VRML: EL LABORATORIO VIRTUAL COMO ALTERNATIVA ECONÓMICA AL LABORATORIO REAL

Arturo Pascual Granged(*), José Angel Bernal Bermejo(**), Rafael Fernández Mateos(***), Eduardo Díaz Delgado(****).

(*) Física Aplicada, (****) Algebra.
(*) Escuela Universitaria Politécnica, (****) Facultad de Informática
(*)(****)Universidad de Sevilla.
(**) Junta de Andalucía
(***) Grupo CruzCampo

RESUMEN

La formación de un Ingeniero Técnico pasa ineludiblemente por dos fases de estudio: El conocimiento teórico y el desarrollo de actividades prácticas. Si bien el conocimiento teórico queda en mayor o peor medida cubierto por las clases de "pizarra", la problemática asociada al desarrollo de actividades prácticas es bien distinta. En esta ponencia planteamos una solución que se ha visto favorecida por el desarrollo de los ordenadores y los resultados obtenidos en el campo de la realidad virtual.

1.- INTRODUCCIÓN

Las Carreras Técnicas, de manera tradicional, deben ser por su concepción, la base de preparación de titulados con una vocación industrial claramente establecida. El sentido eminentemente práctico de su formación no lo pone nadie en entredicho, no obstante, el ingeniero técnico se concibe como una persona con una formación Científico-Técnica capaz de resolver los problemas planteados por la industria en los procesos de desarrollo y fabricación. Partiendo de esta filosofía, los nuevos planes de estudio imponen una gran limitación de horario a las clases tradicionalmente conocidas como "de pizarra", y potencian el trabajo experimental o de laboratorio como un pilar importante en la formación de cualquier Ingeniero Técnico.

Aunque la filosofía de los nuevos planes de estudio es muy interesante en este aspecto, la realidad es bien distinta. De una parte, la masificación que sufren

estas escuelas, impide que los alumnos tengan un acceso al laboratorio fuera de horas de prácticas para poder familiarizarse con los instrumentos o reforzar los conocimientos adquiridos en alguna práctica particular. Por otro lado, la dotación presupuestaria para equipamiento y adaptación de los laboratorios es claramente insuficiente, quedando, en la mayoría de los casos, reducida a una inversión inicial que después no se ve apoyada por inversiones acordes de mantenimiento y renovación de los equipos, sin tener en cuenta los desgastes producidos en los equipos de laboratorio por la alta saturación de alumnos que acceden a ellos semanalmente. Por último, cabe destacar que la formación de un Ingeniero Técnico debe ser enfocada hacia el entorno industrial planteando el problema en asignaturas de marcado carácter terminal. El entrenamiento que debe plantear el laboratorio en estos casos debe ser mucho más flexible, adaptándose constantemente a las nuevas tendencias del mercado así como a las nuevas tecnologías.

El desarrollo de nuevas tecnologías sumado al considerable aumento en la potencia de los ordenadores personales ha llevado a distintos grupos de trabajo a plantear soluciones locales encaminadas en esta línea. La utilización del ordenador como instrumento de apoyo a la docencia y a la educación no es nueva. Muchos autores plantean el uso del laboratorio informático, desde la utilización del ordenador en el laboratorio informático de prácticas con varios sistemas operativos [1], pasando por el uso de aplicaciones multimedia [2] hasta la propuesta de Karlsson [3] del desarrollo de un programa (PLATO) encaminado al aprendizaje de la Física y la Mecánica a distancia mediante el desarrollo de un programa de apoyo que trabajaría en una red local, ...

El gran auge de Internet en los últimos años se ha planteado como la última revolución industrial de este milenio, llevando consigo el gran aumento de ordenadores personales por casa así como la facilidad de conexión entre grupos de investigación y estudiantes, entre empresas y particulares. Esta revolución ha llevado consigo el planteamiento de nuevos retos dando sus frutos en desarrollos espectaculares. De una parte, muchos grupos de investigación han unificado esfuerzos en el desarrollo de un sistema operativo de libre difusión (LINUX) cuya idea parte de Linus Torvalds o en diversos programas bajo licencia GNU o de libre difusión ya planteado en [4]. Por otra parte, el desarrollo de lenguajes independientes de plataforma comienza con la implantación del HTML como formato estándar para páginas Web en Internet, llegando al desarrollo del Proyecto JAVA basado en un lenguaje independiente de plataforma para la ejecución, en principio, de programas incluidos en páginas web.

Un desarrollo muy ambicioso ha sido el Lenguaje de Modelado para Realidad Virtual (VRML) que en 1997 ha llegado a convertirse en un estándar ISO/IEC. La independencia de plataforma de este lenguaje lo hace especialmente útil para la

presentación y simulación de entornos virtuales en Internet. Asimismo, el espectacular aumento de librerías y repositorios existentes en Internet hacen de este lenguaje un arma con una potencia extrema no sólo en el ámbito de la educación sino en el comercio electrónico o en la presentación de proyectos de edificación [5].

Desde el punto de vista de la docencia en las Escuelas Técnicas de Ingeniería el Lenguaje de Modelado de Realidad Virtual presenta una gran cantidad de ventajas ya que el alumno va a poder asistir al laboratorio de forma virtual incluso antes de que deba hacerlo de manera presencial. De esta forma va a poder someter al equipo a pruebas que, de hacerlo físicamente supondrían la destrucción del aparato, permitiendo un aprendizaje más completo, pudiendo incluir o variar los comportamientos según vaya conociendo el proceso. Desde el punto de vista del profesorado, permite entrenar al alumno en maquinaria o procesos que, por su excesivo coste o peligrosidad, sería imposible de llevar a cabo en un laboratorio, permitiendo que los alumnos adquieran la destreza necesaria sin riesgo propio o del material. No obstante, el laboratorio virtual no debe ser sustituto sino más bien como un complemento del laboratorio tradicional.

Desde el punto de vista de la Escuela en general, la realidad virtual modelada mediante este sistema puede plantear soluciones respecto de espacio y ocupación de los laboratorios, ya que gran parte del tiempo de prácticas se podría llevar a cabo bien en una sala de ordenadores de forma independiente al horario de ocupación del laboratorio, bien en casa del propio alumno si éste dispone de un ordenador personal. En ambos casos podríamos conseguir descongestionar el laboratorio sin que ello repercuta en detrimento de la formación especifica del Ingeniero Técnico.

2.- EL LENGUAJE DE MODELADO DE REALIDAD VIRTUAL (VRML)

El Lenguaje de Modelado de Realidad Virtual (VRML-Virtual Reality Modeling Language) es básicamente un formato de intercambio de ficheros en tres dimensiones (3D). Su semántica nos permite describir objetos en una escena de 3D y las interacciones entre éstos [6][7].

Podemos definir, mediante una estructura jerárquica denominada *grafo de la escena* (scene graph), todos los objetos que componen nuestro mundo virtual. VRML integra 3D, texto y multimedia para presentarnos un nuevo tipo de aplicaciones interactivas[8]. Se espera que la interfaz en 3D que nos presenta VRML reemplace en un corto periodo de tiempo el aspecto actual en 2 dimensiones de la World Wide Web.

En 1989 comenzó un proyecto en Silicon Graphics para diseñar y construir una infraestructura para aplicaciones gráficas en 3D interactivas.

El conjunto de herramientas Iris Inventor apareció en 1992 como primer producto de estos esfuerzos. Desde el principio, el formato de los ficheros se diseñó para ser ligero y fácil de usar. La segunda revisión salió a la luz en 1994 con el nombre de Open Inventor, haciendo referencia a su portabilidad a varias plataformas y basarse en OpenGL. El manual de referencia que describe los objetos y el formato de ficheros de Open Inventor fueron usados por Gavin Bell para escribir el primer borrador de la especificación VRML 1.0.

En Octubre de 1994, la especificación VRML 1.0 fue publicada en la Segunda Conferencia Internacional sobre el World Wide Web en Chicago. El objetivo era conseguir un núcleo funcional que contuviese una definición consistente de un espacio en 3D y consiguientemente, evitar aspectos más controvertidos, como comportamientos de los objetos o capacidades multiusuario, con el fin de acelerar el proceso [9].

La especificación VRML 2.0 apareció en Agosto de 1996 en la conferencia SIGGRAPH celebrada en Nueva Orleans, con el objetivo de expandir la especificación inicial con interacción, animación y comportamiento. Con el fin de cumplir este propósito, era necesario permitir que los objetos en el mundo virtual se movieran independientemente e igualmente, que los usuarios interaccionaran con los objetos en el mundo virtual [10].

En 1997, fue publicado y aceptado por ISO el estándar ISO/IEC 14772-1:1997, the Virtual Reality Modeling Language (VRML), conocido como «VRML97» [11].

Actualmente se trabaja en la nueva versión del estándar que recibe el nombre de X3D.

La interacción entre múltiples objetos, la integración de leyes físicas ("de mundo real") y comportamiento autónomo de los objetos que comprenden el mundo virtual, son de las características más importantes para conseguir simulaciones correctas [12][13].

Grafo de la Escena (Scene graph)

VRML describe objetos y sus propiedades dentro de mundos virtuales, mediante una estructura jerárquica conocida como *grafo de la escena*. Las entidades que constituyen el grafo de la escena se denominan *nodos*.

En el estándar VRML 2.0 están definidos 54 tipos diferentes de nodos. Entre ellos se encuentran primitivas geométricas (p.ej. Cone, Sphere), propiedades de apariencia (p.ej: TextureTansform, DirectionalLight), propiedades de sonido (p.ej: Sound) y nodos de agrupación (p.ej: Transform). Sin embargo, podemos definir nuevos nodos si los necesitáramos mediante el mecanismo de *prototipos*.

Los nodos almacenan sus datos en *campos* (fields). Existen 20 tipos diferentes de campos que pueden ser usados para almacenar valores simples (aquellos que tienen como prefijo las letras SF) o valores múltiples (cuyas dos primeras letras son MF). Por ejemplo, un campo de tipo *SFBool* puede contener un solo valor booleano (Verdadero o Falso) y un campo de tipo MFVec3f puede contener múltiples (cero o más) vectores tridimensionales (p.ej: *Normales[1 0 0, 0 54 20]*).

Los campos se encuentran dentro de sentencias de nodos en el fichero VRML y definen el estado persistente de los objetos en el mundo virtual. Cada nodo interpreta los valores de sus campos con respecto a su implementación.

El fichero VRML construye el grafo de la escena como un grafo dirigido acíclico. Esta jerarquía se construye de la manera siguiente. Los nodos padres (que son nodos de agrupación) pueden contener uno o más nodos hijos (children). Los nodos hijos pueden tener uno o más nodos padres.

```
A {
children B {
children  D{} E{} F{}
  }
  C {
     children  F{} G{}
     }
  }
```

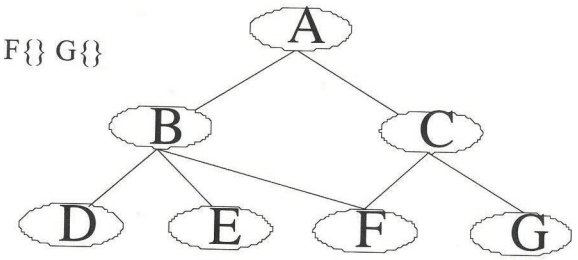

Figura 1: Relaciones padre-hijo en el grafo de la escena.

VRML utiliza un sistema de coordenadas tridimensional. El *punto de vista* (viewpoint) o punto desde donde se observa al mundo virtual, se encuentra sobre el eje positivo de la Z, mirando en la dirección –Z y con el eje +Y hacia arriba. Se puede usar una simple transformación para modificar la proyección por defecto.

Cada nodo posee una definición basada en la orientación a objetos. A continuación mostramos la especificación de la interfaz pública para un nodo de tipo Cube. En ella especificamos los nombres y tipos de los campos del nodo, así como los valores por defecto de los campos. Las palabras en negrita corresponden a la sintaxis del fichero (lo que encontramos en el fichero VRML) para ese nodo.


```
Cone {
  field SFFloat  bottomRadius  1
  field SFFloat  height        2
  field SFBool   side          TRUE
  field SFBool   bottom        TRUE
}
```

El campo *bottomRadius* es del tipo *SFFloat* y su valor por defecto es 1.

La sintaxis del fichero VRML para una instancia (el tipo de nodo nos define un objeto genérico –tipo o clase- que necesita ser instanciado para utilizarse) de un nodo de tipo Cone llamado *micono* con un valor para *bottomRadius* de 2 y un valor para *height* de 4. Los valores de campos no especificados explícitamente (*side* y *bottom*) toman sus valores por defecto; TRUE para ambos).

DEF micono Cone { bottomRadius 2 height 4 }

Cada mundo virtual en VRML posee un *punto de vista* (viewpoint) desde el cual el mundo se observa en un momento determinado. La *navegación* es la acción que realiza el usuario para cambiar la posición y/o la orientación de este punto de vista, cambiando la visión que observa el usuario. Esto permite moverse por el mundo virtual o examinar un objeto.

La extensión de los ficheros VRML está normalizada y es **.wrl**. Existe un tipo oficial MIME para los ficheros VRML que es **model/vrml.**

Eventos

El mecanismo por el cual los nodos en el grafo de la escena pueden transmitir mensajes a otros nodos es por medio de *eventos*. Un tipo de nodo tiene tipos de eventos predefinidos que las instancias del nodo pueden enviar y recibir.

Los nodos VRML generan eventos en respuesta a estímulos producidos por el entorno o el usuario. Una vez generados, los eventos son enviados a sus destinos en orden cronológico. Cada evento posee una *marca de tiempo* (timestamp) que indica el tiempo en el que fue generado y es procesado por el nodo que lo recibe según su marca de tiempo. Este mecanismo nos asegura resultados deterministas, necesarios para una correcta simulación [13].

Este mecanismo, denominado *ruta de eventos* (event routing) puede alterar el estado interno del nodo receptor y transformar el grafo de la escena.

Los caminos entre nodos que son emisores de eventos y los nodos que son receptores de eventos se definen mediante sentencias del tipo ROUTE.

La interfaz pública para un nodo de tipo *Sensor de Contacto* (TouchSensor) es:

```
TouchSensor{
 exposedField SFBool enabled TRUE
 eventOut    SFVec3f hitNormal_changed
 eventOut    SFVec3f hitPoint_changed
 eventOut    SFVec2f hitTexCoord_changed
 eventOut    SFBool  isActive
 eventOut    SFBool  isOver
 eventOut    SFTime  touchTime
}
```

Los eventos definidos para un tipo de nodo específico pueden ser eventIn, eventOut o exposedField. El tipo *eventIn* define eventos que pueden ser recibidos por el nodo. El tipo *eventOut* indica el tipo y el nombre de eventos que este nodo puede enviar a otros nodos. Normalmente, los eventos del tipo eventIn se llaman de la forma *set_XXX* y los eventos del tipo eventOut, de la forma *XXX_changed*. El tipo de evento *exposedField* es una combinación de eventOut, eventIn y campo.

A continuación mostramos parte de la definición de la interfaz pública de un nodo ViewPoint. Mostramos sólo el campo *position* que es del tipo exposedField (esto es, eventIn, eventOut y campo) con valores del tipo SFVec3f.

```
ViewPoint{
exposedField SFVec3f position  0 0 10
...
}
```

En el fichero VRML podemos definir dos nodos *TouchS1* y *ViewP1*, de tipos TouchSensor y ViewPoint, respectivamente. Después definimos una ruta (ROUTE) entre el evento de salida *hitPoint_changed* del sensor de contacto (TouchSensor) al exposedField (que funciona como evento de entrada -eventIn) de ViewPoint.

```
DEF TouchS1 TouchSensor {}
DEF ViewP1  ViewPoint {}
ROUTE TouchS1.hitPoint_changed TO ViewP1.position
```

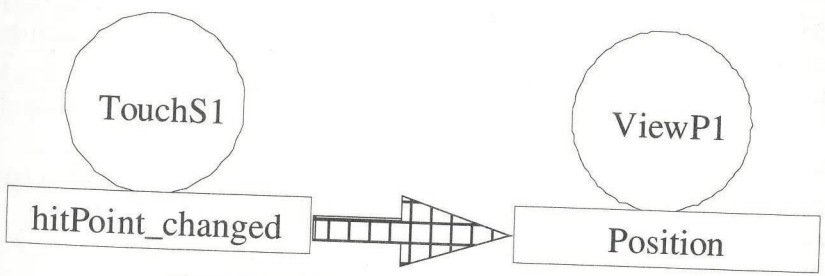

Figura 2: ROUTE entre TouchSensor y ViewPoint

Los nodos reciben eventos en sus campos eventIn. Los eventos que entran son mensajes enviados por otros nodos que afectan al estado del nodo receptor. Los campos eventOut envían mensajes a los nodos destino indicando que el estado ha cambiado en el nodo origen. Los tipos de eventos de salida en el nodo emisor y el tipo del evento de entrada en el nodo receptor deben ser idénticos.

Sensores

Los sensores nos permiten interaccionar con objetos en el mundo VRML. Los sensores generan eventos que son enrutados (por medio de sentencias ROUTE) a otros nodos que reaccionan a esta entrada. Existen diferentres tipos de sensores. Los sensores pueden detectar cambios en el tiempo (el nodo TimeSensor genera eventos con el paso del tiempo y es fundamental para la sincronización de simulaciones), cambios en relación al movimiento del observador u objetos en el mundo virtual (ProximitySensor, VisibilitySensor, Collision group) o cambios en el estado de un dispositivo de entrada (TouchSensor, PlaneSensor).

En el momento actual, la detección de colisiones entre el observador y los objetos del mundo virtual es soportada por el nodo *Collision group*, pero la especificación no trata (ha de especificarse aparte) la detección de colisiones entre objetos dentro del mundo virtual.

Los *sensores de entorno* (environmental sensors) incluyen:

a) ProximitySensor: detecta cuándo un usuario navega (es el término utilizado para los recorridos dentro del mundo virtual) a una región específica.
b) TimeSensor: es un reloj que se utiliza para comenzar o parar nodos dependientes del tiempo o simulaciones.
c) VisibilitySensor: detecta cuándo una parte específica del mundo virtual se hace visible al observador.
d) Collision: nodo de agrupación que detecta cuándo el observador colisiona con objetos en el mundo virtual.

El sensor de tiempo (TimeSensor) genera eventos a medida que transcurre el tiempo. Puede ser usado para múltiples propósitos, como por ejemplo, conducir simulaciones, controlar actividades periódicas o iniciar eventos.

Scripts (Guiones)

Los nodos de tipo script se insertan entre generadores de eventos (nodos sensores u otros nodos de script) y los receptores de eventos. Este tipo de nodo nos permite definir el comportamiento codificado en un determinado lenguaje de pro-

gramación [14][15]. La especificación VRML 2.0 define enlaces para los lenguajes de programación Java [16] y JavaScript.

El visualizador (browser) de VRML es responsable de interpretar (parsing), ejecutar y presentar los resultados de la interacción con el mundo virtual.

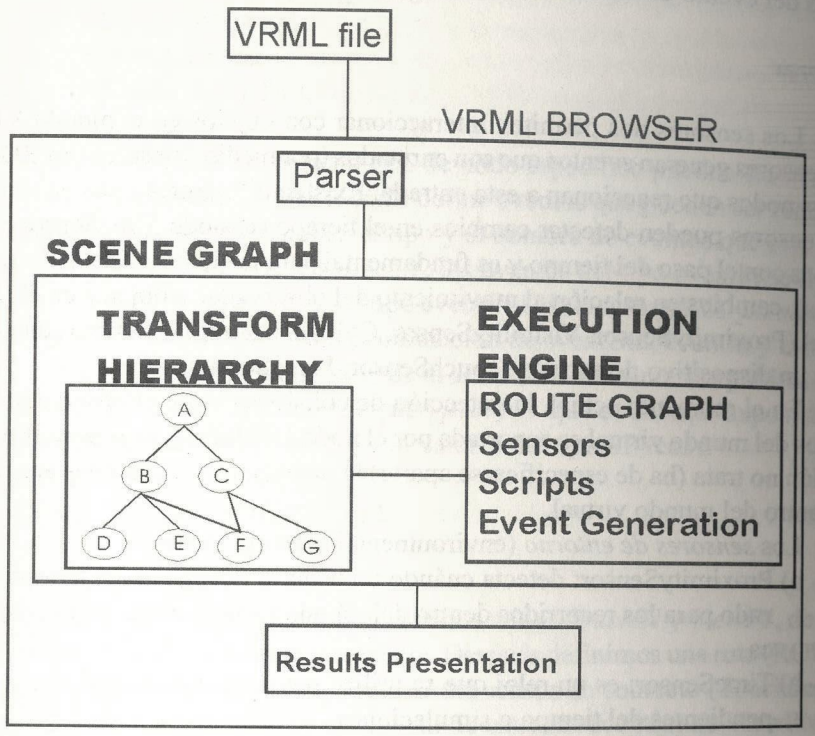

Figura 3: Estructura de ejecución del fichero VRML

El fichero VRML es leído por el browser y genera el grafo de la escena. El grafo de la escena está compuesto por la *Jerarquía de Transformación* (Transform Hierarchy –nodos que describen objetos en nuestro mundo virtual) y el *Grafo de Rutas* (Route Graph) que define las conexiones e interacciones entre los objetos.

Los nodos tipo script son el instrumento que permite a VRML dotar de comportamiento a los objetos. Los nodos script reciben eventos de otros nodos, los procesan adecuadamente y envían eventos de salida a otros nodos. Un nodo script puede retener información de su estado interno entre diferentes ejecuciones. La interfaz pública externa de un nodo de tipo script es la siguiente:


```
Script {
exposedField MFString  url          []
field        SFBool    directOutput FALSE
field        SFBool    mustEvaluate FALSE
[eventIn     eventType eventName]
[eventOut    eventType eventName]
[field       fieldType fieldName initialValue]
}
```

El código asociado en un lenguaje de programación (Java o JavaScript) es referenciado por el campo *url*, que es buscado y ejecutado para llevar a cabo la función del nodo script.

Si un nodo script tiene acceso a un nodo VRML (por medio de un valor de tipo SFNode o MFNode en uno de los campos del nodo script o pasado como un evento de entrada), el script puede leer los contenidos de la interfaz pública de dicho nodo.

Aplicación

A continuación mostramos un ejemplo que presenta la interacción entre un sensor de tiempo que activa un script para comenzar una simulación del movimiento de un robot en un laboratorio virtual. El script toma el control de la ejecución y una de sus actividades es modificar el punto de vista desde el cual el robot observa el mundo (por ejemplo, desde una cámara que éste lleva incorporada) [17].

Definimos un sensor de tiempo con los valores por defecto.

```
DEF TS1 TimeSensor {}
```

Esta es la definición del script llamado StartSimulation.

```
DEF StartSimulation Script {
  url        «http://localhost/start1.class»
  eventIn    SFBool   start
  eventOut   SFVec3f  position
}
```

Definimos una ruta entre el campo isActive del sensor de tiempo y el campo start del Script.

```
ROUTE TS1.isActive TO StartSimulation.start
```

Este es el esqueleto del fichero Java *start1.class:*

```
import vrml.*;
import vrml.field.*;
import vrml.node.*;

public class start1 extends Script {
private SFVec3f position1;
public void processEvent(Event ev) {
if (ev.getName().equals(«start»)){
constSFBool v = (ConstSFBool) ev.getValue();
if (v == true) {
//start simulation
position1.setValue(actual_position);
}
...
}
}
...
}
```

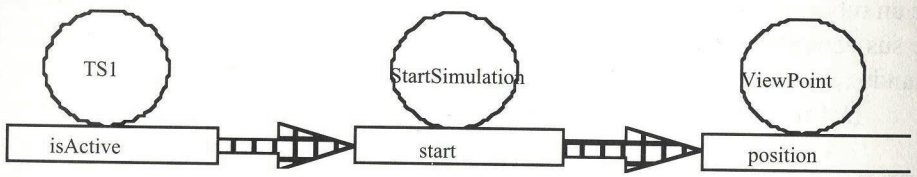

Figura 4: Interacción entre Sensor, Script and ViewPoint

Cuando el nodo script StartSimulation recibe el evento de entrada (eventIn) de otro nodo, un método (nombre Java para una acción asociada a un objeto) *processEvent()* es llamado, el cual recibe el evento de entrada como un objeto Java de tipo Event.

El objeto de tipo evento en Java posee tres campos de información: nombre (name), valor (value) y marca de tiempo (timestamp). El campo name se refiere al nombre del evento de entrada (eventIn), value contiene el valor del evento como tipo ConstField y timestamp contiene el tiempo en el que ocurrió el evento.

3.- SIMULACIÓN DE SISTEMAS Y PROGRAMACIÓN FUERA DE LÍNEA

Un entorno de simulación de robots nos ayuda a comprobar cómo se comportará un robot construido o proyectado, en un entorno de trabajo de la vida real

Paquetes software de bibliotecas reutilizables ayudan a los diseñadores a construir entornos de trabajo virtuales e incluir robots con todas sus especificaciones para comprobar la viabilidad del trabajo del robot en una aplicación específica

El software que demuestra su corrección en el entorno simulado puede se cargado en el robot real. Esto es lo que se conoce como programación fuera de línea (off-line programming) y entre sus beneficios se encuentran los siguientes:

- Elimina un gran porcentaje de órdenes de cambio del sistema robotizado gracias a una mayor calidad de los programas.
- Las bibliotecas reutilizables de componentes software aceleran las fase de prototipado y diseño.
- Disminuye costes del software de simulación debido a su estandarización
- Disminuye la posibilidad de daños en el robot debido a fallos en los programas de control.

4.- EL ENTORNO DE SIMULACIÓN DEL LABORATORIO

A la hora de generar un laboratorio virtual, el primer paso que habrá qu definir es cuál va a ser nuestro entorno virtual. Este entorno se generará indepe dientemente para cada práctica sin elementos superfluos que no contribuyan objetivo de la práctica. El punto de vista debe quedar acorde con el que el alum va a tener en el laboratorio respecto de la práctica correspondiente. Se debe rec dar en todo momento el objetivo fundamental de este laboratorio virtual, que familiarizar al alumno con un entorno de trabajo real existente.

La definición de elementos es la parte fundamental de un laboratorio v tual. Sin perder el objetivo fundamental de este laboratorio, los elementos los p dremos clasificar en dos grupos:

a) Elementos de estudio. Este tipo de elementos serán aquellos de los cu les se propongan para analizar el comportamiento, que será caracterís co de cada práctica. A modo de ejemplo, el estudio de un generador alterna lleva asociado un elemento como es ese generador, un elem motor, ... , que no van a ser compartidos por una práctica de calorimet donde el elemento básico será el modelo de un calorímetro.

b) Elementos de medida. Dentro de esta categoría se van a englobar aquellos objetos que sirven para obtener los resultados de unos determinados sensores situados en los elementos de estudio. Debido a la dificultad o al miedo que, en algunos casos, presentan los alumnos a manejar estos elementos en el laboratorio, el desarrollo de estos objetos virtuales puede representar una clara facilidad pues les permite un entrenamiento "sin riesgo". El desarrollo de estos elementos supone una gran ventaja ya que los elementos de medida no son exclusivos de una única práctica, cambiando solamente los correspondientes sensores asociados.

La elaboración de un ambiente virtual siempre va a llevar asociado la interconexión de ambos elementos.

Creemos que la metodología más adecuada a la hora de implantar este laboratorio llevaría a un desarrollo de las clases prácticas en tres fases.

Una primera fase consistiría en la familiarización con el ambiente virtual por parte del alumno. En este punto los elementos de medida serían de gran utilidad ya que el alumno podría aprender su manejo a la vez que va adaptándose a dicho entorno. También convendría un primer contacto con el laboratorio real a fin de que se vean las semejanzas respecto de aparatos reales y, de esta forma, poder unir ambos laboratorios. Debemos inducir al alumno el hecho de que el laboratorio virtual es una "copia de comportamientos" del laboratorio real y no algo desligado de la realidad. Tampoco debemos transmitir la idea de que el laboratorio virtual va a suplir al laboratorio real ya que el entrenamiento virtual puede ser un gran apoyo a la educación del Ingeniero Técnico pero es incompleto a la hora de establecer su formación práctica.

Una segunda fase consistiría en el planteamiento formal del problema a resolver. En este punto introducimos los elementos de estudio. Descubrir el comportamiento, en general, es el objetivo fundamental de una práctica de laboratorio. Por esta razón, el comportamiento debe permanecer inicialmente oculto al alumno. Dentro del laboratorio virtual, el alumno debe familiarizarse con los elementos a estudiar, con la colocación de los distintos sensores y con la interconexión entre todos los elementos de la escena virtual. Dentro de esta fase es conveniente que el alumno realice diversas medidas y analice los resultados para saberlos interpretar. Llegado a este punto sería recomendable que el alumno realice físicamente la práctica para la cual se ha entrenado, de esta forma al llegar al laboratorio el alumno no se encuentra con un ambiente extraño, los diversos aparatos ya son familiares y su forma de actuar ya no es totalmente extraña frente a equipos que desconozca. El tiempo asignado para su estancia en el laboratorio será, de esta forma, mucho más fructífero ya que no sólo podrá realizar con éxito la práctica sino que podrá llevar a cabo pruebas sobre ella que ya haya realizado de manera virtual.

Una vez superada la fase de realización física del laboratorio llega la fase más interesante desde el punto de vista docente. Al estar las clases de laboratorio íntimamente ligadas a las clases teóricas convendría abrir al alumno las claves de comportamiento de los correspondientes sistemas estudiados en el laboratorio. La finalidad de esta última fase es permitir que el alumno pueda cambiar los comportamientos de los diversos elementos de estudio para poder comprobar los conocimientos adquiridos en clase. En este punto, el alumno podría de forma sencilla comprobar un péndulo con la gravedad lunar o llevar a cabo ensayos de materiales distintos a los existentes en el laboratorio. Al realizar estos experimentos el alumno puede asentar los conocimientos adquiridos de forma teórica e incluso incrementarlos por su propia experimentación.

Como punto final cabe resaltar que el alumno puede buscar en Internet nuevos modelos ya formados en VRML pues existen enlaces que sirven de repositorios de modelos y objetos, además de las generadas en su centro.

Obviamente, esta metodología no siempre se va a poder llevar a la práctica. El Ingeniero Técnico debe tener una formación íntimamente relacionada con los distintos avances tecnológicos que se vayan produciendo en la industria. En este punto, aunque la formación teórica se vaya adaptando a estos cambios, el coste debido a la adaptación de los laboratorios puede ser extremadamente alto e incluso puede plantear problemas de espacio. Es aquí donde el laboratorio virtual se establece como laboratorio único.

Es en asignaturas de marcado carácter terminal donde, a la hora de entrenar a un alumno, se obtienen grandes beneficios. Así, un alumno podrá ver y saber qué debe hacer en el caso de una central térmica si ocurre cualquier incidente sin necesidad de tener una central térmica en el laboratorio, algo imposible por otro lado. Es en este tipo de asignaturas donde los cambios industriales se pueden adaptar mas rápidamente consiguiendo que la formación del alumno se vea revalorizada respecto de los entornos industriales a los que va dirigido.

BIBLIOGRAFÍA

[1] Díaz, E. y Fernández, R., "Laboratorio de Sistema Operativo Múltiple para Prácticas". VI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas. Las Palmas de Gran Canaria, 1998. Pags. 321-330.

[2] Turégano, J.A., Velasco, M.C., Hernández, M.A., Romeo, L.M., Cózar, J.M. y Arciniega, T., "Informatización de una asignatura: Gestor/Autor de Problemas y Cuestiones. Aplicación a la Termodinámica Técnica". VI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas. Las Palmas de Gran Canaria, 1998. Pags. 77-84.

[3] Karlsson, G. "FLIP- Flexible Learning in Physics and Mechanics", ASPEN-MITAPE, Philippines, 1996-10/22.

[4] Pascual, A. y Díaz, E., "Software de dominio público: una alternativa de coste cero para la docencia asistida por ordenador". VI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas. Las Palmas de Gran Canaria, 1998. Pags. 315-320.
[5] O'Leary, T., "On-Line Learning Enviroments in Archirtectural and Constraction Education". ASCILITE'97, Australia 1997.
[6] A.L. Ames, et al. «The VRML 2.0 SourceBook», Wiley & Sons, 1996.
[7] E. Tittel, et al. «Building VRML Worlds», Osborne-McGrawHill, 1996.
[8] W. Grosky. «Multimedia Information Systems». IEEE Multimedia, Vol. 1 nº 1, pags. 12-24, Spring 1994.
[9] G. Bell, A. Parisi, M. Pesce. «VRML 1.0 Specification» http://www.vrml.org 1995.
[10] ISO/IEC CD 14772. «VRML 2.0 Specification» http://www.vrml.org 1996.
[11] ISO/IEC. Information technology — Computer graphics and image processing — The Virtual Reality Modeling Language (VRML) — Part 1: Functional specification and UTF-8 encoding
[12] D. Anderson, «Building Multiuser Interactive Multimedia Environments at MERL». IEEE Multimedia, Vol. 2 nº 4, pags 77-82, Winter 1995.
[13] V. Aunpam & C. Bajaj, «Shastra: Multimedia Collaborative Design Environment». IEEE Multimedia, Vol. 1 nº 2,pags. 39-49, Summer 1994.
[14] O. Hagsand, «Interactive Multiuser VEs in the DIVE System». IEEE Multimedia, Vol. 3 nº 1, pags. 30-39, Spring 1996.
[15] R. Carey & G. Bell., «The VRML 2.0 Reference Manual». Addison-Wesley, 1997.
[16] P. Naugton & H. Schildt, «Java: The Complete Reference». Osborne-McGrawHill, 1997.
[17] A. Del Bimbo & E. Vicario, «Visual Programming of Virtual Worlds Animation». IEEE Multimedia, Vol. 3 nº 1, pags. 40-49, Spring 1996.