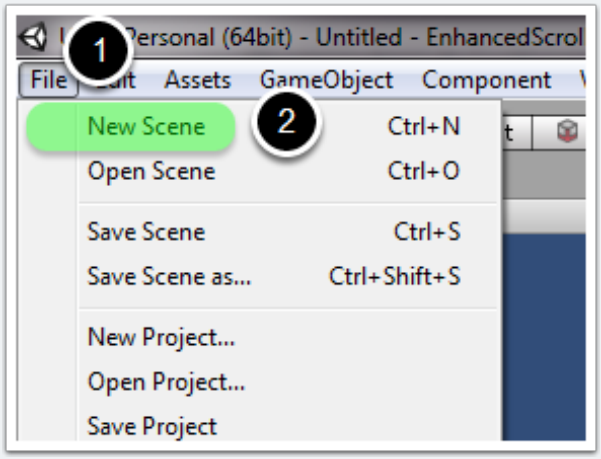
01 - Quick Start Tutorial

Create a new scene

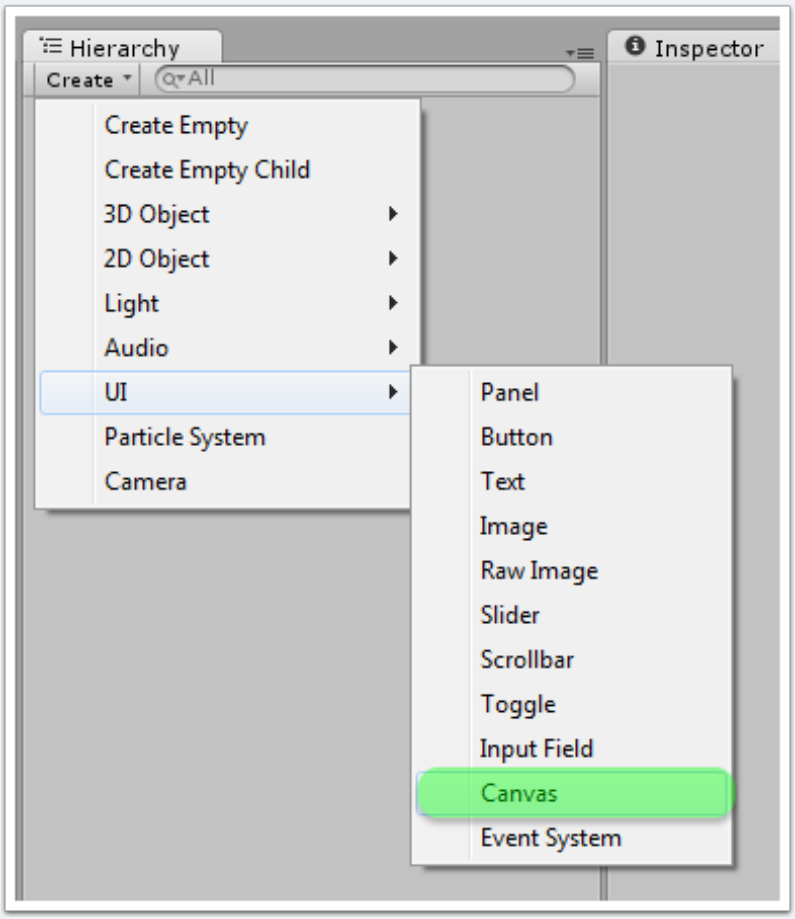
1. Click on **File** in the Unity menu
2. Click **New Scene**



Create a canvas

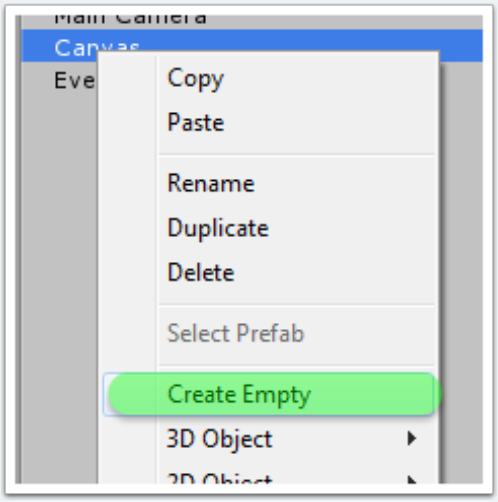
In the editor hierarchy:

1. Click the **Create** button
2. Click **UI** > **Canvas**



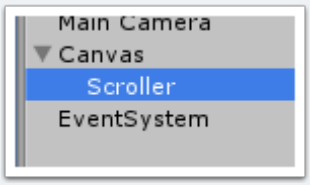
Create an empty GameObject for the scroller

Right-click on the Canvas GameObject and select **Create Empty**



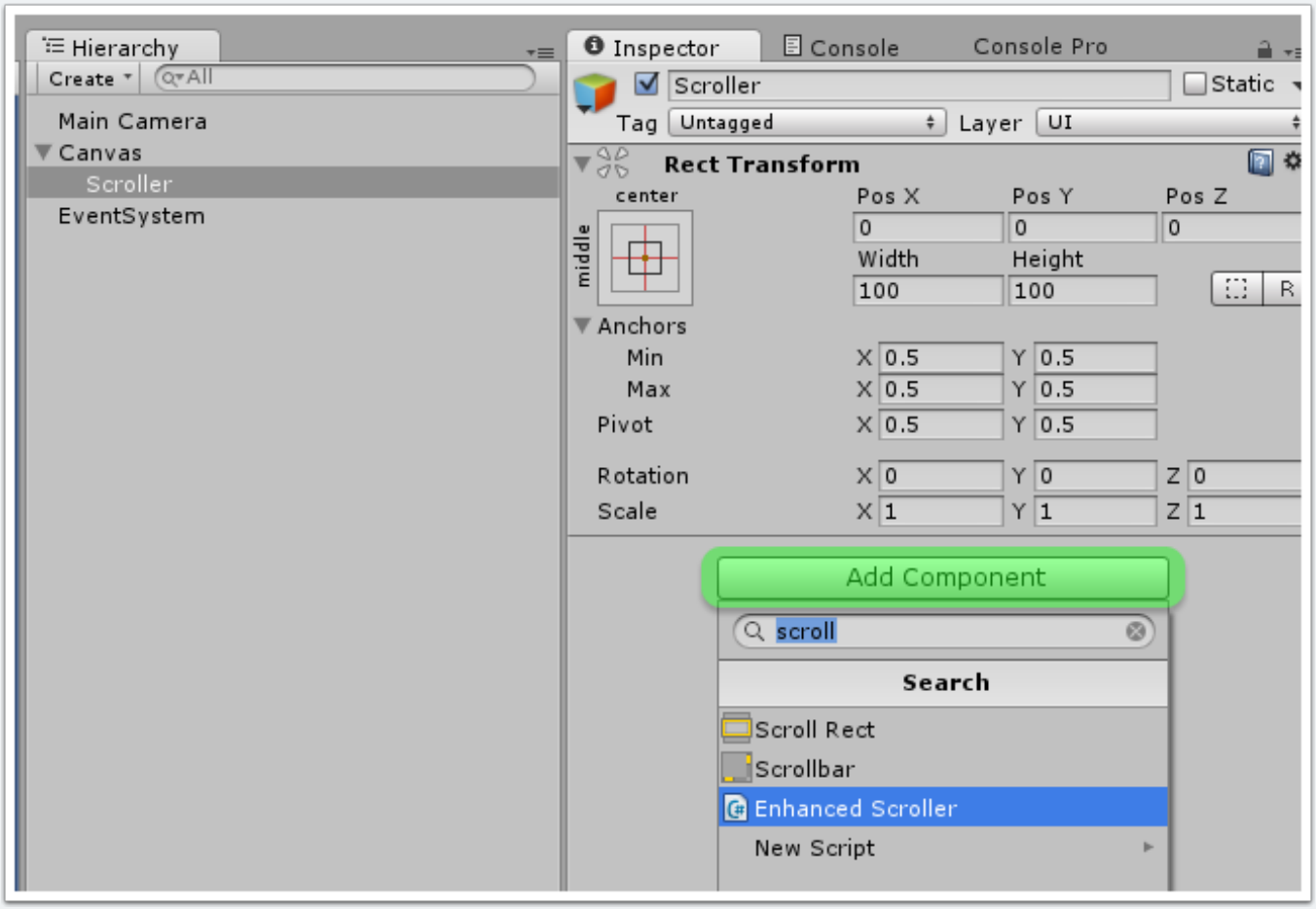
Rename the GameObject

Call your scroller GameObject **Scroller** for this tutorial



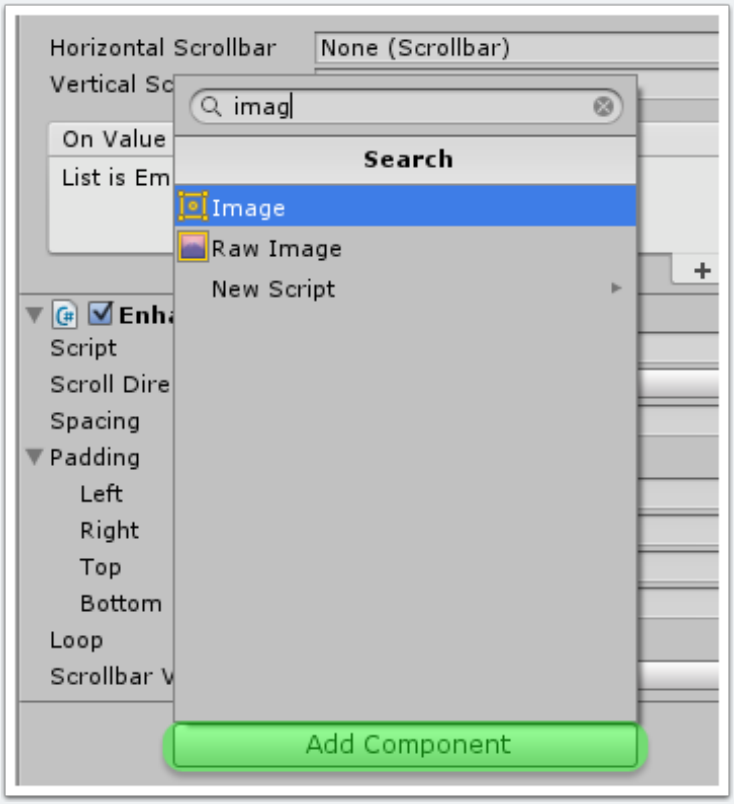
Add the EnhancedScoller component to the scroller GameObject

While the **Scroller** GameObject is selected, click on the **Add Component** button in the inspector. Search for **Enhanced Scroller** and select it to add. Alternatively, you can drag the **EnhancedScroller** script from the EnhancedScroller **Plugins** folder directly to the inspector if you prefer. When you add the EnhancedScroller component, a **Scroll Rect** is automatically added.



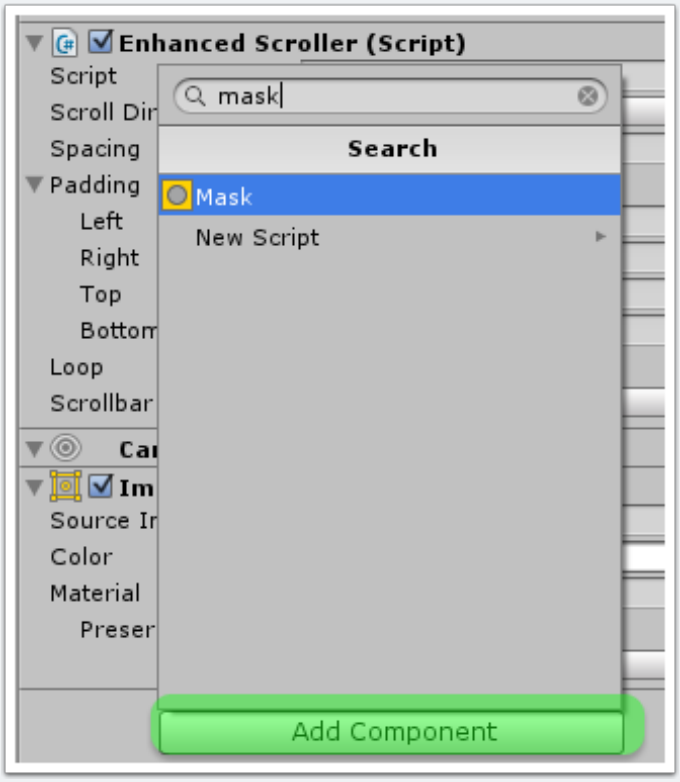
Add an image

While the **Scroller** GameObject is still selected, click on the **Add Component** in the inspector and search for **Image**. Select it to add.



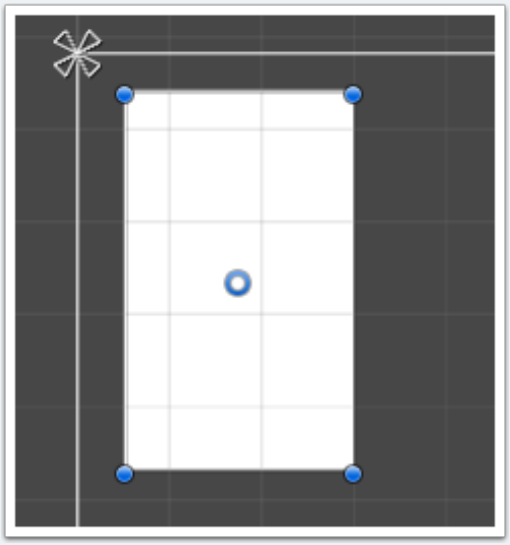
Add a mask

While the **Scroller** GameObject is selected click on the **Add Component** button in the inspector and search for **Mask**. Select to add.



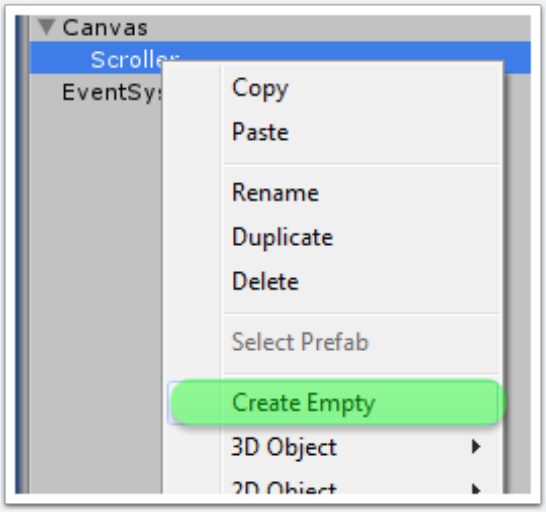
Resize and position the scroller GameObject

In the **Scene** window of the Unity editor, drag the scroller around until you like its position and size



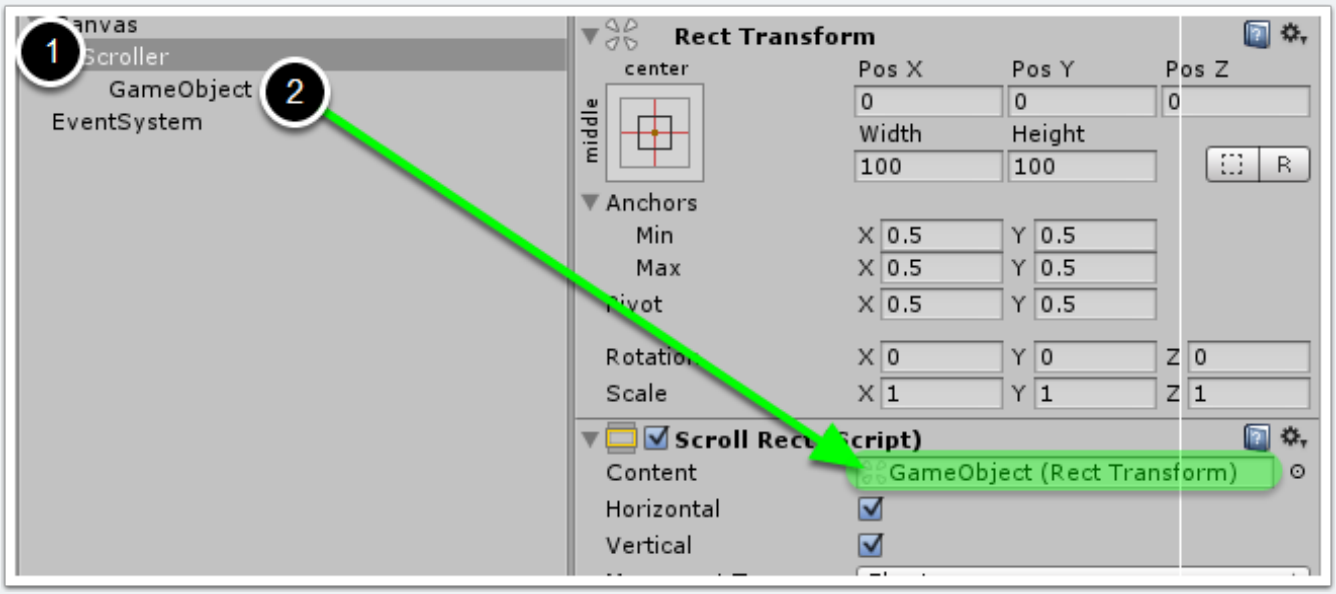
Create a temporary cell view container

Right-click on the **Scroller** GameObject in the scene hierarchy and select **Create Empty**. Don't bother renaming this new GameObject as it will be deleted when the scroller starts up. The only reason we have to create this is because Unity will give errors if the Scroll Rect has no content GameObject. We can also use this GameObject as a way to create and preview the cell view.



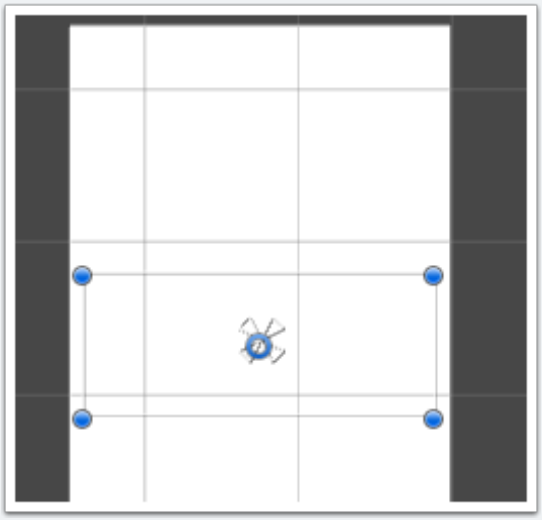
Link the cell view container to the scroller

1. Select the **Scroller** GameObject in the scene hierarchy
2. Drag the new **GameObject** to the **Content** field of the scroller's **Scroll Rect** component in the inspector



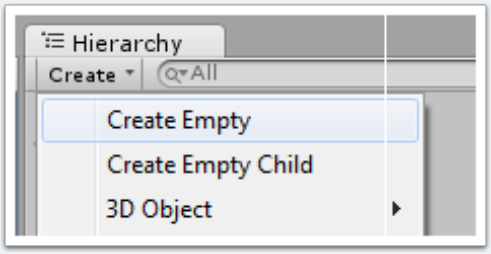
Resize and position the temporary cell view container

Resize the temporary GameObject you just created. This GameObject will be deleted at runtime, so don't worry too much about positioning. You are just resizing and positioning so that you can preview your cell views easily.



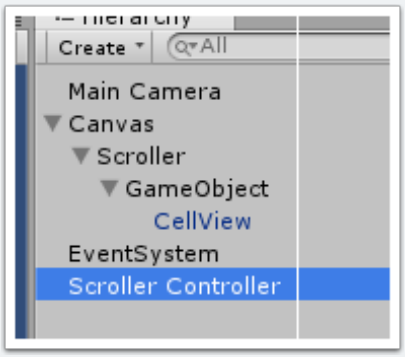
Create a scroller controller GameObject

Click on the **Create** button in the scene hierarchy and select **Create Empty**



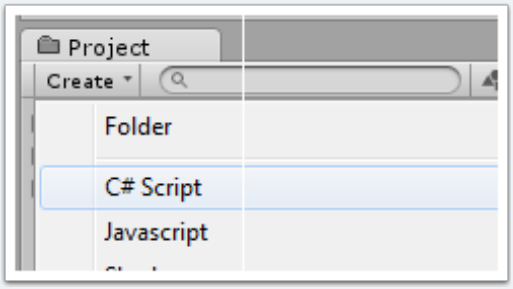
Rename the GameObject

Rename the new GameObject to **Scroller Controller** for this tutorial



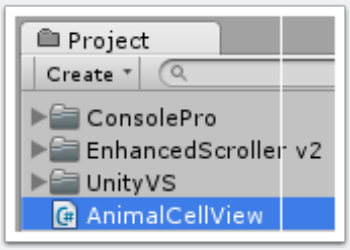
Create a cell view script

Create a new C# script by clicking the **Create** button in the project window.



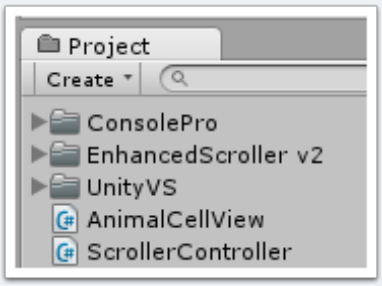
Rename the cell view script

Rename the new script to **AnimalCellView** for this tutorial



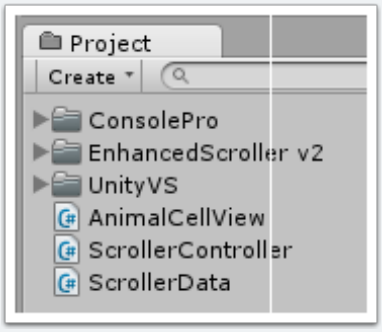
Create a scroller controller script

Create a scroller controller script like you did for the cell view. Rename the script to **ScrollerController** for this tutorial



Create a scroller data script

Create a data script and rename to **ScrollerData** for this tutorial



Set up the data script

Open up the **ScrollerData** script in your script editor and copy this code over what is already there

```
public class ScrollerData
{
    public string animalName;
}
```

Explanation:

This class will hold the data for our list. This should just be pure data without any concern for presentation. We will deal with presentation in the cell view script.

The data class is a representation of one record of our data.

Note: you can use your own data classes from any source. The EnhancedScroller isn't even aware of the data, only your delegate will handle this. This makes the scroller highly flexible and reusable.

Set up the cell view script

Open up the **AnimalCellView** script in your script editor and copy this code over what is already there

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using EnhancedUI.EnhancedScroller;

public class AnimalCellView : EnhancedScrollerCellView
{
    public Text animalNameText;

    public void SetData(ScrollerData data)
    {
        animalNameText.text = data.animalName;
    }
}
```

Explanation:

The AnimalCellView script is the representation of our data in the scene. It will handle how the data is layed out and formatted. This class must inherit from the EnhancedScrollerCellView.

There is one UI field that links the Animal Name Text GameObject to this view script called animalNameText.

The SetData function is optional, but it allows you to pass data to the view so that it can be displayed. In this example, the animalNameText object's text property is updated to the data record's animalName field.

Note: we set up some library references at the top of the class to simplify usage in the class body.

Set up the scroller controller script

Open up the **ScrollerController** script in your script editor and copy this code over what is already there

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using EnhancedUI.EnhancedScroller;

public class ScrollerController : MonoBehaviour, IEnhancedScrollerDelegate
{
    private List<ScrollerData> _data;

    public EnhancedScroller myScroller;
    public AnimalCellView animalCellViewPrefab;
```

```
        void Start ()
    {
        _data = new List<ScrollerData>();

        _data.Add(new ScrollerData() { animalName = "Lion" });
        _data.Add(new ScrollerData() { animalName = "Bear" });
        _data.Add(new ScrollerData() { animalName = "Eagle" });
        _data.Add(new ScrollerData() { animalName = "Dolphin" });
        _data.Add(new ScrollerData() { animalName = "Ant" });
        _data.Add(new ScrollerData() { animalName = "Cat" });
        _data.Add(new ScrollerData() { animalName = "Sparrow" });
        _data.Add(new ScrollerData() { animalName = "Dog" });
        _data.Add(new ScrollerData() { animalName = "Spider" });
        _data.Add(new ScrollerData() { animalName = "Elephant" });
        _data.Add(new ScrollerData() { animalName = "Falcon" });
        _data.Add(new ScrollerData() { animalName = "Mouse" });

        myScroller.Delegate = this;
        myScroller.ReloadData();
    }

    public int GetNumberOfCells(EnhancedScroller scroller)
    {
        return _data.Count;
    }

    public float GetCellViewSize(EnhancedScroller scroller, int dataIndex)
    {
        return 100f;
    }

    public EnhancedScrollerCellView GetCellView(EnhancedScroller scroller, int
dataIndex, int cellIndex)
    {
        AnimalCellView cellView = scroller.GetCellView(animalCellViewPrefab) as
AnimalCellView;
        cellView.SetData(_data[dataIndex]);
        return cellView;
```

```
    }
}
```

Explanation:

This controller is the heart of our tutorial. It handles setting up the data for the scroller and provides some callbacks that the scroller will request when it needs information. The controller can handle any type of data, it doesn't even need a list to work. You could have completely separated objects being used to drive the scroller.

Here is a breakdown:

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using EnhancedUI.EnhancedScroller;
```

Set up some references to our libraries we'll be using.

```
public class ScrollerController : MonoBehaviour, IEnhancedScrollerDelegate
```

Inheriting from the IEnhancedScrollerDelegate interface, we are telling the ScrollerController that we need to set up some callbacks for the EnhancedScroller.

```
private List<ScrollerData> _data;
```

This will be our list of data records

```
public EnhancedScroller myScroller;
public AnimalCellView animalCellViewPrefab;
```

These lines are inspector fields that we will use to link our GameObjects to this class

```
     void Start ()
{
    _data = new List<ScrollerData>();

    _data.Add(new ScrollerData() { animalName = "Lion" });
    _data.Add(new ScrollerData() { animalName = "Bear" });
    _data.Add(new ScrollerData() { animalName = "Eagle" });
    _data.Add(new ScrollerData() { animalName = "Dolphin" });
    _data.Add(new ScrollerData() { animalName = "Ant" });
    _data.Add(new ScrollerData() { animalName = "Cat" });
    _data.Add(new ScrollerData() { animalName = "Sparrow" });
    _data.Add(new ScrollerData() { animalName = "Dog" });
    _data.Add(new ScrollerData() { animalName = "Spider" });
    _data.Add(new ScrollerData() { animalName = "Elephant" });
    _data.Add(new ScrollerData() { animalName = "Falcon" });
    _data.Add(new ScrollerData() { animalName = "Mouse" });

    myScroller.Delegate = this;
    myScroller.ReloadData();
}
```

The Start function occurs when the scene loads. Here we create our list of data and tell the scroller that it should use the ScrollerController as its delegate. By setting this script as the scroller's delegate, we are telling the scroller that when it needs information about our data or views, it should ask this script. Finally, we reload the data to get it to display.

```
public int GetNumberOfCells(EnhancedScroller scroller)
{
    return _data.Count;
}
```

This is one of the delegate callbacks that the scroller will call to get information. GetNumberOfCells just tells the scroller how many list items we are expecting. In this case, we just return the number of items in the _data list.

```
    public float GetCellViewSize(EnhancedScroller scroller, int dataIndex)
    {
        return 100f;
    }
```

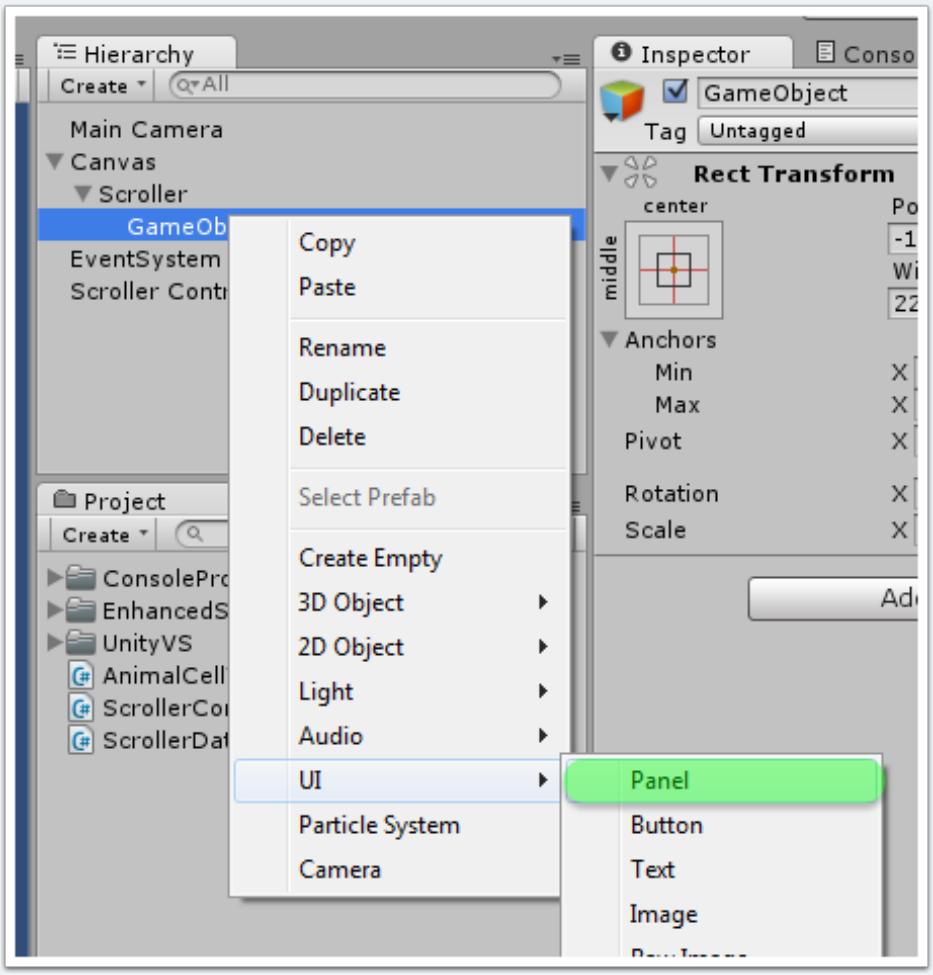
GetCellViewSize is another callback that tells the scroller how large to make each cell. This number could potentially be different for each cell, but in this example we are returning a static 100 pixels for all cells.

```
    public EnhancedScrollerCellView GetCellView(EnhancedScroller scroller, int
dataIndex, int cellIndex)
    {
        AnimalCellView cellView = scroller.GetCellView(animalCellViewPrefab) as
AnimalCellView;
        cellView.SetData(_data[dataIndex]);
        return cellView;
    }
```

GetCellView returns which cell view prefab the scroller should use to display the cell at the given dataIndex. In this example, we are only using a single type of cell, the animalCellViewPrefab. First we ask the scroller to create the view for us. If the view has already been created and is in the recycled list, the scroller will recycle the view instead of creating a new one. Next, the cell view has its data set. This is optional, but in this case it is what drives the view to update its text UI. Finally, we return the cell view to the scroller for processing.

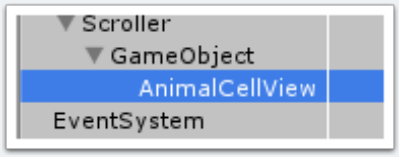
Create a cell view GameObject in the temporary container

In the scene hierarchy right-click on the temporary **GameObject** under the Scroller. Select **UI** > **Panel**



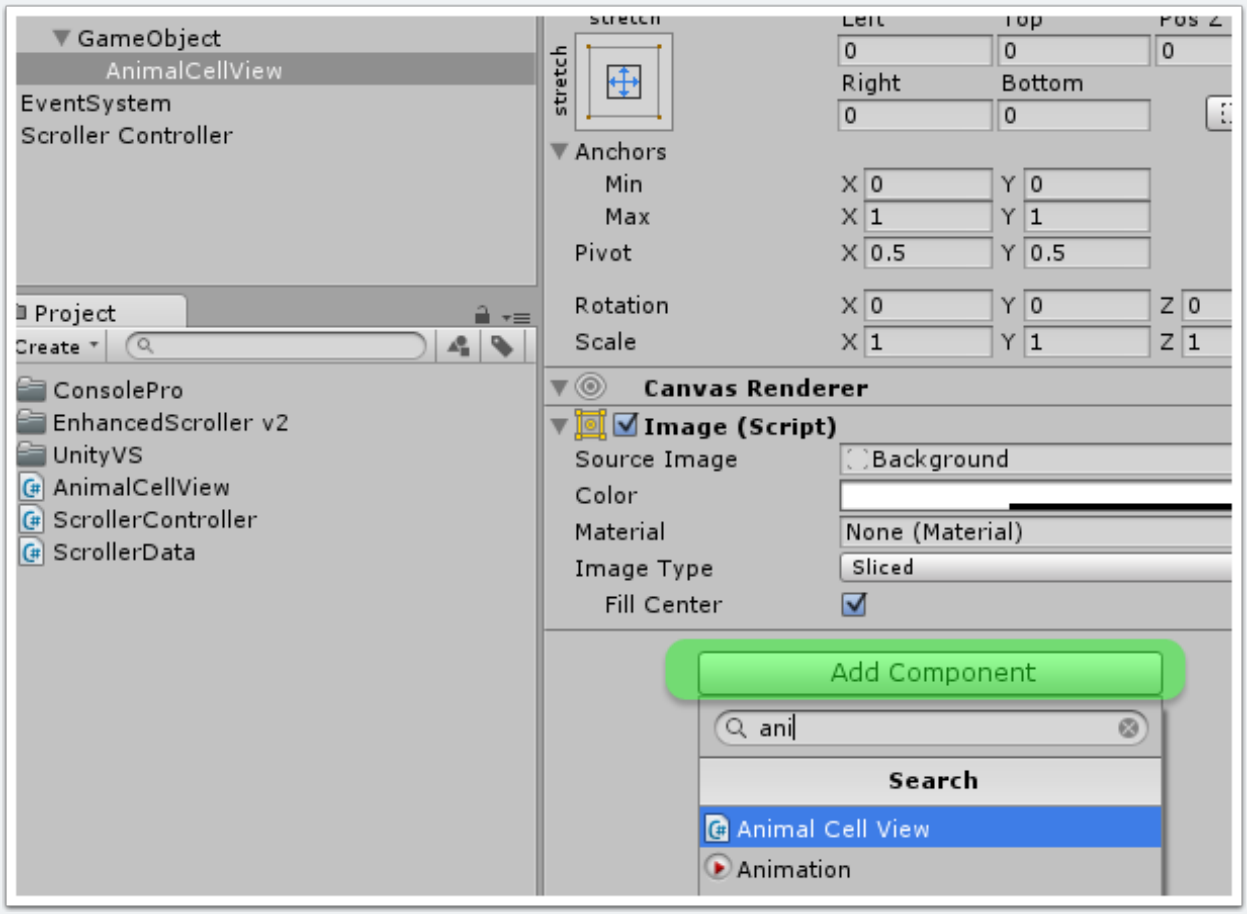
Rename the cell view GameObject

Rename the new GameObject **AnimalCellView**



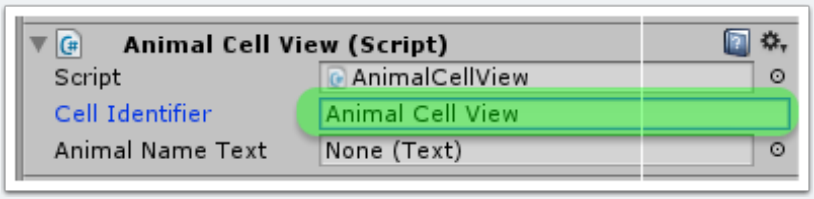
Add the cell view component to the cell view GameObject

While the **AnimalCellView** GameObject is selected, click the **Add Component** button in the inspector and search for **Animal Cell View**. Select it to add the component.



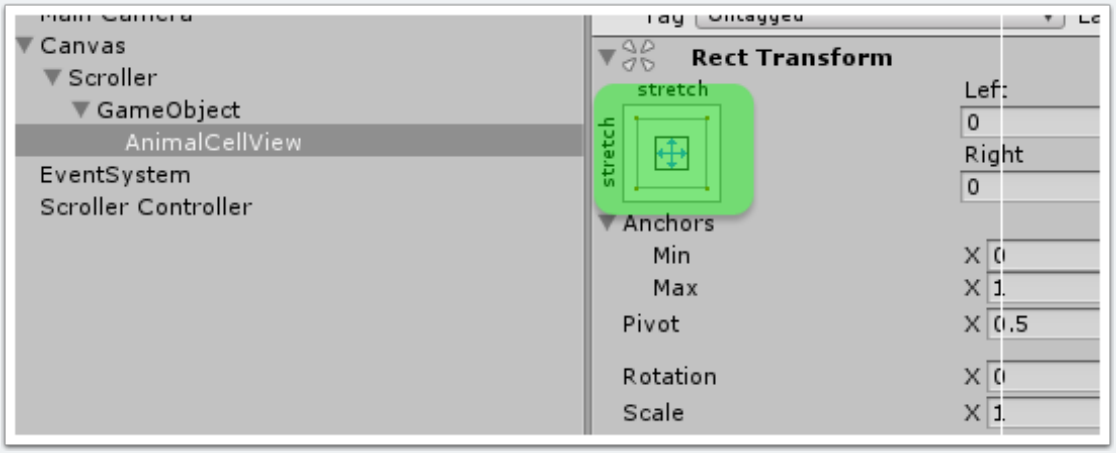
Set the cell view identifier

Change the cell identifier to something useful. In this tutorial we will set it to **Animal Cell View**. Since this is the only cell view used in the tutorial, we could have left it blank. The **Cell Identifier** field should be unique for all the cell prefabs that the scroller will be handling so it can choose the correct GameObject to recycle.



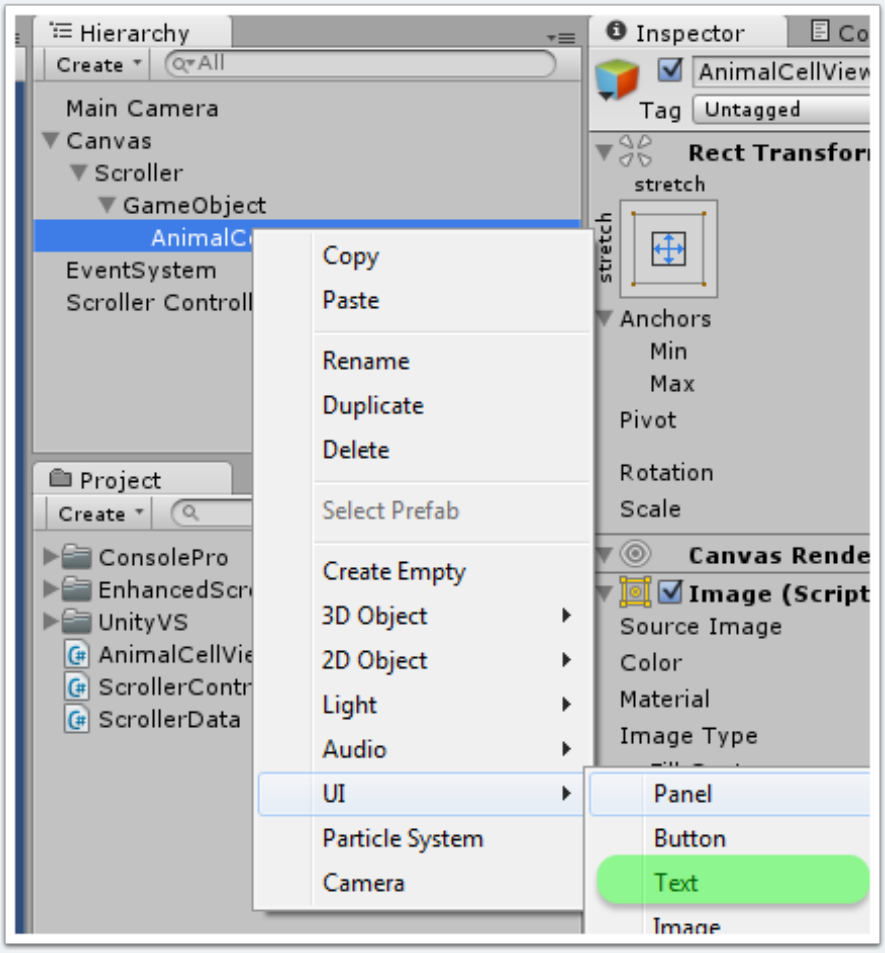
Set the cell view GameObject to stretch

Be sure the cell view is stretching on both axis. The scroller will stretch the cells to fit according to the scroller dimensions and the cell size specified by the delegate.



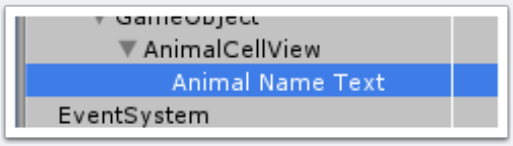
Create a UI Text GameObject for the animal name

In the scene hierarchy, right-click the **AnimalCellView** GameObject and select **UI** > **Text**



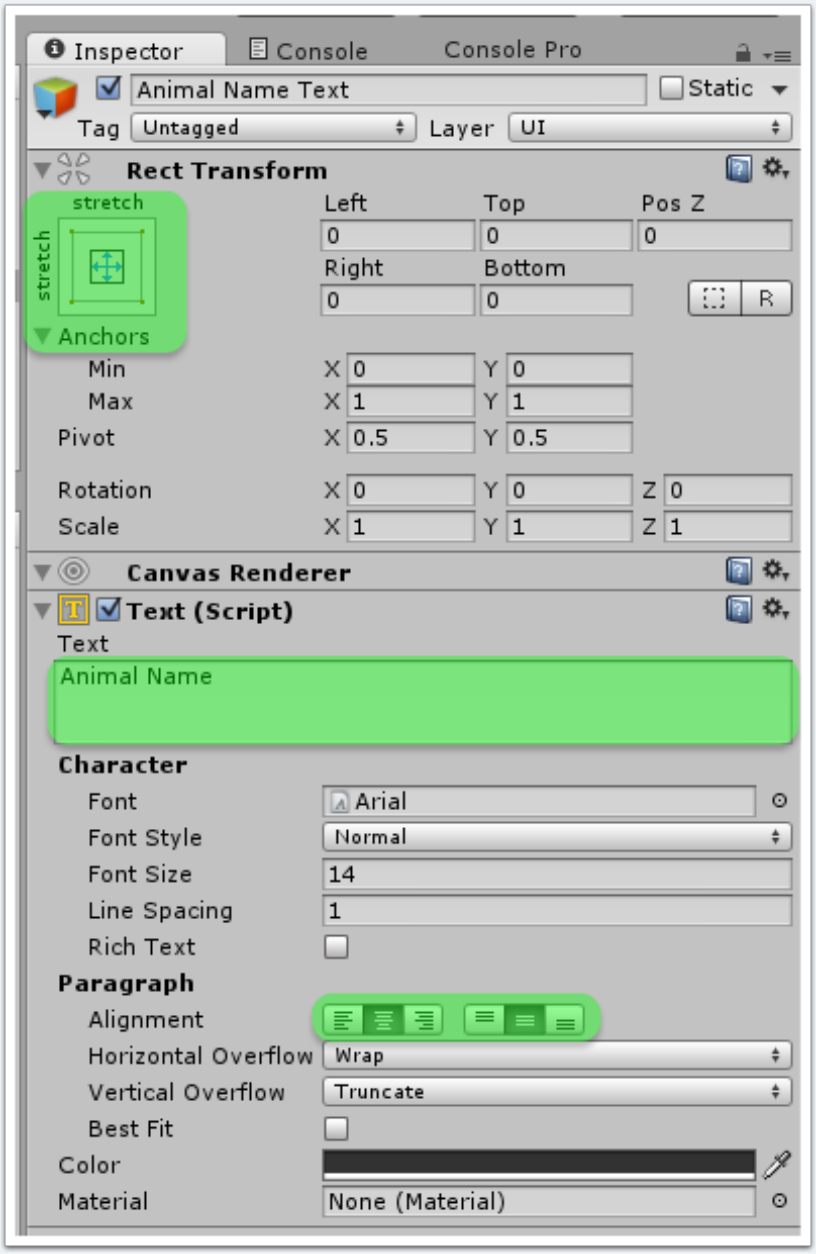
Rename the text GameObject

Rename the new GameObject **Animal Name Text** for this tutorial



Set up the text GameObject's properties

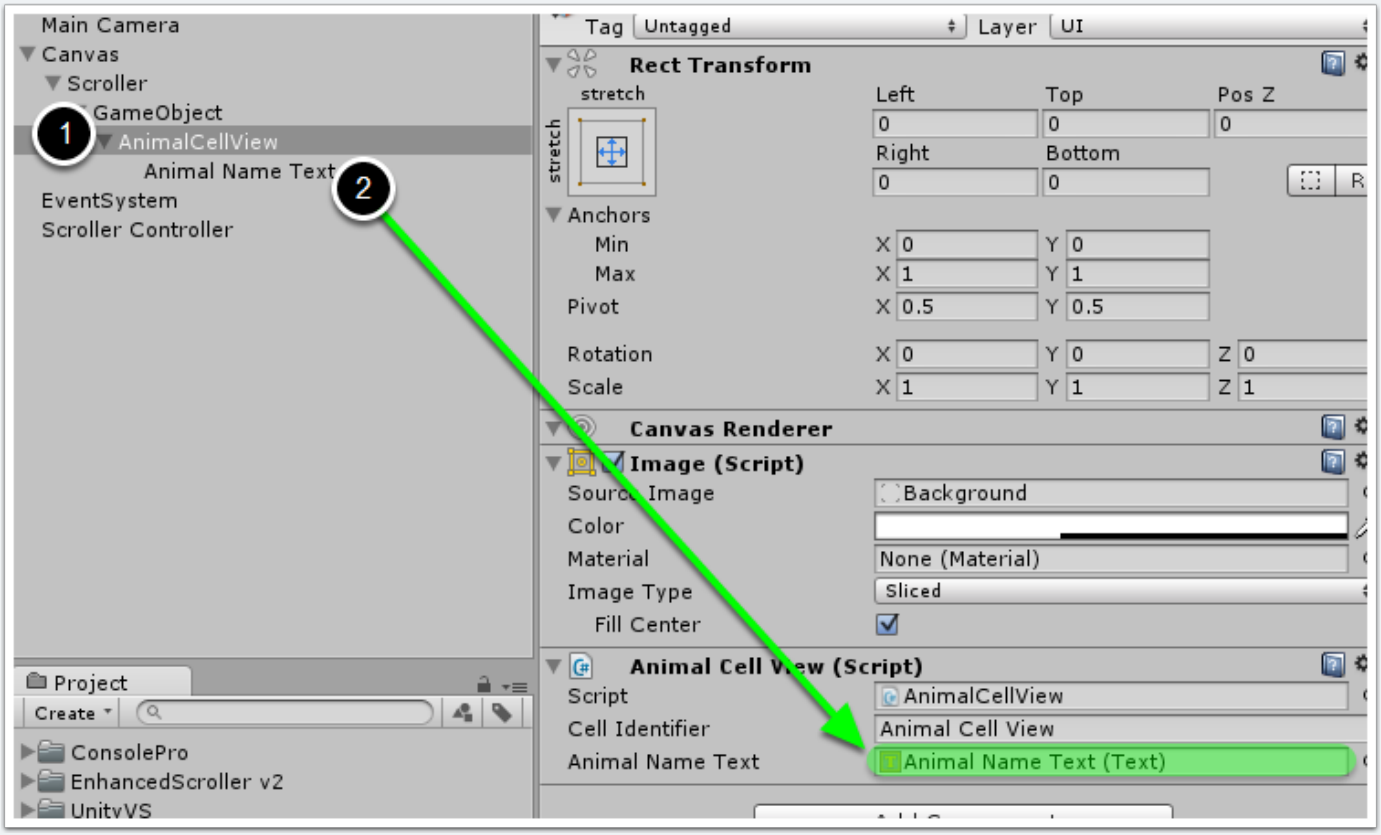
While the text GameObject is still selected, change its properties in the inspector. You can use the following setting or experiment with your own. For this example, we changed the the stretching and anchor to fill the parent panel completely. Also, the alignment was changed to center vertically and horizontally.



Link the text GameObject to the cell view

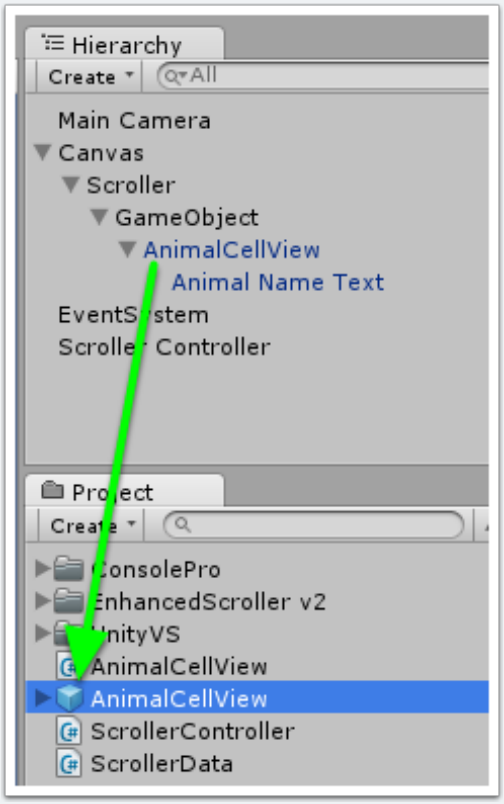
1. Select the **AnimalCellView** GameObject in the scene hierarchy

2. Drag the **Animal Name Text** GameObject to the **Animal Name Text** field of the AnimalCellView component in the inspector



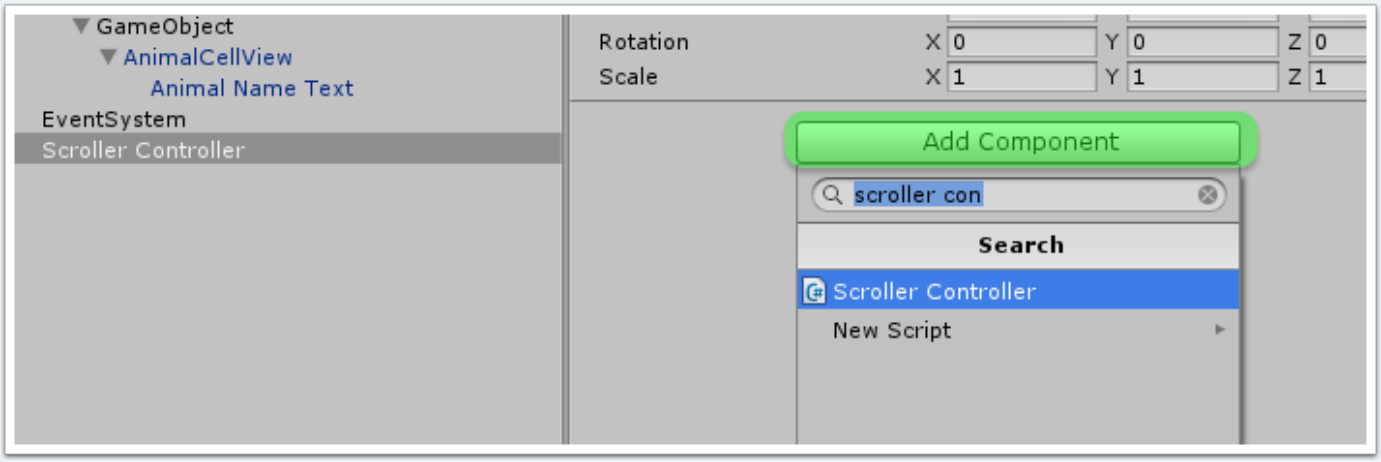
Create a prefab of the cell view

Click on the **AnimalCellView** GameObject in the scene hierarchy and drag it to the project window to create a prefab for this cell view



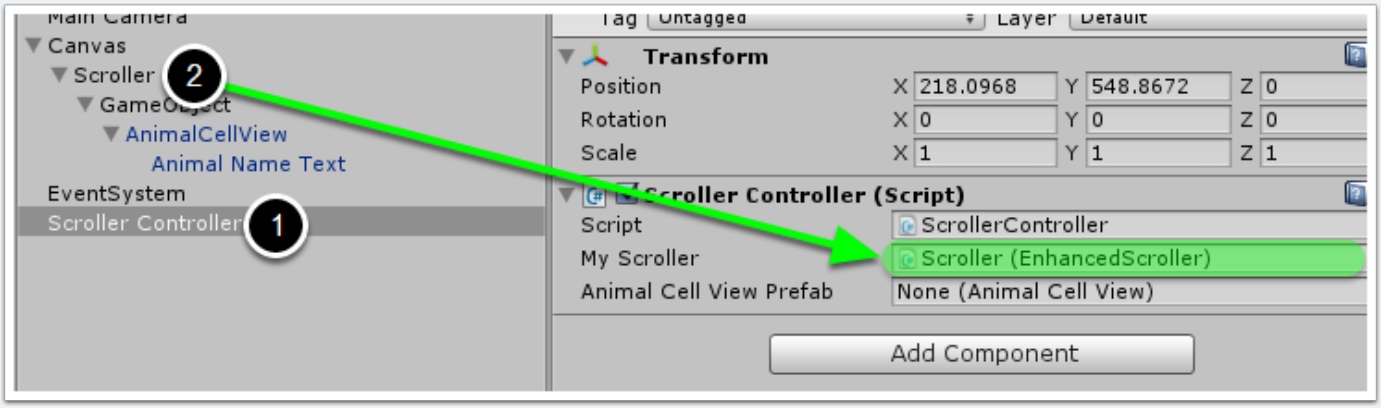
Add the scroller controller script to the Scroller Controller GameObject

Select the **Scroller Controller** GameObject in the scene hierarchy. In the inspectory click the **Add Component** button and search for **Scroller Controller**. Select to add the component.



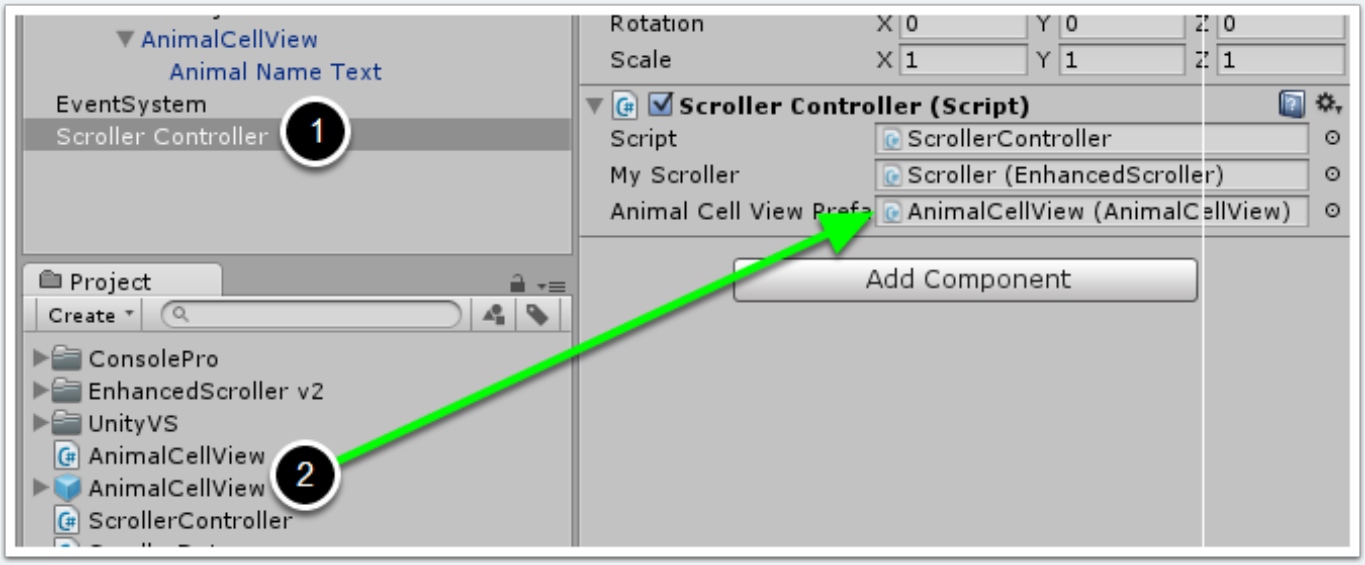
Link the Scroller GameObject to the Scroller Controller

1. Select the **Scroller Controller** GameObject in the scene hierarchy
2. Drag the **Scroller** GameObject to the **My Scroller** field of the Scroller Controller script in the inspector



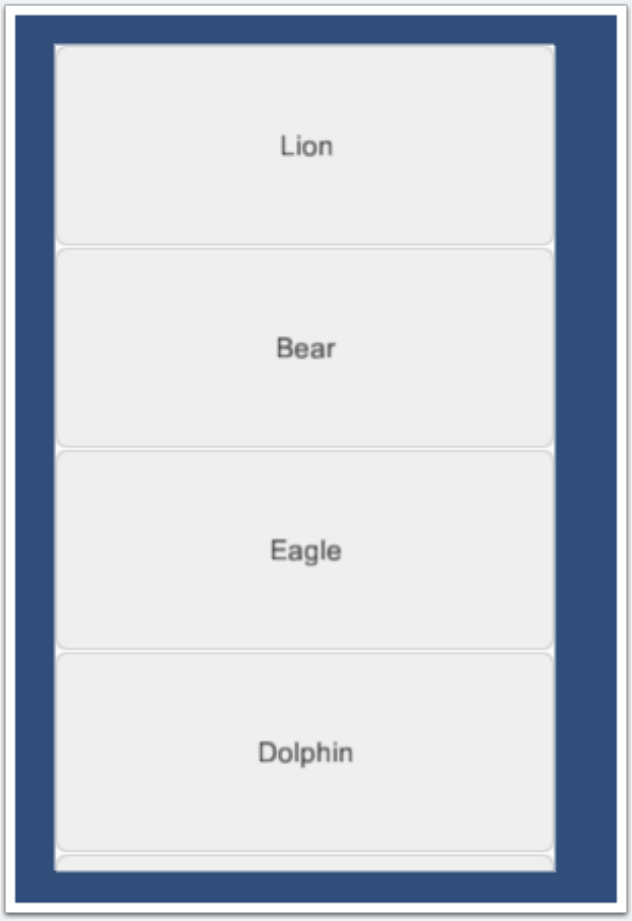
Link the cell view prefab to the scroller component

1. Select the **Scroller Controller** GameObject in the scene hierarchy
2. Drag the **AnimalCellView** prefab from the project window to the **Animal Cell View Prefab** field of the Scroller Controller's inspector



Run the scene

Click the run button to see the scroller in action.



Recycling

While the scene is running you can see the active cell views in your scroller by expanding the **Container** GameObject. As you scroll, this list will be updated. If a cell gets recycled, it will show up under the **Recycled Cells** list. You can see the recycling visually by turning off the **Mask** component of the Scroller if you prefer.