



Papervision3D Effects for Flex



Tink

<http://www.tink.ws>

<http://www.tink.ws/blog>

Contents

- 1. Effects overview**
- 2. How to use effects**
- 3. How effects work**
- 4. EffectPV3DManager**
- 5. EffectPV3D & EffectPV3DInstance**
- 6. CachePV3DEffect and CachePV3DEffectInstance**
- 7. Creating your own EffectPV3D**

Effects Overview

Description and uses

Effects included

Using Effects

Using Effects with Containers

An effect is a visible or audible change to the component occurring over a period of time, measured in milliseconds. Examples of effects are fading, resizing, or moving a component.

Effects are paired with triggers. A trigger is an action such as a mouse click, a component getting focus, or a component becoming visible. Flex uses CSS to define triggers that you reference as a property in an MXML tag or in an ActionScript function. The CSS property names of triggers use the following convention:

`triggerEffect`
where `trigger` is the trigger name. For example, the `show` trigger occurs when a component becomes visible; the CSS property name for the `show` trigger is `showEffect`. The `hide` trigger occurs when a component becomes invisible; its CSS property name is `hideEffect`.

Triggers are not ActionScript events. For example, a `Button` has both a `mouseDownEffect` trigger and a `mouseDown` event. The trigger initiates a Flex effect. The event calls an ActionScript function or object method.

[AnimateProperty](#) - Animates a numeric property of a component, such as height, width, `scaleX`, or `scaleY`.

[Blur](#)

[Dissolve](#) - Modifies the alpha property of an overlay to gradually have to target component appear or disappear.

[Fade](#)

[Glow](#)

[Iris](#) - Animates the effect target by expanding or contracting a rectangular mask centered on the target. The effect can either grow the mask from the center of the target to expose the target, or contract it toward the target center to obscure the component.

[Move](#)

[Pause](#) - Does nothing for a specified period of time. This effect is useful when you need to composite effects.

[Resize](#)

[Rotate](#)

[SoundEffect](#) - Plays an MP3 audio file. For example, you could play a sound when a user clicks a `Button` control. This effect lets you repeat the sound, select the source file, and control the volume and pan.

[WipeLeft](#)

[WipeRight](#)

[WipeUp](#)

[WipeDown](#) - Defines a bar Wipe effect. The before or after state of the component must be invisible.

[Zoom](#) - Zooms a component in or out from its center point.

`Parallel` - The `Parallel` effect plays multiple child effects at the same time.

`Sequence` - The `Sequence` effect plays multiple child effects one after the other, in the order in which they are added.

How Effects Work

Effect

EffectInstance

EffectManager

`Effect.play()` creates an `EffectInstance` (or correct instance required using `createInstance()` method) for each target.

Invokes `startEffect()` on each instance.

The `Effect` instance is the class that actually carries out the effect.

Most visual effects extend `TweenEffect`. This tweens the properties or values and applies the values to the properties.

inside `startEffect()` the instance is added to the effect manager. `EffectManager.effectStarted(this);`

For `TweenEffects`, when a tween is created, the effect instance is passed as a listener.

On completion the `Tween` then invokes `onTweenEnd()` which in turn invokes `finishRepeat()`.

`finishRepeat()` checks to see if the effect needs to repeat. If so `play()` is invoked, if not `finishEffect()` is invoked.

Inside `finishEffect()` the effect is removed from the `EffectManager`. `EffectManager.effectFinished(this);`

Any effect can be stopped by invoking `end()` which in turn invokes `finishEffect()`.

Stores a reference to any effects taking place.

Manages caching targets as `Bitmaps` where effects take place.

Enables you to end all effects for a particular component with static method `endEffectsForTarget(target:IUIComponent):void`

PV3DEffectManager

Singleton

Top level in display list

The code

Future improvements

Create by the first EffectPV3D played.

```
initialize();
```

Added as a child of Application.application.systemManager.addChild(instance);

Appears on top of all model content.

Talk about the option of specifying where the effect is added.

```
location ( top, modal, parent )
```

Instead of effects getting added to EffectPV3DManager and it being a display item, effects will just get added themselves at the point specified.

This would mean making _display a UIComponent is EffectPV3DInstance.

PV3DEffect & PV3DEffectInstance

Core classes to extend Properties

- **hideMode:String (blendMode)**
- **transparent:Boolean (false)**

Methods

- **createBitmapDatas():void**
- **createMaterials():void**
- **createDisplayObject3Ds():void**
- **createPropertiesToTween():void**

1. Creates Sprite to display 3D and positions it.
2. Create Scene3D and adds display.
3. Creates Camera3D.
4. Adds a new DisplayObject3D to scene and stores returned value in `_root3D`
5. Adds the effectInstance to the Effectmanager

Manages hiding the target when the effect starts.

Need to hide actually Flex component, but can't use visible as that could fire an effect.

hideMethod - default blendMode.

1. none - the original item isn't hidden.
2. alpha - self explanatory (doesn't work unless you embed fonts)
3. blendMode - uses BlendMode.ERASE. Works on none embedded fonts, but will remove any prev blendModes

transparent - default false.

createBitmapDatas():void

Create BitmapData objects for use with materials.
Override this method if you want more complex BitmapData objects.
The default creates a single BitmapData that is a snapshot of the target UIComponent.

createMaterials():void

Create materials for DisplayObject3D objects using BitmapData objects.
Override this method if you want complex materials.
The default creates a single BitmapMaterial that is a snapshot of the target UIComponent.

createDisplayObject3Ds():void

Create DisplayObject3D objects.
Create the DisplayObject3D objects using the array of materials,
then add the DisplayObject3D objects to `_root3D`.

default throws an error

createPropertiesToTween():void

Create an array of values to tween from and an Array or corresponding values to Tween to.

default throws an error

Standard PV3DEffects

Flip

- *constrain:Boolean (false)*
- *direction:String (left)*
- *type:String (show)*

FlipInstance

- *onTweenUpdate():void*
- *createDisplayObject3Ds():void*
- *createPropertiesToTween():void*

Composite PV3DEffects

Overview

Cache

- **cache:Boolean (true) - read only**
- **materials:Array - read only**

CacheInstance

- **play():void**
- **initialize():void**
- **createDisplayObject3Ds():void**

Containers such a ViewStack require 2 single effects (hide and show)
Composite effects skip the hide and apply the whole effects in the show (imagine a cube)

CacheEffect enables composite effects
Used as the hideEffect
It has cache set to true.
Creates materials and bitmapDatas but no DisplayObject3D.
Bitmap is display and effect last for a single frame.
When complete it is stored in the Manager (all other effects get cleaned up).
Storage is based on the targets parent (i.e. the container). Only 1 effect can be stored per container.

Users the previous effects materials from the Manager. If a CacheEffect, or an effect that extends CacheEffect is not used for the hideMethod, it will throw an error

Composite EffectPV3Ds

Cube

- **constrain: Boolean (false)**
- **direction: String (left)**

CubeInstance

- **onTweenUpdate(value:Object):void**
- **createBitmapDatas():void**
- **createMaterials():void**
- **createDisplayObject3Ds():void**
- **createPropertiesToTween():void**

constrain - moves Cube on z axis to stop it growing in size due to perspective

direction - rotation direction

play - starts tweening properties

tweenUpdate - to change properties

createBitmapDatas - Creates a mirrored snapshot for the bottom of the cube

createMaterials - gets material from previous effect (Cache).
creates 3 materials (prev, snapshot + mirrored snapshot for bottom) and create a MaterialList for cube.

createDisplayObject3Ds - creates Cube and sets start and end values

createDisplayObject3Ds - creates the properties to tween from and to.

Creating a Zoom Effect

Will zoom and rotate on any axis

Standard Effect

Clone of Rotate & RotateInstance + zoom

QUESTIONS ??

Tink

<http://www.tink.ws>

<http://www.tink.ws/blog>