



OSMF Release Samples

Walkthrough 4: UI Control Bar and Layout Management

Overview:

In this walkthrough we will see how to use the advanced layout controls offered by the OSMF components and how dynamically control the layout of interactive elements outside of the OSMF components conditionally based on the size of the content displayed by OSMF. Specifically we will use the MediaContainer advanced layout capabilities with MetaData for composite media loading and display, and see how to integrate custom graphical user interface controls to make a skinnable control bar, and make it functional by linking the user interactions with the MediaPlayer API, and adjusting the position of the control bar dynamically based off of the media content being displayed via events.

Objectives:

- Understand how to utilize MediaElements (specifically the ImageElement or other forms of Loadable elements) to create more advanced media display experiences
- Understand how to use LayoutMetaData with a MediaElement to control how it will be displayed in a MediaContainer (or MediaPlayerSprite)
- Implements manual loading of MediaElements via the use of traits (LoadTrait) when not controlled by a MediaPlayer
- Display and link user controls with the MediaPlayer to create custom control bars including basic play, pause, and real-time visual progress
- Be able to dynamically adjust the position of non OSMF instances (the control bar) relatively to the size of the media being displayed

Setup

1. Open the file WT04_IntegratingUIControls.as in the {SAMPLES_PROJECT}/src directory.

NOTE: This file has been provided as a starting point for these walkthroughs.

2. Set the class file as the application file to compile. There are two different ways of doing this depending on which program you are building your application in.

Flash Builder

Right-click the WT04_IntegratingUIControls.as file and select Set as Default Application from the context menu that appears. This will add the project to the list of compilable applications. A blue dot on the file icon indicates that the file is the default application file.

Flash Professional

Open the OSMF_SampleTemplate.fla and save it as WT04_IntegratingUIControls.fla.

Then change the document class for the file (in the Properties panel) to WT04_IntegratingUIControls.

Adding an Image Overlay (Bug)

3. Locate the comment that begins with "//Marker 1:"

Create and Setup the Image for the Bug

4. Under the comment create a local URLResource variable named bugUrlResource. Set bugUrlResource equal to a new URLResource object and pass the String 'assets/osmf_stacked.png' to the constructor.

```
//Marker 1: Add Bug Overlay
var bugUrlResource:URLResource = new URLResource( "assets/
osmf_stacked.png");
```

5. Create a local ImageElement variable named 'bug'. Set bug equal to a new ImageElement and pass the bugUrlResource variable to the constructor.

```
var bugUrlResource:URLResource = new URLResource( "assets/
osmf_stacked.png");
var bug:ImageElement = new ImageElement( bugUrlResource );
```

Manage the Layout & Positioning of the Bug with Metadata

6. Create local LayoutMetadata variable named layoutData and set it equal to a new LayoutMetadata object.

```
var bug:ImageElement = new ImageElement( bugUrlResource );
var layoutData:LayoutMetadata = new LayoutMetadata();
```

Set the 'right' property of the layoutData object equal to 10.

```
var layoutData:LayoutMetadata = new LayoutMetadata();
layoutData.right = 10;
```

Set the 'bottom' property of the layoutData object equal to 10.

```
var layoutData:LayoutMetadata = new LayoutMetadata();
layoutData.right = 10;
layoutData.bottom = 10;
```

7. Set the scaleMode property of the layoutData object equal to the static property NONE of the ScaleMode object.

```
var layoutData:LayoutMetadata = new LayoutMetadata();
```

```

layoutData.right = 10;
layoutData.bottom = 10;
layoutData.scaleMode = ScaleMode.NONE;

```

8. Add the layoutMetadata to the bug ImageElement by calling the addValue() method on the metaData property of the bug element. Pass the LAYOUT_NAMESPACE static property of the LayoutMetadata object as the first parameter and the layoutData object as the second parameter.

```

layoutData.scaleMode = ScaleMode.NONE;
bug.metadata.addValue( LayoutMetadata.LAYOUT_NAMESPACE, layoutData );

```

Load the Bug by Accessing the Appropriate Trait and Display the Bug

Since the bug ImageElement isn't going to be used with a MediaPlayer to handle the loading and such you must manage the loading manually via the LoadTrait of the ImageElement

9. Create a local LoadTrait variable named bugLoadTrait.
10. Set bugLoadTrait equal to the result of calling getTrait() on the bug object. To specify the type of trait to load pass the static LOAD property of the MediaTraitType object to the constructor making sure to cast the result as a LoadTrait.

```

bug.metadata.addValue( LayoutMetadata.LAYOUT_NAMESPACE, layoutData );
var bugLoadTrait:LoadTrait = bug.getTrait( MediaTraitType.LOAD ) as LoadTrait;

```

11. Call the load() method on the bug object to load the image.

```

var bugLoadTrait:LoadTrait = bug.getTrait( MediaTraitType.LOAD ) as LoadTrait;
bugLoadTrait.load();

```

12. Add the bug instance to the MediaContainer instance called 'container'

```

container.addMediaElement( bug );

```

13. The completed code should look like the following:

```

//Marker 1: Add Bug Overlay
var bugUrlResource:URLResource = new URLResource( "assets/osmf_stacked.png" );
var bug:ImageElement = new ImageElement( bugUrlResource );
var layoutData:LayoutMetadata = new LayoutMetadata();
layoutData.right = 10;
layoutData.bottom = 10;
layoutData.scaleMode = ScaleMode.NONE;
bug.metadata.addValue( LayoutMetadata.LAYOUT_NAMESPACE, layoutData );
var bugLoadTrait:LoadTrait = bug.getTrait( MediaTraitType.LOAD ) as

```

```
LoadTrait;  
bugLoadTrait.load();  
container.addMediaElement( bug );
```

14. Save the file and run the application. You should see the the bug image of the OSMF logo in the bottom right of the screen. Since the layout MetaData used the anchor properties of right & bottom, no matter what the size of the media played the bug will always be 10 pixels from the right and the bottom.



Implementing a Control Bar Without the OSMF Layout Containers

15. Locate the comment that starts "//Marker 2:" in the constructor.
16. Add a call to the `initControlBar()` method.

```
//Marker 2: Initialize Control Bar  
initControlBar() ;
```

17. Locate the `initControlBar()` method. Review the content of this method. This method creates and positions the controlbar elements and adds listeners for play, pause and seek events. The visual elements are being instantiated from a SWC in the `libs` folder that was created from a Flash (FLA) source file located in the `libs` directory as well.
18. Save the file and run the application. You should see the controlbar and the bug image and a video should begin to play. The controlbar isn't functional yet, we'll add in the control bar interaction and positioning next.



Adding the Control Bar Interactions

19. Locate the event handler method for the playButton, `_onPlayClick()`. Under the comment that begins `//Marker 4:`, call the `play()` method on the player object. Locate the comment that begins `//Marker3:` in the `_onPauseClick()` event handler method. If you remember from the `initControlBar()` method, this is the event handler for the `pauseButton`.

20. Under the comment call the `pause()` method on the player object.

```
protected function _onPauseClick( p_evt:MouseEvent ):void
{
    //Marker 3: Add call to pause method
    player.pause();
}
protected function _onPlayClick( p_evt:MouseEvent ):void
{
    //Marker 4: Add call to pause method
    player.play();
}
```

21. Save and run the application. The control bar should now be positioned under the video, and the pause and play buttons should also work. The progress bar however doesn't reflect the correct position and you cannot seek.

Automate the Positioning of the Control Bar to the LayoutContainer

22. To activate the automatic positioning and progress bar we will need to set up event handler methods for the `'mediaSizeChange'` and `'currentTimeChange'` events on the

player object. Locate the comment that begins "//Marker 5:".

23. Under the comment add an event listener to the player object for the static MEDIA_SIZE_CHANGE property of the DisplayObjectEvent class and provide the _onSizeChange function as the handler.

```
//Marker 5: Add MediaPlayer listeners for media size and current time  
change  
player.addEventListener( DisplayObjectEvent.MEDIA_SIZE_CHANGE,  
_onSizeChange );
```

24. Add an event listener for the static CURRENT_TIME_CHANGE property of the TimeEvent class and provide the _onProgress function as the handler.

```
//Marker 5: Add MediaPlayer listeners for media size and current time  
change  
player.addEventListener( DisplayObjectEvent.MEDIA_SIZE_CHANGE,  
_onSizeChange);  
player.addEventListener( TimeEvent.CURRENT_TIME_CHANGE, _onProgress );
```

25. Locate the comment that begins "//Marker 6:" in the _onSizeChange() method.

26. Under the comment set the y property of the controlBar object equal to the newHeight property of the p_evt object.

```
//Marker 6: Reposition appropriately when the media size changes  
controlBar.y = p_evt.newHeight;
```



Activate the Active Progress Bar Display and Seeking

27. Locate the comment that begins "//Marker 7:" in the `_onProgress()` event handler method.
28. Under the comment set the `scaleX` property of the `progressCurrent` object equal to the `time` property of the `p_evt` object divided by the `duration` property of the `player` object.

```
//Marker 7: Set the scalex of the current progress bar to the current  
time divided by the duration of the media  
progressCurrent.scaleX = p_evt.time / player.duration;
```

29. Save the file and run the application. The progress bar should now reflect the media's current position.



30. Locate the comment that begins "//Marker 8:" in the `_onSeek()` method.

Under the comment create a new local Number variable named `seekTo` and set it equal to the `duration` property of the player object multiplied by the result of dividing the `mouseX` property of the event object's target property by the width of the event objects target property.

```
//Marker 8: Call the seek method of the MediaPlayer passing it the
appropriate time determined by the click on the progress track
var seekTo:Number = player.duration * ( p_evt.target.mouseX/
p_evt.target.width );
```

31. Call the `seek()` method on the player object passing it the `seekTo` variable.

```
//Marker 8: Call the seek method of the MediaPlayer passing it
the appropriate time determined by the click on the progress track
var seekTo:Number = player.duration * ( p_evt.target.mouseX/
p_evt.target.width );
player.seek( seekTo );
```

32. Save the file and run the application. You should now be able to seek by clicking on the progress bar.