Papervision3D 2.0 Alpha-Great While

翻译：FireYang　　来源：http://pv3d.org

持续翻译中……待续……

内容目录

0. 安装环境（略）

Flash Develop + Flex 2 SDK

1. 下载 Papervision3D 2.0 Alpha-Great While

译自：http://pv3d.org/2007/12/12/1-downloading-papervision3d-20-alpha-great-white/

这篇文章介绍你使用 Subversion(SVN)从 Google Code 下载 Papervision3D 2.0 Alpha。SVN 通常使用到经常更新代码的项目（用于版本管理），同样 Papervision3D 也允许开发人员更新主机上当前版本到你的文件夹上。

前提条件：

无

步骤：

1. 下载和安装 Tortoise SVN。
2. 在你的硬盘上创建和命名一个新的文件夹来保存 Papervision3d。我命名为”Papervision3D”

3. 右击文件夹选择“SVN Checkout”
4. 在存储文件夹的 URL 中输入“http://papervision3d.googlecode.com/svn/”
5. 单击 OK 并等待下载（可能需要一段时间，因为大约有 35MB）
6. 当你看到“Completed At revision: xxx”，Papervision 就下载完毕了（在翻译这篇文章的时候，当前版本已经是 424

注意：

- 当你在 Flash，Flex 或者 FlashDevelop 创建一个新的项目，你要把 papervision 的分支版本的文件夹指定为库类路径。
- Papervision 2.0 alpha 命名为“Great White”。你可以在“trunk\branches\GreatWhite\src”路径中找到“Great White”

2. 创建 pv3d 2.0 框架类

译自：http://pv3d.org/2007/12/14/3-creating-a-pv3d-20-skeleton-class/

创建一个 pv3d 项目是非常简单的。在项目中创建一个 viewport 作为容器。这个 viewport 将控制设置宽度，高度，裁剪，消除和相互作用（有许多不同类型的摄影机[cameras]，这将涵盖在以后的教程），然后 renderer 将所有的对象一起渲染到场景里，并可以根据你的喜好来设置它们。

前提条件：

配置项目

步骤：

1. 继承自 Sprite(详情查看相关文档)

```
class Main extends Sprite;
```

2. 声明所有的必要 pv3d 变量

```
private var viewport:Viewport3D;
private var scene:Scene3D;
private var camera:FreeCamera3D;
private var renderer:BasicRenderEngine;
```

3. 设置你的函数

```
private function init():void
{
        //这里初始化你工程里必须的一些函数
        //例如: initPapervision
 }
private function initPapervision():void
{
//所有初始化Papervision的东西都放在这里
```

```
    }
```

4. 在 initPapervision 初始化你定义的 pv3d 变量

```
    private function initPapervision():void
    {
            viewport = new Viewport3D();
            //如果你忘记把viewport “addChild”，
            //你的工程依然执行，但是你将看不到任何东西！
            addChild( viewport );
            scene = new Scene3D();
            camera = new FreeCamera3D();
            renderer = new BasicRenderEngine();
    }
```

5. 渲染场景

```
    //这个方法在以后的教程中将放置到ENTER_FRAME事件里
    renderer.renderScene( scene, camera, viewport );
```

6. 调用函数依照固定的顺序：主构造函数(Constructor)调用 init，init 调用 initPapervision
7. 点击“Ctrl+Enter”测试你的影片（这里是在 FlashDevelop 中的快捷键）

最终代码：

```
package
{
	import flash.display.Sprite;
	import org.papervision3d.cameras.FreeCamera3D;
	import org.papervision3d.render.BasicRenderEngine;
	import org.papervision3d.scenes.Scene3D;
	import org.papervision3d.view.Viewport3D;

	public class Main extends Sprite
	{
		private var viewport:Viewport3D;
		private var scene:Scene3D;
		private var camera:FreeCamera3D;
		private var renderer:BasicRenderEngine;

		public function Main()
		{
			init();
		}

		private function init():void
		{
			initPapervision();
		}

		private function initPapervision():void
		{
			viewport = new Viewport3D();
			addChild( viewport );
			scene = new Scene3D();
			camera = new FreeCamera3D();
			renderer = new BasicRenderEngine();
			renderer.renderScene( scene, camera, viewport );
		}
	}
}
```

3. 创建你的第一个 pv3d 中的 3d 对象（A.K.A DisplayObject3D）

A.K.A 原来是 Also Known As 的缩写，可能是众所周知的意思，所以这里就不讲 DisplayObject3D 了，^_^！

译自：http://pv3d.org/2007/12/14/4-creating-your-first-pv3d-3d-object-aka-displayobject3d/

类似在工程创建其他对象一样创建 3d 对象然后添加到场景里。在对象被创建后不要忘记去渲染(render)到场景(scene)上！

前提条件：

创建 pv3d 2.0 框架类

步骤：

1. 声明你的 3d 对象。此示例中，我们将使用“Plane”基本图形。（我们将在以后的教程中探讨更高级 3d 对象）

```
private var plane:Plane;
```

2. 用 new 函数，实例化你的 3d 对象

```
plane = new Plane();
```

3. 在你的场景中添加你新增的 3d 对象

```
scene.addChild( plane );
```

4. 3d 对象被添加以后，渲染你的场景

```
renderer.renderScene(scene,camera,viewport);
```

5. 在 init 函数里轻松的调用你的 3d 对象生成函数（在你 papervision 的安装函数之后）

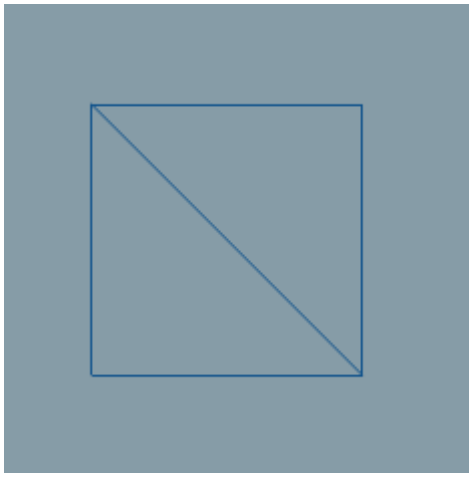
```
private function init():void
{
        initPapervision();
        initObjects();
}
```

最终代码：

```
package
{
        import flash.display.Sprite;
        import org.papervision3d.cameras.FreeCamera3D;
        import org.papervision3d.objects.primitives.Plane;
        import org.papervision3d.render.BasicRenderEngine;
        import org.papervision3d.scenes.Scene3D;
        import org.papervision3d.view.Viewport3D;

        public class Main extends Sprite
        {

        private var viewport:Viewport3D;
        private var scene:Scene3D;
        private var camera:FreeCamera3D;

        private var renderer:BasicRenderEngine;

        private var plane:Plane;

                public function Main()
```

```
		{
			init();
		}

		private function init():void
		{
			initPapervision();
			initObjects();
		}

		private function initPapervision():void
		{
			viewport = new Viewport3D();
			addChild( viewport );

			scene = new Scene3D();
			camera = new FreeCamera3D();

			renderer = new BasicRenderEngine();
		}

		private function initObjects():void
		{
			plane = new Plane();
			scene.addChild( plane );

			renderer.renderScene( scene, camera, viewport );
		}
	}
}
```

效果：



4. 在 pv3d 3D 对象应用材质（皮肤技术／纹理技术）

译自：http://pv3d.org/2007/12/18/5-applying-materials-to-a-pv3d-3d-object-skinningtexturing/

在此前的教程中，你创建的 3d 对象都被默认的设置了“WireFrameMaterial”。还有很多不同类型的材质类（materials）可以应用到你的 3d 对象上。这篇文章将向你介绍如何让你把一个简单的“ColorMaterial”材质应用到你的 3d 对象上，这意味这你的 3d 对象实体将拥有“ColorMaterial”的颜色。你将在这里学习到将材质应用到 3d 对象的原因可以举一反三到所有的材质和所有的 3d 对象。

前提条件：

3.创建你的第一个 pv3d 中的 3d 对象

步骤：

1. 声明你的材质

```
private var material1:ColorMaterial1;
```

2. 在 initMaterials 函数里初始化你的材质。在这里，ColorMaterial 接受 16 进制的颜色（Wikipedia on web colors)作为第一个参数。

```
private function initMaterials():void
{
        material1 = new ColorMaterial1( 0xcc0000 );
}
```

3. 通过第一个参数将材质应用到你的 3d 对象

```
plane = new Plane( material1 );
```

4. 调用“initMaterials” 函数要在“initPapervision”之后和“initObjects”之前

```
private function init():void
{
        initPapervision();
        //调用initMaterials在initPapervision之后，但在initObjects之前
        initMaterials();
        initObjects();
}
```

最终代码：

```
package
{
        import flash.display.Sprite;
        import org.papervision3d.cameras.FreeCamera3D;
        import org.papervision3d.materials.ColorMaterial1;
        import org.papervision3d.objects.primitives.Plane;
        import org.papervision3d.render.BasicRenderEngine;
        import org.papervision3d.scenes.Scene3D;
        import org.papervision3d.view.Viewport3D;
```

```
public class Main extends Sprite
{

private var viewport:Viewport3D;
private var scene:Scene3D;
private var camera:FreeCamera3D;

private var renderer:BasicRenderEngine;

private var material:ColorMaterial;

private var plane:Plane;

        public function Main()
        {
                init();
        }

        private function init():void
        {
                initPapervision();
                initMaterials();
                initObjects();
        }

        private function initPapervision():void
        {
                viewport = new Viewport3D();
                addChild( viewport );

                scene = new Scene3D();
                camera = new FreeCamera3D();

                renderer = new BasicRenderEngine();
        }

        private function initMaterials():void
        {
                material = new ColorMaterial( 0xcc0000 );
        }

        private function initObjects():void
        {
                plane = new Plane( material );
                scene.addChild( plane );

                renderer.renderScene( scene, camera, viewport );
```

```
            }
        }
}
```

5. 3d 中的基本运动

译自：http://pv3d.org/2008/01/01/6-basic-movement-in-3d/

如果你知道在 AS3 中如何添加一个 Event.ENTER_FRAME 监听器，那么在 3d 中移动对象简直是小菜一碟（a piece of cake，估计是这个意思^o^）

前提条件：

4.在 pv3d 3D 对象应用材质

学习工具：

3d 对象运动管理器

步骤：

1. 当 plane 渲染到场景上时，创建 Event.ENTER_FRAME 处理器（也可以叫句柄吧）使用一个运动方法。看 3d 对象运动管理器中方法和属性是如何工作的。

```
        private var material:ColorMaterial;
```

2. 创建一个“initListeners()”函数并添加到 Event.ENTER_FRAME 监听器里。

```
        private function initListeners():void
        {
                addEventListener( Event.ENTER_FRAME, onEnterFrame );
        }
```

3. 在你的步骤中调用 initListeners()函数

```
        private function init():void
        {
                initPapervision();
                initMaterials();
                initObjects();
                initListeners();//<--------
        }
```

最终代码：

```
package
{
        import flash.display.Sprite;
        import flash.events.Event;
        import org.papervision3d.cameras.FreeCamera3D;
        import org.papervision3d.materials.ColorMaterial;
        import org.papervision3d.objects.primitives.Plane;
```

```
import org.papervision3d.render.BasicRenderEngine;
import org.papervision3d.scenes.Scene3D;
import org.papervision3d.view.Viewport3D;

public class Main extends Sprite
{

private var viewport:Viewport3D;
private var scene:Scene3D;
private var camera:FreeCamera3D;

private var renderer:BasicRenderEngine;

private var material1:ColorMaterial1;

private var plane:Plane;

        public function Main()
        {
                init();
        }

        private function init():void
        {
                initPapervision();
                initMaterials();
                initObjects();
                initListeners();
        }

        private function initPapervision():void
        {
                viewport = new Viewport3D();
                addChild( viewport );

                scene = new Scene3D();
                camera = new FreeCamera3D();

                renderer = new BasicRenderEngine();
        }

        private function initMaterials():void
        {
                material1 = new ColorMaterial1( 0xcc0000 );
        }

        private function initObjects():void
        {
```

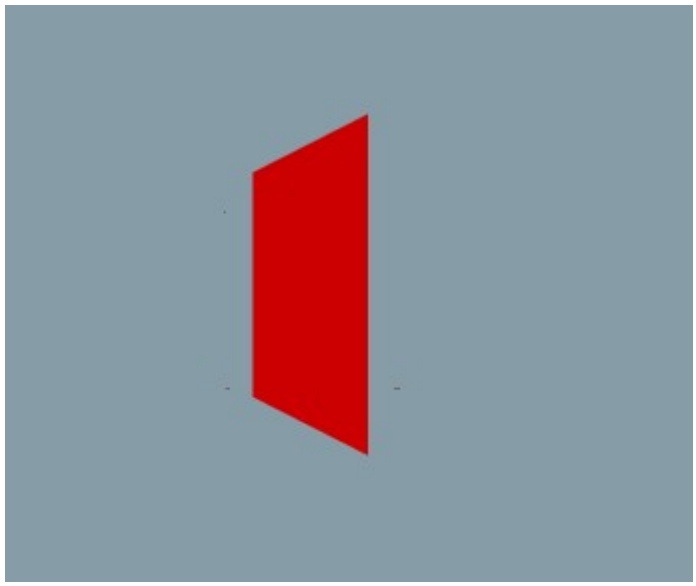
```
                plane = new Plane( material1 );
                scene.addChild( plane );
        }

        private function initListeners():void
        {
                addEventListener( Event.ENTER_FRAME, onEnterFrame );
        }

        private function onEnterFrame( e:Event ):void
        {
                plane.yaw( 2 ); // or try plane.rotationY += 2;
                renderer.renderScene( scene, camera, viewport );
        }
    }
}
```

注意：

- 如果你的面板已经旋转了，你会发现你的面板的背部什么都没有（背部的时候你看不到任何东西）。可以设置你的材质的背部，只要设置你的材质的 doubleSided 属性为“true”。

```
    private function initMaterials():void
    {
            material1 = new ColorMaterial( 0xcc0000 );
            material1.doubleSided = true;
    }
```

- 虽然你现在看到“initListeners()”函数里面没什么东西，但是以后的所有交互监听器全放在这里。
- 现在你已经掌握了 pv3d 的主要的过程：设置 pv3d,创建材质，创建对象（附加了材质），创建监听器和可视对象（render 不知道怎么去翻译好）。



6. 基本交互

译自：http://pv3d.org/2008/01/06/7-basic-interactivity/

开始 pv3d 时，你必须让你的视口（viewport）和材质(material)的 interactivity（交互）属性值为 enable，这样你才能让各种 click 事件能被监听。看看它是如何实现的：

前提条件：

#5.3d 中的基本运动|outline

步骤：

1. 改变 **viewport** 的 **interactivity** 属性（**AKA** InteactiveSceneManager 的一个实例）

   第一步可以有两种方法实现。第一种就是新建 viewport 的时候直接在第 4 个参数传递一个 true 进去。这些参数分别是：width(宽度),height(高度),autoScaleToStage(是否有滚动条),interactive(是否可以交互)。代码便是：

```
//Viewport3D( width, height, autoScaleToStage, interactive )
viewport = new Viewport3D( 640, 480, false, true );
```

   第二种方法跟直接传递一个参数给 viewport 相比，代码会多一点，主要是新建一个用于交互的 InteactiveSceneManager 对象（第一种这些是自动生成的）。

```
viewport = new Viewport3D();
viewport.interactive = true;
viewport.interactiveSceneManager = new InteractiveSceneManager( viewport );
```

2. 允许你的材质接收事件：

```
material.interactive = true;
```

3. 创建一个函数当你事件在触发的时候发生（press,release,click,等等）。下面 InteractiveScene3DEvent.OBJECT_PRESS 监听器的实例就是让它向下旋转 20 度：

```
private function objectPress( e:InteractiveScene3DEvent ):void
{
        plane.pitch(20);
}
```

4. 给 plane 添加一个监听（或其他的 DisplayObject3D 对象）：

```
plane.addEventListener( InteractiveScene3DEvent.OBJECT_PRESS, objectPress );
```

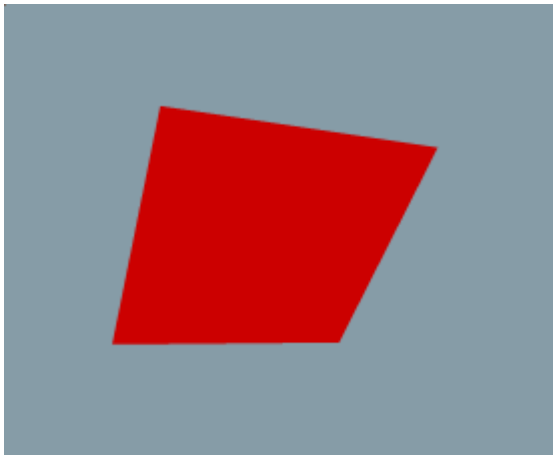
最终代码：

```
package
{

import flash.display.Sprite;
import flash.events.Event;
```

```
import org.papervision3d.cameras.FreeCamera3D;
import org.papervision3d.events.InteractiveScene3DEvent;
import org.papervision3d.materials.ColorMaterial;
import org.papervision3d.objects.primitives.Plane;
import org.papervision3d.render.BasicRenderEngine;
import org.papervision3d.scenes.Scene3D;
import org.papervision3d.view.Viewport3D;

public class Main extends Sprite
{

private var viewport:Viewport3D;
private var scene:Scene3D;
private var camera:FreeCamera3D;

private var renderer:BasicRenderEngine;

private var material:ColorMaterial;

private var plane:Plane;

        public function Main()
        {
                init();
        }

        private function init():void
        {
                initPapervision();
                initMaterials();
                initObjects();
                initListeners();
        }

        private function initPapervision():void
        {
                viewport = new Viewport3D( 640, 480, false, true );
                addChild( viewport );

                scene = new Scene3D();
                camera = new FreeCamera3D();

                renderer = new BasicRenderEngine();
        }

        private function initMaterials():void
        {
                material = new ColorMaterial( 0xcc0000 );
```

```
                material1.interactive = true;
                //material1.doubleSided is not necessary for interactivity
                material1.doubleSided = true;
        }

        private function initObjects():void
        {
                plane = new Plane( material1 );
                scene.addChild( plane );
        }

        private function initListeners():void
        {
                addEventListener( Event.ENTER_FRAME, onEnterFrame );
                plane.addEventListener( InteractiveScene3DEvent.OBJECT_PRESS,
onPress );
        }

        private function onPress( e:InteractiveScene3DEvent ):void
        {
                plane.pitch( 20 );
        }

        private function onEnterFrame( e:Event ):void
        {
                plane.yaw( 2 );
                renderer.renderScene( scene, camera, viewport );
        }
    }
}
```

注意：这只是个简单的交互示例，以后会更件深入的介绍



7. 基本模板和功能修饰解释

译自：http://pv3d.org/2008/01/12/template-basic-template-and-function-modifier-explanations/

前提条件：

0.安装环境（略）

基本模板：

曾经自言自语（原作者说的），“约翰，我不想在 Papervision3d 中总是为了些基础的东西写这么多代码”。其实我已经为你写了个好东东了。

一个 BasicTemplate 的子类，把 Papervision3d 的设置单独分离到一个类文档里。看看下面的代码如何将一个 plane 添加到场景（scene）里去：

```
package
{
	import org.papervision3d.objects.primitives.Plane;
	import org.pv3d.tutorials.BasicTemplate;

	public class Main extends BasicTemplate
	{
		private var plane:Plane;

		override protected function createObjects():void{
			plane = new Plane();
			scene.addChild( plane );
		}
	}
}
```

现在是不是很简单了？你以前是否没有看过“override”，在 Basic Template 里面有“override”标记的重写函数。因此当你更改函数后，模板依然会替换调用这个函数。这就是神奇的模板了^_^。

函数功能修饰符：

在以后的教程中我将会使用“private”，“protected”和“public”，因此我要先在这里做一下解释。首先，对你来说“protected”可能是全新的，“protected”是介于“private”和“public”之间的，例如，“保护”一个函数后你将不允许如此操作：

```
	//NO NO NO NO NO 你不能实例化一个类并调用他受保护的函数
	var p:BasicTemplate = new BasicTemplate();
	p.createObjects();
	//^^^ 保护的函数是不被执行的
```

但是“保护”的函数是允许你像上面那样在一个子类中覆盖掉，也允许它子类中的非覆盖函数去调用它（“私有的”是限制你子类中使用的）。你可以安如下的标准

去限制类中函数的使用：

- private
  - 同一个类中调用
- protected
  - 同一个类中调用
  - 子类调用，或者 override 调用
- public
  - 同一个类中调用
  - 子类调用，或者 override 调用
  - 类的实例直接调用

不管怎样，这样的解释足够了……

结尾

下载模板和示例

提供的示例中包含了更多的方法和属性变量。这个 BasicTemplate 虽然不是完美的，但它还是能很好的运行的，在以后的教程中我们就可以只关注材质，交互等方面了。不要认为我（作者）写这个仅仅是为了教程的示例而觉得它有局限性，其实它可以用到你想使用的任何地方的！

附录：

一些示例(有源代码的)

- *FreeCamera3D控件*
- *InteractiveScene3DEvent OBJECT_PRESS（一个交互事件类，交互必须的）*
- *3d Object Movement Explorer（3d对象运动探测器）*
- *3d Camera Movement Explorer（3d摄像机探测器）*
- *Interactive Cube（交互立方体，根据鼠标点击更改颜色）*
- *Basic Pixel3D（基本3D像素，满天星斗的感觉）*
- *Effects（3D特效，^o^ cool!）*
- *PhongMaterial(另一种特效，不知道怎么称呼)*
- *Reflection（3D的倒影）*
- *Shooter（射击效果）*