Directions for Gallery Scene

NOTICE: This section assumes you have fully completed the General Setup, all Space Instructions, and all prior Music instructions.

Getting Started

This is the last scene we need to work on for the music project. These instructions will contain open-ended guidelines; however, most of this scene's technical details will be left as a challenge for you.

Creating the spawner

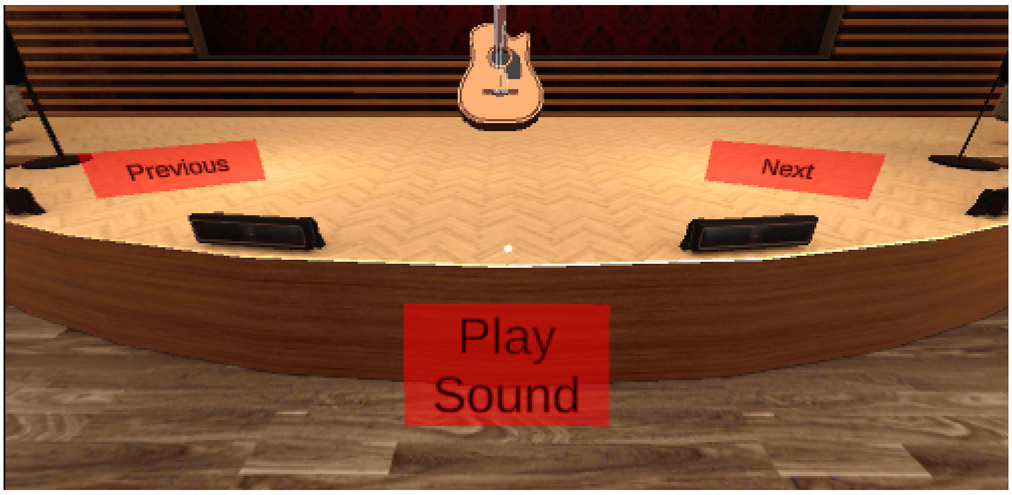
1. You will start by creating a spawner just like you did for the Finite scene creation. The only difference this time is that you will only have one spawn location in the middle of the stage as we will only spawn one instrument at a time. Ensure that you have both the parent Spawner object and the child Spawner Location object.
2. Position your spawner in such a way that is far enough away to still be on the stage but close enough to see the instrument clearly. *HINT: You can drag some of the instrument prefabs into the scene and change their position to find what position looks best to you. You will then delete these prefabs from the scene afterwards and use the position values you liked for them for your spawner.*

Code will be added to this object later in these instructions.

Adding (Editing) the Buttons

3. Now that we have our spawner location, we can create the buttons that will surround the spawned instrument. Copy and paste in the MusicMainMenuButtonDrop game object and all children into this scene from the Finite game scene. Delete the following children of the MusicMainMenuButtonDrop game object: ScoreText, PanelGreatWork, and ButtonRestart.
4. Rename the Left button to Previous Button, the Middle Button to Play Sound Button, and the Right Button to Next Button.
5. Change the position and text content of the Previous, Play Sound, and Next buttons to match how they look in our game. Remove the current method actions from these buttons; new actions will be assigned after the code is created.

An example of how our buttons are positioned is shown below:

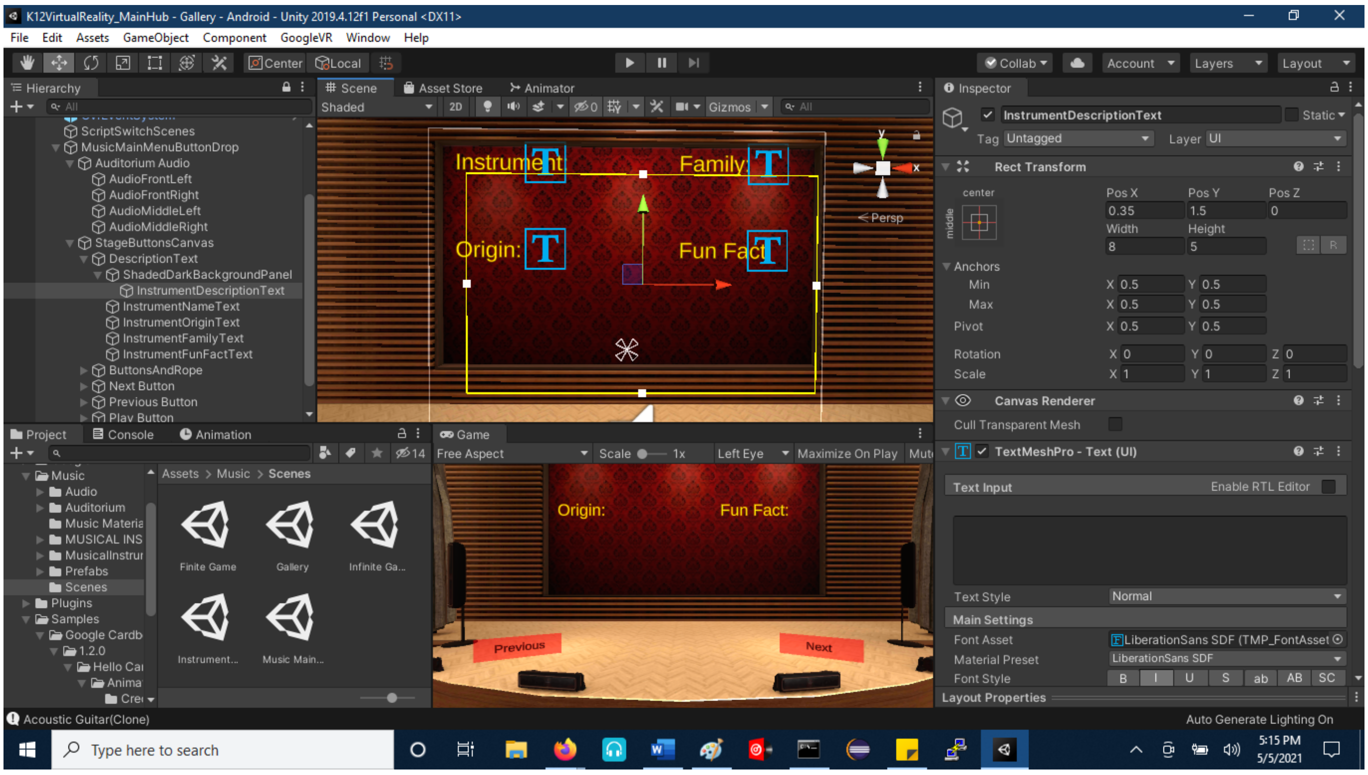


6. Next, edit the Start button and instructions panel to state information relevant to the Gallery scene. Remove the current method actions from these buttons; new actions will be assigned after the code is created. This will be assigned to work with the previously created animations later on. For now, you can move the RopeAndButtons object to the objects raised position so that it is not in your way.

Adding the Instrument Description UI

7. We can now add the Text Descriptions for each instrument. Start by creating an empty game object and name it DescriptionText. This will serve as a holder for all text objects.
8. In order to have our text stand out more, we chose to add a mostly transparent black panel over the red area of the back stage wall. This was purely a style choice; therefore, you may choose to do the same or not.
9. Create 4 text objects and name them as follows: InstrumentNameText, InstrumentFamilyText, InstrumentOriginText, and InstrumentFunFactText and make the text content "Instrument:", "Family:", "Origin:", and "Fun Fact:", respectively. Position each text object on the panel in a way similar to ours that allows room for descriptions for each instrument to be displayed and for the instrument itself to take up a good portion of the middle of the panel.
10. Next, as a child of the panel (or not if you did not create a panel), create another text object and name it InstrumentDescriptionText. Ensure that this text box is large and positioned in such a way that it covers enough of the panel to overlay the Text Headers already on the panel. You can use sample text to test different positions and font sizes, but ultimately, this text content will be left empty to allow dynamic assigning of text descriptions at the location of this text description.

An example of how your hierarchy and scene should look at this time is shown below:


K12VirtualReality_MainHub - Gallery - Android - Unity 2019.4.12f1 Personal <DX11>
File Edit Assets GameObject Component GoogleVR Window Help
Center Local
Collab Account Layers Layout
Hierarchy
Scene Asset Store Animator
Shaded 2D Gizmos
Instrument: Family:
Origin: Fun Fact:
Persp
Inspector
Project Console Animation
Game
Free Aspect Scale 1x Left Eye Maximize On Play
Assets > Music > Scenes
Finite Game Gallery Infinite Ga... Instrument... Music Main...
Previous Next
Acoustic Guitar(Clone)
Auto Generate Lighting On
Type here to search
5:15 PM 5/5/2021

Hierarchy:

```
ScriptSwitchScenes
MusicMainMenuButtonDrop
  Auditorium Audio
    AudioFrontLeft
    AudioFrontRight
    AudioMiddleLeft
    AudioMiddleRight
  StageButtonsCanvas
    DescriptionText
      ShadedDarkBackgroundPanel
        InstrumentDescriptionText
      InstrumentNameText
      InstrumentOriginText
      InstrumentFamilyText
      InstrumentFunFactText
    ButtonsAndRope
    Next Button
    Previous Button
    Play Button
```

Project folders:

```
Music
  Audio
  Auditorium
  Music Materia
  MUSICAL INS
  MusicalInstru
  Prefabs
  Scenes
Plugins
Samples
  Google Cardb
    1.2.0
      Hello Ca
        Anima
          Cre
```

Inspector:

```
InstrumentDescriptionText                          Static
Tag Untagged        Layer UI
Rect Transform
center     Pos X 0.35    Pos Y 1.5    Pos Z 0
           Width 8       Height 5
Anchors
  Min      X 0.5   Y 0.5
  Max      X 0.5   Y 0.5
Pivot      X 0.5   Y 0.5
Rotation   X 0   Y 0   Z 0
Scale      X 1   Y 1   Z 1
Canvas Renderer
  Cull Transparent Mesh
TextMeshPro - Text (UI)
  Text Input                Enable RTL Editor
  Text Style                Normal
  Main Settings
  Font Asset                LiberationSans SDF (TMP_FontAsset)
  Material Preset           LiberationSans SDF
  Font Style                B I U S ab AB SC
Layout Properties
```

Creating the Text Description Prefabs

11. The next step is the create the actual descriptions for each instrument that will be called dynamically in the code. We chose to do this by creating prefabs for each instrument description and then displaying those prefabs when the corresponding instrument was currently displayed. This is not hard to do;

however, it is tedious as it has to be done for each instrument. You can use the same instrument descriptions we did or you are free to gather your own data for the descriptions.

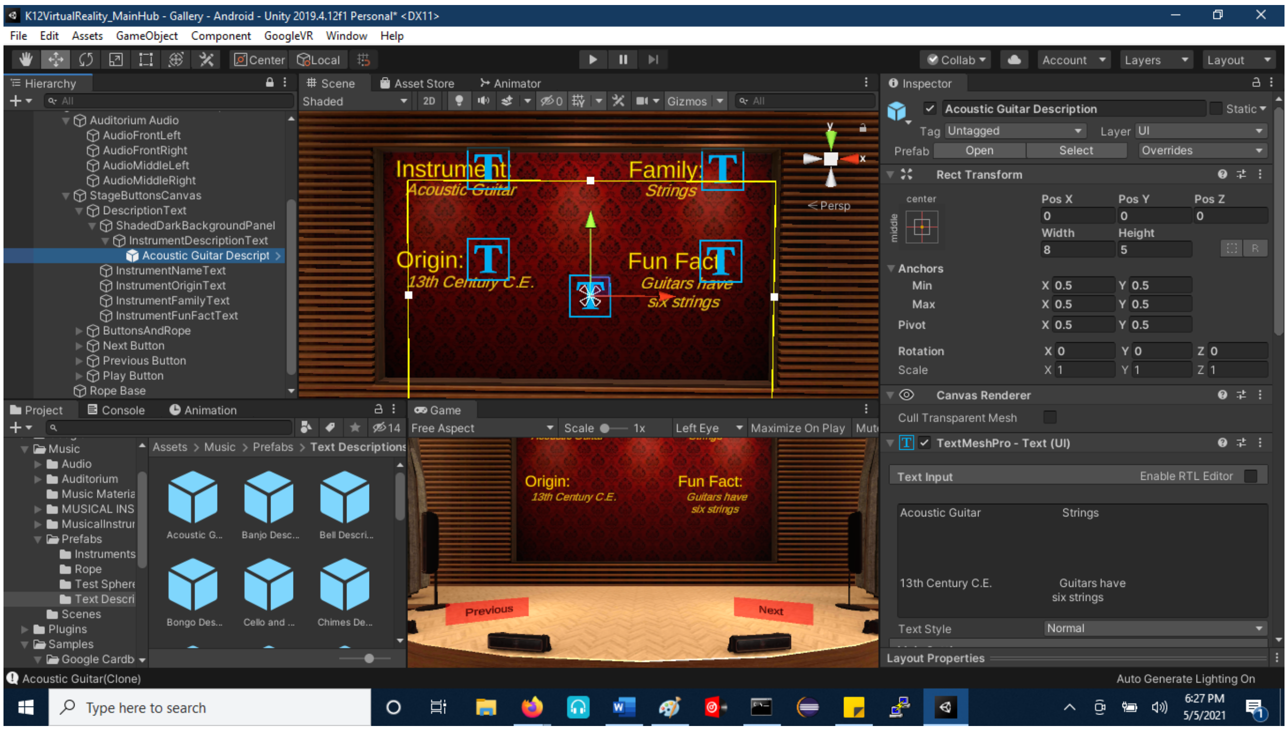
We will cover how to do this for one instrument and then you can use the same logic for the remaining instruments.

12. Go to the text content box for the InstrumentTextDescription text object and enter the description for the Acoustic Guitar. The complete description will be contained in the same area so you have to space your text out accordingly to make it look correct in the scene window. Once the text is spaced correctly and has proper formatting and font size, it is time to make this a prefab.

13. Locate your previously created Prefabs folder in your project window. This folder should already contain your instrument prefabs. Create a new folder and name it Text Descriptions. Open the Text Descriptions folder.

14. Drag the InstrumentTextDescription text object from the hierarchy into the Text Descriptions folder. This will make the InstrumentTextDescription into a prefab. In the Text Descriptions Folder, rename the newly created prefab "Acoustic Guitar Description". Return to the hierarchy, right click the highlighted blue InstrumentTextDescription object, and select "Unpack Prefab Completely". You can now clear the text content box of the InstrumentTextDescription so it is again ready to be used as the Prefabs placeholder. Return to the Acoustic Guitar Description prefab and zero out the position values. The position of the prefabs in the game will be controlled by the InstrumentTextDescription object itself.

15. To test that you created the prefab correctly, drag the prefab you created to be a temporary child of the InstrumentTextDescription object. If you did it correctly, it should look like the image below:


K12VirtualReality_MainHub - Gallery - Android - Unity 2019.4.12f1 Personal* <DX11>

Hierarchy:
- Auditorium Audio
  - AudioFrontLeft
  - AudioFrontRight
  - AudioMiddleLeft
  - AudioMiddleRight
- StageButtonsCanvas
  - DescriptionText
    - ShadedDarkBackgroundPanel
      - InstrumentDescriptionText
        - Acoustic Guitar Descript
      - InstrumentNameText
      - InstrumentOriginText
      - InstrumentFamilyText
      - InstrumentFunFactText
  - ButtonsAndRope
  - Next Button
  - Previous Button
  - Play Button
- Rope Base

Project: Assets > Music > Prefabs > Text Descriptions — Acoustic G..., Banjo Desc..., Bell Descri..., Bongo Des..., Cello and ..., Chimes De...

Inspector: Acoustic Guitar Description
Tag Untagged, Layer UI
Rect Transform: Pos X 0, Pos Y 0, Pos Z 0; Width 8, Height 5
Anchors: Min X 0.5 Y 0.5; Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
TextMeshPro - Text (UI)
Text Input:
Acoustic Guitar   Strings

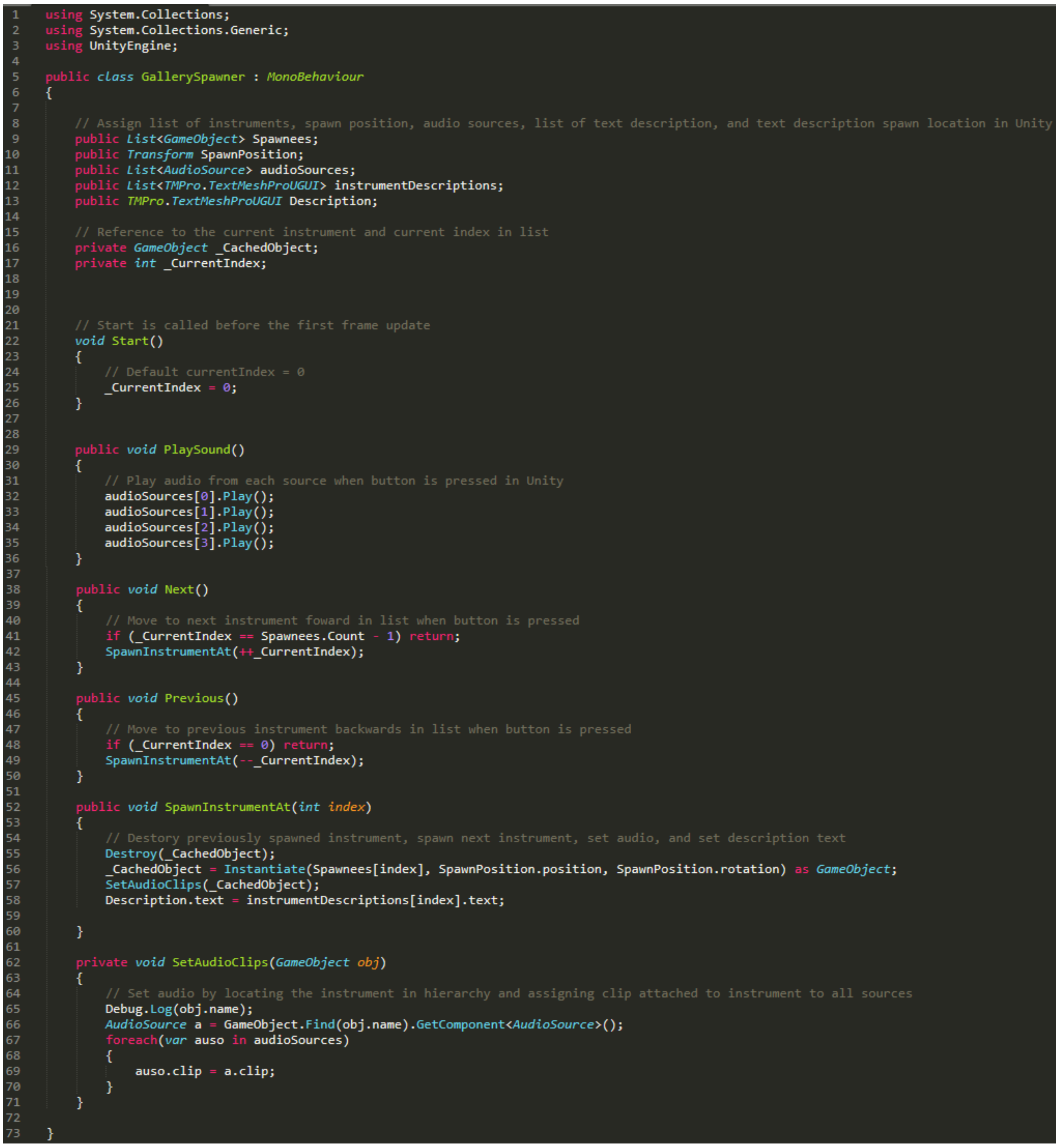
13th Century C.E.   Guitars have
                          six strings
Text Style Normal

16. Use this logic to create the prefabs for the remaining instruments.

Creating the Code

17. We are now ready to create the code for the Gallery Scene. Navigate to the parent Gallery Spawner object and attach a new script component and name the script GallerySpawner.

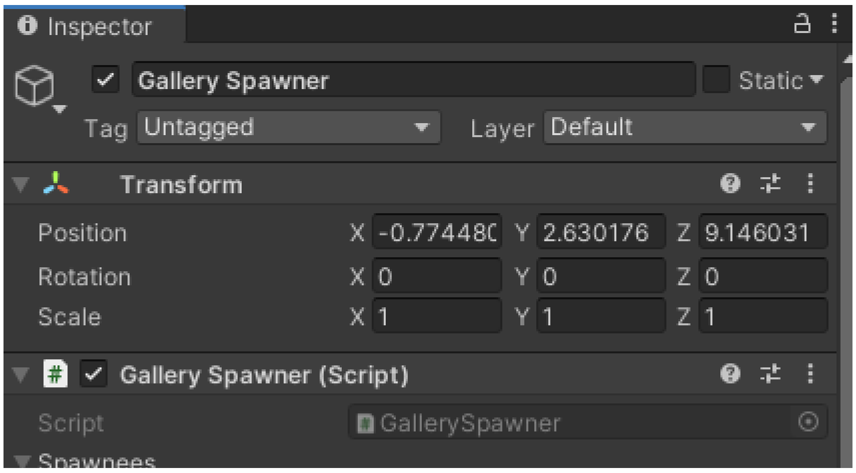
18. Make your script look like the code below:



```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GallerySpawner : MonoBehaviour
{

    // Assign list of instruments, spawn position, audio sources, list of text description, and text description spawn location in Unity
    public List<GameObject> Spawnees;
    public Transform SpawnPosition;
    public List<AudioSource> audioSources;
    public List<TMPro.TextMeshProUGUI> instrumentDescriptions;
    public TMPro.TextMeshProUGUI Description;

    // Reference to the current instrument and current index in list
    private GameObject _CachedObject;
    private int _CurrentIndex;



    // Start is called before the first frame update
    void Start()
    {
        // Default currentIndex = 0
        _CurrentIndex = 0;
    }


    public void PlaySound()
    {
        // Play audio from each source when button is pressed in Unity
        audioSources[0].Play();
        audioSources[1].Play();
        audioSources[2].Play();
        audioSources[3].Play();
    }

    public void Next()
    {
        // Move to next instrument foward in list when button is pressed
        if (_CurrentIndex == Spawnees.Count - 1) return;
        SpawnInstrumentAt(++_CurrentIndex);
    }

    public void Previous()
    {
        // Move to previous instrument backwards in list when button is pressed
        if (_CurrentIndex == 0) return;
        SpawnInstrumentAt(--_CurrentIndex);
    }

    public void SpawnInstrumentAt(int index)
    {
        // Destory previously spawned instrument, spawn next instrument, set audio, and set description text
        Destroy(_CachedObject);
        _CachedObject = Instantiate(Spawnees[index], SpawnPosition.position, SpawnPosition.rotation) as GameObject;
        SetAudioClips(_CachedObject);
        Description.text = instrumentDescriptions[index].text;

    }

    private void SetAudioClips(GameObject obj)
    {
        // Set audio by locating the instrument in hierarchy and assigning clip attached to instrument to all sources
        Debug.Log(obj.name);
        AudioSource a = GameObject.Find(obj.name).GetComponent<AudioSource>();
        foreach(var auso in audioSources)
        {
            auso.clip = a.clip;
        }
    }

}
```

Setting the Public Variables for the Script in Unity

19. Assign your public variables in Unity like below:



Gallery Spawner

Static

Tag Untagged

Layer Default

Transform

Position X -0.77448C Y 2.630176 Z 9.146031

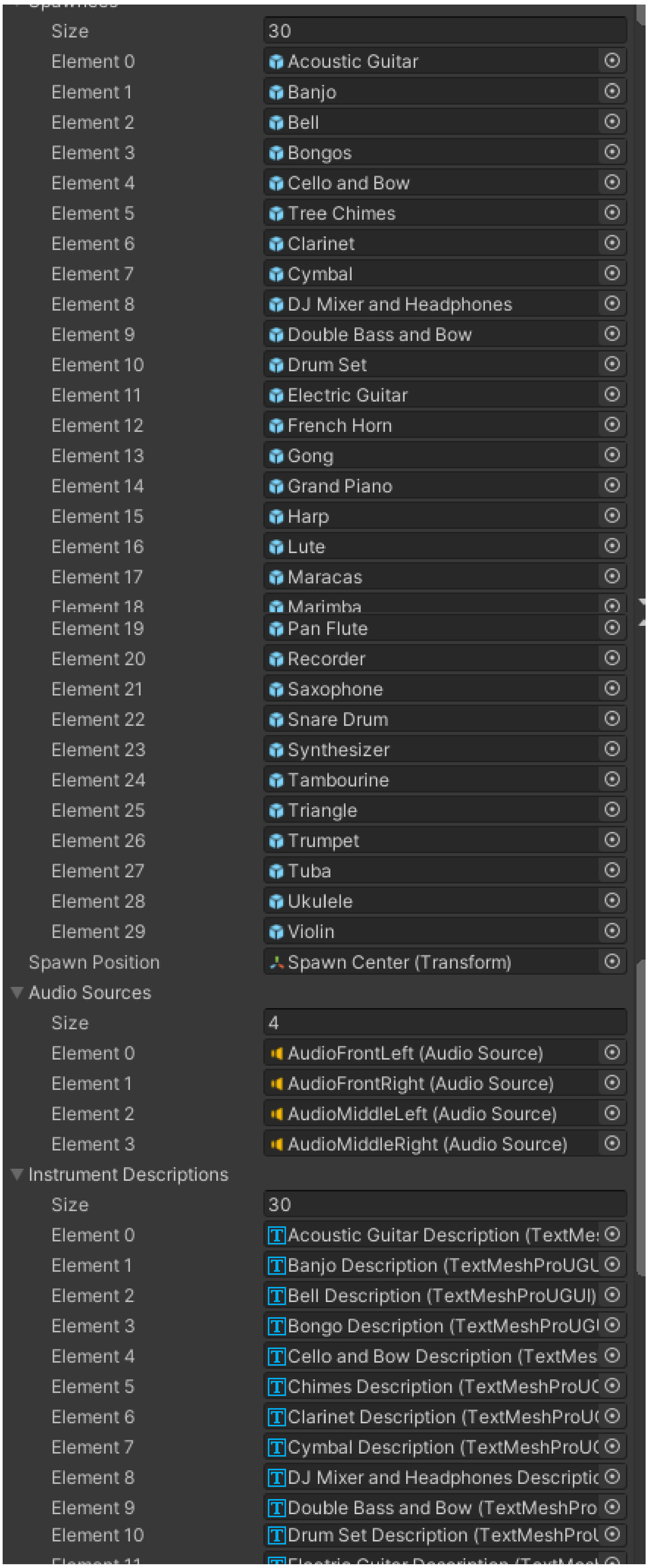
Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

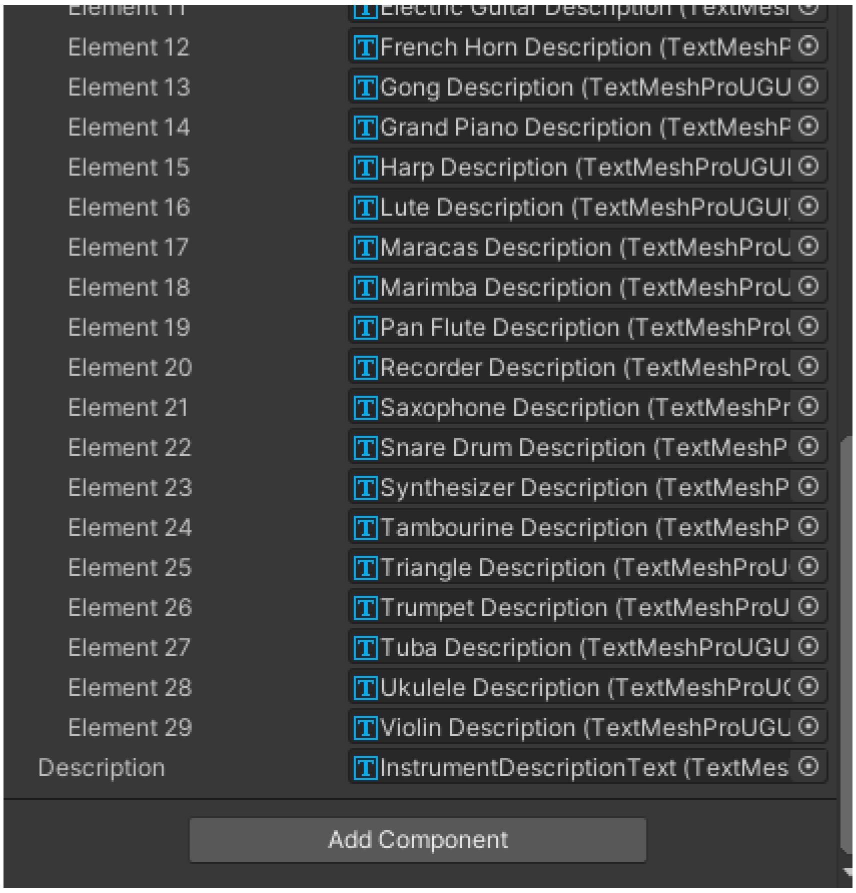
Gallery Spawner (Script)

Script GallerySpawner

Spawnees



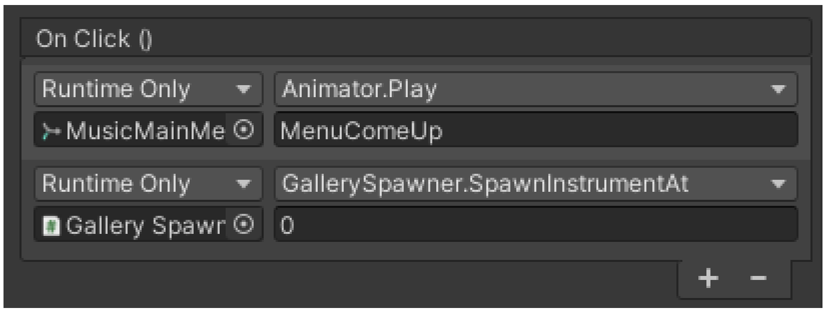
Size	30
Element 0	Acoustic Guitar
Element 1	Banjo
Element 2	Bell
Element 3	Bongos
Element 4	Cello and Bow
Element 5	Tree Chimes
Element 6	Clarinet
Element 7	Cymbal
Element 8	DJ Mixer and Headphones
Element 9	Double Bass and Bow
Element 10	Drum Set
Element 11	Electric Guitar
Element 12	French Horn
Element 13	Gong
Element 14	Grand Piano
Element 15	Harp
Element 16	Lute
Element 17	Maracas
Element 18	Marimba
Element 19	Pan Flute
Element 20	Recorder
Element 21	Saxophone
Element 22	Snare Drum
Element 23	Synthesizer
Element 24	Tambourine
Element 25	Triangle
Element 26	Trumpet
Element 27	Tuba
Element 28	Ukulele
Element 29	Violin
Spawn Position	Spawn Center (Transform)
Audio Sources	
Size	4
Element 0	AudioFrontLeft (Audio Source)
Element 1	AudioFrontRight (Audio Source)
Element 2	AudioMiddleLeft (Audio Source)
Element 3	AudioMiddleRight (Audio Source)
Instrument Descriptions	
Size	30
Element 0	Acoustic Guitar Description (TextMe:
Element 1	Banjo Description (TextMeshProUGL
Element 2	Bell Description (TextMeshProUGUI)
Element 3	Bongo Description (TextMeshProUGI
Element 4	Cello and Bow Description (TextMes
Element 5	Chimes Description (TextMeshProU(
Element 6	Clarinet Description (TextMeshProU(
Element 7	Cymbal Description (TextMeshProU(
Element 8	DJ Mixer and Headphones Descriptic
Element 9	Double Bass and Bow (TextMeshPro
Element 10	Drum Set Description (TextMeshProl
Element 11	Electric Guitar Description (TextMesl



Element 11	Electric Guitar Description (TextMesh
Element 12	French Horn Description (TextMeshP
Element 13	Gong Description (TextMeshProUGU
Element 14	Grand Piano Description (TextMeshP
Element 15	Harp Description (TextMeshProUGUI
Element 16	Lute Description (TextMeshProUGUI
Element 17	Maracas Description (TextMeshProU
Element 18	Marimba Description (TextMeshProU
Element 19	Pan Flute Description (TextMeshProU
Element 20	Recorder Description (TextMeshProU
Element 21	Saxophone Description (TextMeshPr
Element 22	Snare Drum Description (TextMeshP
Element 23	Synthesizer Description (TextMeshP
Element 24	Tambourine Description (TextMeshP
Element 25	Triangle Description (TextMeshProU
Element 26	Trumpet Description (TextMeshProU
Element 27	Tuba Description (TextMeshProUGU
Element 28	Ukulele Description (TextMeshProU(
Element 29	Violin Description (TextMeshProUGU
Description	InstrumentDescriptionText (TextMes

Assigning Action to the Buttons

20. The next step is to assign what each button does in the scene. Assign the Previous, Next, and Play Sound buttons to their corresponding method in the GallerySpawner Code.
21. Assign the below actions to the View Gallery button:



On Click ()

Runtime Only	Animator.Play
MusicMainMe	MenuComeUp
Runtime Only	GallerySpawner.SpawnInstrumentAt
Gallery Spawr	0

Updating the Animations to Work with the Gallery Scene

22. This is left as a personal challenge for you. Use the same logic you learned in prior uses of the animator to have the proper elements fade in and out in this scene.

That's all there is to it! Congratulations on finishing your music project!