Make This

Chapter 8 – Exploring the VideoPlayer Component

Integrating video into apps is a common feature. You usually have a choice of packaging the video with an app or linking out to video posted on a hosting site (such as YouTube). Both have their own unique advantages. In this exercise, we'll explore including a video within your App and using the VideoPlayer component to view it.

This tutorial teaches the following skills:

- **Using the VideoPlayer component**
- **Integrating user controls into an app**

Special Requirements of the VideoPlayer Component

The VideoPlayer component has some limitations:

- Only certain types of video files are supported – specifically videos should be in 3GPP (.3gp) or MPEG-4 (.mp4) formats.
- Only video files under 1MB in size are supported. (In addition, App Inventor apps themselves must be under 5MB in size). If your media files are too large, you may receive errors when trying to export the files for installation on other devices. You can always use video editing software such as Windows Movie Maker or Apple iMovie to make your videos smaller (cutting the length) or converting them to an appropriate format.
- VideoPlayer is not adept at accessing streaming video. If a video file (say an MP4 file) is directly accessible on a website, then the Source for VideoPlayer can be set to a URL that accesses that file. But URLs that direct users to a video player (not the actual file), such as YouTube, will not work as the Video Player source. For accessing YouTube videos, it is best to use the WebViewer component (covered in the Chapter 3 exercise).

Note: Before attempting this exercise, complete the Chapter 2 and 3 exercises to familiarize yourself with the App Inventor interface, getting your Android device connected, and starting a new project.

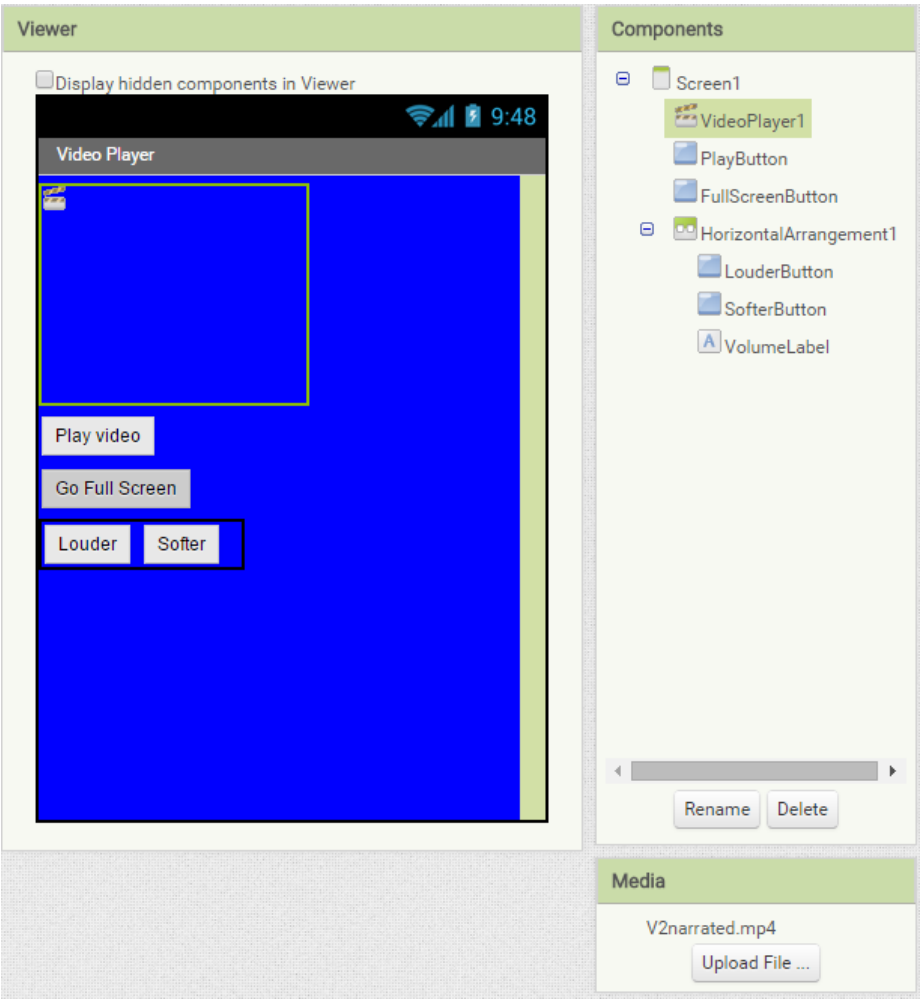
Building the VideoPlayer App

1. Navigate to http://appinventor.mit.edu/explore/. If necessary, sign in with your Google Account.
2. Start a new project named *VideoPlayerApp*. Change the **Title** of **Screen1** to *Video Player*.
3. Connect App Inventor to your Android device.

Part 1 – Constructing the Interface

You are constructing an app that will display videos. The interface in this example is simple: it is designed to play a single video, make the video display full screen, and include a button to control the volume of the sound playback.

From previous chapter exercises, you should be familiar with constructing an interface. Your layout should look like this:



Build the interface shown above by dragging out the components shown in the following table:

Component	Palette Group	Component Name	Function
VideoPlayer	Media	VideoPlayer1	Used to play a video clip
Button	User Interface	PlayButton	Starts video playback
Button	User Interface	FullScreenButton	Switched to full screen view
HorizontalArrangement	Layout	HorizontalArrangement1	Organize the sound buttons
Button	User Interface	LouderButton	To increase the

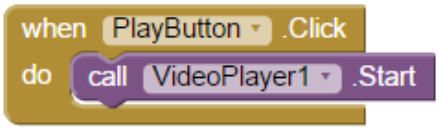
			volume of the playback
Button	User Interface	SofterButton	To decrease the volume
Label	User Interface	VolumeLabel	To display the current volume setting

In the **Media** window, click the **Upload File** button and upload the *V2narrated.mp4* file provided (or upload your own video file). **Alert:** Don't forget, the file must be under 1MB in size!

Set the properties of each component in the following ways:

- Set the **Source** of **VideoPlayer1** to the *V2narrated.mp4 file* you uploaded.
- Change the **BackgroundColor** of **Screen1** to *Blue*.
- Change the **Text** of the **PlayButton** to *Play video*.
- Change the **Text** of the **FullScreenButton** to *Go Full Screen*.
- Change the **Text** for **LouderButton** to *Louder*.
- Change the **Text** for **SofterButton** to *Softer*.
- Delete the default **Text** for **VolumeLabel** (should be blank). Change the **TextColor** to *Red*.

Part 2 – Playing a Video



1. Switch to the **Blocks** view.
2. Drag out the blocks (as indicated in the table below) and arrange them as shown above.

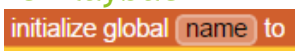
Block	Drawer	Function
when PlayButton .Click	PlayButton	Controls what happens when PlayButton is clicked
call VideoPlayer1 .Start	VideoPlayer1	Starts the VideoPlayer component
when FullScreenButton .Click	FullScreenButton	Controls what happens when FullScreenButton is clicked
set VideoPlayer1 . FullScreen to	VideoPlayer1	Sets the VideoPlayer to full screen if true
true	Logic	Returns the Boolean true

Now test your app by pressing the **Play button**. The video should play. If you touch the video, the app will display controls at the bottom of the screen to forward, rewind, or pause the video.

Pressing the **Go Full Screen Button** should display the video in full screen mode with the controls visible. Hitting the return option on your device will exit full screen mode.

Alert: The video will not appear in the emulator, although you may hear audio and see the control buttons to control the video.

Part 2 – Controlling the Volume of the Playback

1. From the **Variables** drawer, drag out a 

block.
2. Click on a blank spot in the **Viewer** window. Type 50 and this drop down will appear:

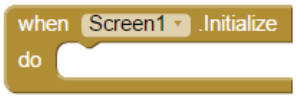
. Click the lower 50 and a 

block will appear. Drop the 

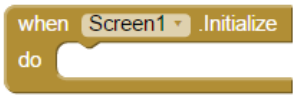
block in the open socket on the 

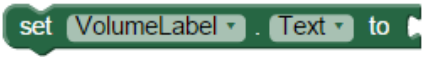
block.
3. Change the name of the **global variable** to *VideoVolume*. Your block should look like:

This block creates a global variable with an initial value of 50. We will use this variable to set the volume of the VideoPlayer. The volume settings on Android devices fall between the range of zero and 100.

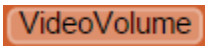
4. From the **Screen1** drawer, select the 

block. Anything placed in this block will happen as soon as the screen is initialized (after global variables are set, but before anything else happens).
5. From the **VideoPlayer1** drawer, select the 

block and insert it in the 

block.
6. From the **VolumeLabel** drawer, select the 

block and insert it underneath the 

block.
7. Point to 

in the global variable and drag out two 

blocks and insert them in the two empty sockets. Your block should look like this:

This block sets the VideoPlayer volume to the initial value of the VideoVolume variable and displays the value of the variable in the VolumeLabel field.

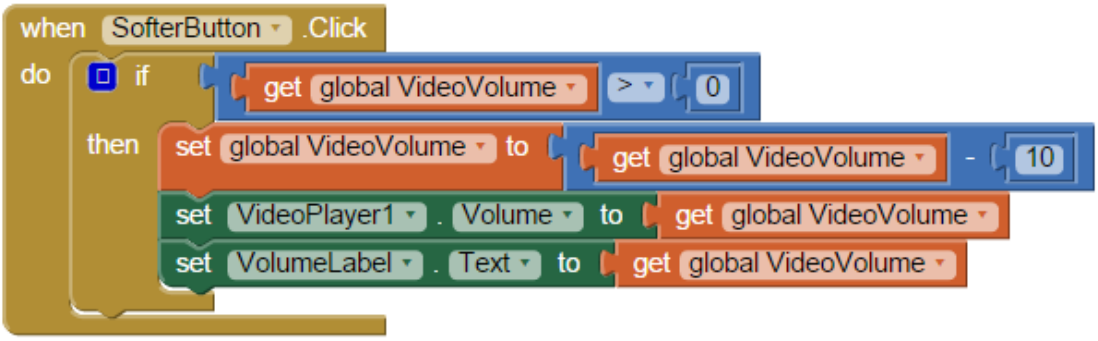
Now you should create the block for the Louder button. It will look like this:

8. Drag out the blocks (as indicated in the table below) and arrange them as shown above.

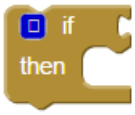
Block	Drawer	Function
when LouderButton .Click	LouderButton	Controls what happens when LouderButton is clicked
if then	Control	If a value is true, then do some statements
<	Math – Drag out a = and change it to <	Returns *true* if the first value is smaller than the second value
get global VideoVolume	Variables	Returns the value of the VideoVolume variable
100	Math	Represents maximum volume setting
set global VideoVolume to	Variables	Sets the value of the VideoVolume variable to value computed
+	Math	Adds two values
get global VideoVolume	Variables	Returns the value of the VideoVolume variable
10	Math	Increases volume level by 10
set VideoPlayer1 . Volume to	VideoPlayer1	Sets volume to given value
get global VideoVolume	Variables	Returns the value of the VideoVolume variable
set VolumeLabel . Text to	VolumeLabel	Sets the value of the VolumeLabel field
get global VideoVolume	Variables	Returns the value of the VideoVolume variable

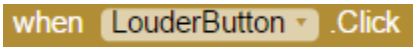
This block checks the volume level and if it is less than 100 (the maximum volume) it increases the volume level by an increment of 10. It also displays the current value of the VideoVolume variable in the VolumeLabel field.

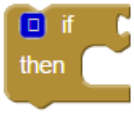
Now you should construct the block that controls the Softer button which will look like this:



9. From the SofterButton drawer, select the 

block.
10. Right click on the 

block from the 

block and select Duplicate from the dropdown menu. This should duplicate the 

block and all attached blocks. Insert the duplicate set of blocks into the 

block.
11. Make the following changes to this new set of blocks to make it look like the completed block above step 9:
    - Change the 

block to a 

block.
    - Change the value in the number block from 

to 

.
    - Replace the 

block with a 

block.

This block checks the volume level and if it is greater than 0 (the minimum volume) it decreases the volume level by an increment of 10. It also displays the current value of the VideoVolume variable in the VolumeLabel field.

Now start the video playing and click the volume buttons to adjust the volume. The value of the VolumeLabel field should change accordingly.

Alert: Again, the video will not appear in the emulator.

Extensions to This Project

1. It is never a good idea to have a program that can generate numbers out of an acceptable range. Modify the app so that the volume cannot be set higher than 100 nor lower than zero.
2. Upload another video to the app. Create a second screen with a play button that plays that video clip. Make sure you can return to Screen 1 from Screen2.
    a. Hint: Check out the Colored Dots for App Inventor 2 app documentation for an explanation of using multiple screens in App Inventor.

Resources

- AI2 Media Components: VideoPlayer
- AI2 If Blocks
- User Guide for App Inventor 2
- Guide to Understanding Blocks