Directions for Animator Set Up

NOTICE: This section assumes you have fully completed all previous instructions. At the end of these instructions, you will be able to duplicate the Finite scene to create the Infinite scene and apply this logic to the Gallery scene

Getting Started

Whether you chose to use an animator in your project or not is up to you. In our project, animators controlled the lowering and raising of main menu buttons, fading in and out of buttons, and fading in and out of UI elements. The project will still work correctly without these buttons movements and transitions; however, an animator can make a project much cleaner and more streamlined.

In the music spawner instructions, you may have noticed that we created a reference to an animator and had it be triggered by the Clear() method. In this tutorial, we will show you how to create this animator and animation, along with the raising and lowering of the main menu buttons. How you chose to handle the main menu is up to you. If you want it to work like ours with the buttons attached to the rope, simply place the buttons where you want their lowered place to be and place several rope objects behind them to give the look of it being held by the rope. Place all elements you want to move under a single parent game object with a fitting name, such as ButtonsAndRope, so that you can control the movement of all elements at once. For creation of the rope, try searching "how to create a rope in blender" in YouTube and you will find multiple tutorials on the subject. All of the creation of the initial Main Menu will be done in the Music Main Menu Scene; however, the animator logic will be applied to all scenes. The buttons created for the Music Main Menu scene will only have the function of switching to the scenes indicated by the button.

Creating the animator

Once you have decided how you want your main menu to look, you can add an animator to it to control movement. Like you did in the space scene, create a new animator, name it whatever you like, and attach it as a component to the previously created MusicMainMenuDrop game object. Ours will be referred to as MusicMainMenuDrop.

NOTE: *All elements you want to be controlled by the animator must be a child of the game object that you attached the animator to. The background music element of the SceneChanger game object should be turned off at this time to avoid confusion in game.*

MusicMenuDrop animation

1. With the new animator created, create an animation named MusicMenuDrop. This will be the animation that is played at the start of each music scene. You can use this same animation in all scenes and it will only change the elements added that exist in the current scene. For this animator to work properly, multiple elements will be controlled and initialized. The first thing you will do is ensure you are at the 0 second mark of the animator and start recording.
2. While recording, set the location of where you want the raised place of the menu to be and ensure that it is recorded in the animation window. Then move to the 10 second mark and set the location of the final lowered menu. Also set the button to be disabled at the 0 second mark and enabled at the 10

second mark. We do not want this button to be available for use until the menu has dropped fully. If you stop recording and play the animation at this time, you should be able to see the menu slowly drop down. Ensure that your animation is not set to loop.

3. For the Music Main Menu scene, that is all that is needed for this animation. However, we need this animation to work across all scenes so we have to make sure it covers all necessary elements to initialize each scene. Copy the ButtonsAndRope game object and paste it as a child of the StageButtonsCanvas object in the **Finite** scene. Make sure that you attach the animator to the MusicMainMenuDrop object in this scene as well. Edit the buttons to say and read what you like. Do not change the name of the ButtonsAndRope object or the animator will not work anymore. You can use a panel rather than a button to state the instructions for the game. Playing the game at this point should have the menu lowering to the proper place as expected. However, you will notice that the scene appears very cluttered as the UI is visible at all times and the left, right, and middle buttons are always in the scene. We can use the animator to initialize these elements in the scene so that they are not always visible.
4. To continue, navigate to the 0 second mark of the MusicMenuDrop animation and begin recording again. You must go through each selection button (both the button image and button itself) and UI element to set them as existing in the scene, but currently disabled and transparent. Doing so correctly will result in a proper, uncluttered opening to the scene. Other than adding the "Great Work!" panel and restart button (which you may have already done in the last instructions), this animation is finished.

MenuComeUp animation

5. The next task is to create the MenuComeUp animation. This will work with the same logic but in reverse. For instance, instead of lowering the menu, you now want it to go back up. Also set the button to be disabled at the .1 second mark. We want this button disabled as soon as it is clicked. Once you set the menu to come back up, we can cover the selection buttons, UI, and audio for the game.
6. We begin with the audio. Set the volume for all four audio sources to 0 at the 0 second mark. Next, set the audio to 100 at the two second mark. This gives the scene time to lift the menu slightly and bring in the other elements and instruments.
7. Again, we must initialize the elements in the scene. Therefore, you must go through each selection button (both the button image and button itself) and UI element to set them as existing in the scene, but currently disabled and transparent, again at the 0 second mark. This is tedious, but failure to do so will cause unexpected results. Set the buttons to become visible and activated at the 5 second mark. Set the UI text to become visible at the 10 second mark. Once you have the "Great Work!" panel and Restart button, they will also be initialized here, but they will not become visible yet. *NOTE:* If you want to use background music like ours before the game starts, you can add an audio source to the MusicMainMenuDrop game object itself and control it in the animator from there like you did background music in the Space scene.
8. Once this animation is finished, you can set the methods for the start game button. Attach the PauseBeforeSpawn method with input spawnRandomFinite to the button. This is what will start the game. Next, attach the animator.Play method with input MenuComeUp to the button. This will trigger the animation.

FiniteGameEnd animation

8. If you haven't done so already, create the "Great Work!" panel and restart button in a position similar to that of our game. You can set the restart button to use the scene changer script and set it to change to

the Finite scene. Having it change to the scene we are already in is simply an easy way to "reset" the scene.

9. Create a new animation, name it FiniteGameEnd, and start recording. We must, again, initialize current elements in the scene; however, this time it will be a little different. The menu is still raised from the earlier animation; however, in order to prevent errors, we must tell the animator that it is still there. At the 0 second mark, set the location of the menu to be still raised and the button to be disabled. No other changes need to be made to the menu.
10. Similar to the last animation, but reversed, the selection buttons will start as existing at the 0 second mark and then fade out completely by the 1 second mark. The selection buttons need to be set as disabled at the 0 second mark and throughout to prevent errors.
11. Next, initialize the "Great Work!" panel and restart button (currently disabled) and their text objects to exist but be invisible. Set them to exist and the button to be enabled at the 2 second mark.
12. In order to prevent UI layering issues, there is one more thing we must do. At the 0 second mark, record the location of the middle button in the animator, and at the 1 second mark, have it moved slightly to the left. This will prevent it overlapping the Restart button so that the restart button can be clicked properly.
13. Refer to the FiniteSpawnerObject code and uncomment the lines we told you to comment in the last set of instructions.

Congratulations, you have now created the three animations that were used in this project. These animations will work with each music scene and only use the proper elements active in each scene.

Next Steps

The animations created will be used throughout the music project.

Infinite Game Scene

You can now duplicate the Finite game scene and name the new scene the Infinite Game scene. In this scene, you will have to alter the UI text to state just the scores rather than the round and remove the "Great Work!" panel and restart button. As long as you do not change the name of the UI elements, all animations should still work properly. You will not use the FiniteGameEnd animation in this scene. You can return to the previous Spawner Instructions if you need help implementing the game logic for this scene.

Gallery Game Scene

The next instructions will cover the Gallery game scene. The same animations, MusicMenuDrop and MenuComeUp, will be applied to this scene as well. We leave it as a personal challenge to you to have the proper elements be added to the animator for it to work correctly. *HINT: Use the same logic you did for the other buttons.*