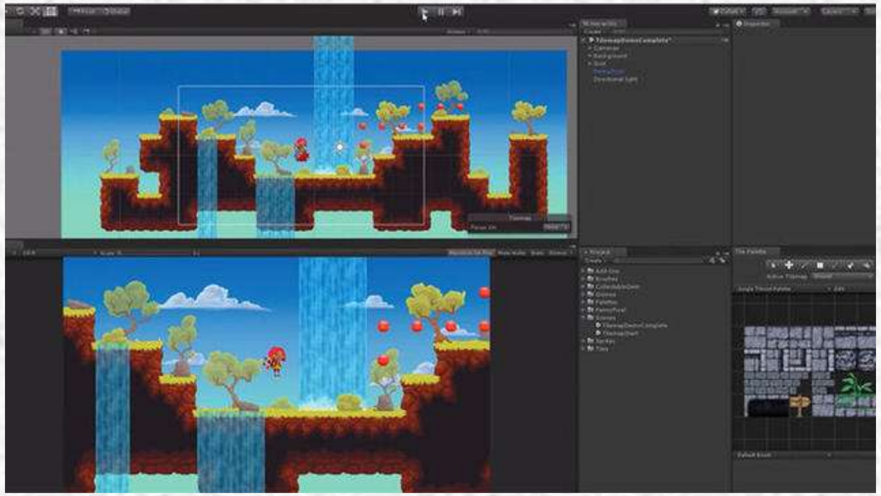
Computer Game Programming (SE3173)

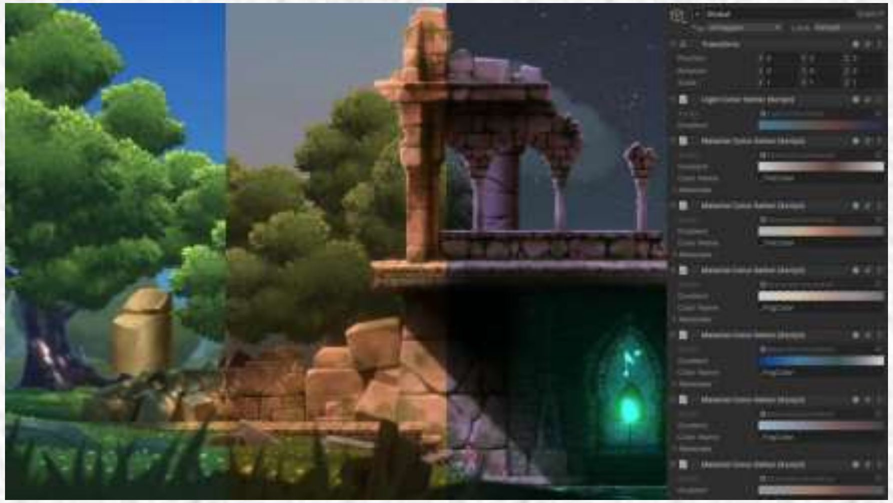
Ibrar Arshad

Ibrar.arshad@cust.edu.pk

Gameplay in 2D

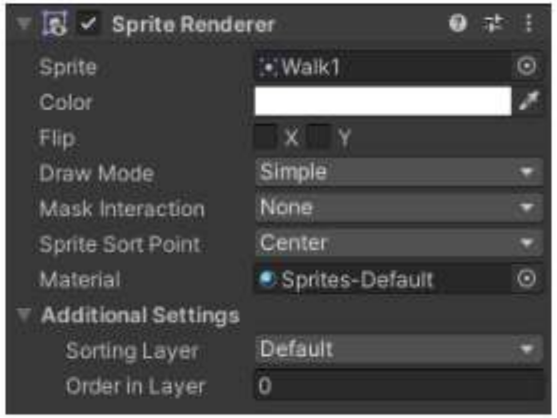
- While famous for its 3D capabilities, Unity can also be used to create 2D games
- The familiar functions of the editor are still available but with helpful additions to simplify 2D development





2D Sprites

- Sprites are 2D Graphic objects (images)
  - If the project mode is set to 2D, all imported images are automatically set as a Sprite
- The Sprite Renderer component renders the Sprite and controls how visually appears in a Scene



☑ Sprite Renderer

Sprite: Walk1

Color

Flip: □ X □ Y

Draw Mode: Simple

Mask Interaction: None

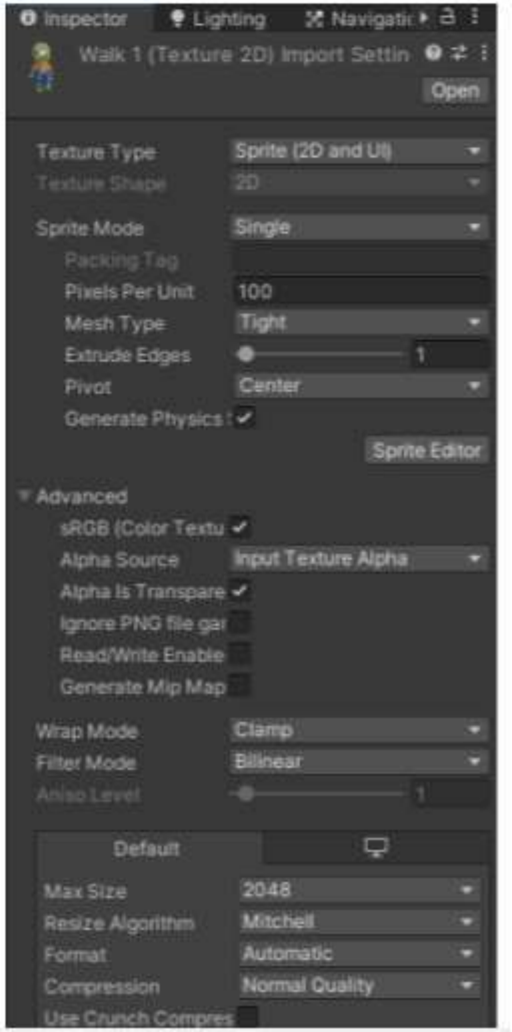
Sprite Sort Point: Center

Material: Sprites-Default

Additional Settings

Sorting Layer: Default

Order in Layer: 0



Walk 1 (Texture 2D) Import Settin

Texture Type: Sprite (2D and UI)

Texture Shape: 2D

Sprite Mode: Single

Packing Tag

Pixels Per Unit: 100

Mesh Type: Tight

Extrude Edges: 1

Pivot: Center

Generate Physics ☑

Advanced

sRGB (Color Textu ☑

Alpha Source: Input Texture Alpha

Alpha Is Transpare ☑

Ignore PNG file gar □

Read/Write Enable □

Generate Mip Map □

Wrap Mode: Clamp

Filter Mode: Bilinear

Aniso Level: 1

Max Size: 2048

Resize Algorithm: Mitchell

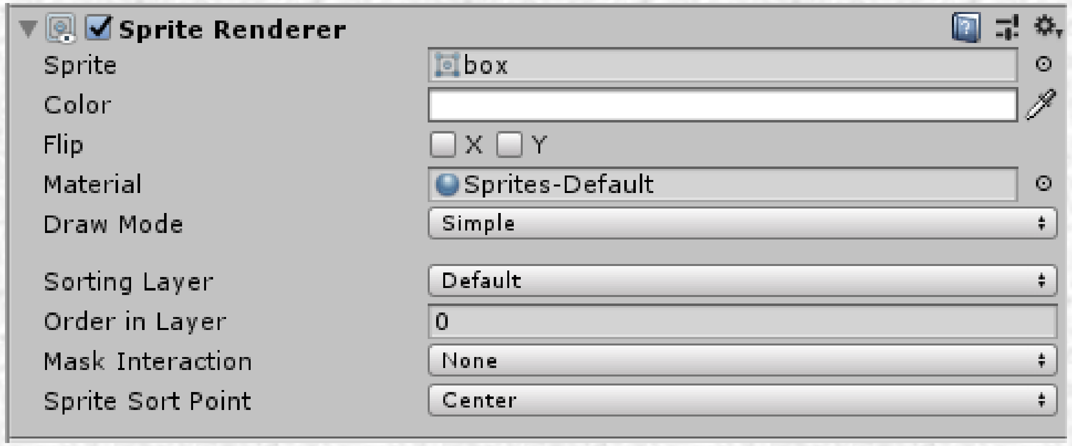
Format: Automatic

Compression: Normal Quality

Use Crunch Compres □

2D Sprites

- The **Sprite Renderer** component renders the **Sprite** and controls how it visually appears in a **Scene** for both 2D and 3D projects
- When you create a Sprite (**GameObject > 2D Object > Sprite**), Unity automatically creates a GameObject with the **Sprite Renderer** component attached
- You can also add the component to an existing GameObject via the **Components** menu (**Component > Rendering > Sprite Renderer**).



☑ **Sprite Renderer**

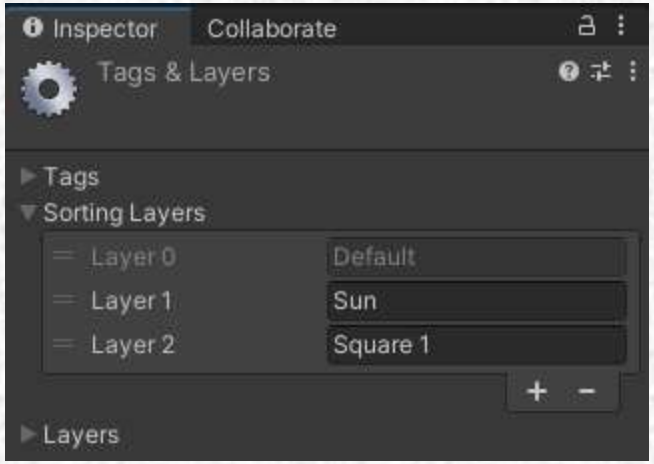
Sprite	box
Color	
Flip	□ X □ Y
Material	Sprites-Default
Draw Mode	Simple
Sorting Layer	Default
Order in Layer	0
Mask Interaction	None
Sprite Sort Point	Center

Sprite Renderer Properties

Property	Function
Sprite	Define which Sprite texture the component should render. Click the small dot to the right to open the object picker window, and select from the list of available Sprite Assets.
Color	Define the vertex color of the Sprite, which tints or recolors the Sprite's image. Use the color picker to set the vertex color of the rendered Sprite texture. See the Color section below this table for examples.
Flip	Flips the Sprite texture along the checked axis. This does not flip the Transform position of the **GameObject**
Material	Define the Material used to render the Sprite texture.
Draw Mode	Define how the Sprite scales when its dimensions change. Select one of the following options from the drop-down box.
Sorting Layer	Set the Sorting Layer of the Sprite, which controls its priority during rendering. Select an existing Sorting Layer from the drop-down box, or create a new Sorting Layer.
Order In Layer	Set the render priority of the Sprite within its Sorting Layer. Lower numbered Sprites are rendered first, with higher numbered Sprites overlapping those below.

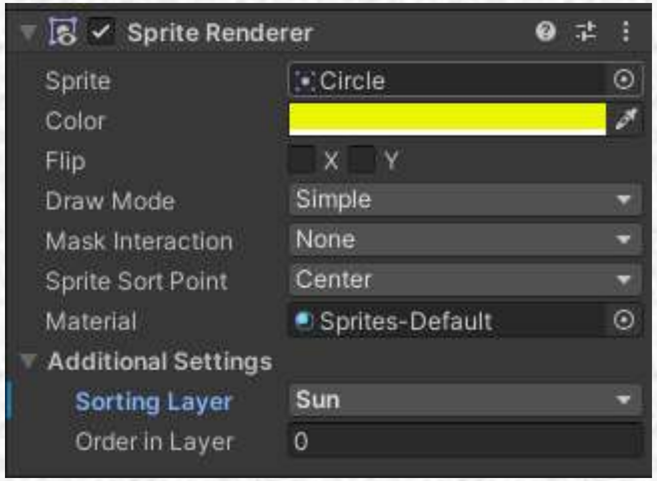
2D Sorting Layers

- The Sorting Layer and Order in Layer are available to all 2D Renderers through the Inspector window
- Set the Renderer to an existing Sorting Layer or create a new one to determine its priority in the rendering queue
- Change the value of the Order in Layer to set the Renderer's priority among other Renderers within the same Sorting Layer



Sorting Layers

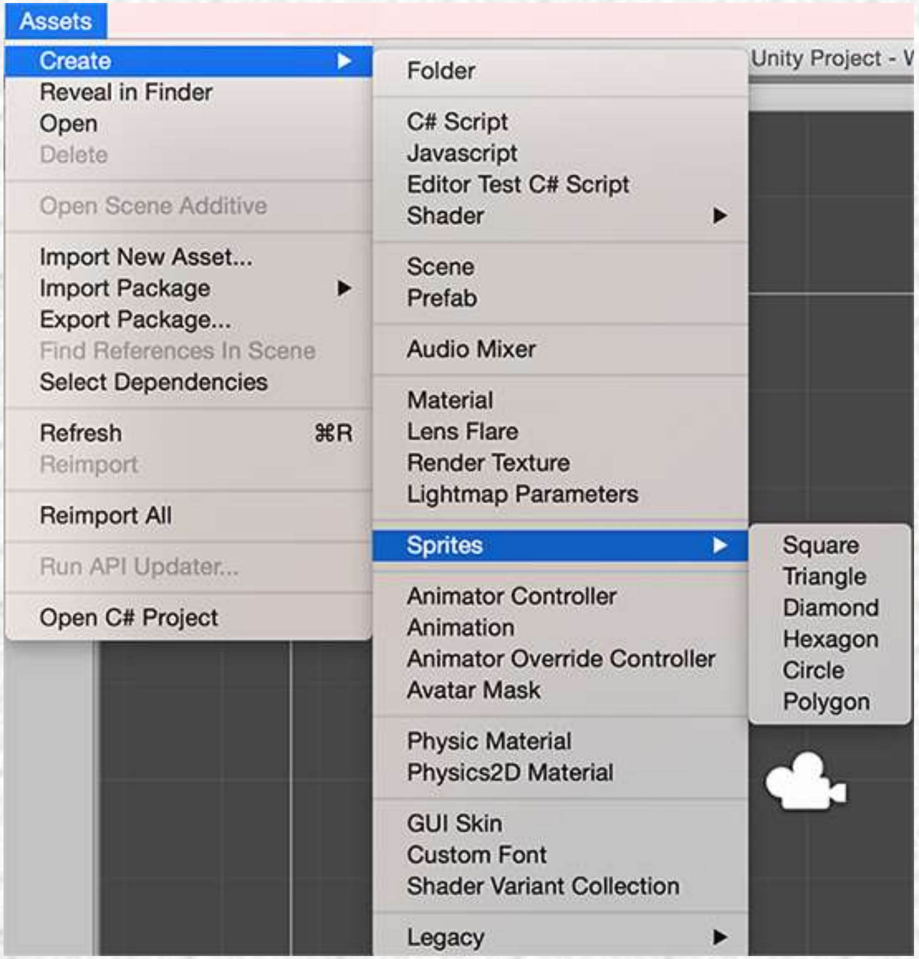
Layer 0	Default
Layer 1	Sun
Layer 2	Square 1



Sprite	Circle
Color	
Flip	□ X □ Y
Draw Mode	Simple
Mask Interaction	None
Sprite Sort Point	Center
Material	Sprites-Default
Sorting Layer	Sun
Order in Layer	0

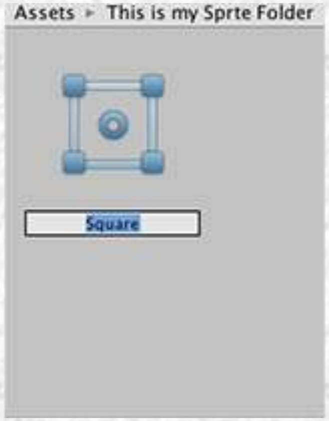
Sprite Creator

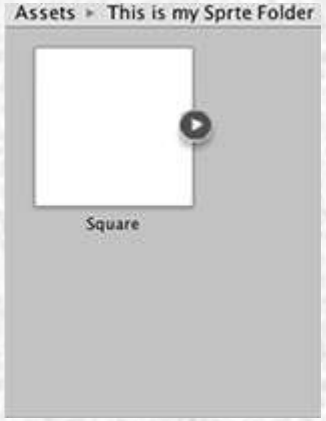
- With this tool you can create temporary placeholder **sprite** (2D) graphics. You can use these in your project during development and then replace them with the graphics you want to use.
- Select **Assets>Create>Sprites** and then select the placeholder sprite you want to make (square, triangle, diamond, hexagon, or polygon).



Using the Sprite

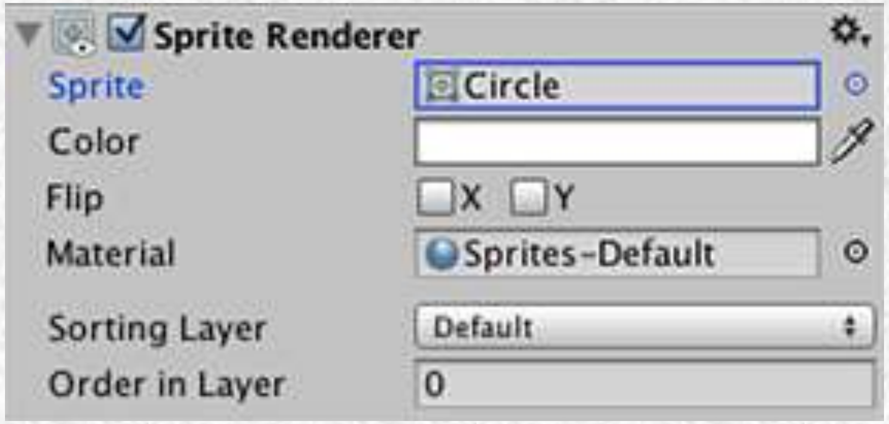
- Your new placeholder sprite appears as a white shape in the asset folder you currently have open.
- The new sprite’s name defaults to its shape name but you have the option to rename your sprite when it is first created
- If you are not sure what you want to call it, leave it as the default; you can change it later by clicking on it





Replacing your Placeholder Sprite

- To change your placeholder sprite, click on it in the **Scene View** and then edit via the **Sprite Renderer Component** in the **Inspector**
- A Unity window that displays information about the currently selected GameObject, asset or project settings, allowing you to inspect and edit the values.
- Edit the **Sprite** field: You can click on the small circle to the right of the input field to bring up the **Sprite Selector** where you can browse and select from a menu of available 2D graphic assets.



☑ Sprite Renderer

Sprite: Circle ○

Color:

Flip: □ X □ Y

Material: Sprites-Default ○

Sorting Layer: Default

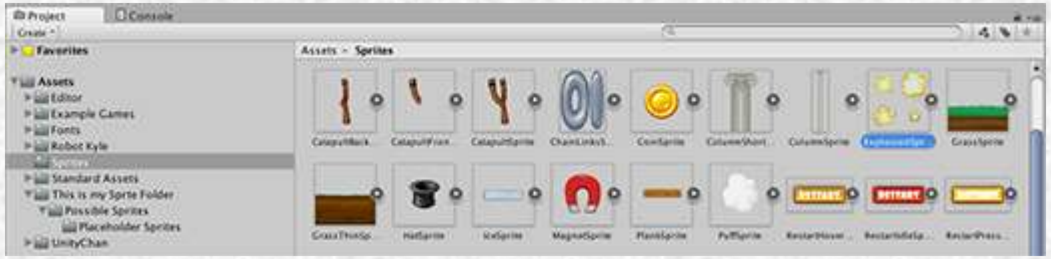
Order in Layer: 0

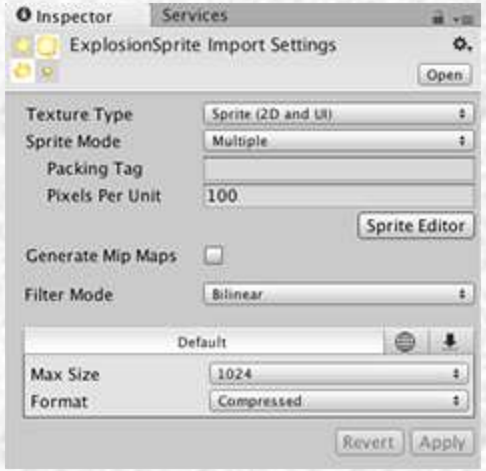
Sprite Editor

- Sometimes a Sprite Texture contains just a single graphic element but it is often more convenient to combine several related graphics together into a single image
  - For example, the image could contain component parts of a single character, as with a car whose wheels move independently of the body.
- Unity makes it easy to extract elements from a composite image by providing a Sprite Editor for the purpose

Opening the Sprite Editor

- To open the Sprite Editor:
  - Select the 2D image you want to edit from the Project View
  - Click on the **Sprite Editor** button in the Inspector
- You can only see the Sprite Editor button if the Texture Type on the image you have selected is set to Sprite (2D and UI).



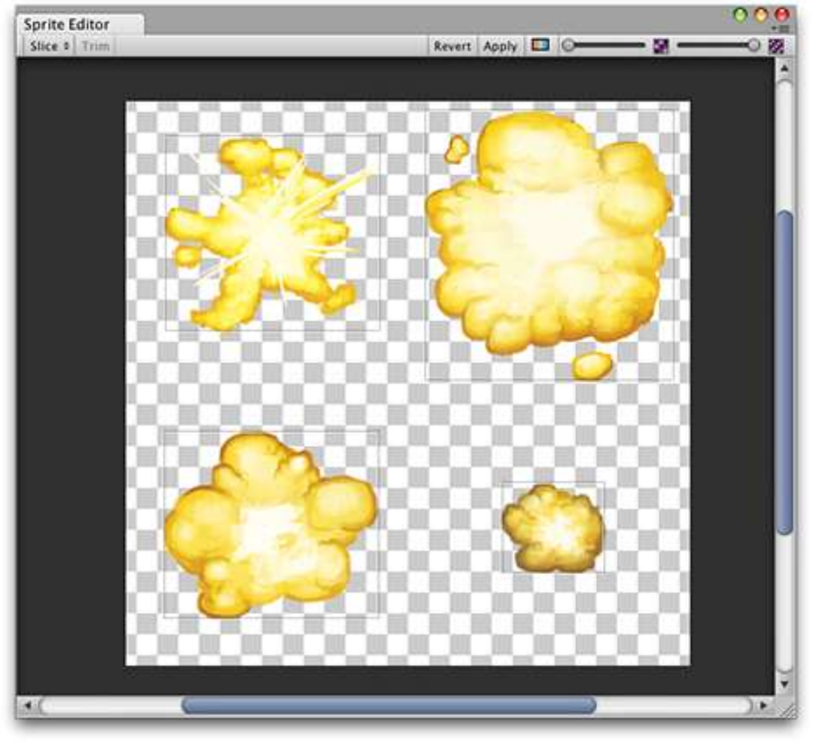


ExplosionSprite Import Settings

Texture Type: Sprite (2D and UI)
Sprite Mode: Multiple
Packing Tag:
Pixels Per Unit: 100
Generate Mip Maps: □
Filter Mode: Bilinear
Max Size: 1024
Format: Compressed

Opening the Sprite Editor

- Set the Sprite Mode to Multiple in the Texture Import Inspector if your image has several elements.

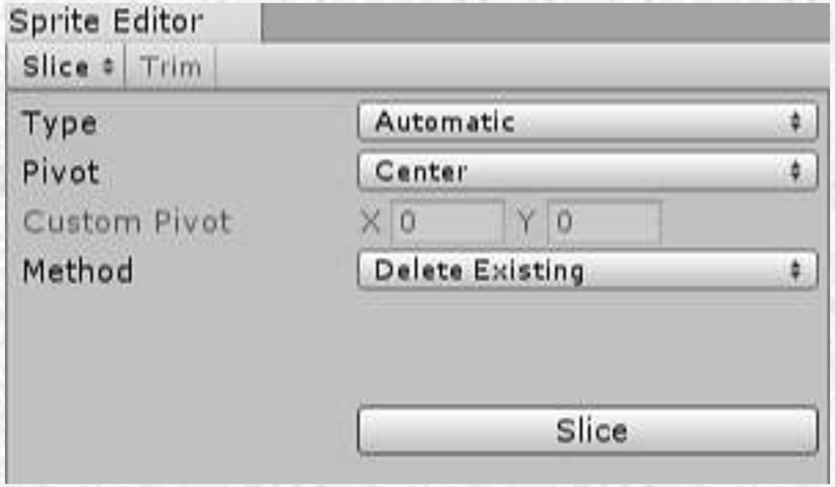


Opening the Sprite Editor

- The most important control in the sprite editor is the Slice menu at the top left, which gives you options for separating the elements of the image automatically.
- Apply and Revert buttons allow you to keep or discard any changes you have made.

Automatic Slicing

- Isolating the Sprite rectangles manually works well but in many cases, Unity can save you work by detecting the graphic elements and extracting them for you automatically
- If you click on the **Slice** menu in the control bar, you will see this panel



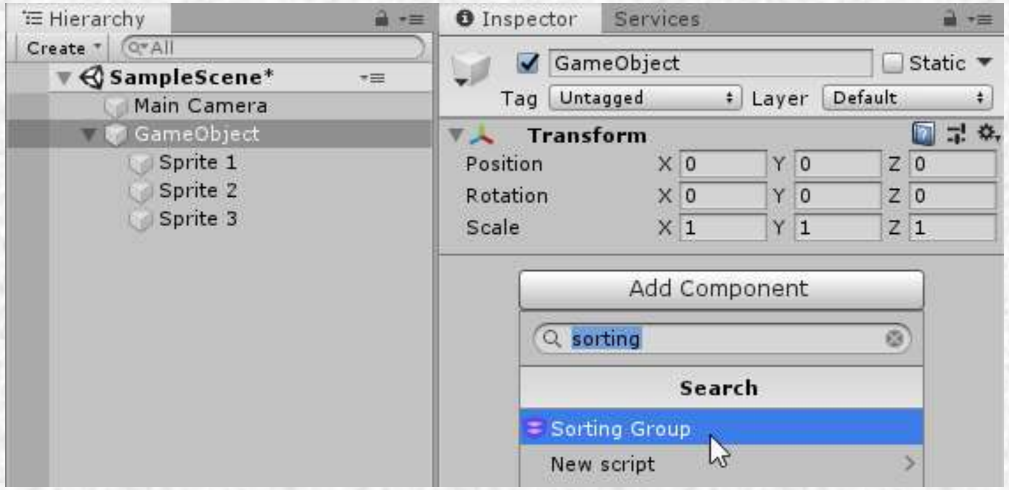
Type: Automatic
Pivot: Center
Custom Pivot: X 0 Y 0
Method: Delete Existing

Automatic Slicing

- With the slicing type set to **Automatic**, the editor will attempt to guess the boundaries of Sprite elements by transparency. You can set a default pivot for each identified Sprite
- The **Method** menu lets you choose how to deal with existing selections in the window.
- The **Delete existing** option will simply replace whatever is already selected, **Smart** will attempt to create new rectangles while retaining or adjusting existing ones, and **Safe** will add new rectangles without changing anything already in place.

Sorting Groups

- Sorting Groups allow you to group GameObjects with Sprite Renderers together, and control the order in which they render their Sprites
- Unity renders Sprite Renderers in the same Sorting Group together, as if they are a single GameObject.
- To place a GameObject into a Sorting Group, add the Sorting Group component to it. To do this, select the GameObject and go to **Component > Rendering > Sorting Group**



☑ GameObject □ Static

Tag Untagged Layer Default

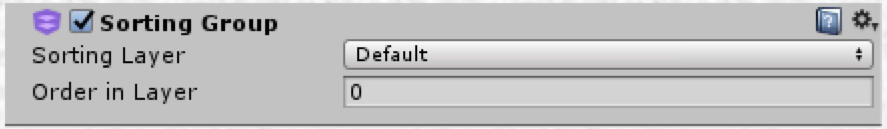
Transform

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

sorting

Sorting Group properties

- Unity uses a **Sorting Group's Sorting Layer and** Order in Layer values to determine its priority in the rendering queue among other Sorting Groups and GameObjects in the Scene



☑ Sorting Group

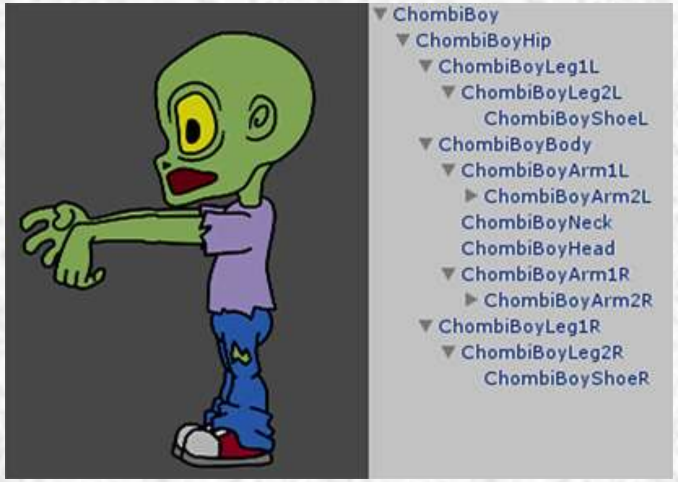
Sorting Layer: Default

Order in Layer: 0

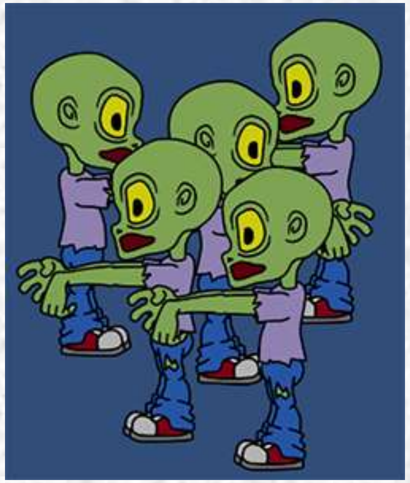
Property	Function
Sorting Layer	Select or add a Sorting Layer from this drop-down menu to determine the Sorting Group's position in the render queue.
Order in Layer	Set the render order of this Sorting Group within its **Sorting Layer**. Unity queues Renderers with lower values first in the render queue, so they appear before Renderers with higher values.

Using Sorting Groups

- The most common way to create a 2D multi-Sprite character is to arrange and parent multiple Sprite Renderers together in the Hierarchy window to form a character
- You can use **Sorting Groups** to help manage this kind of complex multi-Sprite character.

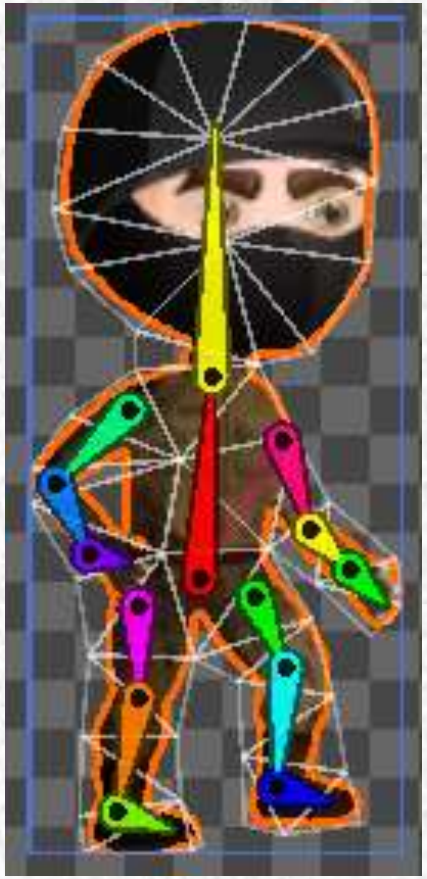


```
▼ ChombiBoy
  ▼ ChombiBoyHip
    ▼ ChombiBoyLeg1L
      ▼ ChombiBoyLeg2L
          ChombiBoyShoeL
    ▼ ChombiBoyBody
      ▼ ChombiBoyArm1L
        ▶ ChombiBoyArm2L
        ChombiBoyNeck
        ChombiBoyHead
      ▼ ChombiBoyArm1R
        ▶ ChombiBoyArm2R
    ▼ ChombiBoyLeg1R
      ▼ ChombiBoyLeg2R
          ChombiBoyShoeR
```



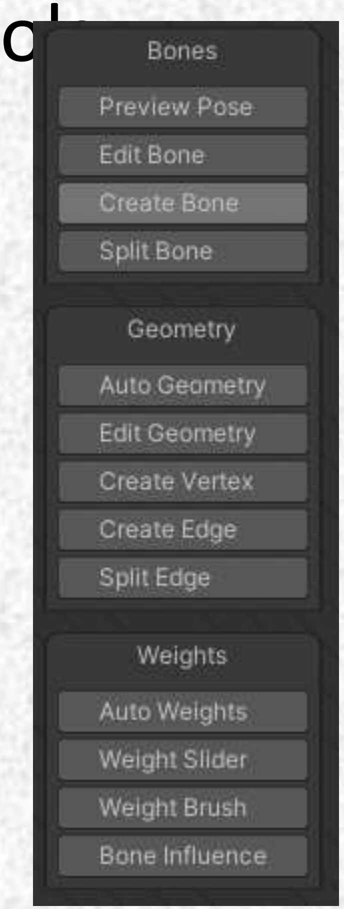
Rig a 2D Character using Skinning Editor

- In order to start rigging a 2D character Sprite Asset, your character will already be prepared and saved into the Assets folder of your project
- The Skinning Editor is used to create a skeleton in the character rigging process.
- The skeleton, or ‘bones’, (also known as a rig,) will be bound to the Sprite itself
- The Sprite’s graphics act as a sort of ‘skin,’ giving the Skinning Editor its name
- The Skinning Editor is a subset of the Sprite Editor

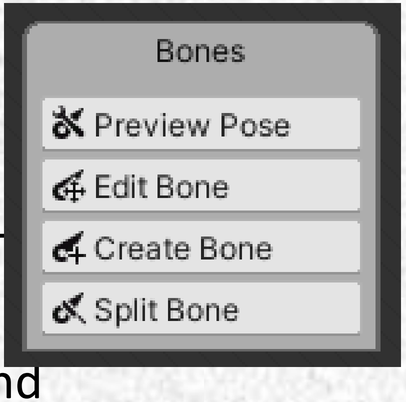


Understanding the Skinning Editor

- The Skinning Editor consists of three groups of contro
  - Bones
  - Geometry
  - Weights

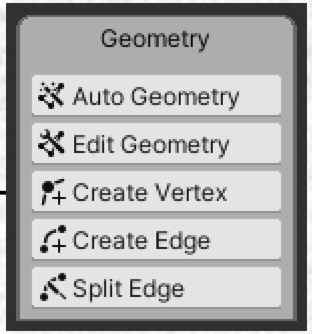


Bones



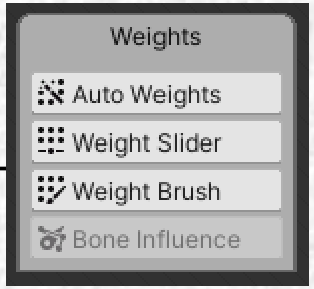
- Preview Pose
  - This allows you to move and rotate bones to see how the rigging affects posing and animation. You can connect, disconnect, and reposition bones, but these changes are not permanent and are only active during the preview.
- Edit Joints
  - Move, reposition, disconnect, and reconnect bones.
  - To move a bone without disconnecting its joints, click-and-drag either the root or the tip of the bone
  - To disconnect a joint, click-and-drag the bone until it separates
- Create Bone
  - This allows you to create a bone.
  - If you are switching to this tool from another, the bone will by default be a child of the root of the previously selected bone chain, starting a new branch in the Hierarchy
- Split Bone
  - This splits a single bone into two connected bones.

Geometry



- Auto Geometry
  - This automatically creates a vertex Mesh for the Sprite, assigns the existing bones to vertices, and generates bone weighting information.
  - Bone weights are set per vertex, and determine how much influence a bone has over a vertex.
- Edit Geometry
  - This allows you to move and delete edges and vertices.
  - If you delete a border edge, causing an open Mesh, Unity only retains the vertices and other border edges.
  - Interior edge information is lost until the surface is closed again or until Auto Geometry is used.
- Create Vertex
  - This allows you to create a vertex, automatically creating edges to connect it to neighboring vertices.
- Create Edge
  - This allows you to create an edge between two existing vertices, or one existing vertex and a new one placed where you click.
- Split Edge
  - This splits the edge nearest your mouse by creating a new vertex where you click, connecting it with both endpoints of the original edge.

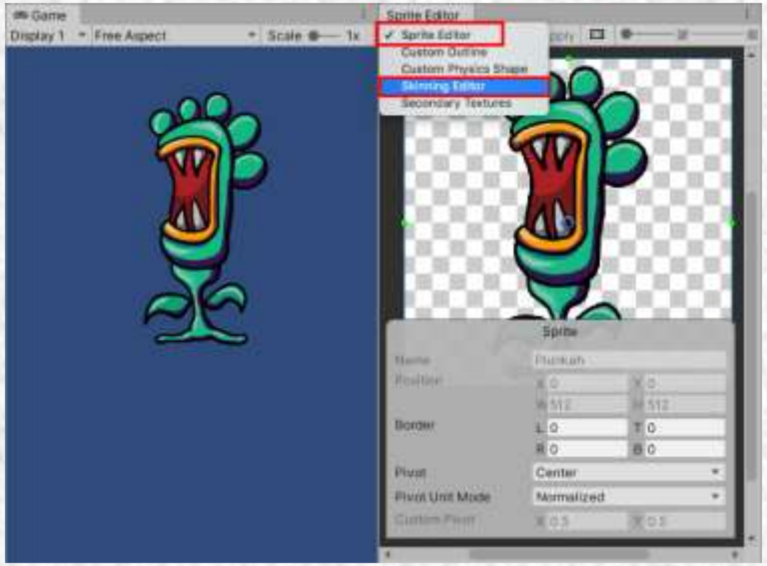
Weights



- Auto Weights
  - This will open a panel that can automatically generate, normalize, or clear bone weights
- Weight Slider
  - Selected vertices can be adjusted via a set of sliders. Select vertices one at a time by clicking on them.
- Weight Brush
  - This allows you to select a bone and paint its area of influence using a traditional digital paintbrush

Using the Skinning Editor

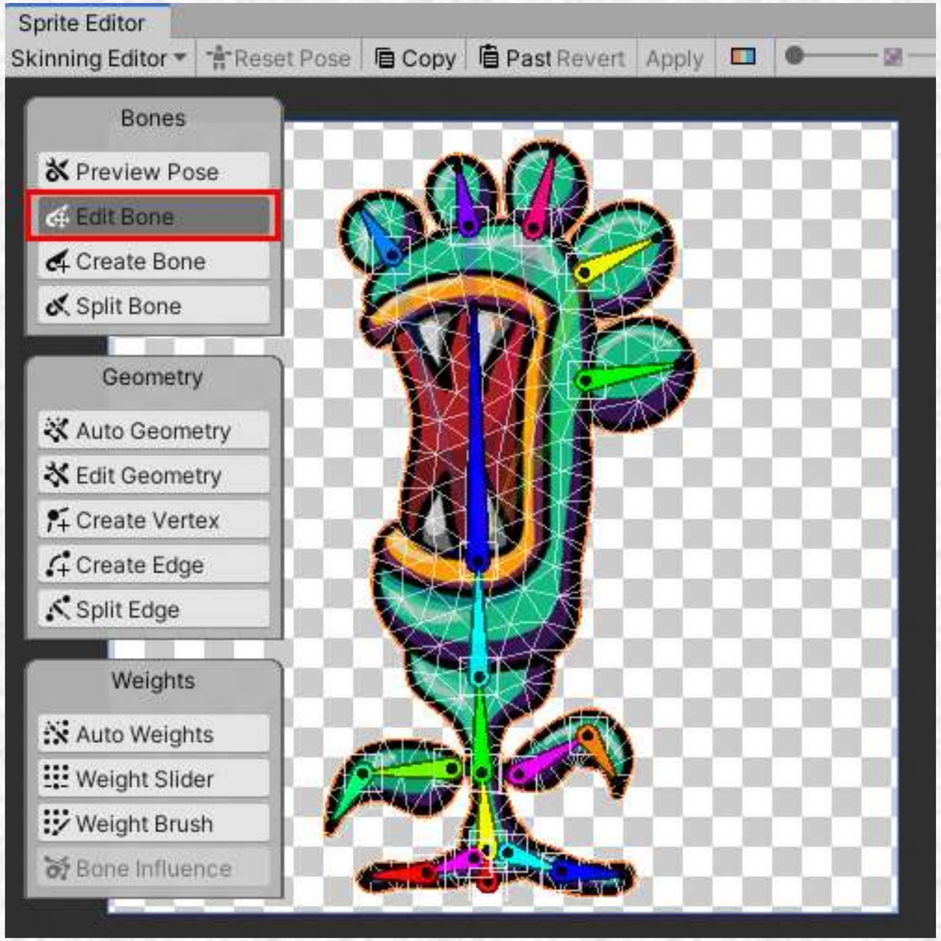
- To start rigging a 2D Sprite, select the Sprite added to your project in the Project Window, and from the dropdown menu select
  - Window > 2D > Sprite Editor
  - Select the dropdown triangle to enter the Skinning Editor



Name Plunkah
Position X 0 Y 0 W 512 H 512
Border L 0 T 0 R 0 B 0
Pivot Center
Pivot Unit Mode Normalized
Custom Pivot X 0.5 Y 0.5

Creating Bone Skeleton Cont.

- Select the Edit Bone Button, and double-click on the Character Sprite to make the bones visible

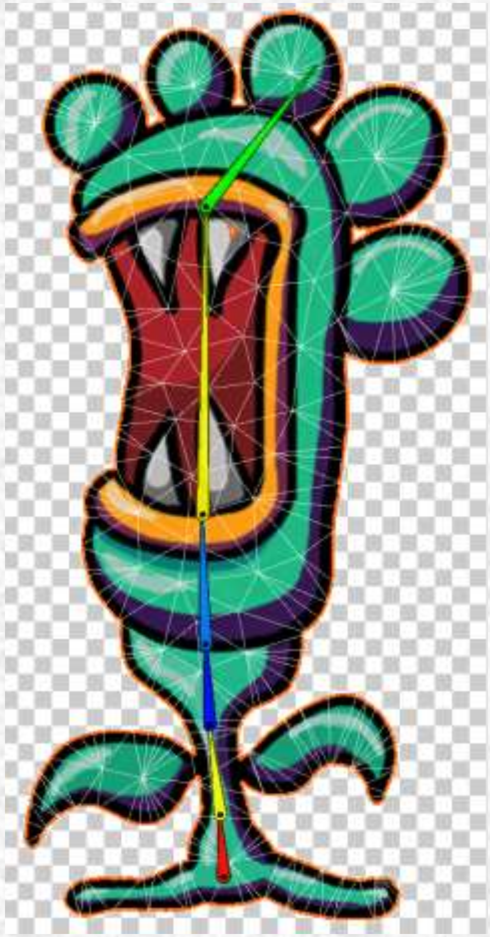


Creating Bone Skeleton Cont.

- Select the Create Bone button
- Click and drag to create new bones, clicking along the character’s skin, pressing the ESC key when finished
- If clicking doesn’t place the root of a bone, double-click your Sprite to enable editing mode

Creating Bone Skeleton Cont.

- Place the root of the first bone in the bottom center of your character’s body
- This bone will serve as the Sprite’s handle
- Continue up the central column, with the last bone roughly in the center leaf on the character’s head

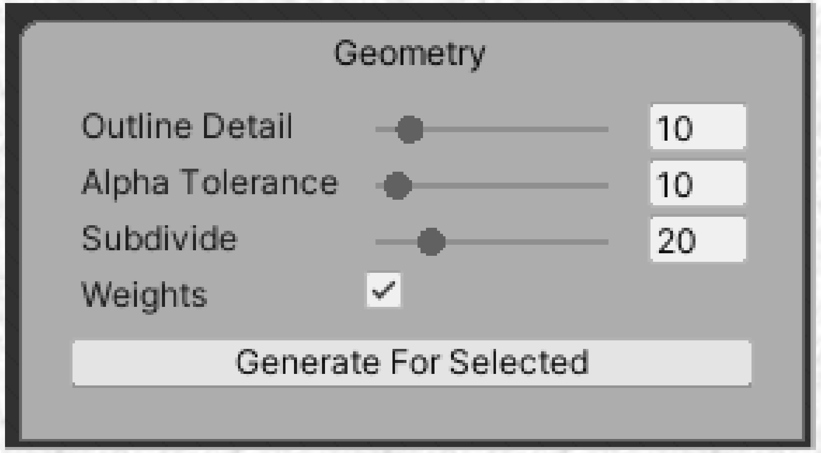


Creating Bone Skeleton Cont.

- Right-click or press the ESC key to break the chain, and click in another section of your character’s pose, such as an arm, leg, leaf or tendril, and beginning near the central column, add a couple of more bones to add to the skeleton.

Creating the Bone Geometry

- Select the Auto Geometry button
- In the popup Geometry panel, select Generate for Selected
- Select Apply at the top of the Sprite / Skinning Editor window to apply these changes



Geometry

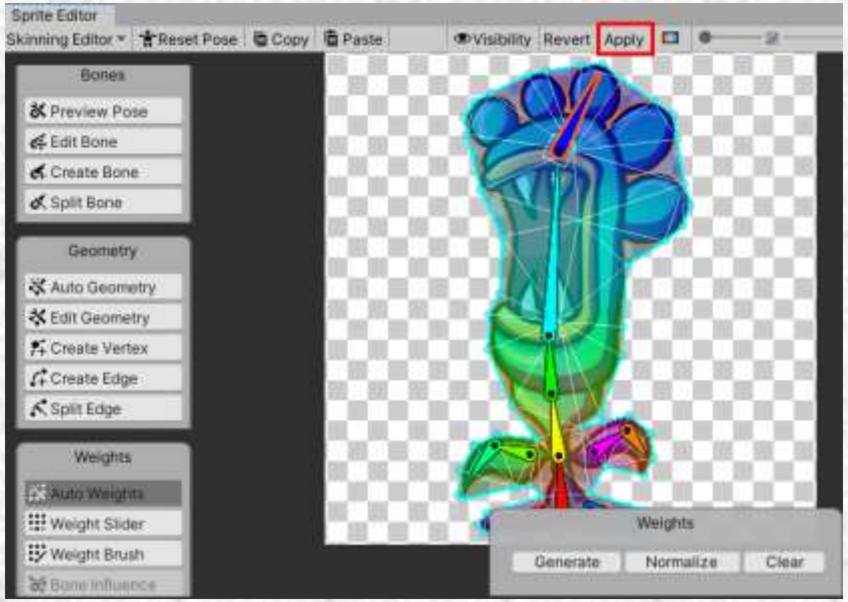
Outline Detail 10

Alpha Tolerance 10

Subdivide 20

Weights ☑

Generate For Selected

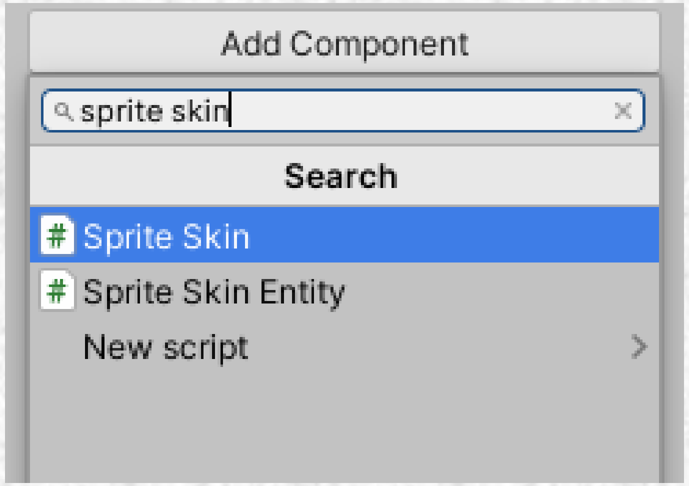


Using Weights

- Try using the Weight Slider controls to adjust vertex weights.
- Remember that the more a vertex is influenced by a bone, the more it will take on that bone’s color in the weight map.
- Any changes made since the last enabling of Apply can be undone by selecting Revert

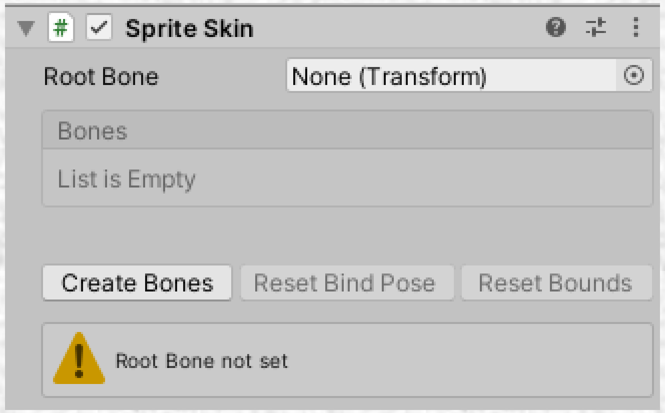
Using Rig in the Game Scene

- Drag your rigged Sprite into the Hierarchy view to create a GameObject.
- Add a Sprite Skin component to your sprite, by first selecting your Sprite in the Hierarchy Window, and then selecting the Add Component Button, and then searching for Sprite Skin



Using Rig in the Game Scene Cont.

- The Sprite Skin component will detect the joints as configured in the Skinning Editor, but there are no corresponding bones

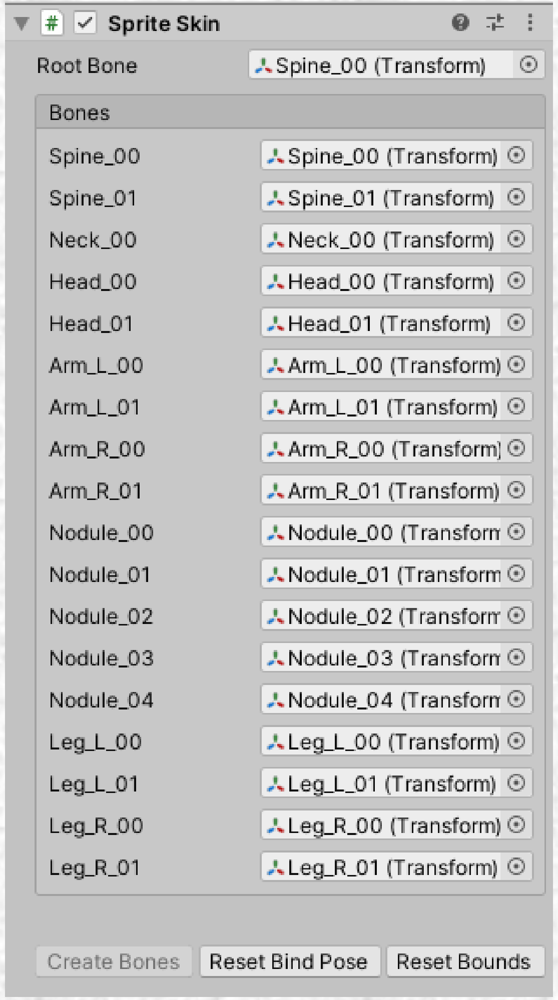


Root Bone: None (Transform)

Bones

List is Empty

Root Bone not set

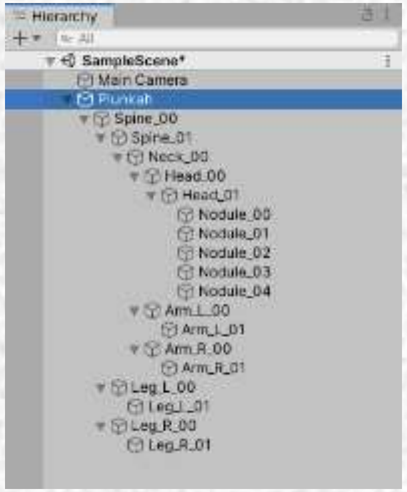


Root Bone: Spine_00 (Transform)

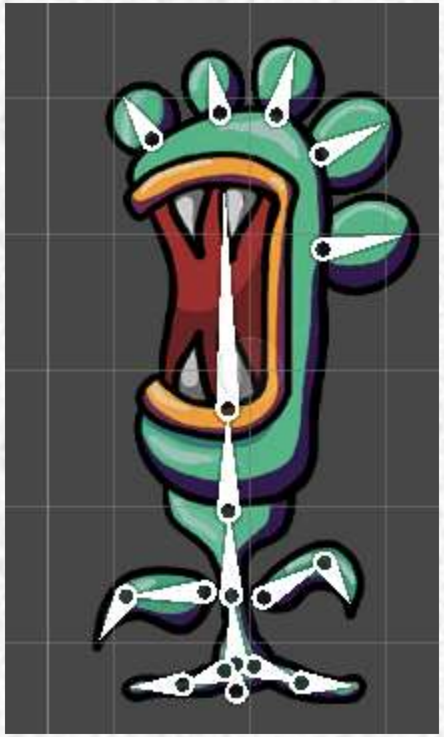
Bones	
Spine_00	Spine_00 (Transform)
Spine_01	Spine_01 (Transform)
Neck_00	Neck_00 (Transform)
Head_00	Head_00 (Transform)
Head_01	Head_01 (Transform)
Arm_L_00	Arm_L_00 (Transform)
Arm_L_01	Arm_L_01 (Transform)
Arm_R_00	Arm_R_00 (Transform)
Arm_R_01	Arm_R_01 (Transform)
Nodule_00	Nodule_00 (Transform
Nodule_01	Nodule_01 (Transform
Nodule_02	Nodule_02 (Transform
Nodule_03	Nodule_03 (Transform
Nodule_04	Nodule_04 (Transform
Leg_L_00	Leg_L_00 (Transform)
Leg_L_01	Leg_L_01 (Transform)
Leg_R_00	Leg_R_00 (Transform)
Leg_R_01	Leg_R_01 (Transform)

Using Rig in the Game Scene Cont.

- Select the Create Bones button to populate the Root and Bones of the Sprite Skin



```
SampleScene*
  Main Camera
  Plunkah
    Spine_00
      Spine_01
        Neck_00
          Head_00
            Head_01
              Nodule_00
              Nodule_01
              Nodule_02
              Nodule_03
              Nodule_04
          Arm_L_00
            Arm_L_01
          Arm_R_00
            Arm_R_01
      Leg_L_00
        Leg_L_01
      Leg_R_00
        Leg_R_01
```



References

- https://learn.unity.com/tutorial/rigging-a-sprite-with-the-2d-animation-package?uv=2019.4#
- https://docs.unity3d.com/Packages/com.unity.2d.animation@9.0/manual/PreparingArtwork.html
- https://www.youtube.com/watch?v=EZtpACxCTEE&t=73s