Flex in a Week, Flex 4.5

Video 4.11: Animating states with transitions

In this Day of training, you have created pages in your application using Flex states and you learned how to animate components using effects.

In this video, you will combine your knowledge of states and effects to create animated transitions between states.

You will learn how to create multi-component composite effects and to control the timing for when components appear and disappear during the animations.

Here is the starter application file.

I have located the states block and you can see that this application has two states named portalState and loginState.

As you know, the default state of any component is the state that is defined first.

However, in this case, you have set the currentState property of the Application component to the loginState, which is why it appears by default.

To create a transition, you define one or more states to transition between.

You also define the order in which you want the effects and the nature of the effects.

The effects must define the components that they target.

All of your transitions should be defined in the transitions property of the component.

This is a single block of code that contains an array of transition definitions.

Each transition is defined with a Transition instance, which declares the states to transition between using the fromState and toState properties.

The fromState property declares the name property of the state that the component should be in for the animation to start.

The toState property declares the name property of the state that the component should be in for the animation to end.

Note that both the fromState and toState property values must match for the transition to play.

Each Transition instance also defines the effects and components to animate.

The first Transition block in this code declares that the animation should only run if the application is moving from the loginState to the portalState.

The second Transition says the opposite: its defined effects should only run if the application is moving from the portalState to the loginState.

Note that you can refer to any state by using the asterisk in the fromState or toState properties of the Transition instance.

For instance this transition says that the animation should run if any state switches to the portalState.

Back in Flash Builder I am locating the states property block again and am creating the transitions property block of the application below it.

Within the transitions block, I am creating a Transition instance and assigning the fromState property a value of loginState state.

I am setting the toState property's value to the portalState state.

I will create an animation for the reverse transition later.

To define the components you want to animate and how they will move, you define the effects within the Transition instance.

The rules for using effects are the same as you learned in the last video and exercise.

Remember that you use the target property with binding syntax to reference the id property of the component that you want to animate.

You use the targets – with an s – property with the array bracket syntax, inside of the curly braces for the binding, to declare multiple components for the effect.

You can also use the Parallel and Sequence composite effects to group two or more effects that you would like to play in parallel or in sequence, respectively.

If you define the targets for a composite effect, you can still further refine the movement of each component by also defining target or targets properties for the nested effects.

In this code, you can see that the Parallel composite effect targets multiple components, but then each component also is modified by its own specific effect.

Note that the Fade effect does not have a target property defined and will, therefore, target all the components listed in the Parallel effect.

Some of the effects I have shown you have starting or ending values and some do not.

If an effect does not explicitly define a start or ending value, Flex will do its best to determine them for you based on the property's value in either the starting or ending state.

If you do not get the expected results from your transition definition, try explicitly defining starting and ending values for the properties you are animating.

When the application switches from the loginState to the portalState, I want to fade and move the Employee of the Month Panel container at the same time.

I have accidentally created the Transition block as a single tag, so I am changing it to block syntax.

Now, between the Transition tags, I am creating a Parallel effect tag block with a target property bound to the employeeOfTheMonth instance.

Within the Parallel effect tags, I am adding a Fade effect followed by a Move effect.

Once you have your transitions defined, the only thing left to do is to play it.

You do not need to apply any special events or triggers to play a transition because transitions are automatically played when the application switches between the states defined in a transition's fromState and toState properties.

When a state changes, Flex searches for and runs the Transition instance that matches those two values.

If more than one transition matches, Flex uses the first match it finds.

Back in Flash Builder, I am saving the file, running the application.

Now I’m logging in.

You can see that the Login container disappeared and that the employeeOfTheMonth Panel container moved into place, but it didn’t fade since it exists in both states.

The other Panel containers in the portalState just appeared.
I want the Employee Directory, Cafeteria Special and Monthly Events containers to also fade and animate into place.

Back in Flash Builder, I am converting the target property of the Parallel effect to the targets property.

Next, I am assigning the targets property value to the four Panel instances inside of the array syntax square brackets.

I’m adding the search, cafeteriaSpecial and monthlyEvents instances.

Note that I want all of the Panel instances to fade in except for the employeeOfTheMonth panel, which already appears in both states.

By simply leaving the Fade effect without any properties, the application will automatically determine the alpha property of each target component and extrapolate from there how to make it fade from one state to the other.

As I just mentioned, since the employeeOfTheMonth instance is visible in each state already, it won't fade but the other three will.

The employeeOfTheMonth instance also has an obvious beginning and ending position while the other three Panel instances do not.

To handle the movement of the employeeOfTheMonth instance, I am assigning the target property of the existing Move effect to the employeeOfTheMonth instance.

I want each of the other three Panel instances to move in from a different direction.

Unfortunately, they do not exist in the transition's beginning state, so I cannot explicitly position them in that state and then have Flex move them to the new location.

Instead, I am going to explicitly define a start position for each one of them in their own Move effect instances.

This figure shows the relative start position of the search, cafeteriaSpecial and monthlyEvents instances.

Below the Move effect for the employeeOfTheMonth instance, I am adding another Move effect instance with a target property value set to the search instance.

I am setting its xFrom property value to -166, which will place the container off stage to the left.

Next, I am adding a third Move effect and assigning the target property value to the cafeteriaSpecial instance with a yFrom property value set to -329 pixels.

This will start the Move effect for the cafeteriaSpecial instance off stage to the top.

Lastly, I am adding a fourth Move effect and assign its target property value to the monthlyEvents instance with an xFrom property of 833, which is off stage to the right.

I am saving the file, running the application and logging in.

You can see that the Login container disappears and the employeeOfTheMonth panel moves to the left while the other three panels fade in and move into place.

I have just handled the animation for the content in the portalState, but before that state appears, I should first handle the content in the loginState.

Instead of just having the login Panel disappear, I want to fade it out while moving the EmployeeOfTheMonth panel to the left.

To accomplish this, I must nest composite effects.

I am surrounding the Parallel effect with a Sequence effect tag block and then highlighting the Parallel effect and using the tab key to indent it.

Between the opening Sequence tag and the Parallel effect, I am creating another Parallel effect instance to handle the animation for the loginState elements.

I've indented the new Parallel block too far, so I'm highlighting it and unindenting it by typing Shift + tab.

Within new the Parallel effect tags, I am adding a Fade effect instance with the target value set to the login instance.

Under the Fade effect, I am adding a Move effect and assigning the target property value to the login instance.

I am also adding the xTo property with a value of -266.

I could have put the target property on the Parallel effect instance instead of on the nested effect.

However, I will next add more effects inside of this composite effect that target another component.

I am locating the Move effect for the employeeOfTheMonth instance and cutting it from the beginning of the second Parallel effect and pasting it at the end of the first.

Now I am removing the employeeOfTheMonth Panel instance name from the targets property value of the second Parallel effect.

The changes that I just made will fade the login panel out while the employeeOfTheMonth panel moves into its new location.

Only after that happens, will the second set of effects run, which animate the rest of the portalState panels into place.

I am saving the file, and then running the application and logging into the Employee Portal.

When I run it, you can see that the animation is not correct.

The Panel containers within the portalState are visible in their final locations before the animations occur.

Let me run that again by logging out and then logging back in.

You are seeing this problem because Flex cannot always determine when, during a transition, a component that doesn't exist in both states should be added or removed from the animation.

In my example, the login container exists in the loginState but not the portalState and the search, cafeteriaSpecial and monthlyEvents containers exist in the portalState but not the loginState.

Strange animation behavior like you just saw is a common symptom of Flex needing more information to properly animate the components.

The best practice for handling components that don't exist in both states is to explicitly state when to add or remove them during the animation.

You use the AddAction effect to define when the specified targets should be added to the animation and the RemoveAction effect to define when the targets should be removed from the animation.

Back in Flash Builder, I am placing my cursor between the two nested Parallel composite effects.

The first Parallel effect fades the login instance off screen, so I can now tell Flex to remove the login screen from the application.

I am adding a RemoveAction tag with a target property bound to the login instance.

The next Parallel effect block handles the three new Panel instances for the portalState, so the first line of the effect will be the AddAction effect to add the three panels to the application before they are animated.

Remember that this AddAction effect will target the three components defined within the Parallel effect.

Now when I save the application and run it and log in, you can see that the components all animate properly except that I want to make some slight fixes to the Employee of the Month container.
Back in Flash Builder, I am locating the first Parallel effect tag and Move effect that targets the employeeOfTheMonth instance.

I am giving it an xFrom property with a value of 298 and an xTo property with a value of 24.

Above the code I just modified, I am adding a Resize effect that also targets the employeeOfTheMonth instance.

I am explicitly directing the Panel instance to resize based on a widthFrom property value of 390 to a widthTo property value of 250.

Note that what I am doing is refining the animation on the employeeOfTheMonth instance during the first part of the animation.

As the Login panel transitions out, the Employee of the Month panel will resize as it moves to its new position in the portalState.

When I save the application, run it, and log in, you can see that the Employee of the Month panel moves and resizes on its way to its new position

in the portalState.

Now I will reverse the animation from the portalState back to the loginState.

To the opening Transition tag, I am adding the autoReverse property with a value of true.

I am saving the file, running the application, and logging in.

Notice that the transition that we have created in this video reverses when I log out, but the animation does not look quite right.

Back in Flash Builder, I am removing the autoReverse property.

The last task for this video is to create a second transition to animate the components when the user logs out of the application.

Within the transitions tag block, below the existing Transition block, I am adding another Transition block.

To the new Transition instance, I am assigning the fromState property value to the portalState state and the toState property to the loginState state.

When the user logs out of the Employee Portal application, I will fade out the search, cafeteriaSpecial and monthlyEvents panels while the employeeOfTheMonth instance resizes and moves to its proper position in the loginState.

Only after that happens will I handle the login panel.

Therefore, I must again use a Sequence composite effect with a nested Parallel effect to handle the components in the portalState and a Fade effect to handle the login panel.

I am adding the Sequence tag block with the nested Parallel tag block.

Within the Parallel effect I am adding a Fade effect and setting the targets property value to the search, cafeteriaSpecial and monthlyEvents panels.

Next I am adding a Resize effect to target the employeeOfTheMonth instance and reversing the width from 250 pixels to 390 pixels.

Lastly, I am adding a Move effect that also targets the employeeOfTheMonth panel instance.

It has an xFrom property value of 24 and an xTo property value to 434.

This will move the employeeOfTheMonth instance to its proper position in the loginState at the same time that it resizes the panel and fades out the other three panels.

Below the Parallel composite effect, but still inside the Sequence effect, I am creating a Fade effect that targets the login instance.

This will fade the login instance into view.

To recap, in the previous steps, I have defined the animations that should occur in sequence.

First of all the panels in the portalState will fade, except the employeeOfTheMonth panel, which will move and resize into its new position.

Then the login panel will fade into view in the loginState.

Lastly I need to define when the panels that do not exist in both states need to be added or removed during the transition.

Between the Parallel composite effect and the Fade effect, I am adding the RemoveAction tag and assigning the targets property value to the search, cafeteriaSpecial and monthlyEvents components.

Once these Panel instances are animated and fade out, they can be removed from the application.

Next, I am adding an AddAction tag and assigning the target property value to the login Panel container.

Since the login instance doesn't exist in the portalState, I must explicitly add it before I use it in the Fade effect.

I am saving the file and running the application.

After I log in, you can see that the application animates into the portalState properly.

Now when I click the Logout button, you can see that the employeeOfTheMonth Panel container moves and resizes while the other panels fade away.

Only after that does the login Panel container fade into view.

For your next step, work through the exercise titled “Applying transitions to view states”.