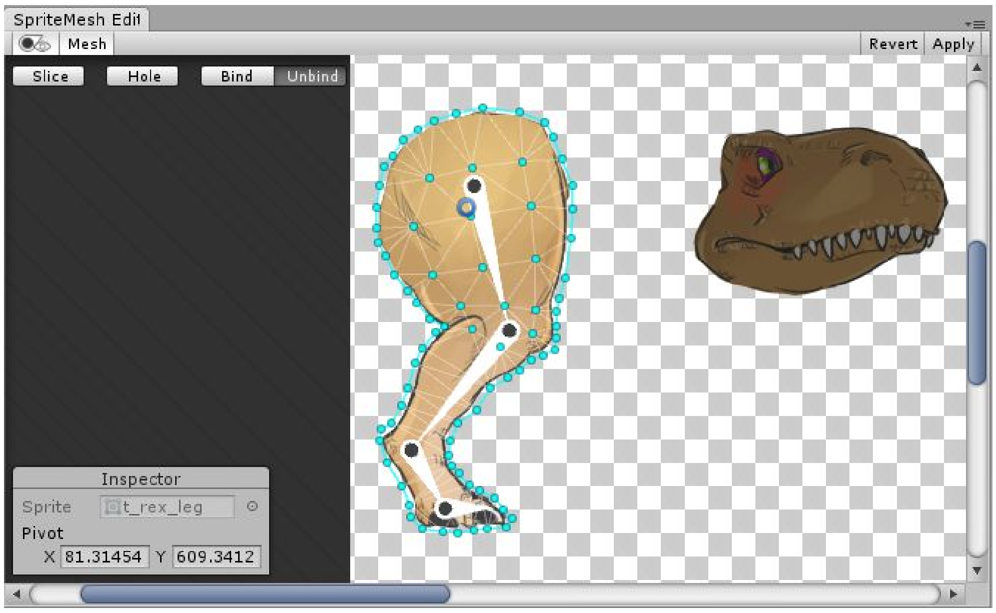
Warning:
Number of SpriteMesh Bind Poses and number of Bones does not match.

T_rex_leg (Mesh Filter)
Mesh Renderer
Add Component

A SpriteMeshInstance with bones.

Another way to add bones is by editing the list manually and dragging the bones to the list's fields.

Open the **SpriteMesh Editor** window again and select your SpriteMeshInstance. This time the bones will appear in the editor.

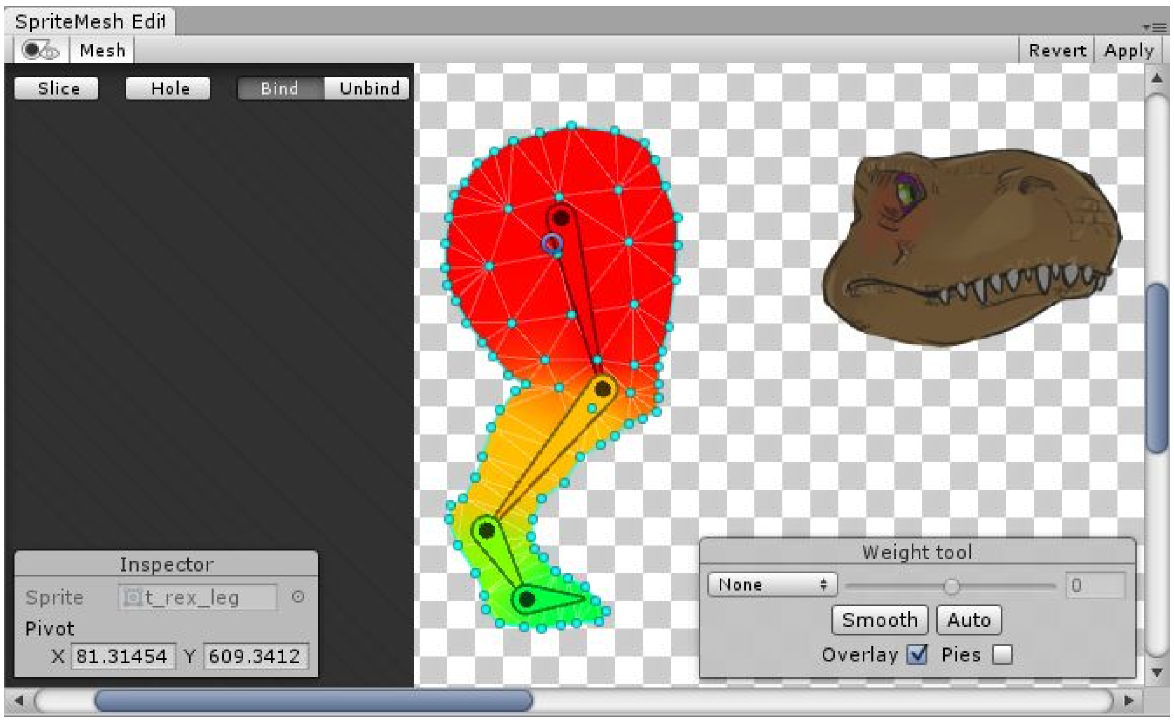


Sprite t_rex_leg
Pivot
X 81.31454 Y 609.3412

SpriteMesh Editor showing the referenced bones from the current SpriteMeshInstance.

6. **Bind** bones

Once the bones appear in the Editor click the **Bind** button to automatically calculate the bone's weights. Bones must be **inside** the Mesh. Select the **Overlay** checkbox to visualize the results.



SpriteMesh Editor showing a colored overlay representing the vertex weights.

You can now adjust the weights by selecting vertices and adding/subtracting bone influence using the **Weight Editor** (see section 5.2 for more detailed information).

Weights can be smoothed by clicking the **Smooth** button. You can smooth only a group of vertices by selecting them first.

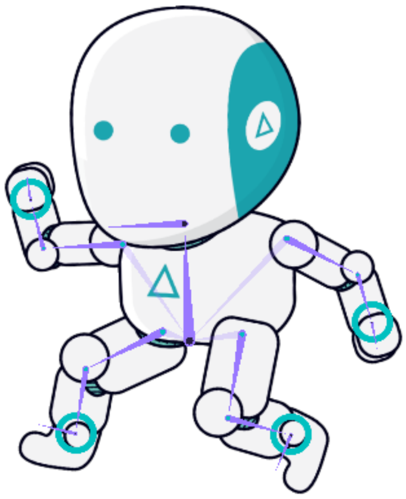
To recalculate weights click on the **Auto** button. This is useful when new vertices are created after the Bind process.

Click the **Apply** button to finalize the changes and your SpriteMeshInstance will do the rest automatically and enable skinning. Now the setup is complete and your mesh will deform properly as bones rotate and move.

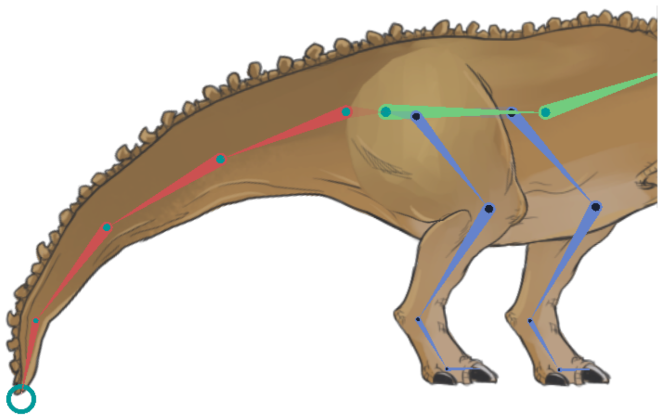
5.2 … USE INVERSE KINEMATICS

To use Inverse Kinematics select a bone in the hierarchy and select **2D Object -> IK Limb** or **2D Object -> IK CCD**. A new GameObject will be created with the corresponding IK component. Move the newly created GameObject to see IK changing bone's pose.

IK Limb needs a 2-bone chain to work and is better suited for character's limbs. It uses a direct law-of-cosine's solver.


Inverse Kinematics controlling the robot's limbs.

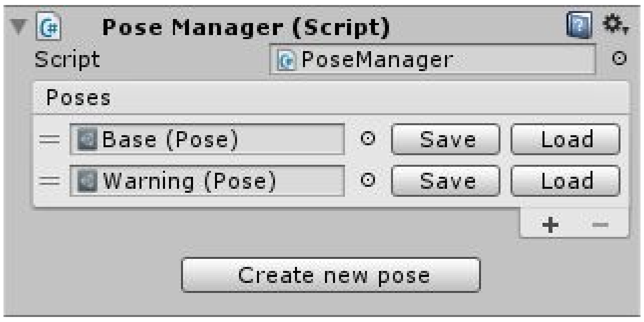
IK CCD can solve any bone chain length by using a Cyclic Coordinate Descent solver.


CCD Inverse Kinematics with a long bone chain.

IK components have a **Weight** field to specify the IK influence to the final pose.

5.3 … SAVE AND LOAD POSES

Once we have our character rig we can save our skeletal objects poses by adding the component *PoseManager* to the object and clicking on the **Create new pose** button. It will store all the child bone's poses in a separate asset. Restore a pose by clicking on the desired pose **Load** button. Click on the **Save** button to overwrite the pose.

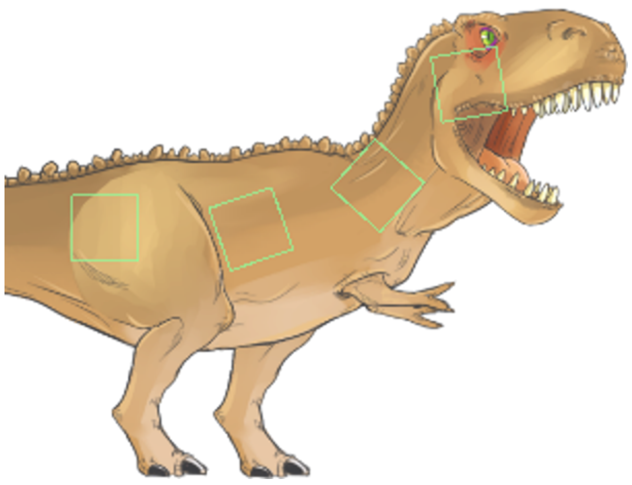


PoseManager component.

5.4 … CREATE CONTROLS

Controls are objects that expose **handles** that can change bone's position and rotation. Controls are useful to expose the main subset of bones that are needed to animate and hide or lock the others. That way you can create your animations in a cleaner interface.

To create controls, right-click on a bone in the hierarchy and select **2D Object -> Control**. You can always set the bone reference manually afterwards.



Controls.

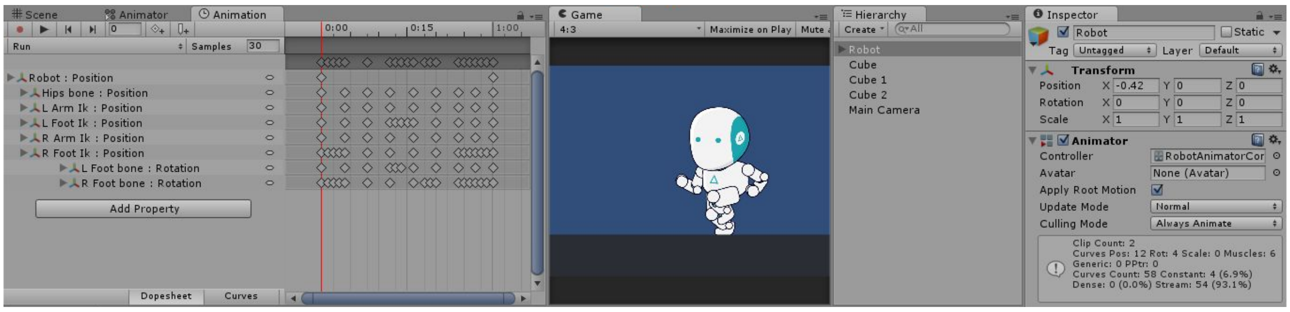
Select the **Rotation Tool** in order to switch to rotation handles.

5.5 … CREATE ANIMATIONS

Finally we can start creating animations using Unity's Animation Window. In this section we will explore how to create animations using **IK** and how to **bake the animation into bones.**

1. Create an **animation file**

   Open the Animation Window (**Window -> Animation**), select your character's root GameObject and select **[Create New Clip]** in Animation Window dropdown to create a new clip and AnimatorController.



Run
Samples 30

Robot : Position
Hips bone : Position
L Arm Ik : Position
L Foot Ik : Position
R Arm Ik : Position
R Foot Ik : Position
L Foot bone : Rotation
R Foot bone : Rotation

Hierarchy: Robot, Cube, Cube 1, Cube 2, Main Camera

Inspector: Robot, Tag Untagged, Layer Default

Transform
Position X -0.42 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Animator
Controller RobotAnimatorCor
Avatar None (Avatar)
Apply Root Motion ✓
Update Mode Normal
Culling Mode Always Animate

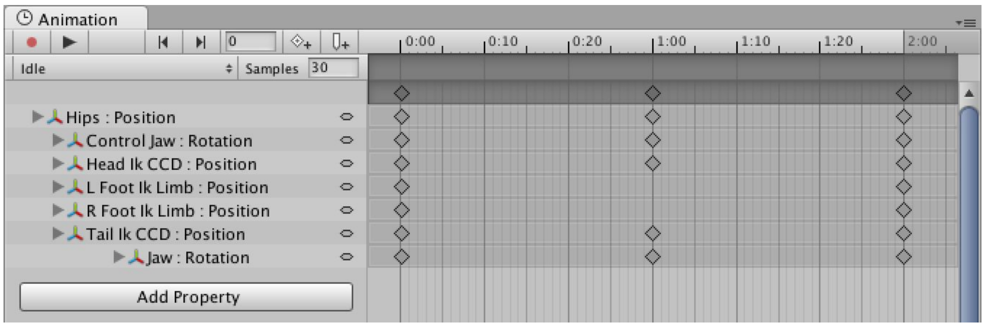
Clip Count: 2
Curves Pos: 12 Rot: 4 Scale: 0 Muscles: 6
Generic: 0 PPtr: 0
Curves Count: 58 Constant: 4 (6.9%)
Dense: 0 (0.0%) Stream: 54 (93.1%)

Creating animations using Unity’s Animation Window.

   For further information regarding Unity’s Animation Window please read Using the Animation View chapter from the Unity Manual.

2. Building animations using **IK**

   Building your animation just by moving bones directly can get complicated when your character have a complex hierarchy. Fortunately IK allow you to pose your character by setting the target position of each bone chain.



Idle
Samples 30

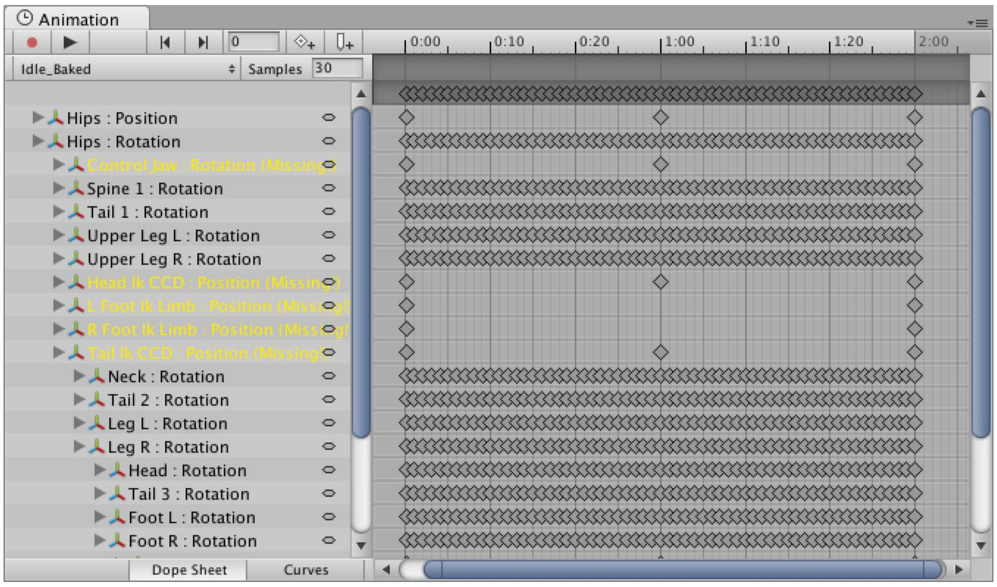
Hips : Position
Control Jaw : Rotation
Head Ik CCD : Position
L Foot Ik Limb : Position
R Foot Ik Limb : Position
Tail Ik CCD : Position
Jaw : Rotation

Animation created using IK objects.

Move and rotate IK objects to create new keyframes and **build your animation using mostly IK curves.** This way you will deal with less curves and will solve complicated movements faster using less keyframes.

3. **Bake animations** into bones

Now that you have your animations built using IK objects it is time to create curves for all the affected bones. This way we will avoid making expensive calculations at runtime. In the other hand this process will add a lot of keyframes and increase the storage size.



Animation baked to bones.

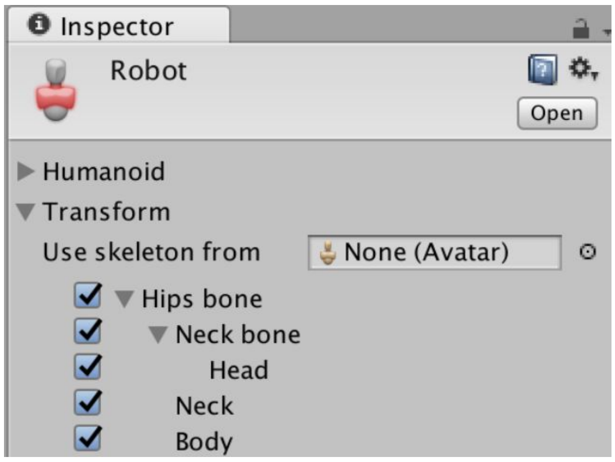
To bake animations into bones you need to select the animation in the Animation Window and then select **Window -> Anima2D -> Bake Animation.** Before doing this process it is recommended to **store a copy** of your animation somewhere. That way you will be able to make changes in the future and repeat the baking. You can now **disable the IK** objects and test your new baked animation.

There is also the possibility to **use IK to record bone keyframes**. This is useful to create simple animations that can work without IK and you don't want to bake in order to keep them as small as possible.

If that is the case, enable the **Record** toggle in your IK component and all the bones affected by your IK will be keyframed too.

5.6 … CREATE AVATAR MASKS

Anima2D allows you to create Avatar Masks for your characters. Avatar Masks are used in Animator Controller's layers to specify which objects should be animated. For more information regarding Avatar Masks check the Avatar Masks section of the Unity Manual.



Avatar Mask.

To create an Avatar Mask, select an object with an Animator component attached and select Window -> Anima2D -> Create Mask. Save your mask anywhere in your project using the Save Dialog.

6. Reference

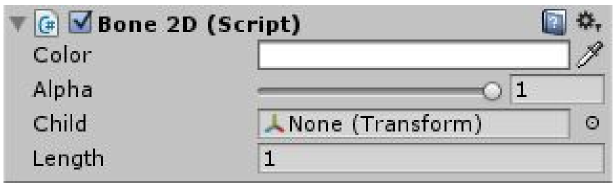
In this section you will find detailed explanations of all elements included within Anima2D, from menu elements to components and editor windows.

6.1. Components

In Anima2D you can find the following components:

Bone2D

Component to define 2D bone hierarchies in your scene.

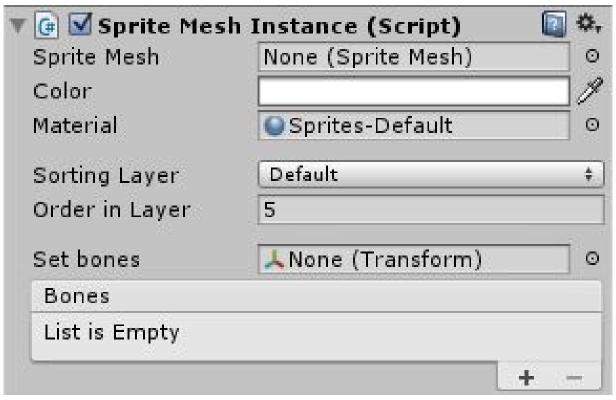


Bone 2D (Script)
Color
Alpha 1
Child None (Transform)
Length 1

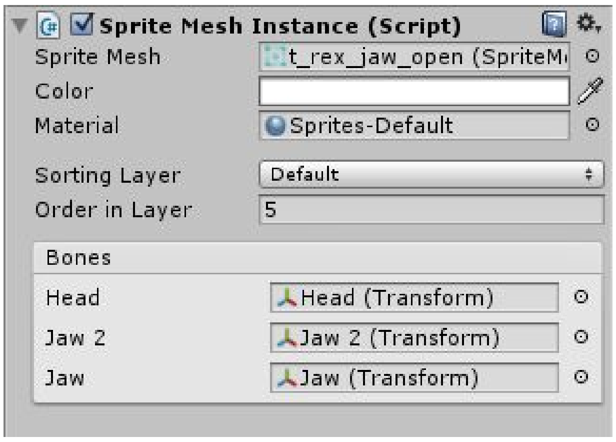
- **Color:** Set the color of the bone.
- **Alpha:** Set the opacity of the bone.
- **Child**: Set a child bone to create a chain of bones. The child bone must be a direct child of the bone. Moving the child bone in the editor will orientate the parent bone accordingly and adapt its length.
- **Length**: The length of this bone.

Sprite Mesh Instance

Component responsible for managing the renderers.



Sprite Mesh Instance (Script)
Sprite Mesh None (Sprite Mesh)
Color
Material Sprites-Default
Sorting Layer Default
Order in Layer 5
Set bones None (Transform)
Bones
List is Empty



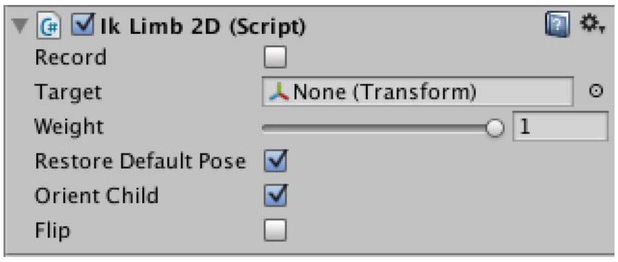
Sprite Mesh Instance (Script)
Sprite Mesh t_rex_jaw_open (SpriteM
Color
Material Sprites-Default
Sorting Layer Default
Order in Layer 5
Bones
Head Head (Transform)
Jaw 2 Jaw 2 (Transform)
Jaw Jaw (Transform)

- **Sprite Mesh**: Reference to the *SpriteMesh* asset.
- **Color:** The color tint of this instance.

- **Material:** Sprite material of this instance.
- **Sorting Layer**: Sets the sorting layer property to the current renderer.
- **Order in layer**: Sets the sorting order property to the current renderer.
- **Set Bones**: Set a hierarchy of bones to the bone list.
- **Bones:** List of bone references that will be used for skinning.

Ik Limb 2D

IK component that operates two bones.

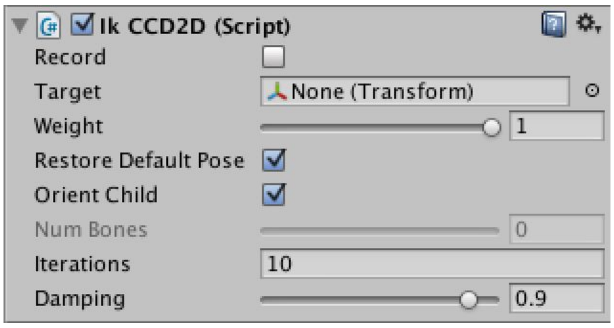

Ik Limb 2D (Script)

Record	☐
Target	None (Transform)
Weight	1
Restore Default Pose	✓
Orient Child	✓
Flip	☐

- **Record:** Record bone keyframes during animation mode.
- **Target**: The target bone to apply IK to.
- **Weight**: The amount of contribution of the IK solver to the final pose.
- **Restore Default Pose:** Set initial pose before performing calculations.
- **Orient Child:** Rotate the target's child bone (if available) to math IK rotation.
- **Flip**: Use the other pose solution for the limb.

Ik CCD2D

IK component that operates a chain of linked bones.


Ik CCD2D (Script)

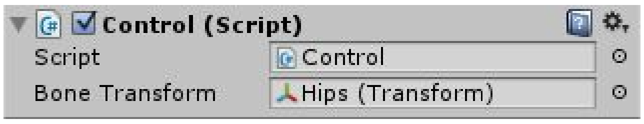
Record	☐
Target	None (Transform)
Weight	1
Restore Default Pose	✓
Orient Child	✓
Num Bones	0
Iterations	10
Damping	0.9

- **Record:** Record bone keyframes during animation mode.
- **Target**: The target bone to apply IK to.
- **Weight**: The amount of contribution of the IK solver to the final pose.
- **Restore Default Pose:** Set initial pose before performing calculations.
- **Orient Child:** Rotate the target's child bone (if available) to math IK rotation.
- **Num bones**: The number of bones affected by the IK solver. The maximum is the root of the bone chain.
- **Iterations**: The number of iterations of the solver.

- **Damping**: Controls the step velocity of the solver. Small values will produce big steps each iteration. Larger values will produce smaller steps. A high damping produces a more natural look.

Control

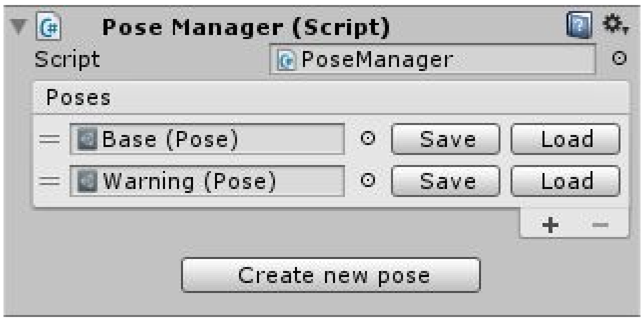
Control components will expose handles for your bones.


Control (Script)
Script: Control
Bone Transform: Hips (Transform)

- **Bone Transform**: reference to the target bone transform.

Pose Manager

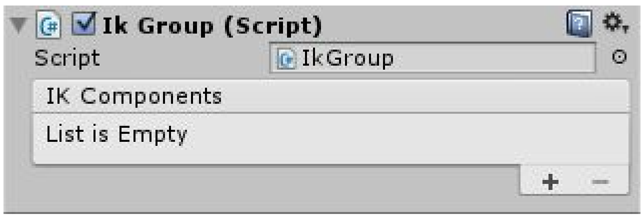
Add a Pose Manager component to the root of your bone hierarchy.


Pose Manager (Script)
Script: PoseManager
Poses
Base (Pose) — Save — Load
Warning (Pose) — Save — Load
Create new pose

- **Save button**: Overwrite the pose. Will save all the GameObject's child bone poses.
- **Load button**: Restore the pose.
- **Create new pose button**: Creates a new asset containing the current pose.

IkGroup

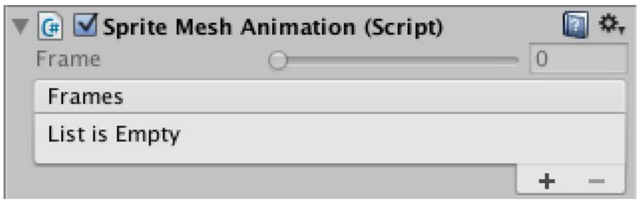
This component will update a group of IK objects. This is useful when you want to specify the execution order of IK elements in runtime.


Ik Group (Script)
Script: IkGroup
IK Components
List is Empty

- **IK Components**: List of IK references.

SpriteMesh Animation

Use this component to swap the SpriteMesh of a SpriteMeshInstace from an animation by animating the **frame** property.

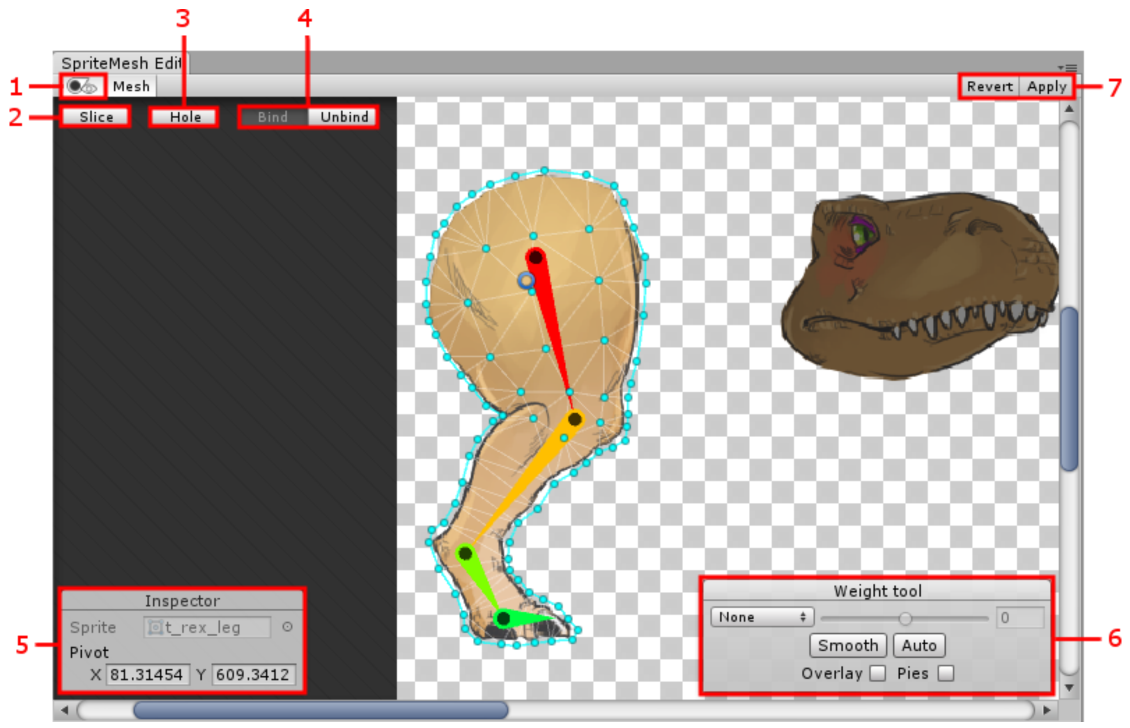

Sprite Mesh Animation (Script)
Frame 0
Frames
List is Empty

- **Frame**: Index of the frame to swap.
- **Frames**: List of SpriteMesh to swap during the animation.

6.2. SpriteMesh Editor Window

The SpriteMesh Editor Window is Anima2D’s main tool for mesh editing. You will be able to create any mesh topology, set bind poses, weights, etc.

The Editor is composed by the following elements:



Inspector
Sprite t_rex_leg
Pivot
X 81.31454 Y 609.3412

Main editor elements.

1. Toggle bone's visualization.
2. Open/close the Slice tool.
3. Toggle add holes.
4. Bind/Unbind bones.
5. The Inspector.
6. The Weight tool.
7. Apply/Revert changes.

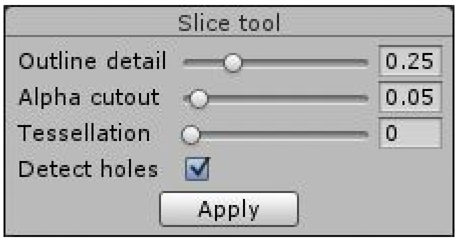
6.2.1 Editing the geometry:

- **Move vertices**: left-click and drag a vertex.
- **Add vertices**: double-click on the image.
- **Delete vertices**: press the Delete / Backspace key to remove a vertex selection.
- **Select vertices:** Click and drag to use the selection rectangle. Hold Ctrl/Cmd to add to current selection.
- **Clear selection:** Do right-click anywhere.

- **Split edges**: Clear selection first. Then hold Shift and left-click to split the closest edge.
- **Add edges**: Select a single vertex first. Then hold Shift and left-click over another vertex. Otherwise a new vertex will be created.
- **Remove edges**: Select the edge you want to remove and press Delete / Backspace key on the keyboard.
- **Add holes**: Toggle the Hole button and double-click where you want to add the hole to.
- **Remove bones/bind poses**: Select a bone or bind pose and press Delete / Backspace key.
- **Move pivot point**: drag the blue circle. Hold Ctrl/Cmd to snap to pixel.

6.2.2. Slice tool:

The Slice tool allows you to cut the mesh automatically from the image outline. Click on the button **Slice** to open the tool:


Slice tool

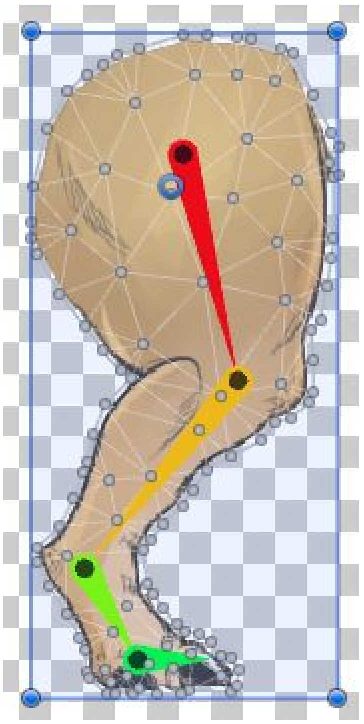
Outline detail	0.25
Alpha cutout	0.05
Tessellation	0
Detect holes	✓

Slice tool parameters.

- **Outline detail**: The bigger this parameter is the more number of vertices will be created in the outline.
- **Alpha cutout**: The alpha tolerance used to discard pixels.
- **Tessellation**: If you want inner vertices in your mesh topology increase this parameter:
- **Detect holes**: Hole areas will be detected (notice that you still need to place a hole in that area).

Use the rectangle handles to adjust the area of your sprite and configure the parameters.

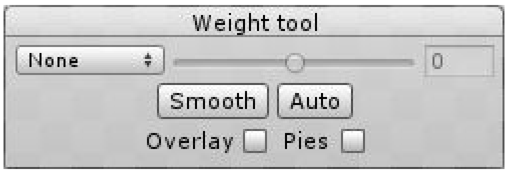
Then press the **Apply** button of the tool.



Rectangle handles and calculated geometry.

6.2.3 Weight tool:

The Weight tool will let you add and subtract bone influence from the selected vertices.



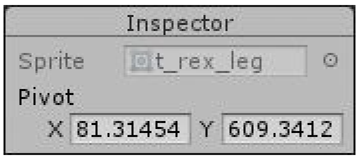
Weight tool.

The Weight tool will show a dropdown list and a disabled slider. Select a bone from the dropdown list or by left-clicking a bone in the window. Move the slider to the right to add influence. Move the slider to the left to subtract.

- **Smooth**: smooth weights from a group of neighbouring vertices by averaging them.
- **Auto**: calculates weights from selected vertices automatically. If no vertices are selected all of them will be used.
- **Overlay**: shows a coloured mesh representing the per-vertex bone influences.
- **Pies**: shows a pie chart to each vertex representing its weights.

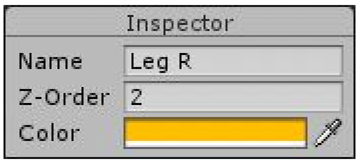
6.2.4 Inspectors:

Depending on the selected element the Inspector will show different information and options:


Inspector
Sprite t_rex_leg
Pivot
X 81.31454 Y 609.3412

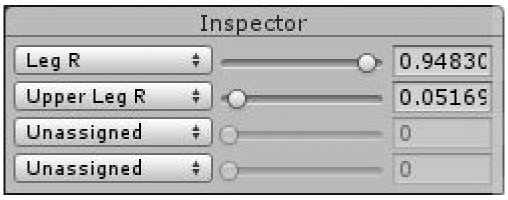
Empty selection.

- **Empty selection:** if nothing is selected the inspector will show the sprite and pivot point of the SpriteMesh.


Inspector
Name Leg R
Z-Order 2
Color

Bind pose selection.

- **Bind pose selection:** if a bind pose is selected you will be able to edit the name, Z-Order and display color. The Z-Order is used to calculate which vertices go in front or behind the others in case of self-overlap.

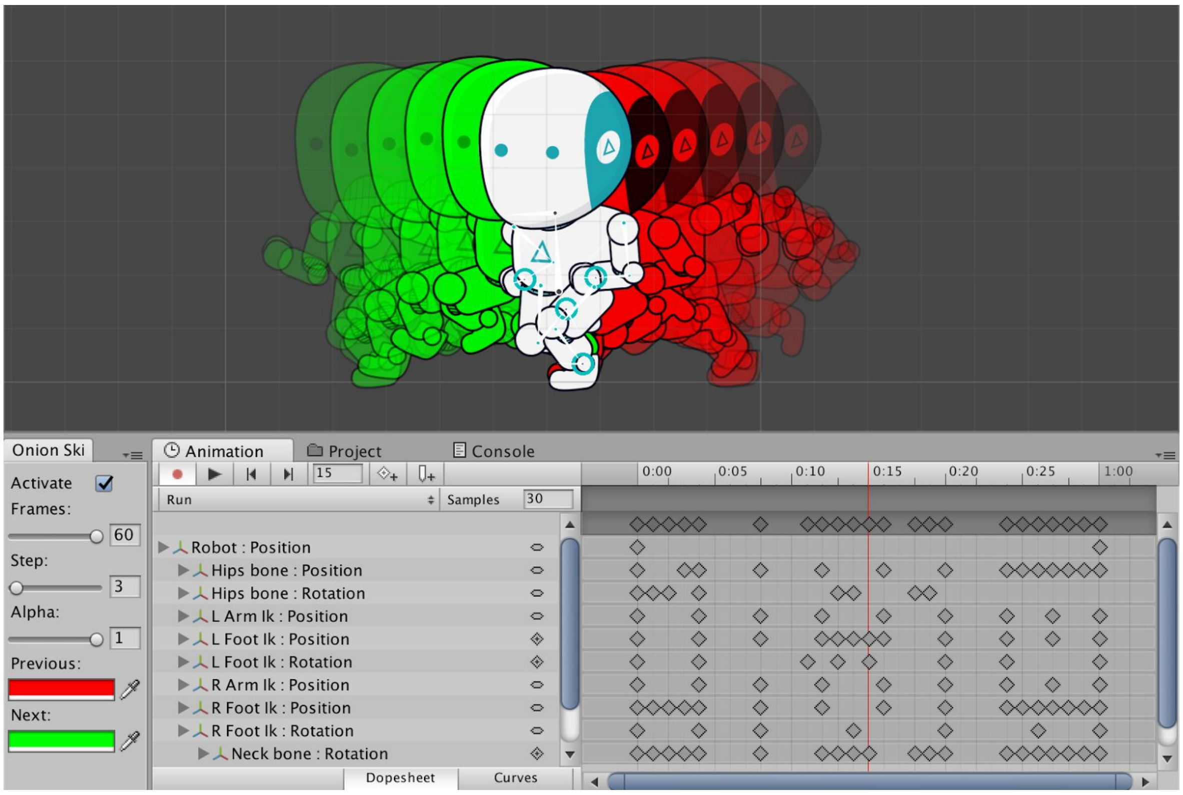

Inspector
Leg R 0.94830
Upper Leg R 0.05169
Unassigned 0
Unassigned 0

Weighted vertex selection

- **Vertex (bound) selection**: will show the vertex's weights as sliders and each associated bone. This allows for manual single vertex editing. Move the sliders to change the weights and associate to other bones by selecting a new bone from the dropdown list.

6.3. Onion Skin

Onion Skin is a tool that shows the previous and following frames of the current frame in order to get a better picture of the motion.



Onion Ski

Activate ✓

Frames: 60

Step: 3

Alpha: 1

Previous: 🟥

Next: 🟩

Run — Samples 30

Onion Skin integrated with Unity’s Animation Window.

Open the Onion Skin Window select **Window -> Anima2D -> Onion Skin.** Dock the window next to your Animation Window for a more convenient access. The window has the following options:

- **Activate**: Enable or disable the preview.
- **Frames**: Number of frames to show.
- **Step**: Number of frames to skip before showing a preview.
- **Alpha**: Global preview transparency.
- **Previous (color)**: Color of the previous frames.
- **Next (color)**: Color of the next frames.

Tip: Select an object with a renderer component to preview that one alone.

7. Frequently Asked Questions

Q: I found a bug in Anima2D. Where can I report it?

A: Please, drop us an email to support@anima2d.com if you found any bugs, have an issue with the plugin or would like to share with us an improvement suggestion. We'd love to hear from you!

Q: How is the mobile performance?

A: Anima2D doesn't have any impact on the performance as it uses only Unity components in runtime (SkinnedMeshRenderer). It is mainly an Editor Tool.