ANIMA 2D

ADVANCED ANIMATION FOR UNITY

USER GUIDE

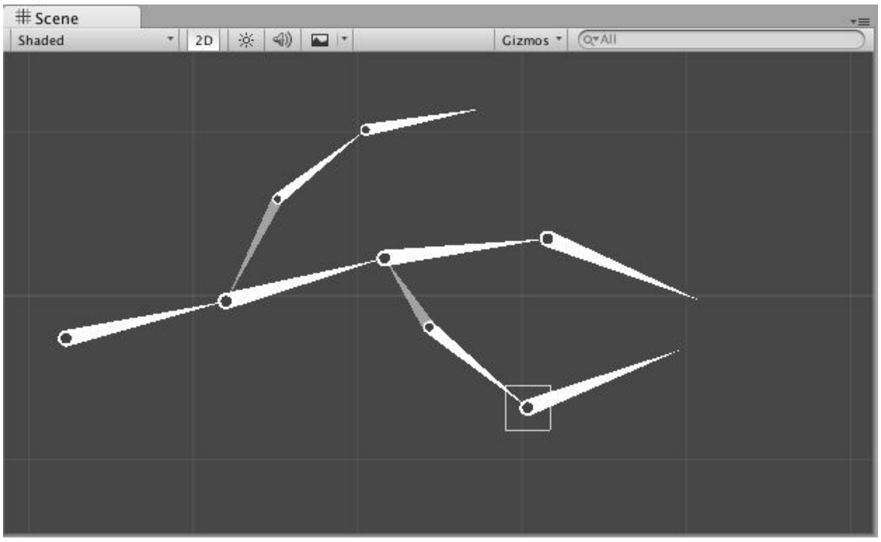
1 Table of Contents

2 Features

Anima2D is a 2D Skeletal Animation plugin for use with Unity 5.x. Anima2D helps you improve your 2D animation workflow "the Unity way" by including the following features:

- **Bones**

2D Bones can be created easily from the context menu **2D Objects**. Bones have a length property and can be linked as chains. They can be rotated using the regular rotation tool or by dragging their body.



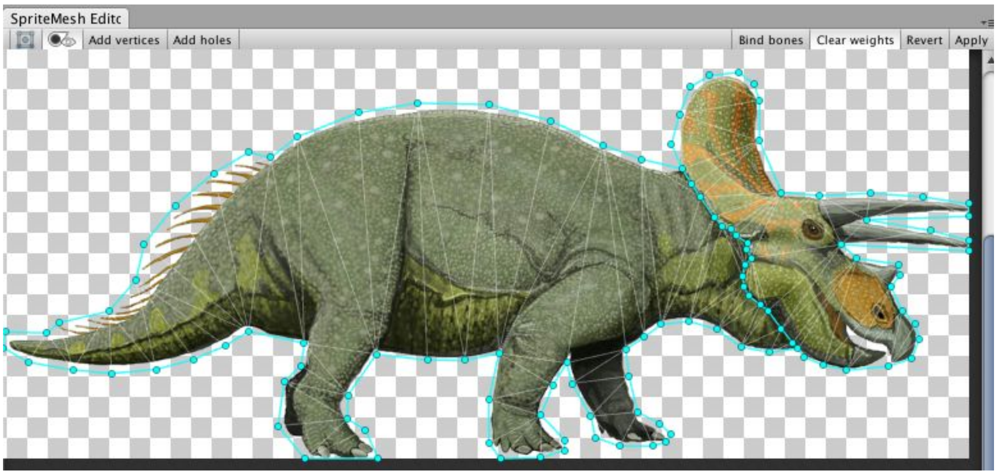
Bone hierarchy.

- **Sprite to mesh conversion**

Sprites can be converted to meshes with a single click from the Project View. It generates new asset files (mesh, default material and custom *SpriteMesh* asset).

- **SpriteMesh Editor**

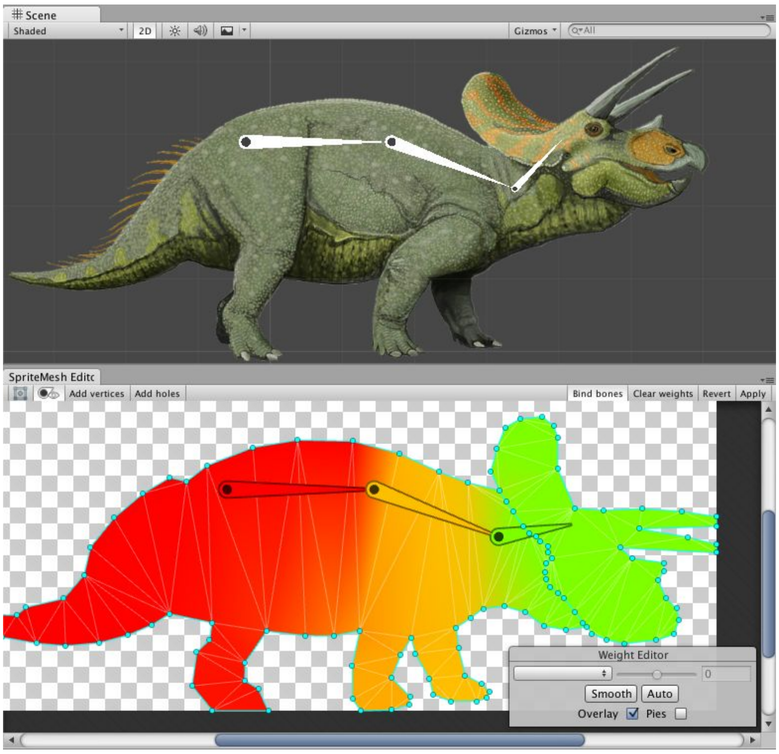
Meshes can be fine-tuned from a proper editor window. You can add and delete vertices, holes, edges and edge constraints to achieve the desired triangulation.



Sprite Mesh editor window.

- Automatic weights

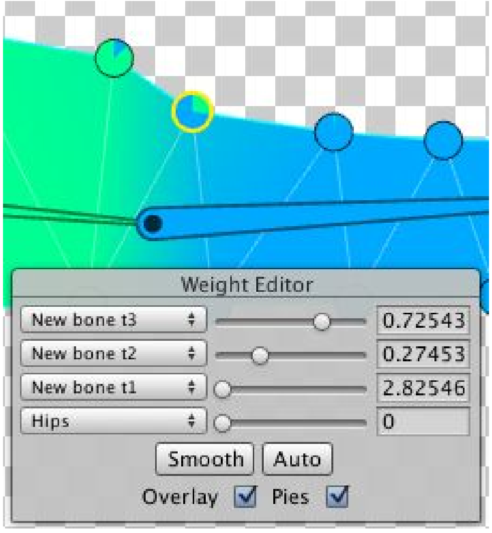
Automatic weights are calculated by complex state of the art algorithms. Calculated weights work out of the box in most of the situations. You can visualize the weights as a coloured overlay in the editor window.



Skinning using out of the box automatic weights.

- **Weight Editor**

You can edit weights by setting the desired value or by selecting vertices and adding/subtracting influence of a bone.



Weight Editor

New bone t3	0.72543
New bone t2	0.27453
New bone t1	2.82546
Hips	0

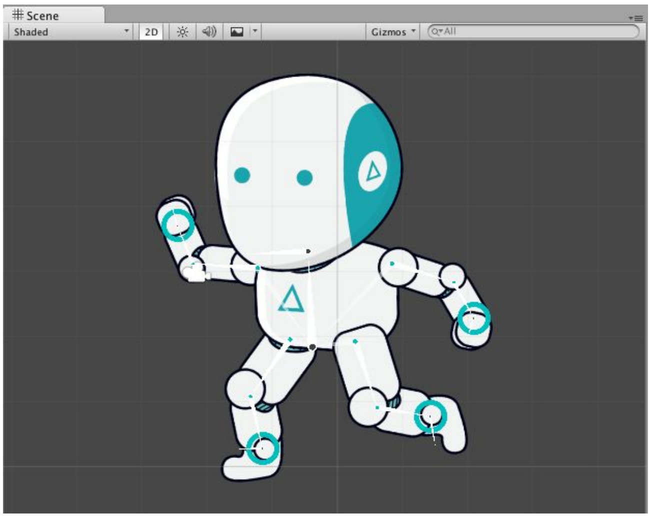
Smooth Auto

Overlay ☑ Pies ☑

Weight Editor.

- **Inverse Kinematics**

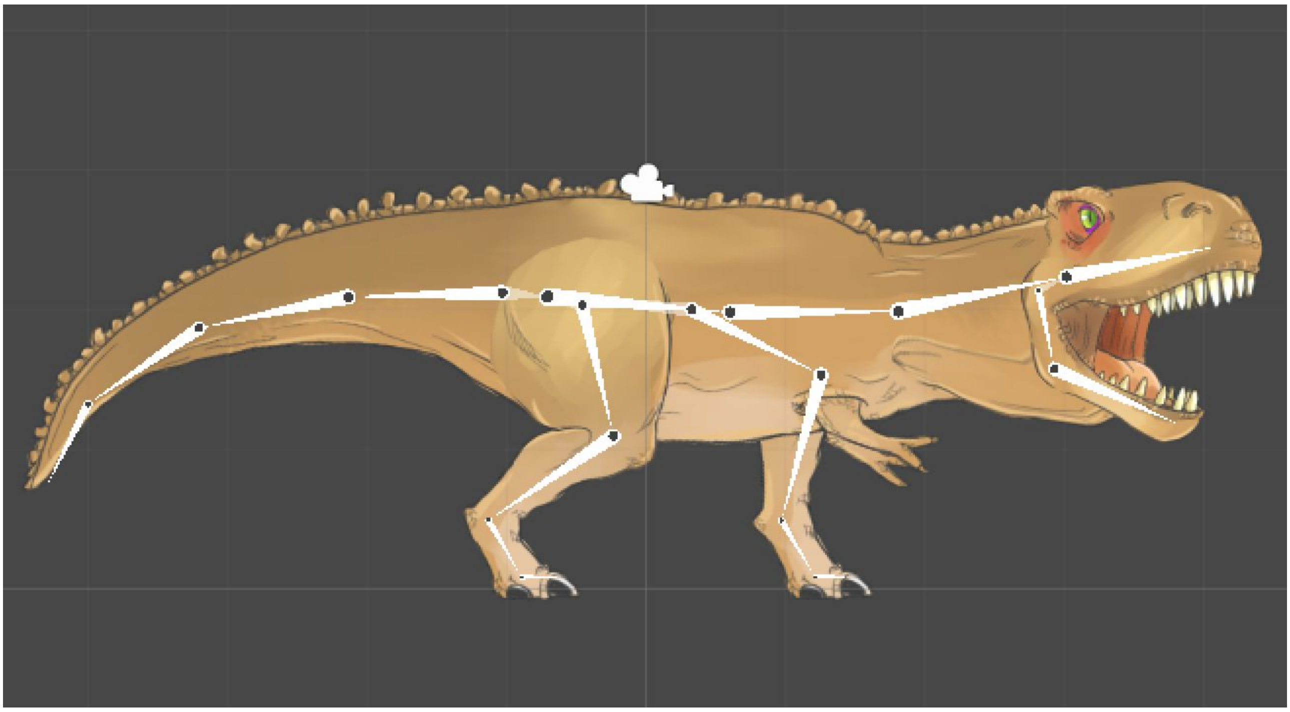
Currently we have IK for limbs and arbitrary chain length (using CCD solver). Gizmos show the affected bones.



Robot pose created using Inverse Kinematics.

- **Save / load poses**

Save the bone's position as a separate asset. Restore the pose at your will anytime. Useful to keep track of the binding pose or to set keyframes while creating animations.



T-Rex result after loading "Warning" pose.

3 First Steps

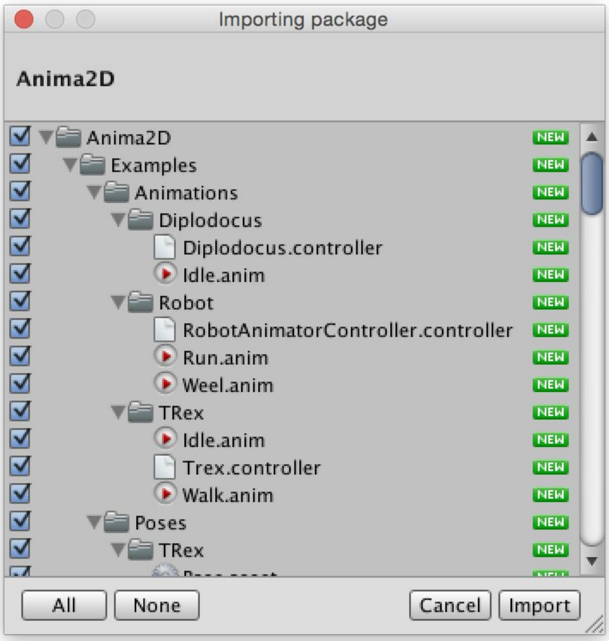
To setup Anima2D please follow the next steps:

3.1 Step 1 – Download Anima2D from Unity Assets Store

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

3.2 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.



Importing package

Anima2D

- ☑ Anima2D NEW
- ☑ Examples NEW
- ☑ Animations NEW
- ☑ Diplodocus NEW
- ☑ Diplodocus.controller NEW
- ☑ Idle.anim NEW
- ☑ Robot NEW
- ☑ RobotAnimatorController.controller NEW
- ☑ Run.anim NEW
- ☑ Weel.anim NEW
- ☑ TRex NEW
- ☑ Idle.anim NEW
- ☑ Trex.controller NEW
- ☑ Walk.anim NEW
- ☑ Poses NEW
- ☑ TRex NEW

Package import dialog.

3.3 Step 3 – Start enjoying Anima2D!

You may want to check the tutorials on our youtube channel https://www.youtube.com/mandarinagames, 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!

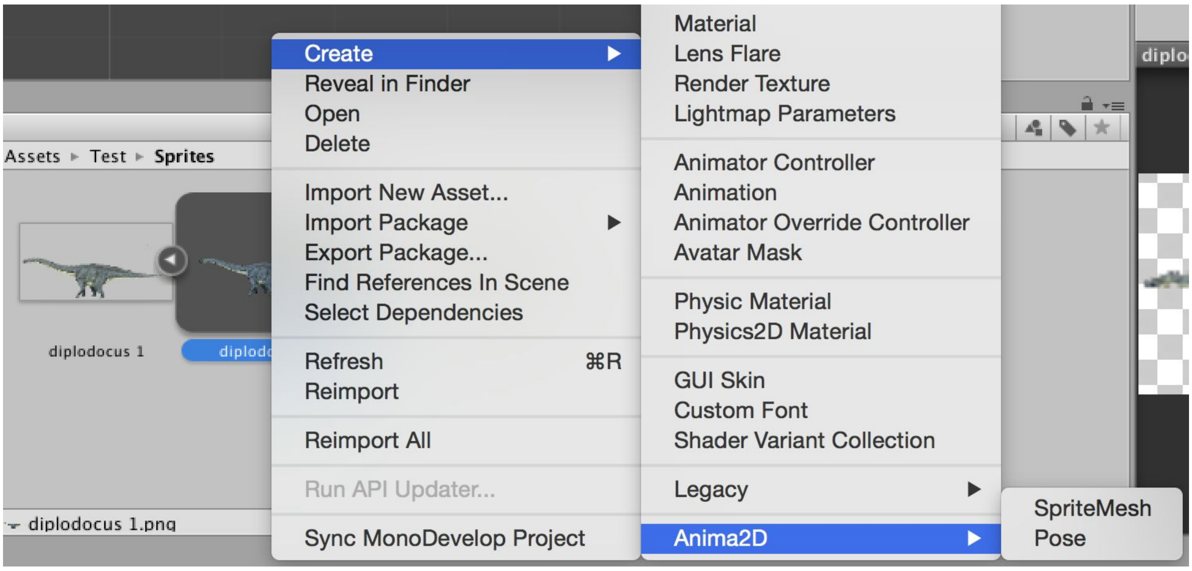
4 How to...

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

4.1 ... Skin sprites to bones

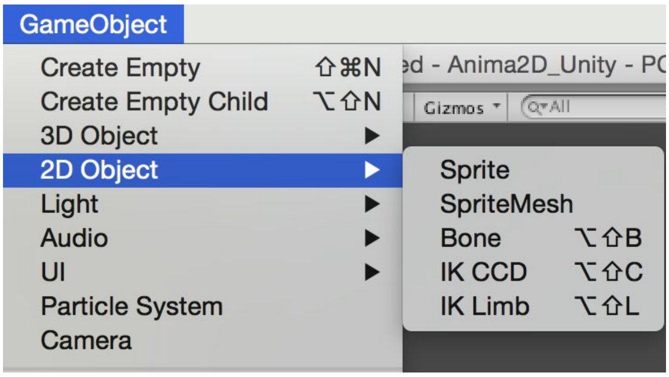
1. Create a mesh

   First of all we need to create a mesh from a Unity Sprite. 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.



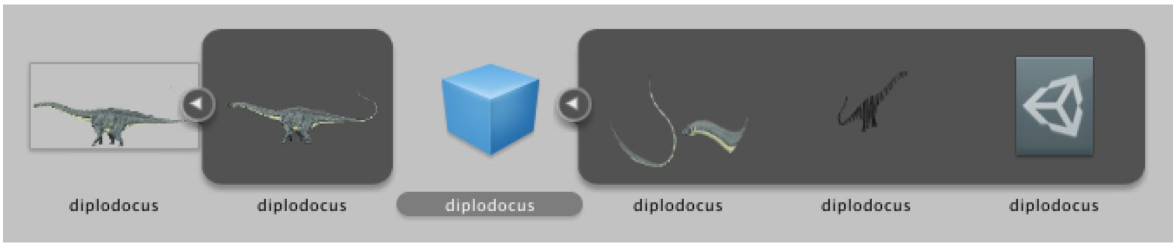
Project View Anima2D context menu.

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.



New 2D Objects added to the menu.

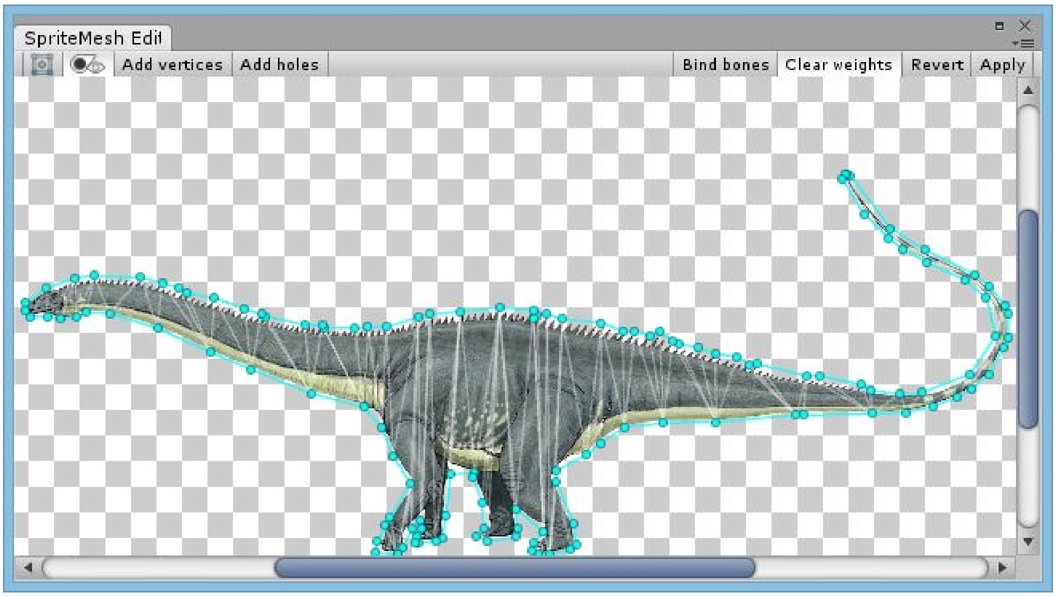
When a *SpriteMesh* is created, Anima2D will add a new asset with a child Mesh, Material and a custom *SpriteMesh*.



Assets created from the original Sprite.

2. Edit the *SpriteMesh*

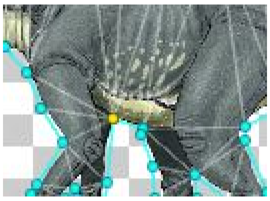
   After creating the *SpriteMesh* you may want to edit the default triangulation to suit your skinning purposes. To do that, start by opening the *SpriteMesh* Editor on the menu option **Window -> SpriteMesh Editor** and select the previously created asset in the Project View. The Sprite will be shown in the editor ready for some editing (read section 5 for more detailed information).



SpriteMesh Editor using the sprite's default triangulation.

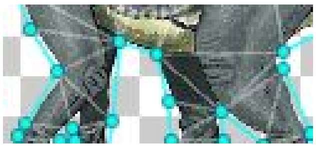
In the *SpriteMesh* Editor you have several options to edit your *SpriteMesh* to suit your needs:

- **Move vertices**: Simply click and drag the existing vertices to the place where you want them to be. For example, you may want to place a vertex closer to the image.



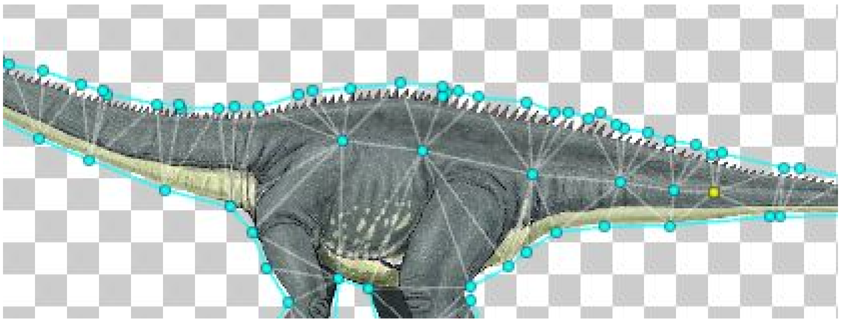
Move the vertices by dragging them.

- **Delete vertices**: To remove an unwanted vertex just click on it to select it and press the **Delete** / **Backspace** key on your keyboard. For example, you may want to create a simpler mesh topology.

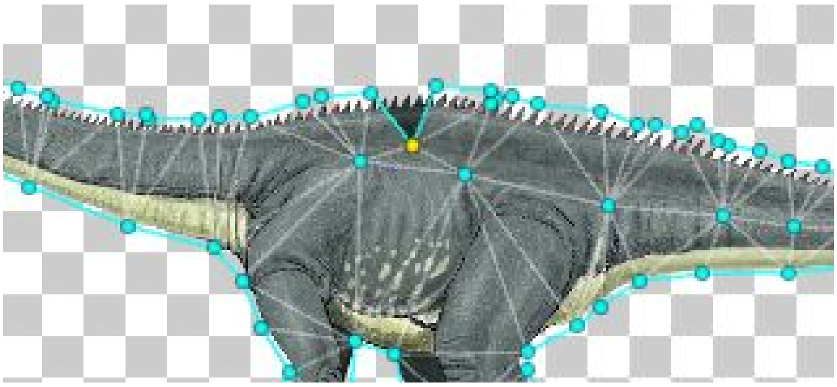


Vertex removed.

- **Add vertices**: To add new vertices to the mesh just click on the **Add vertices** button and click on the image to place a new vertex. If you click with the shift key pressed, the closest edge will be split. For example, we may want to add vertices inside the body.

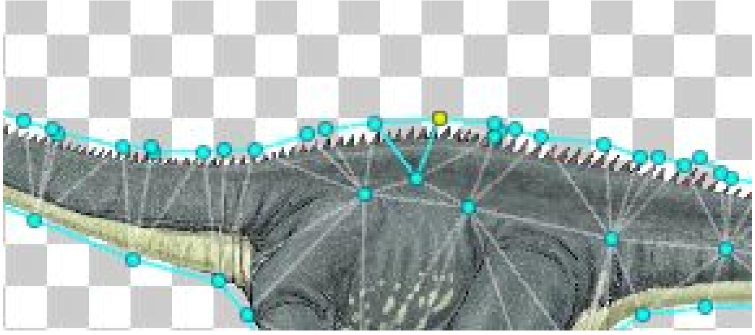


New vertices inside the outline path.



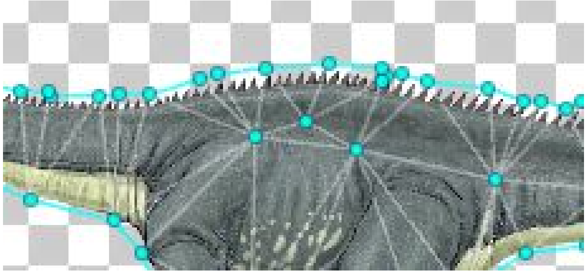
Using the shift key the new vertex will split the closest edge.

- **Add edges**: To join vertices with an edge you will have to uncheck the **Add Vertices** button, click on the first vertex you want to join, then leave **Shift** pressed and click the second vertex to join. For example, in the next image we will join again the two vertices we split in the previous example.



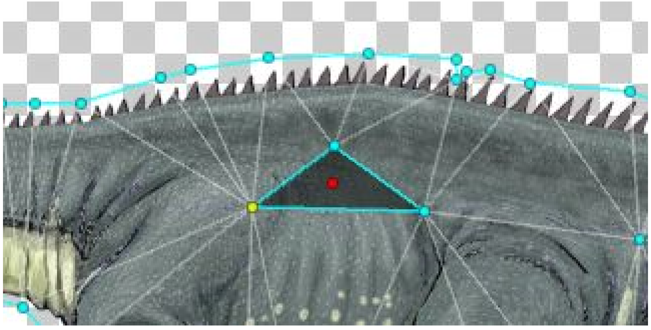
New edge created.

- **Remove edges**: Select each edge you want to remove and press **Delete** / **Backspace** key on the keyboard. For example, we will remove the edges created on the split edge in the previous example.



Removed unnecessary edges.

- **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 **Add holes** button and then clicking at the place you want to add the hole to. For example, after adding edges to create an inner edge path, we have added a hole to eliminate that area.

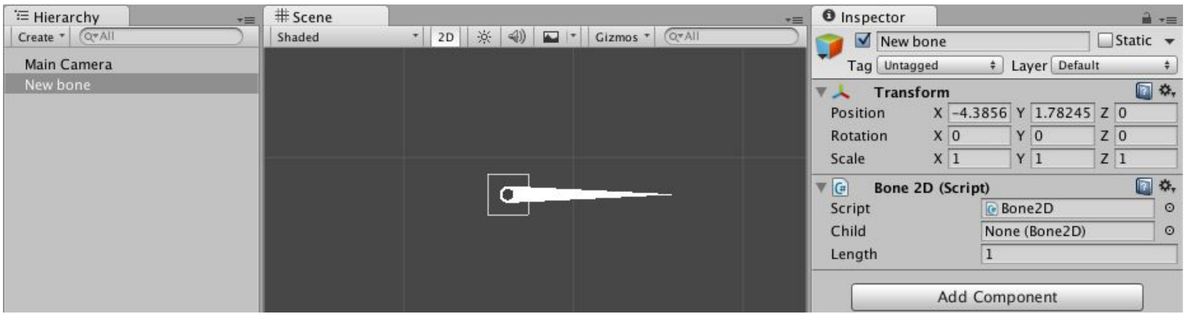


Hole in the mesh.

- **Apply changes**: Once you have your desired topology press the **Apply** button, so the changes done to our *SpriteMesh* and *Mesh* are applied.

3. Create bones hierarchy

   To create a bone right-click on a GameObject and select **2D Object -> Bone** from the popup menu, or alternatively select a GameObject and press **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

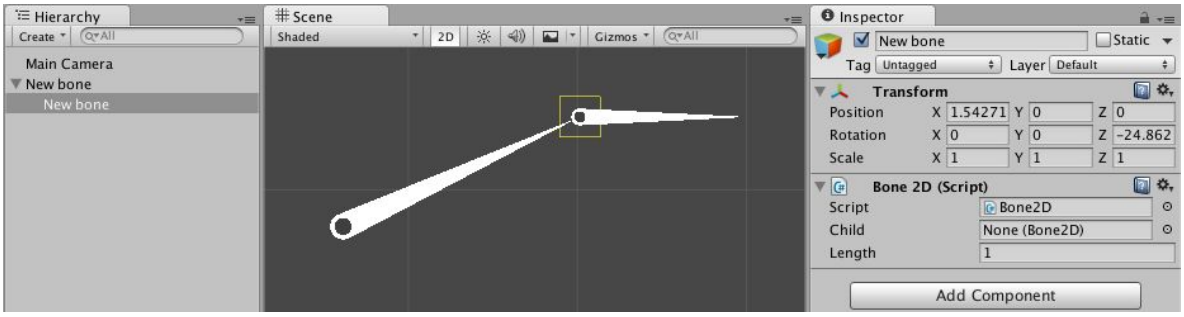
	X	Y	Z
Position	-4.3856	1.78245	0
Rotation	0	0	0
Scale	1	1	1

Bone 2D (Script)

Script	Bone2D
Child	None (Bone2D)
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 (Editor only) and adjust its length to match its end-point to child's position.



New bone
☐ Static
Tag Untagged
Layer Default

Transform

	X	Y	Z
Position	1.54271	0	0
Rotation	0	0	-24.862
Scale	1	1	1

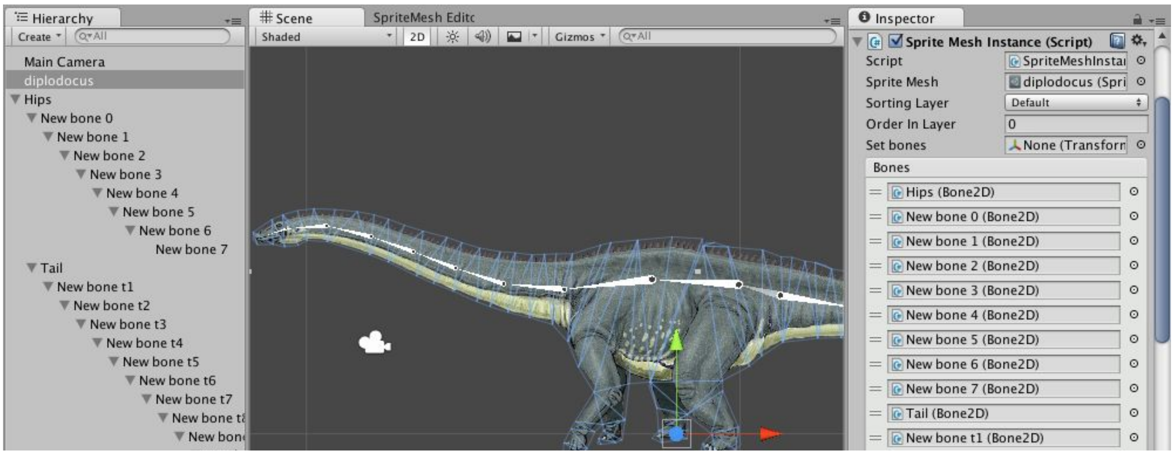
Bone 2D (Script)

Script	Bone2D
Child	None (Bone2D)
Length	1

Bone with a linked child bone.

4. Set bones to *SpriteMesh* instance

   Once you have both a *SpriteMeshInstance* and a bone hierarchy, 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. Another way to add bones is by editing the list manually and dragging the bones to the list's fields.

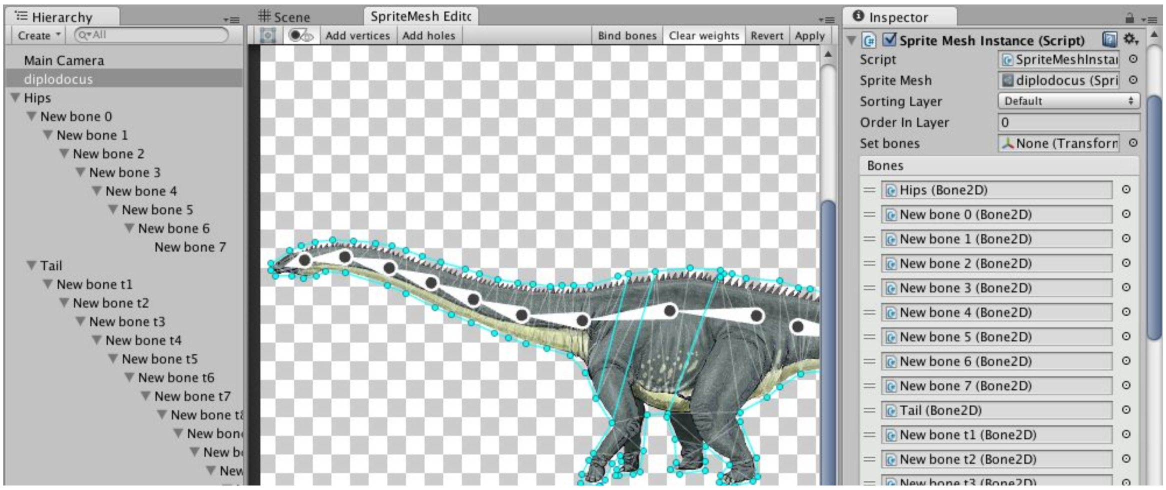


Script: SpriteMeshInstai
Sprite Mesh: diplodocus (Spri
Sorting Layer: Default
Order In Layer: 0
Set bones: None (Transforn

Bones
- Hips (Bone2D)
- New bone 0 (Bone2D)
- New bone 1 (Bone2D)
- New bone 2 (Bone2D)
- New bone 3 (Bone2D)
- New bone 4 (Bone2D)
- New bone 5 (Bone2D)
- New bone 6 (Bone2D)
- New bone 7 (Bone2D)
- Tail (Bone2D)
- New bone t1 (Bone2D)

SpriteMeshInstance after setting the bones.

Open the *SpriteMesh* Editor window again and select your *SpriteMeshInstance*. This time the bones will appear in the editor. You can toggle the bone's visualization on and off by hitting the button 👁.



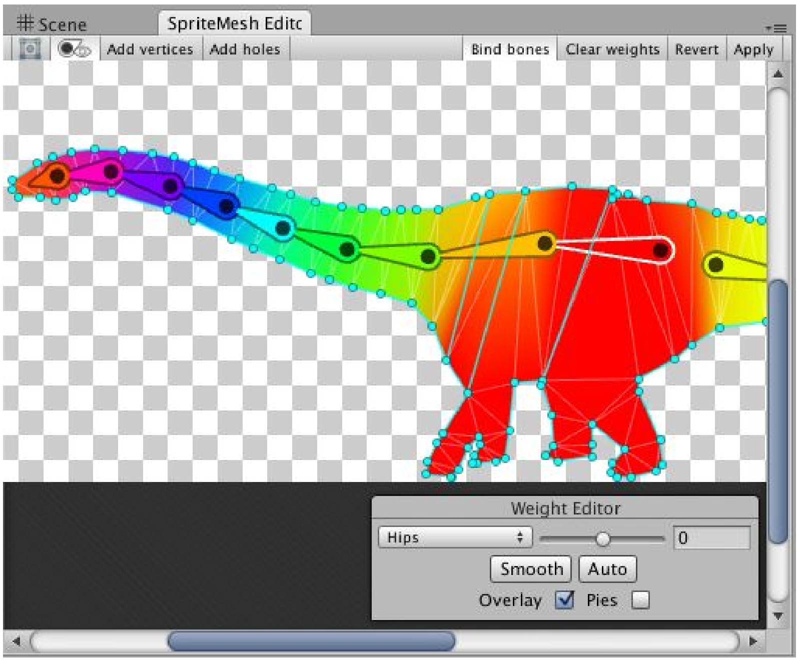
Script: SpriteMeshInstai
Sprite Mesh: diplodocus (Spri
Sorting Layer: Default
Order In Layer: 0
Set bones: None (Transforn

Bones
- Hips (Bone2D)
- New bone 0 (Bone2D)
- New bone 1 (Bone2D)
- New bone 2 (Bone2D)
- New bone 3 (Bone2D)
- New bone 4 (Bone2D)
- New bone 5 (Bone2D)
- New bone 6 (Bone2D)
- New bone 7 (Bone2D)
- Tail (Bone2D)
- New bone t1 (Bone2D)
- New bone t2 (Bone2D)
- New bone t3 (Bone2D)

SpriteMesh Editor showing the referenced bones from the current *SpriteMeshInstance*.

5. Bind bones in editor

Once the bones appear in the Editor click the **Bind bones** 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.

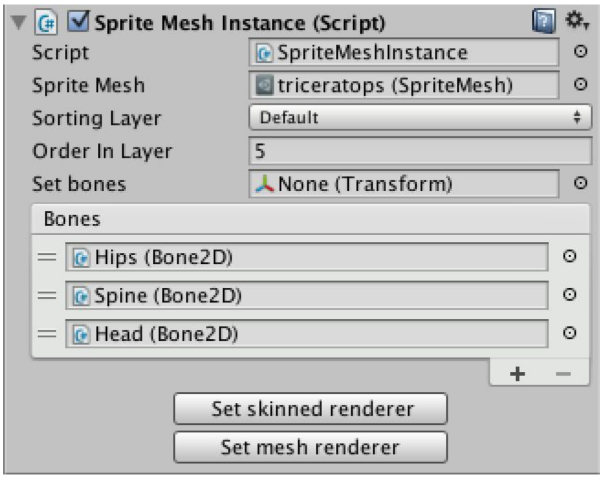
You can now adjust the weights by selecting vertices and adding/subtracting bone influence using the **Weight Editor** (see section 5.2 for more detailed information).

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 Bones process.

Click the **Apply** button to finalize the changes.

6. Set skinned renderer



☑ Sprite Mesh Instance (Script)

Script: SpriteMeshInstance

Sprite Mesh: triceratops (SpriteMesh)

Sorting Layer: Default

Order In Layer: 5

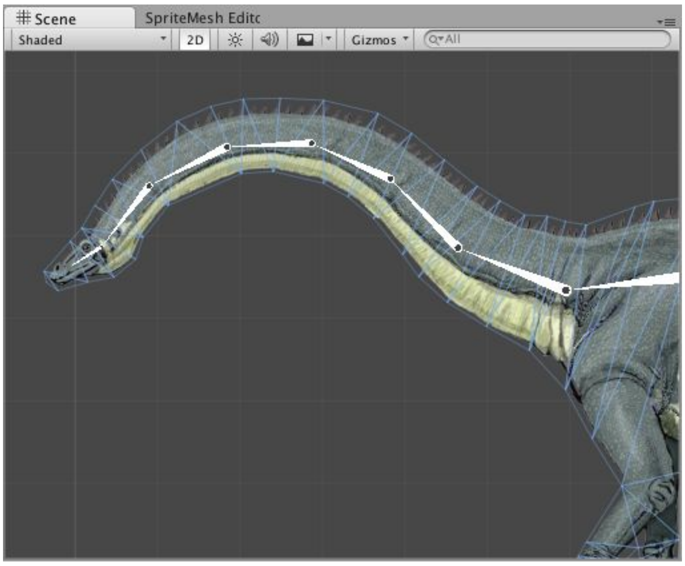
Set bones: None (Transform)

Bones

- Hips (Bone2D)
- Spine (Bone2D)
- Head (Bone2D)

SpriteMeshInstance. Use the buttons to set the appropriate *Renderer*.

Select your *SpriteMeshInstance* and hit the **Set skinned renderer** button to switch to a *SkinnedMeshRenderer* component. Now the setup is complete and your mesh will deform properly as bones rotate and move.

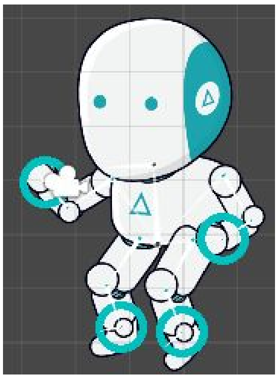


SpriteMeshInstance with a SkinnedMeshRenderer set.

4.2 … USE INVERSE KINEMATICS

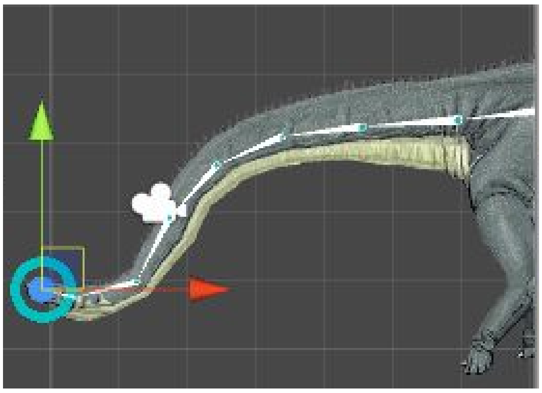
To use Inverse Kinematics select a bone in the hierarchy and select **2D Object -> IK Limb** or **2D Object -> IK CCD**. A new GameObject will be created with the corresponding IK component. Move the newly created GameObject to see IK changing bone's pose.

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.

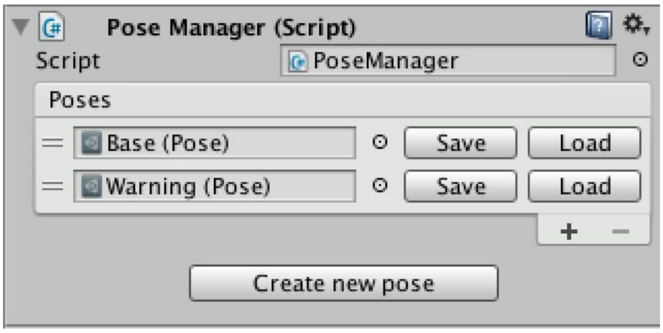


CCD Inverse Kinematics with a long bone chain.

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

4.3 … 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.



Script: PoseManager

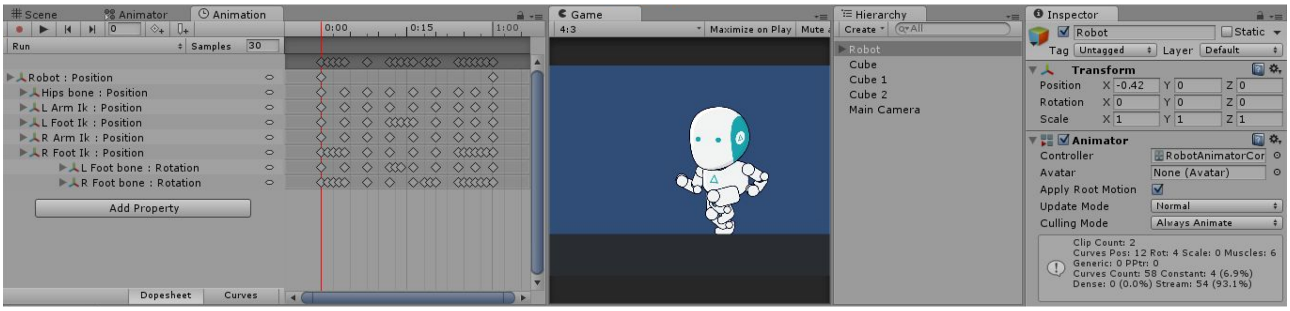
Base (Pose)

Warning (Pose)

PoseManager component.

4.4 … CREATE ANIMATIONS

Finally we can start creating animations using Unity's Animation Window. 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

Robot
Cube
Cube 1
Cube 2
Main Camera

☑ Robot □ Static
Tag: Untagged; Layer: Default
Position X -0.42 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
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.

Now you can add new Keyframes, rotate bones, move IK objects to create awesome new animations. Anima2D will record all the bone's changes produced by the Inverse Kinematics system. Loading poses will also trigger new Keyframes.

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

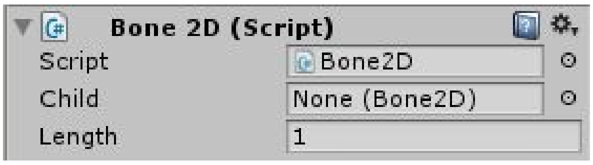
5. Reference

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

5.1. Components

In Anima2D you can find the following components: Bone2D, Sprite Mesh Instance, IK Limb 2D and IK CCD2D.

Bone2D

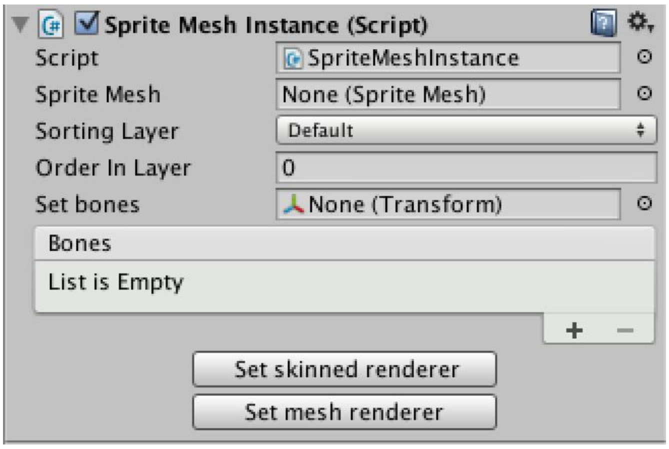


Bone 2D (Script)

Script	Bone2D
Child	None (Bone2D)
Length	1

- **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



☑ Sprite Mesh Instance (Script)

Script	SpriteMeshInstance
Sprite Mesh	None (Sprite Mesh)
Sorting Layer	Default
Order In Layer	0
Set bones	None (Transform)

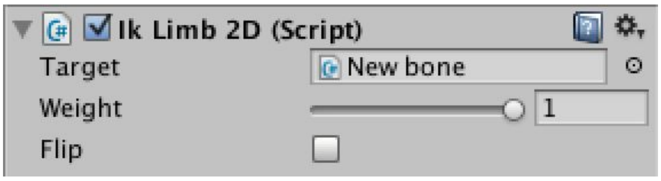
Bones

List is Empty

- **Sprite Mesh**: Reference to the *SpriteMesh* asset.
- **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.

- **Set skinned renderer**: Set a *SkinnedMeshRenderer* and set up the bones.
- **Set mesh renderer**: Set a *MeshRenderer*. Useful when the *Mesh* still has no weights.

Ik Limb 2D



☑ Ik Limb 2D (Script)

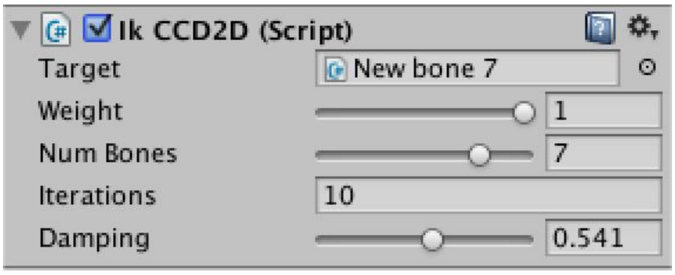
Target: New bone

Weight: 1

Flip: □

- **Target**: The target bone to apply IK to.
- **Weight**: The amount of contribution of the IK solver to the final pose.
- **Flip**: Use the other pose solution for the limb.

Ik CCD2D



☑ Ik CCD2D (Script)

Target: New bone 7

Weight: 1

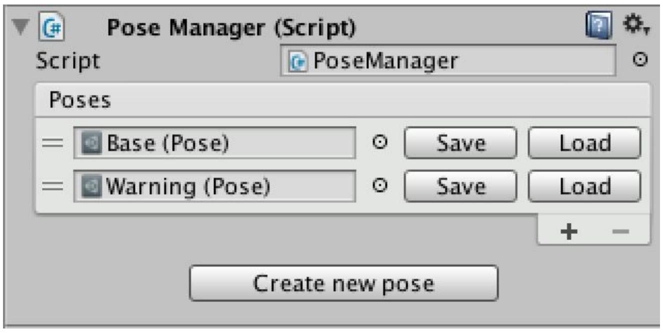
Num Bones: 7

Iterations: 10

Damping: 0.541

- **Target**: The target bone to apply IK to.
- **Weight**: The amount of contribution of the IK solver to the final pose.
- **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.

Pose Manager



Pose Manager (Script)

Script: PoseManager

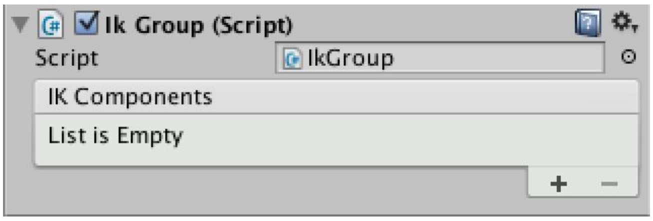
Poses

Base (Pose)

Warning (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



☑ Ik Group (Script)

Script: IkGroup

IK Components

List is Empty

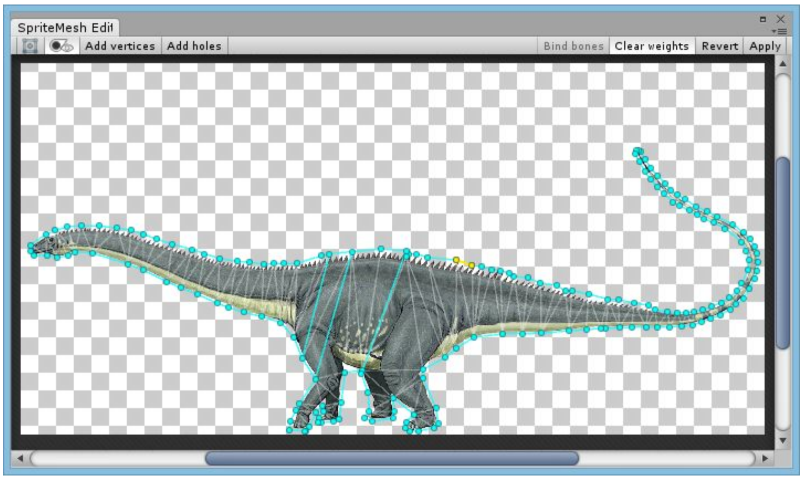
This component will update a group of IK objects. This is useful when you want to specify the execution order of IK elements in runtime.

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

5.2. Editor Window

In Anima2D you can find the following editors: *SpriteMesh Editor* and *Weight Editor*. *Weight Editor* appears inside the *SpriteMesh Editor Window* the bones are bound.

SpriteMesh Editor

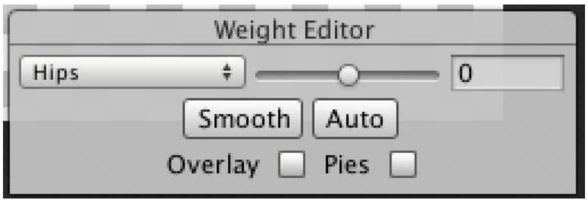


The SpriteMesh Editor.

- Resets the *SpriteMesh* using Unity's default information or a simple quad.
- Toggle bone's visualization.
- **Add vertices**: Add vertices anywhere in the mesh. Can be also activated by pressing the **V** key. Hold down the **Shift** key in order to add the vertex to the closest edge. Left-click to add the vertex. Right-click anywhere to toggle off.
- **Remove vertices**: Select one or more vertices and press the **Backspace** or **Delete** key.
- **Add edge**: Select a vertex, hold the 'Shift' key and left-click to another vertex. If no other vertex is detected a new one will be created.
- **Remove edge**: Select an edge and press the **Backspace** or **Delete** key.
- **Add holes**: Add a hole in the mesh. Can be also activated by pressing the **H** key. A hole will appear as a red spot and will remove any triangles inside a closed edge path.
- **Remove holes**: Select a hole and press the **Backspace** or **Delete** keys.
- **Selection tool**: Click on any empty space and drag to select multiple vertices. Hold **Ctrl** or **Cmd** to append more vertices to the current selection. Selected vertices can be dragged around and deleted.

Weight Editor

The *Weight Editor* will show different options depending on the selected vertices:



Weight Editor.

- **Single vertex mode:**

The *Weight Editor* 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.

- **Multiple vertex mode:**

In multiple vertex selection mode we can add and subtract influence from a bone.

The *Weight Editor* 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.

Important: A vertex that is not associated to the selected bone by any of its weights will not be affected.

- **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. 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!