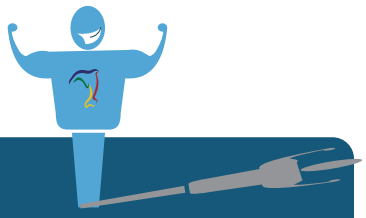
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

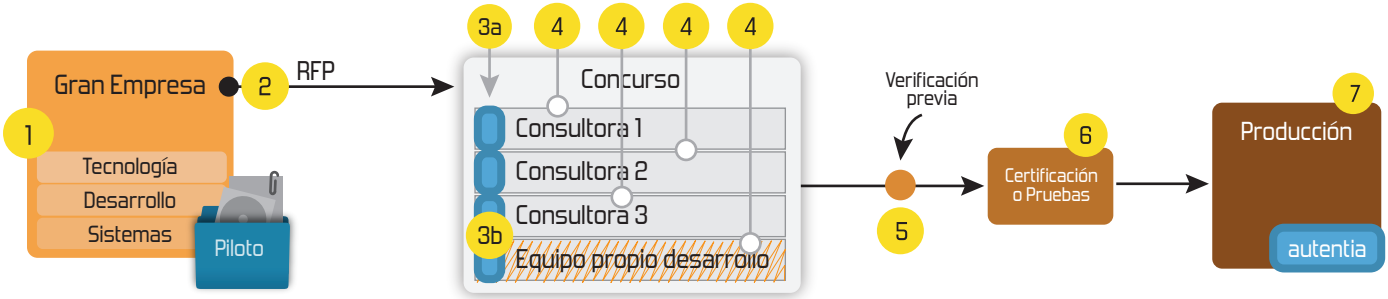
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Ultimas Noticias

» Lanzamiento del nuevo Web de Autentia
» Historia de la Informática. Capítulo 70. 1993
» Historia de la Informática. Capítulo 69. 1992
» Si se pregunta ¿Qué ofrece este Web?
» Autentia en la Sun Open Communities Forum
» Autentia cumple 6 años
» Comentario del libro: El economista naturalista de Robert Frank
» Contratos ágiles: Vendiendo Scrum a tus clientes.
» Alimarket.es: Primera aplicación pública del framework wuija by Autentia

+Noticias Destacadas

» Lanzamiento del nuevo Web de Autentia
» Contratos ágiles: Vendiendo Scrum a tus clientes.
» Quinta charla Autentia + Proyectalis + Agile Spain: Contratos ágiles: Vendiendo Scrum a tus clientes
» Lo mejor de esta semana: Curso de Scrum con Ángel Medinilla

+Comentarios Cómic

+Enlaces

Tutorial desarrollado por

Javier Ceballos Fernández

Consultor tecnológico de desarrollo de proyectos informáticos.

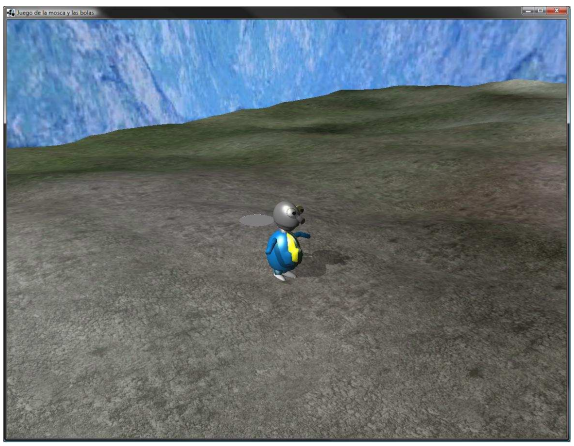
Ingeniero en Informática por la Universidad de Alcalá de Henares.

Puedes encontrarme en Autentia

Somos expertos en Java/J2EE

Fecha de creación del tutorial: 2009-07-30

Funciones esenciales para crear un juego.



En tutoriales anteriores se ha mostrado como podíamos crear un juego a partir de uno existente, en este tutorial intentaremos facilitaros las cosas, para ello os enseñaremos algunas funciones de utilidad (crear escenarios propios, cargar personaje propio y como dotarle de movimiento, etc.).

Empezaremos por crear el espacio donde se producirán todas las acciones del juego, para ello nos podemos crear una clase que sea derivada de Skybox, en ella tendremos el constructor que lo usaremos para montar las caras de nuestro Skybox y otra función que nos cárgara las texturas.

```
view plain  print  ?
01. public class MiSkyBox extends Skybox {
02.
03.     public MiSkyBox() {
04.         super("universo", 100, 100, 100);
05.         Texture north =     cargaTextura("north.jpg");
06.         Texture south =     cargaTextura("south.jpg");
07.         Texture east =      cargaTextura("east.jpg");
08.         Texture west =      cargaTextura("west.jpg");
09.         Texture up =        cargaTextura("top.jpg");
10.         Texture down =      cargaTextura("bottom.jpg");
11.
12.         setTexture(Skybox.Face.North, north);
13.         setTexture(Skybox.Face.West, west);
14.         setTexture(Skybox.Face.South, south);
15.         setTexture(Skybox.Face.East, east);
16.         setTexture(Skybox.Face.Up, up);
17.         setTexture(Skybox.Face.Down, down);
18.         Callable<Object> preload = new Callable<Object>() {
19.
20.             public Object call() throws Exception {
21.                 preloadTextures();
22.                 return null;
23.             }
24.         };
25.         GameTaskQueueManager.getManager().getQueue(GameTaskQueue.RENDER).enqueue(preload);
26.         updateRenderState();
27.     }
28.
29.     Texture cargaTextura(String nombre)
30.     {
31.         return TextureManager.loadTexture(
32.                         mostermosca.class.getClassLoader().getResource(
33.                         nombre),
34.                         Texture.MinificationFilter.BilinearNoMipMaps,
35.                                 Texture.MagnificationFilter.Bilinear);
36.
37.     }
38.
39. }
```

Con esta clase siempre podremos crear cualquier Skybox a nuestro gusto y de este modo tendremos nuestro mundo creado.

En el update() del juego tendremos que añadir este código.

```
view plain  print  ?
01. skybox.setLocalTranslation(cam.getLocation());
02. skybox.updateGeometricState(0, true);
```

Con esto haremos que el Skybox este siempre donde la cámara de ese modo nunca nos saldremos de él.

Ahora os mostrare como creamos el suelo por el que queremos que se mueva nuestro jugador.

```
view plain  print  ?
01. private void buildTerrain() {
02.
03.
04.         MidPointHeightMap heightMap = new MidPointHeightMap(64, 1f);
05.         Vector3f terrainScale = new Vector3f(4, 0.0575f, 4);
06.         tb = new TerrainBlock("Terrain", heightMap.getSize(), terrainScale,
07.                 heightMap.getHeightMap(), new Vector3f(0, 0, 0));
08.
09.         tb.setModelBound(new BoundingBox());
10.         tb.updateModelBound();
11.         ProceduralTextureGenerator pt = new ProceduralTextureGenerator(
12.                 heightMap);
13.         pt.addTexture(new ImageIcon(TestTerrain.class.getClassLoader().getResource("adictosaltrabajo/texturas/grassb.png")), -128, 0, 128);
14.         pt.addTexture(new ImageIcon(TestTerrain.class.getClassLoader().getResource("adictosaltrabajo/texturas/dirt.jpg")), 0, 128, 255);
15.         pt.addTexture(new ImageIcon(TestTerrain.class.getClassLoader().getResource("adictosaltrabajo/texturas/highest.jpg")), 128, 255,
16.                 384);
17.         pt.createTexture(32);
18.
19.         TextureState ts = display.getRenderer().createTextureState();
20.         Texture t1 = TextureManager.loadTexture(pt.getImageIcon().getImage(),
21.                 Texture.MinificationFilter.Trilinear, Texture.MagnificationFilter.Bilinear,true);
22.         ts.setTexture(t1, 0);
23.
24.         Texture t2 = TextureManager.loadTexture(
25.                 TestTerrain.class.getClassLoader().getResource(
26.                 "adictosaltrabajo/texturas/Detail.jpg"),
27.                 Texture.MinificationFilter.Trilinear,
28.                 Texture.MagnificationFilter.Bilinear);
29.
30.         ts.setTexture(t2, 1);
31.         t2.setWrap(Texture.WrapMode.Repeat);
32.
33.         t1.setApply(Texture.ApplyMode.Combine);
34.         t1.setCombineFuncRGB(Texture.CombinerFunctionRGB.Modulate);
35.         t1.setCombineSrc0RGB(Texture.CombinerSource.CurrentTexture);
36.         t1.setCombineOp0RGB(Texture.CombinerOperandRGB.SourceColor);
37.         t1.setCombineSrc1RGB(Texture.CombinerSource.PrimaryColor);
38.         t1.setCombineOp1RGB(Texture.CombinerOperandRGB.SourceColor);
39.
40.         t2.setApply(Texture.ApplyMode.Combine);
41.         t2.setCombineFuncRGB(Texture.CombinerFunctionRGB.AddSigned);
42.         t2.setCombineSrc0RGB(Texture.CombinerSource.CurrentTexture);
43.         t2.setCombineOp0RGB(Texture.CombinerOperandRGB.SourceColor);
44.         t2.setCombineSrc1RGB(Texture.CombinerSource.Previous);
45.         t2.setCombineOp1RGB(Texture.CombinerOperandRGB.SourceColor);
46.
47.         tb.setRenderState(ts);
48.         tb.setDetailTexture(1, 16);
49.         tb.setRenderQueueMode(Renderer.QUEUE_OPAQUE);
50.
51.         scene.attachChild(tb);
52.
53.
54.     }
```

Este método lo podemos añadir en el fichero java donde estemos creando la escena del juego.

Si no queremos que la cámara se sitúe por debajo del terreno debemos añadir el siguiente código en el método update() del juego.

```
view plain  print  ?
01. if (cam.getLocation().y < (tb.getHeight(cam.getLocation()) +2)) {
02.      cam.getLocation().y = tb.getHeight(cam.getLocation()) +2;
03.      cam.update();
04. }
```

Entonces por el momento hemos montado el cielo, es decir la caja donde transcurrirá el juego y el suelo del juego donde se moverá nuestro jugador.

Ahora crearemos nuestro jugador.

Para hacer esto primero tenemos que hacer es cargar el modelo, para ello nos creamos una clase llamada CargadorModelos y le ponemos este código que sera el que nos permita cargar un xml para nuestro muñeco.

```
view plain  print  ?
01. class CargadorModelos {
02.
03.     private final Logger logger = Logger.getLogger(CargadorModelos.class.getName());
04.     private URL urlmodelo = null;
05.     protected XMLImporter xmlImporter = XMLImporter.getInstance();
06.
07.     CargadorModelos(URL path) {
08.         urlmodelo = path;
09.     }
10.
11.     protected Spatial CargaModelo() {
12.         // May also be called during update() loop to add to scene.
13.         Spatial espacioCargado = null;
14.
15.         try {
16.             espacioCargado = (Spatial) xmlImporter.load(urlmodelo);
17.         } catch (Exception e) {
18.             logger.log(Level.INFO, "Error al cargar fichero: " + urlmodelo, e);
19.             throw new IllegalArgumentException("Error al cargar el modelo: " + urlmodelo, e);
20.         }
21.
22.         espacioCargado.setModelBound(new BoundingBox());
23.         //espacioCargado.setLocalScale(0.1f);
24.
25.         espacioCargado.updateModelBound();
26.
27.         // The default update loop will update world bounding volumes.
28.
29.         return espacioCargado;
30.     }
31. }
```

Después no crearemos una clase para poder definir lo que queremos que haga el jugador en nuestro caso la hemos llamado Mosca.

LWUIT: Una librería gráfica tipo AWT o Swing para J2ME

2009-06-10
Mapas mentales con XMind

2009-02-26
Redimensionar Imagenes en Windows Vista

2009-06-08
UploadFile con Icefaces + Hibernate + Anotaciones

2009-06-05
Habilitar exportación en Liferay

2009-06-01
Registrar Liferay en Eclipse

2009-05-29
Liferay Social Office

2009-05-28
Broadcast con Ustream

2009-05-25
Tabla datos accesible con ordenación y paginación

2009-05-21
Primeros pasos con Audacity: Un editor de sonido libre y multiplataforma.

2009-05-11
Introducción a TortoiseSVN

2009-05-07
Hacer 'scp' de varios ficheros sin solicitud de clave

2009-05-02
Plugin Hibernate3 para Maven

2009-04-26
AgileDraw: una técnica rápida de modelado

2009-04-24
Spring AOP: Cacheando aplicaciones usando anotaciones y aspectos con Aspectj

2009-04-20
Modelos de conocimiento con CmapTools

2009-04-16
Informes Crosstab con iReport

2009-04-16
Registro de un fichero de datos personales con el formulario NOTA

2009-04-15
Estadísticas de www.adictosaltrabajo.com Abril 2009

2009-04-15
Iniciación a OSWorkflow con Spring

2009-04-14
Tests de Selenium con librerías de componentes JSF: Apache Tomahawk.

2009-04-13
JTAPI. El API de Telefonía para Java

2009-04-13
Registro de Web Services con Apache jUDDI. Configuración y ejemplo

2009-04-13
Cómo hacer UML con Eclipse y el plugin UML2

2009-04-09
Spring WS: Servicios Web a través del correo electrónico

2009-04-02
Creación de cursos con Moodle

Últimas ofertas de empleo

2009-06-29
Atención a cliente - Call Center - BARCELONA.

2009-06-25
Atención a cliente - Call Center - BARCELONA.

2009-06-20
Comercial - Ventas - CASTELLON.

2009-06-19
Otras - Ingenieria (minas, puentes y puertos) - VALENCIA.

2009-06-17
Comercial - Ventas - ALICANTE.

```
view plain  print  ?
```

```
01. public class Mosca extends Node {
02. 
03.     Spatial mosca;
04.     private Quaternion q1 = new Quaternion();
05.     private Quaternion q2 = new Quaternion();
06.     private Quaternion q3 = new Quaternion();
07.     private Vector3f leanAxis = new Vector3f(0, 1, 0);
08.     private Vector3f vaux = new Vector3f();
09.     private Vector3f vmov = new Vector3f();
10.     private Vector3f aux = new Vector3f(0f, 0f, 1f);
11.     private float Velocidad = 3f;
12.     private boolean primeravez = true;
13.     float xini, zini, xfin, xac, zac, zfin, incx, incz;
14. 
15.     public Mosca(String id, Spatial pmosca) {
16. 
17.         super(id);
18.         this.detachChild(this.mosca);
19.         mosca = pmosca;
20. 
21.         this.attachChild(this.mosca);
22.     }
23. 
24.     public void update(float time) {
25. 
26.         MoverAdelante(vmov);
27.     }
28. 
29.     public void Stop() {
30.         this.localTranslation.addLocal(0, 0, 0);
31.     }
32. 
33.     public void GoTo(Vector3f destino) {
34. 
35. 
36.         mosca.worldToLocal(destino, vaux);
37. 
38.         if (destino != mosca.getLocalTranslation()) {
39. 
40. 
41.             if (primeravez == true) {
42.                 vmov = vaux.divide(10);
43.                 primeravez = false;
44. 
45.                 RotarMosca(vaux);
46. 
47. 
48.              }
49.             MoverAdelante(vmov);
50. 
51.         }
52.     }
53. 
54.     public void MoverAdelante(Vector3f donde) {
55.         this.localTranslation.addLocal(donde);
56. 
57. 
58.     }
59. 
60.     public void MoverAtras() {
61.         this.localTranslation.addLocal(0.2f, 0.0f, -0.2f);
62.     }
63. 
64.     public void DesplazarMosca() {
65.         int Direccion;
66. 
67.         Random rnd = new Random();
68. 
69.         Direccion = (int) (rnd.nextDouble() * 2);
70. 
71.         if (Direccion == 1) {
72.             this.getLocalTranslation().addLocal(1, 0, 0);
73.         } else {
74.             this.getLocalTranslation().addLocal(-1, 0, 0);
75.         }
76. 
77. 
78.     }
79. 
80.     public void mueveMosca(int accion) {
81. 
82. 
83.         if (accion == MoverMoscaKeyInputAction.DERECHA) {
84.             this.getLocalTranslation().addLocal(-1, 0, 0);
85. 
86.         }
87.         if (accion == MoverMoscaKeyInputAction.IZQUIERDA) {
88.             this.getLocalTranslation().addLocal(1, 0, 0);
89.         }
90.         if (accion == MoverMoscaKeyInputAction.ABAJO) {
91.             this.getLocalTranslation().addLocal(0, 0, -1);
92.         }
93.         if (accion == MoverMoscaKeyInputAction.ARRIBA) {
94.             this.getLocalTranslation().addLocal(0, 0, 1);
95.         }
96. 
97. 
98.     }
99. 
100.     public void RotarMosca(Vector3f donde) {
101. 
102.         this.getLocalRotation().fromAngleAxis(aux.angleBetween(donde.normalize()),this.leanAxis);
103.     }
104. }
```

Como se puede comprobar es una clase derivada de Node. En esta clase esta implementado algunos métodos esenciales como el método update() que se ejecuta cada vez que el juego realiza su update(), métodos interesantes de este código sería el GoTo ya que haya la dirección a la que queremos que nuestro jugador se mueva, el método RotarMosca() ya que en el se muestra como se hace para rotar un personaje sobre su eje y, se puede observar que el vector aux esta inicializado en la dirección que mira el jugador (este debería haber sido pintado mirando hacia un eje especifico) en nuestro caso el eje Z, entonces normalizamos el vector dirección para que exista una intersección entre los dos vectores. Y por último el método mueveMosca que implementa movimientos que la mosca realizara cuando se pulsen las teclas correspondientes.

Si queremos que el jugador siempre que se mueva por el mapa esté siempre sobre la superficie debemos añadir este código.

```
view plain  print  ?
```

```
01. float characterMinHeight = tb.getHeight(player.getLocalTranslation()) + agl;
02. 
03. if (!Float.isInfinite(characterMinHeight) && !Float.isNaN(characterMinHeight)) {
04. 
05.     player.getLocalTranslation().y = characterMinHeight;
06. }
07. 
08. tb.getSurfaceNormal(player.getLocalTranslation(), normal);
09. if (normal != null) {
10.     player.rotateUpTo(normal);
11. }
```

La variable agl que es de tipo float contiene el valor de la altura del BoundingBox del jugador, entonces al sumárselo a la altura del mapa, conseguimos el punto donde tenemos que situar al personaje.

Para crear las etiquetas encima de nuestros personajes, encontré por internet una clase que se llama TextLabel2D y me solucionó este problema.

view plain print ?

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.awt.image.ConvolveOp;
import java.awt.image.Kernel;
import java.util.Arrays;

import com.jme.image.Texture;
import com.jme.image.Texture.MagnificationFilter;
import com.jme.image.Texture.MinificationFilter;
import com.jme.math.FastMath;
import com.jme.math.Vector2f;
import com.jme.renderer.Renderer;
import com.jme.scene.BillboardNode;
import com.jme.scene.Spatial.LightCombineMode;
import com.jme.scene.shape.Quad;
import com.jme.scene.state.BlendState;
import com.jme.scene.state.TextureState;
import com.jme.scene.state.BlendState.TestFunction;
import com.jme.system.DisplaySystem;
import com.jme.util.TextureManager;

public class TextLabel2D {
    private String text;
    private float blurIntensity = 0.1f;
    private int kernelSize = 5;
    private ConvolveOp blur;
    private Color foreground = new Color(1f, 1f, 1f);
    private Color background = new Color(0f, 0f, 0f);
    private float fontResolution = 40f;
    private int shadowOffsetX = 2;
    private int shadowOffsetY = 2;
    private Font font;

    public TextLabel2D(String text) {
        this.text = text;
        updateKernel();
        setFont(Font.decode("Sans PLAIN 40"));
    }

    public void setText(String text)
    {
        this.text=text;
    }

    public String getText()
    {
       return this.text;

    }
    public void setFont(Font font){
        this.font = font;
    }

    public void setShadowOffsetX(int offsetPixelX){
        shadowOffsetX = offsetPixelX;
    }
    public void setShadowOffsetY(int offsetPixelY){
        shadowOffsetY = offsetPixelY;
    }
    public void setBlurSize(int kernelSize){
        this.kernelSize = kernelSize;
        updateKernel();
    }
    public void setBlurStrength(float strength){
        this.blurIntensity = strength;
        updateKernel();
    }
    public void setFontResolution(float fontResolution){
        this.fontResolution = fontResolution;
    }

    private void updateKernel() {
        float[] kernel = new float[kernelSize*kernelSize];
        Arrays.fill(kernel, blurIntensity);
        blur = new ConvolveOp(new Kernel(kernelSize, kernelSize, kernel));
    }

    /**
      *
      * @param scaleFactors is set to the factors needed to adjust texture coords
      * to the next-power-of-two- sized resulting image
      */
    private BufferedImage getImage(Vector2f scaleFactors){
        BufferedImage tmp0 = new BufferedImage(10, 10, BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2d = (Graphics2D) tmp0.getGraphics();
        Font drawFont = font.deriveFont(fontResolution);
        g2d.setFont(drawFont);
        Rectangle2D b = g2d.getFontMetrics().getStringBounds(text, g2d);

        int actualX = (int)b.getWidth()+kernelSize+1+shadowOffsetX;
        int actualY = (int)b.getHeight()+kernelSize+1+shadowOffsetY;

        int desiredX = FastMath.nearestPowerOfTwo(actualX);
        int desiredY = FastMath.nearestPowerOfTwo(actualY);

        if(scaleFactors != null){
            scaleFactors.x = (float)actualX/desiredX;
            scaleFactors.y = (float)actualY/desiredY;
        }

        tmp0 = new BufferedImage(desiredX, desiredY, BufferedImage.TYPE_INT_ARGB);

        g2d = (Graphics2D) tmp0.getGraphics();
        g2d.setFont(drawFont);
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING, RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

        int textX = kernelSize/2;
        int textY = g2d.getFontMetrics().getMaxAscent() - kernelSize2;

        g2d.setColor(background);
        g2d.drawString(text, textX + shadowOffsetX, textY + shadowOffsetY);

        BufferedImage ret = blur.filter(tmp0,null);

        g2d = (Graphics2D) ret.getGraphics();
        g2d.setFont(drawFont);
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        g2d.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING, RenderingHints.VALUE_TEXT_ANTIALIAS_ON);

        g2d.setColor(foreground);
        g2d.drawString(text, textX, textY);

        return ret;
    }

    public Quad getQuad(float height){
        Vector2f scales = new Vector2f();
        BufferedImage img = getImage(scales);
        float w = img.getWidth() * scales.x;
        float h = img.getHeight() * scales.y;
        float factor = height / h;
        Quad ret = new Quad("textLabel2d", w * factor , h * factor);
        TextureState ts = DisplaySystem.getDisplaySystem().getRenderer().createTextureState();
        Texture tex = TextureManager.loadTexture(img, MinificationFilter.BilinearNoMipMaps, MagnificationFilter.Bilineartrue);



        ts.setTexture(tex);
        ts.setEnabled(true);
        ret.setRenderState(ts);

        ret.setRenderQueueMode(Renderer.QUEUE_TRANSPARENT);

        BlendState as = DisplaySystem.getDisplaySystem().getRenderer().createBlendState();
        as.setBlendEnabled(true);
        as.setTestEnabled(true);
        as.setTestFunction(TestFunction.GreaterThan);
        as.setEnabled(true);
        ret.setRenderState(as);

        ret.setLightCombineMode(LightCombineMode.Off);
        ret.updateRenderState();
```

Y la verdad es muy útil, con que se cree y se añada como hijo al elemento que queramos etiquetar ya esta.

El código fuente de las clases implementadas las puede descargar de: fuentes.zip

Espero que les haya sido útil este tutorial, seguiremos haciendo más tutoriales sobre esta tecnología analizando más ejemplos algo más complicados, todo el que quiera hacer una aportación será bien recibida. Para comunicarme cualquier problema o sugerencia de mejora podéis utilizar la zona de comentarios, de este modo todo el mundo se podrá aprovechar de las respuestas.

Saludos.

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Directorio de ejemplos de jMonkey Engine	Directorio de los ejemplos más relevantes de jMonkeyEngine, que nos ayudarán a crear juegos en 3D	2009-07-20	312	Muy bueno	4	
Como implementar el Scene Monitor para analizar las escenas en jMonkeyEngine	Implementación de SceneMonitor de jMonkeyEngine en nuestra aplicación.	2009-07-16	283	Muy bueno	4	
Continuación del Tutorial: JMonkeyEngine, Creación de nuestro primer juego.	En este turorial implementaremos colisiones en jMonkeyEngine, en el juego 3D que estamos analizando	2009-07-16	374	Muy bueno	8	
Detalles del juego de la moto en jMonekyEngine.	Detalles del juego de la moto creado en jMonkeyEngine, donde se definen movimientos y chaserCameras	2009-07-15	565	Muy bueno	12	
JMonkeyEngine, Creación de nuestro primer juego.	Intentaremos enseñaros a crear vuestro primer juego, partiremos de un ejemplo hecho de JMonkeyEngine, que trata sobre el manejo de una moto por un escenario.	2009-07-14	479	Bueno	15	
Ajax tests con Selenium: prototype.js e ICEfaces.	En este tutorial se va habla de cómo escribir tests funcionales con Selenium IDE sobre aplicaciones que realizan recargas controladas de la interfaz de usuario con Ajax.	2009-07-13	564	Bueno	3	
AOP con AspectJ y Maven	Programacion orientada a aspectos con AspectJ y Maven	2009-07-08	567	Bueno	1	
Iniciarse en el manejo de JME, Creación de un Cloth.	Primeros pasos con Jmonkey engine, crearemos una pequeña aplicación animada basada en un Cloth con colisiones	2009-07-07	687	Muy bueno	6	
Primeros pasos con Blender: Pintando nuestra mascota en 3D	Inicio del manejo de blender, os mostraremos paso a paso como se dibuja en blender	2009-07-06	686	Bueno	18	
Juegos 3D en Java: Blender y JMonkeyEngine	En este tutorial se abordara una pequeña iniciación a los juegos 3D en java usando la herramienta Blender y como motor gráfico JMonkeyEngine	2009-07-02	1898	Bueno	24	