DT−VAM : un Tutoriel Distribué pour l'Analyse Virtuelle de Mécanismes. Application aux suspensions automobiles

DT−VAM : a Distributed Tutorial for Virtual Analysis of Mechanisms. Application to car suspension mechanisms

Jean−Christophe FAUROUX [1], Alain VAXELAIRE [2]

(1) : IFMA / LaRAMA,
Campus Universitaire de Clermont−Ferrand / Les Cézeaux,
BP 265, 63175 AUBIERE Cedex, FRANCE
Phone : 00.33.4.73.28.80.50 / Fax : 00.33.4.73.28.81.00
E−mail : Jean−Christophe.Fauroux@ifma.fr
Web : www.ifma.fr/recherche/larama

(2) : MICHELIN,
Centre de Technologies Ladoux, 63040,
CLERMONT−FERRAND Cedex 9
Phone : 00.33.4.73.10.78.87 / Fax : 00.33.4.73.10.78.49
E−mail : Alain.Vaxelaire@fr.michelin.com
Web : www.michelin.com

Résumé : DT−VAM est un tutoriel distribué destiné à permettre la bonne compréhension de mécanismes complexes, tels que les trains de suspensions automobiles. Il est utilisable dans un client web sans adjonction de plugin et offre une visualisation 3D animée et interactive du mécanisme grâce au protocole X3D. Ce papier décrit la création de DT−VAM et démontre l'intérêt de son utilisation pour propager les connaissances au sein de l'entreprise.

Mots clés : tutoriel, distribué, analyse de mécanisme, X3D, suspension.

Abstract : DT−VAM is a distributed tutorial for a better understanding of complex mechanisms, such as car suspension mechanisms. It is directly usable in a web browser without any plugin and displays an animated and interactive 3D view of the mechanism via X3D protocol. This paper describes how DT−VAM was created and demonstrates its interest for propagating knowledge inside enterprise.

Key words : distributed, tutorial, mechanism analysis, X3D, suspension**.**

1− Introduction

Chaque métier technologique nécessitant des connaissances spécifiques, il arrive de plus en plus que les grandes entreprises prennent conscience de la nécessité de synthétiser leur savoir−faire propre et de le rendre disponible pour tous en interne. C'est le constat fait par la société Michelin et qui a donné lieu à ce travail effectué à l'Institut Français de Mécanique Avancée (IFMA), sur une durée de deux années et dans le cadre de trois projets d'étudiants consécutifs [1, 2, 3].
L'objectif à atteindre était de créer des outils innovants permettant de comprendre le fonctionnement de mécanismes complexes tels que les suspensions automobiles. Ces outils devaient pouvoir s'intégrer sans difficulté dans l'intranet Michelin et être facilement extensibles de sorte que l'on puisse imaginer à terme une véritable bibliothèque de mécanismes de suspensions.
Cet article présente en détails la démarche ayant permis de construire l'application DT−VAM (Distributed Tutorial for Virtual Analysis of Mechanisms), un logiciel hybride répondant au problème évoqué précédemment.

2− Problématique et objectifs

2.1 − Les suspensions automobiles

Il est apparu assez tôt que chaque mécanisme, du fait de sa complexité, devrait être analysé à l'aide d'une maquette virtuelle 3D animée et interactive. La *troisième dimension* permet de bien observer les formes des assemblages complexes sous tous les angles ; l'*animation* facilite la compréhension du fonctionnement même du mécanisme ; l'*interactivité* permet de soumettre le mécanisme à différents cas de charges.

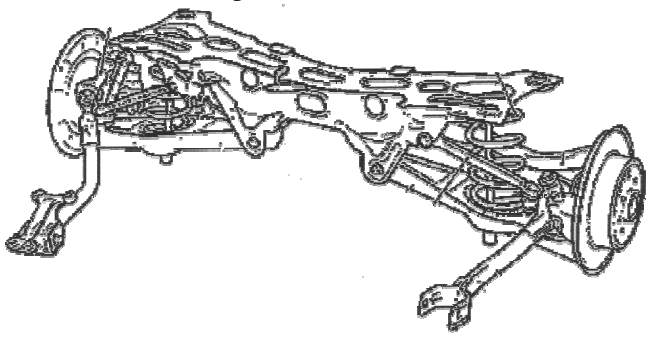

Fig. 1 : Schéma d'un essieu multibras [4].

La Figure 1 représente la vue d'ensemble d'un mécanisme de suspension tel qu'il est décrit dans un ouvrage spécialisé [4]. Dans notre optique pédagogique et simplificatrice, il est évident que nous devrons fortement alléger l'apparence d'un tel mécanisme. La Figure 2 nous montre ce même train arrière tel qu'il apparaît dans DT−VAM après simplification : seules les pièces principales apparaissent (plus de vis, de ressorts...) ; leurs formes sont épurées ; les couleurs aident à les différentier.

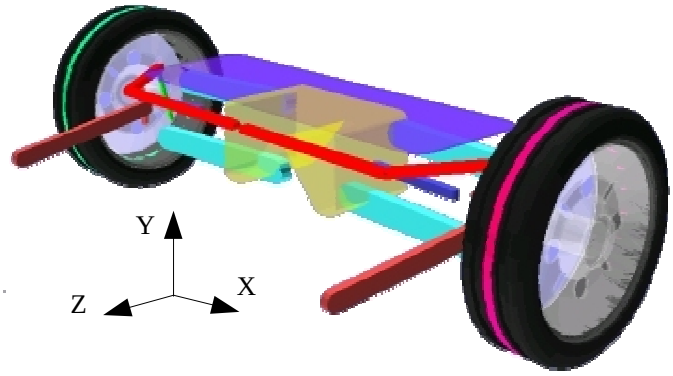

Fig. 2 : Essieu multibras dans DT−VAM.

2.2 − Sollicitations appliquées

Après discussion avec des spécialistes, il est apparu que notre outil métier devrait être capable de soumettre chaque suspension à *4 cas de charges typiques* :

- Débattement vertical simultané : les roues sont animées d'un déplacement vertical simultané d_Y selon l'axe *Y*, les autres mobilités étant bloquées.
  $$d_Y = -10 + 90.sin(t) \quad (1)$$
  La variable temps *t* évolue de *0* à *2*Π (soit une période) et d_Y est exprimé en millimètres.
- En roulis : la roue droite est soumise à un déplacement vertical d_{YD} en opposition de phase avec celui de la roue gauche d_{YG}..
  $$d_{YD} = -30.sin(t) \quad (2)$$
  $$d_{YG} = +30.sin(t)$$
- Sous effort latéral : un effort latéral F_X variable dans le temps est appliqué à la base de chaque roue (F_X en newtons).
  $$F_X = 3000.sin(t) \quad (3)$$
- En freinage : un effort longitudinal F_Z variable dans le temps est appliqué à la base de chaque roue pour simuler le freinage.
  $$F_Z = 4000.sin(t) \quad (4)$$

2.3 − Grandeurs caractéristiques

De même, il est apparu que les spécialistes analysent le comportement d'une suspension à l'aide de *4 grandeurs caractéristiques* :

- Variation de voie : c'est la variation de l'écartement transversal des roues (selon l'axe *X*).
- Braquage induit : lors de la sollicitation, c'est la rotation des roues selon l'axe *Y*, perpendiculaire à la route.
- Carrossage : c'est la rotation des roues selon l'axe longi−tudinal *Z*.
- Enroulement : c'est la rotation des roues selon leur axe de rotation *X*.

DT−VAM se doit donc de faire apparaître clairement ces 4 grandeurs, voire de les mettre en valeur par des moyens adaptés si leur amplitude s'avère trop faible.

2.4 − Un mécanisme innovant : la suspension OCP

La suspension OCP (Optimized Contact Patch), dont on voit le modèle CAO en Figure 3 est un type de suspension innovant. Il s'agit d'un mécanisme original impliquant conjointement la société Michelin et l'IFMA [5] et breveté récemment. Il assure une meilleure tenue de route en virage en inclinant les roues comme celles d'une moto, ce qui assure un meilleur contact pneu−sol que sur un train classique. Ce mécanisme comporte des pièces supplémen−taires, telles que les bielles rouges de la Figure 3. Ce train OCP nous servira par la suite d'exemple de démonstration.

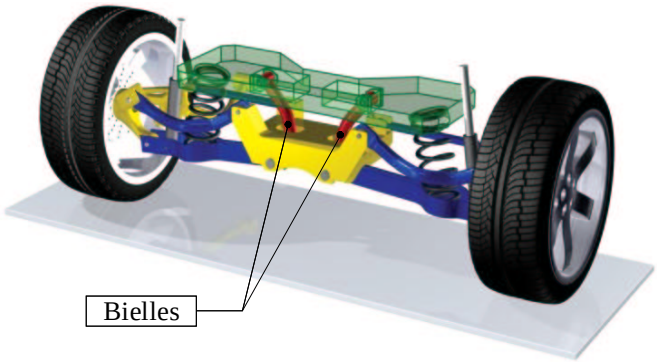

Fig. 3 : Modèle CAO de la suspension OCP [5].

Comme de nombreux autres trains, le train OCP comporte des articulations rigides mais aussi des articulations élastiques (*bushings*), qui devront être prises en compte dans la modélisation mécanique.

2.5 −Cahier des charges final

En guise de synthèse, nous pouvons dégager un cahier des charges comportant les contraintes suivantes :

- C_1 : Développer un tutoriel portable, i.e. exécutable par les ordinateurs ayant accès à l'intranet ;
- C_2 : Assurer une grande simplicité d'utilisation et donc intégrer un contenu visuel, 3D, animé et interactif ;
- C_3 : Concevoir une solution ne nécessitant pas de procédure d'installation particulière et notamment celle d'un *plugin* (composant supplémentaire à installer manuellement sur chaque client web) ;
- C_4 : Veiller à la compatibilité avec les versions du navigateur web imposées (Netscape Communicator 4 ou plus) ;
- C_5 : Ne pas être trop gourmand en puissance de calcul et en mémoire ;
- C_6 : Limiter la taille des fichiers et le flot de données à transmettre à une bande passante raisonnable.

Mis à part C_2 , contrainte pédagogique, toutes les autres contraintes sont implicitement liées à la difficulté de partager simplement une information 3D entre un grand nombre de personnes au sein d'une entreprise vaste. C'est

certainement la contraint C_3 concernant l'interdiction d'installer un quelconque plugin qui a le plus pesé sur les choix techniques effectués.

3− Choix des composants logiciels

3.1− Choix généraux

Nous nous sommes donc décidés à réaliser un tutoriel sous forme de site web, avec d'une part un contenu statique et informatif en HTML, et d'autre part un contenu interactif apporté par une maquette virtuelle. Celle−ci doit comprendre les éléments suivants :

- une fenêtre graphique contenant le mécanisme virtuel ;
- la possibilité de naviguer grâce à la souris ;
- la possibilité de revenir à des vues prédéfinies ;
- la possibilité d'animer le mécanisme selon les 4 sollicitations désirées ;
- la possibilité d'afficher les 4 grandeurs caractéristiques désirées à l'aide d'indicateurs amovibles.

3.2− Choix de composants logiciels pour le rendu 3D

Ces fonctionnalités 3D à développer soulèvent évidemment un grand nombre de problèmes techniques. L'ensemble des technologies permettant d'ajouter un contenu 3D dans des pages WEB sans plugin est encore récent. Il a fallu les évaluer en terme de pérennité et de potentialités (modifica−tions ultérieures, ajout d'options, optimisations, etc.)
Un travail de veille technologique a permis de faire un bilan sur les outils de développement informatique pouvant répondre aux exigences du client et à nos attentes.

1.1.2 − Composants nécessitant un plugin

- Visionneuses VRML

VRML (Virtual Reality Modeling Language), est un langage normalisé qui permet la création et la diffusion de mondes virtuels et interactifs en 3D sur Internet. La dernière norme en date s'intitule VRML 97 (ISO/IEC DIS 14772−1) [6] et a été élaborée par le Consortium Web 3D [7]. VRML est basé sur un format textuel de fichiers pour la description d'objets et de scènes 3D. Ce format neutre est indépendant de la plate−forme. De nombreux programmes de CAO ou de modélisation 3D peuvent exporter leurs données en format VRML.

Une visionneuse VRML est habituellement nécessaire pour visualiser une scène VRML. C'est un logiciel qui traduit un fichier VRML en une représentation 3D et calcule les images de la scène vue par l'utilisateur. Il peut être intégré dans un navigateur web (plugin) ou fonctionner comme un logiciel indépendant. Nous avons évalué deux navigateurs VRML [8, 9].

Pour DT−VAM, la contrainte C_3 de ne pouvoir procéder à aucune installation logicielle nous a évidemment conduits à chercher des solutions autres que les visionneuses VRML. Par contre, nous avons cependant privilégié l'utilisation du format VRML pour modéliser tous nos mécanismes. Ce standard de fait s'avère mature, riche en possibilités et largement répandu. Les mécanismes modélisés dans DT−VAM sont donc tous strictement conformes à la norme VRML 97.

- Composants écrits en Java 3D

Java 3D est une bibliothèque complémentaire du langage de programmation Java destinée à écrire des programmes capables de simuler un environnement graphique 3D interactif [10]. Comme tout programme Java, le programme résultant fonctionne sur bon nombre de plates−formes matérielles et peut prendre la forme d'une petite application intégrable dans une page web (*applet*). Java 3D fournit une collection de classes de haut−niveau pour créer, manipuler et présenter un rendu graphique d'une géométrie 3D.

Voici quelques caractéristiques intéressantes pour nous :

- Modélisation et affichage 3D de scènes virtuelles avec tous les raffinements de l'informatique graphique (brouillard, anti−crénelage, etc.) ;
- Existence de méthodes pour importer des fichiers au format VRML (parmi de nombreux autres formats) ;
- Représentation d'une scène 3D sous forme d'un *graphe scénique*, agencement d'objets 3D organisés sous forme arborescente inversée, spécifiant le contenu de l'univers virtuel et comment il doit être rendu ;
- Possibilité d'implanter des comportements spécifiques programmés (*behaviors*) ;
- Notion d'*interpolateurs* : possibilité de lier les variations d'une variable à l'horloge du système ;
- Détection des collisions entre objets ;
- Sélection d'objets et gestion de périphériques de saisie.

En conclusion, Java 3D comporte de nombreux avantages mais ne pourra pas être retenu pour développer DT−VAM car cette bibliothèque n'existe que sous forme de plugin. Par contre, le langage Java nous servira de liant entre composants et s'avérera précieux par sa portabilité.

1.1.2 − Composants sans plugin

- Norme X3D

X3D est le nom donné au projet coordonné par le Consortium Web3D [7] pour créer un outil de développement 3D reposant sur la technologie Java et ne nécessitant pas l'installation de plugin [11]. Il repose sur une partie des standards définis pour VRML 97 et reprend bon nombre des fonctionnalités de Java 3D.

- Outils basés sur la norme X3D

Les outils de développement suivants sont basés sur les spécifications X3D [12, 13, 14, 15]. Après comparaison de ces différents outils, notre choix s'est finalement porté sur Blaxxun 3D [13], une bibliothèque intéressante sur de nombreux points :

- qualité du rendu ;
- faible taille des bibliothèques ;
- gestion des horloges et des animations ;
- gestion des zones sensibles et de la souris ;
- possibilité d'importer les fichiers VRML compressés ;
- compatibilité avec les principaux navigateurs du marché (dont Netscape) ;
- documentation fournie, simplicité d'apprentissage ;
- gratuité.

On signalera également quelques limitations :

- absence des fichiers sources ;

- performances moins élevées qu'avec Java 3D ;
- quelques fonctionnalités manquantes.

Néanmoins, Blaxxun 3D s'est avéré être le composant le plus adapté à nos besoins.

4− Démarche de création de l'application

Les choix de composants logiciels étant faits, voici maintenant la démarche pour créer un nouveau modèle de mécanisme dans DT−VAM.

4.1 − Modélisation mécanique simplifiée

On commence par réaliser un modèle mécanique simplifié du mécanisme à l'aide d'un logiciel de dynamique (dans notre cas, ADAMS [16]). On veille à ce que le modèle intègre les efforts et / ou déplacements imposés par les sollicitations (Figure 4).

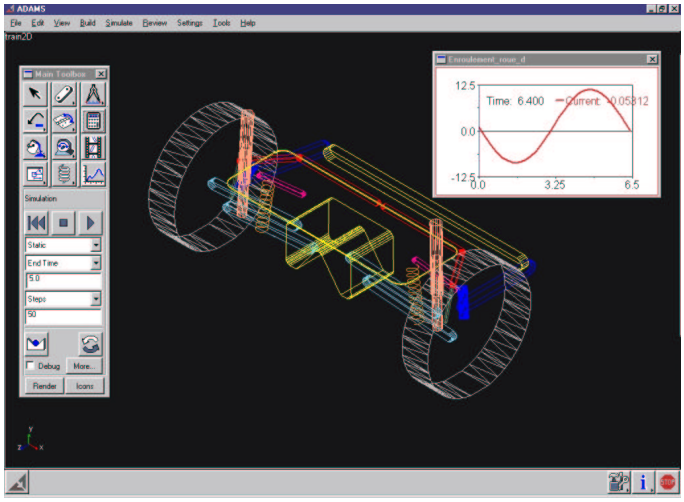

Fig. 4 : Modèle mécanique de la suspension.

On crée alors des marqueurs (repères locaux) attachés à chaque pièce afin de récupérer les valeurs de position et d'angles dans chaque configuration intermédiaire calculée par le solveur dynamique. Nous avons opté pour une discrétisation du mouvement en 33 positions, précision nécessaire pour espérer apercevoir les effets non linéaires des *bushings*. Il ne reste plus alors qu'à exporter les données sous forme d'une table de valeurs en format textuel. Diverses manipulations automatisables permettent de transformer les données brutes issues d'ADAMS en arguments correctement formatés pour les commandes « PositionInterpolator » et « OrientationInterpolator » à insérer dans les fichiers VRML d'assemblage du mécanisme. Un grande vigilance sur les conversions d'unités (distances et angles) est recommandée.

4.2 − Modélisation géométrique

Dans un second temps, la géométrie du modèle simplifié est éventuellement affinée. L'opération consiste, via un format neutre tel qu'IGES, à importer des formes depuis ADAMS dans un logiciel de CAO (en l'occurrence Solid Edge) pour les retravailler. On peut également repartir de zéro, comme dans le cas des roues présentées en Figure 5. Une fois la géométrie figée, on passe à l'exportation au format VRML.

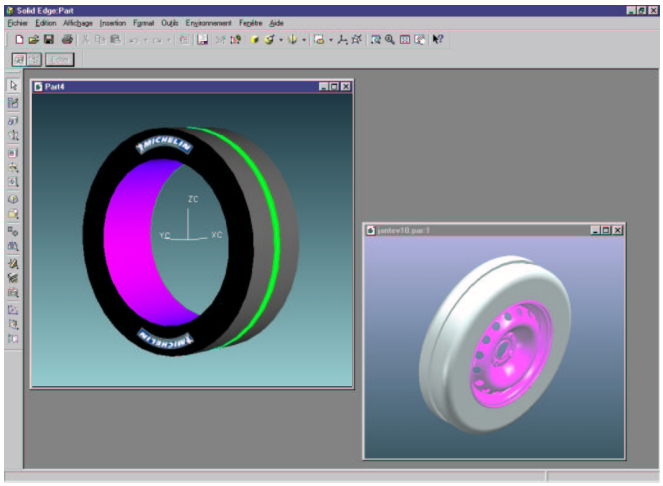

Fig. 5 : Modélisation géométrique d'une roue.

Notons que la conversion s'opère pièce par pièce car la plupart des logiciels de CAO exportent un assemblage comme un VRML monolithique impossible à animer. La création du fichier VRML d'assemblage s'opère en manuel. Ce fichier contient :

- La définition des points de vue ;
- Le graphe scénique obtenu en imbriquant les différentes pièces de l'assemblage à l'aide des mots−clés VRML « children » et « Inline ». Ce dernier permet d'importer directement un fichier VRML externe dans l'assemblage ;
- Les tables d'interpolation.

Il est important de veiller à la bonne orientation et position du modèle CAO par rapport au repère global, car on la retrouve généralement inchangée dans le fichier VRML. Il ne faut pas tomber non plus dans le piège consistant à dessiner des formes trop complexes ou inutilement riches en détails. Par exemple, des entités CAO telles que les congés génèrent un grand nombre de facettes minuscules dans le fichier VRML, ce qui l'alourdit inutilement. On pourra également utiliser avec profit des logiciels permettant de délester le fichier VRML de ses facettes inutiles. Le format VRML 97 compressé (avec *gzip*) est également recommandé. Dans le cas de la suspension OCP traité plus loin, nous avons choisi de délibérément supprimer les jantes des roues pour alléger le modèle et le rendre plus lisible. On peut également jouer sur les transparences de certaines pièces pour rendre les assemblages plus compréhensibles. Enfin, il est possible de rajouter des textures, comme on peut le voir sur les flancs de la roue (Figure 5).

4.3 − Programmation

Cette phase étant terminée, il reste à lier toutes ces données par quelques développements complémentaires. La Figure 6 représente les différents modules mis en oeuvre ainsi que le langage informatique associé :

- L'utilisateur communique avec le programme via une interface graphique écrite en Java AWT ;

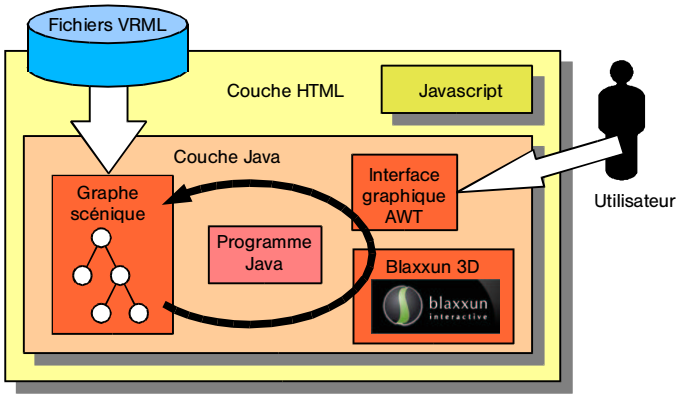

Fig. 6 : Architecture logicielle de DT−VAM.

- La fenêtre graphique est prise en charge par Blaxxun 3D, qui est capable d'importer dans le graphe scénique de l'application des fichiers VRML externes via un filtre d'import
- Le programme écrit en Java a pour rôle d'interconnecter les différents composants et notamment d'assurer la liaison entre la structure de données issue du graphe scénique VRML et les commandes Blaxxun.
- Un petit module écrit en Javascript permet de lancer des fonctions annexes (choix de la résolution graphique, menus d'aide, etc.)
- Le tout est encapsulé dans une page HTML qui intègre l'applet.

Le programme Java est doté de nombreuses caractéristiques :

- C'est une application multi−processus :
  L'applet développée est capable de gérer plusieurs tâches simultanées. Cela permet à la fois de scruter les événements de rendu émis par le navigateur Blaxxun, les événements liés à la navigation avec la souris, les événements liés à l'interface de contrôle et l'affichage des courbes dynamiques.
- Elle comporte divers observateurs d'événements :
  L'utilisation de ces objets permet de détecter le chargement d'un fichier, l'appui sur un bouton, les clics de souris, le changement des options d'un menu, etc. Ils détectent tous les changements qui peuvent intervenir dans le modèle 3D et émettent une information en retour qui sera interprétée et permettra au programme de réagir en conséquence. Les détecteurs d'événements détectent deux types d'événements :
  − événements dans l'interface utilisateur AWT (appui de boutons) ;
  − événements dans le monde VRML (changements de vue, démarrage ou arrêt des horloges d'animation, clic sur des zones sensibles, etc.)
- La gestion des horloges et le déclenchement contrôlé d'événements :
  L'applet Java prend en charge la gestion des horloges créées dans les fichiers VRML et associe à leur déclenchement l'animation des différentes simulations enregistrées.
- La gestion des noeuds et des champs du graphe scénique :
  Grâce aux méthodes incluses dans la librairie Blaxxun 3D, l'applet Java est en mesure de gérer les noeuds du graphe scénique, autrement dit, elle est en mesure de manipuler les objets affichés dans l'univers virtuel. Il est alors possible de contrôler leur position, leur orientation, leur échelle et leur apparence. C'est par la manipulation des noeuds relatifs aux différents points de vue enregistrés, que l'utilisateur peut observer sous des angles différents le train de suspension. Il en est de même pour les animations qui sont réalisées grâce aux noeuds contenant les différentes tables d'interpolation en orientation et en position.

5− Résultats

Au bilan, on obtient une page web intégrant du contenu 3D animé, comme on peut le voir sur la Figure 7. L'interface de DT−VAM comporte les éléments suivants :

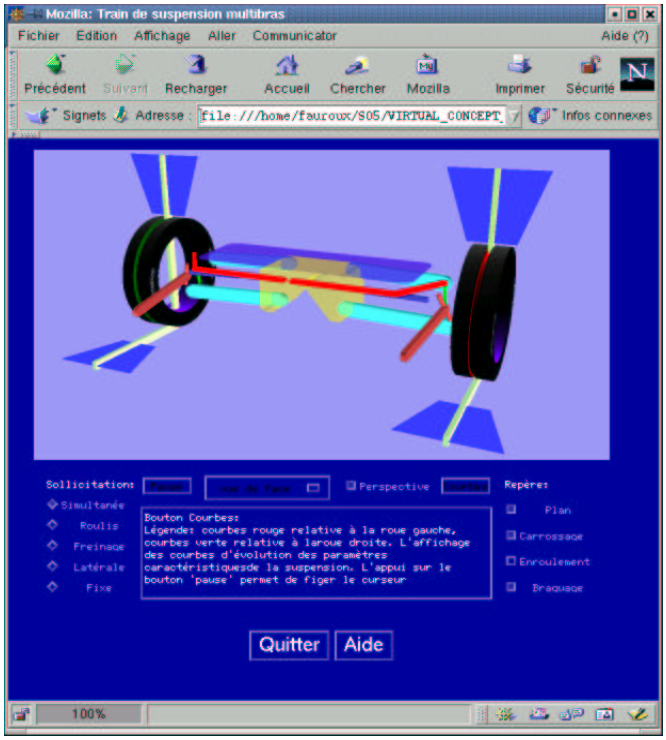

Fig. 7 : Interface de DT−VAM affichant un train OCP.

- une fenêtre 3D pilotée à la souris (rotations activées par défaut, zoom avec la touche Maj, translations avec la touche Ctrl) ;
- 4 boutons radio permettant de choisir un type de sollicitation à appliquer ;
- un bouton « Pause » permettant de figer l'animation ;
- une liste déroulante offrant plusieurs vues prédéfinies (face, dessus, droite, isométrique, etc.) ;
- 3 boutons à cocher permettant d'activer / désactiver l'affichage de trois longues aiguilles vertes destinées à visualiser et amplifier les angles de carrossage, enroulement et braquage induit. A noter que les aiguilles suivent la roue dans ses translations, mettant ainsi mieux

en évidence les rotations. Au niveau du VRML, l'affichage et la suppression des aiguilles se fait au moyen de la commande « switch », qui permet de rajouter ou enlever des élément au graphe scénique ;

- un bouton à cocher permettant d'activer / désactiver l'affichage de plans destinés à rendre plus lisibles les déplacements des aiguilles ;
- un bouton permettant d'afficher une fenêtre complémentaire représentant les courbes d'évolution des grandeurs caractéristiques au cours du temps, avec un curseur animé indiquant l'instant *t* de l'animation (Figure 8) ;
- une fenêtre d'aide en ligne pour rappeler les principales fonctionnalités et options disponibles.

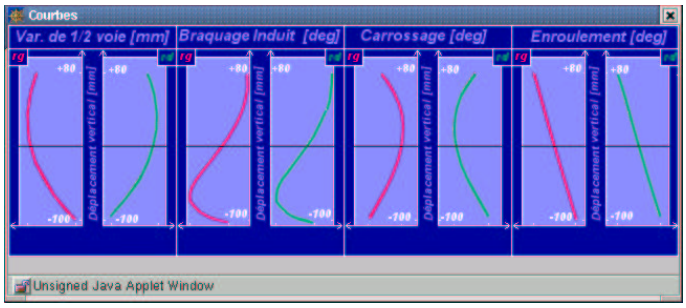

Fig. 8 : Fenêtre des courbes d'évolutions.

D'un point de vue concret, DT−VAM se présente comme un répertoire d'environ 1200 Ko contenant divers fichiers et qu'il suffit d'installer sur un serveur web. Les fichiers VRML occupent environ 800 Ko, les plus gros d'entre eux étant sans conteste les fichiers d'assemblage, lourdement chargés en valeurs d'interpolation. Le code source Java pèse environ 100 Ko pour 2000 lignes de code, les binaires résultants occupant environ le même espace. Le code HTML reste extrêmement léger, avec moins de 10 Ko occupés.
Lors de l'exécution, l'animation reste fluide sur la plupart des machines, même modestes (de la catégorie d'un Pentium II à 300 Mhz). On note cependant des ralentissements possibles selon le navigateur web utilisé.

6− Conclusion

Nous avons présenté le tutoriel DT−VAM destiné à faciliter la compréhension de mécanismes complexes dans un environnement web 3D distribué. Actuellement, DT−VAM comporte deux mécanismes de suspensions (trains OCP et multibras) dans sa bibliothèque, qui est naturellement extensible.

Voici plusieurs différences notables entre DT−VAM et une visionneuse VRML classique :
- il s'agit d'un programme exécutable très léger ;
- aucun besoin d'installation préalable ;
- animation multi−corps ;
- intégration de résultats de calculs de dynamique solide et élastique ;
- présence d'outils complémentaires pour illustrer les phénomènes mécaniques (aiguilles et plans de références amovibles, flèches animées d'application d'efforts) ;

Au nombre des limitations, la fluidité des animations n'est pas garantie pour de grandes fenêtres graphiques et le traitement des corps déformables ne semble pas simple en VRML (pour modéliser un essieu de torsion par exemple).

Une des perspectives les plus alléchantes consisterait à tirer parti des zones cliquables en VRML pour permettre des zooms contextuels sur certaines parties du mécanisme ou des déclenchements d'autres actions.

En conclusion, ce travail démontre la faisabilité et l'intérêt d'outils distribués de réalité virtuelle qui, par leur richesse graphique et leur contenu 3D, propagent efficacement la connaissance au sein même de l'entreprise.

7− Bibliographie

[1] Blanc O. and Jeanne F. Modélisation en réalité virtuelle d'une suspension automobile. Rapport de projet de 5ème semestre, IFMA, janvier 2001, 32 p.

[2] Ahmoy Y. Modélisation en réalité virtuelle d'un train de suspension automobile. Rapport de projet de fin d'études, IFMA, juin 2001, 120 p.

[3] Caekebeke L. and Deruy S. Modélisation d'une suspension en réalité virtuelle. Rapport de projet de 5ème semestre, IFMA, janvier 2002, 35 p.

[4] Halconruy T. Les liaisons au sol. Editions ETAI, ISBN 2−7268−8250−1, 1995

[5] Heuze L., Ray P., Gogu G., Serra L. and André F. Development of an optimised automotive axle. In IDMME'2002, Clermont−Ferrand, France, 14−16 mai 2002

[6] VRML 97, The Virtual Reality Modeling Language, International Standard ISO/IEC 14772−1:1997, www.web3d.org/Specifications/VRML97

[7] Web 3D Consortium homepage, www.web3d.org

[8] Cosmoplayer, Platinum Technologies Inc. www.cai.com/cosmo

[9] Cortona, Parallel Graphics Inc. www.parallelgraphics.com

[10] Java 3D homepage, java.sun.com/products/java−media/3D

[11] X3D draft specification, International Standard ISO/IEC xxxxx:200x, July 21, 2002, www.web3d.org/fs_x3d.htm

[12] Anfy 3D, Anfy Team Inc. www.anfyteam.com

[13] Blaxxun3D, Blaxxun Interactive Inc. www.blaxxun.com

[14] Shout3D, Eyematic Inc. www.shout3d.com

[15] Sumea Inc. www.sumea.com

[16] ADAMS, MSC Software Inc., www.adams.com