

# Les composants Asphyre

par Pierre RODRIGUEZ (Pedro) ([Pages de Pedro](#))

Date de publication : 22 octobre 2005

Dernière mise à jour : 18 décembre 2005

Nombreux sont ceux qui, comme moi, ont été déçus par DelphiX lorsqu'ils ont voulu utiliser DirectX sous Delphi.

Aide incomplète, erreurs dans l'adaptation en Delphi, bref ce ne sont pas les problèmes qui manquent... Et pour couronner le tout, son développement est arrêté depuis bien longtemps.

La dernière alternative qui s'offrait alors aux développeurs désireux de s'essayer aux joies de la 3D était GLScene qui encapsulait OpenGL.

Heureusement, est apparu il y a peu de temps: **Asphyre Pro**.

Une série de composants encapsulant parfaitement DirectX et ce, dans sa version 9.0 s'il vous plait!

Ces composants sont vraiment très efficaces. Et leur utilisation est très aisée car leur structure est bien pensée. Voici donc un tour d'horizon des possibilités de ces composants.

Pour réagir et commenter cet article, [cliquez ici](#).

- 1 - Introduction
- 2 - La logique
- 3 - Les composants de base
  - 3.1 - TAsphyreDevice
  - 3.2 - TAsphyreCanvas
  - 3.3 - TAsphyreTimer
  - 3.4 - TVTDb
  - 3.5 - TAsphyreImages
  - 3.6 - TAsphyreFonts
  - 3.7 - TAsphyreKeyboard, TAsphyreMouse et TAsphyreJoysticks
- 4 - Les composants 3D
  - 4.1 - TAsphyreLights
  - 4.2 - TAsphyreCamera
  - 4.3 - TAsphyreModels
  - 4.5 - TAsphyreMaterials
  - 4.6 - TAsphyreTextures
  - 4.7 - TAsphyreFog
  - 4.8 - TAsphyreParticles
- 5 - Conclusion
- 6 - Téléchargement et liens utiles
- 7 - Remerciements

## 1 - Introduction

La collection de composants Asphyre permet aux développeurs Delphi d'utiliser DirectX très facilement dans leur application.

Jusqu'à maintenant, le seul moyen d'utiliser l'API DirectX *simplement* était d'utiliser DelphiX. Un package de composants dont la dernière version (2000.07.17) commençait à dater un peu. D'autant que cette version ne prenait en charge que la version 7 de DirectX.

DelphiX proposait une série de composants qui encapsulait DirectX et facilitait grandement le développement des applications 3D et 2D.

Au départ, **Asphyre Pro** a été développé par 2 personnes : Yuriy Kotsarenko et Humberto Andrade.

La nouvelle version de Asphyre (Extreme) est développée uniquement par Yuriy Kotsarenko. Celui-ci est un ukrainien qui vit au Mexique actuellement. Il prépare un master en imagerie informatique.

Il a donc développé ces composants tout d'abord en version Pro. A l'heure où vous lisez ces lignes, la version Extreme est en préparation et bientôt finalisée.

Les seules choses qu'on puisse reprocher à Asphyre sont :

- L'aide n'existe pas (en cours de rédaction pour la version Extreme)
- Les composants conteneurs de ressources ne sont pas protégés (corrigé dans la version Extreme)
- Le code n'est pas très propre (programmé plus ou moins dans l'urgence selon les dires du créateur)

Les points forts sont:

- Une encapsulation très bien pensée
- Des outils très efficaces (image, police, etc..)
- Des fichiers d'exemple très explicites
- Reconnaissance de nombreux formats d'image
- Et j'en oublie d'autres...

 *La version Pro est maintenant remplacée par la version Extreme. Pour la télécharger, reportez-vous à la **Section 6 : Téléchargements et liens utiles***

*Un article sera bientôt disponible sur ma page d'accueil.*

## 2 - La logique

Lorsque l'on utilisait DelphiX, l'affichage se faisait sur un composant nommé TDXDraw. Le système utilisé par Asphyre est très proche de DelphiX à ceci près qu'il n'y a pas de TDXDraw, autrement dit, pas de surface sur laquelle on peut dessiner directement.

Le composant principal est le **TAsphyreDevice**.

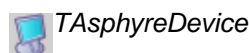
C'est lui qui centralise tout. C'est avec ce composant que l'on définit l'affichage et quelques autres options... Grâce à ce composant, on peut, soit afficher directement sur la TForm, soit afficher dans un TPanel par exemple en spécifiant son Handle.

Chaque composant utilise le **TAsphyreDevice** pour fonctionner.

## 3 - Les composants de base

Voici donc une présentation sommaire des composants les plus importants.

### 3.1 - TAsphyreDevice



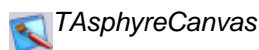
Comme je le disais plus haut, c'est le composant central à tout projet utilisant Asphyre.

Un tel projet nécessite systématiquement la présence de ce composant. Il permet de paramétrer:

- Plein écran ou non
- La taille (hauteur et largeur) de la zone d'affichage
- L'activation ou non des fonctions Transform and Lightning de votre carte vidéo (prise en charge des fonctions de calculs 3D plus poussés normalement dédiés au microprocesseur)
- La zone d'affichage (propriété **WindowHandle** : laisser **0** si la zone d'affichage est la TForm)
- etc.

Les composants qui affichent des images, des textes ou des objets sont directement liés à ce composant. Et c'est par son intermédiaire que se fait l'affichage.

### 3.2 - TAsphyreCanvas



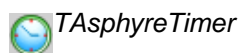
Voilà donc le canvas sur lequel on va pouvoir dessiner.

Même si ce composant n'est pas un dérivé de TCanvas, la plupart des fonctions qui s'y trouvent ressemblent beaucoup aux fonctions de ce dernier. On retrouve ainsi les fonctions Rectangle, LineTo, FillRect, Etc

Toutefois, il faut noter que la plupart de ces fonctions ont des paramètres supplémentaires. Par exemple, il est possible (et très facile) de dessiner un rectangle avec un liseré noir opaque et une couleur blanche transparente en son centre. On trouve de même des effets d'anti-aliasing (effet anti escaliers: sorte de flou cachant les contrastes trop forts entre pixels) pour dessiner des ellipses.

Il faut noter également le principe assez particulier du TAsphyreCanvas: lorsque l'on dessine une forme, on enregistre en fait la forme et ses paramètres dans une sorte de base de données. Lorsque le *canvas* est dessiné, la base de données est parcourue et chaque entrée est dessinée.

### 3.3 - TAsphyreTimer



Les utilisateurs reconnaîtront ce fameux Timer si particulier. Pour les autres, il s'agit d'un Timer plus précis que le TTimer de la VCL. Ce Timer est utilisé pour les traitements des entrées de l'utilisateur (clavier, souris et joystick(s)), pour le calcul des positions des différents objets de la scène à un moment  $t$  et d'afficher le résultat. Il s'utilise *presque* de la même façon que le TDXTimer de DelphiX. A une différence notable près, on y trouve 2 événements:

- OnTimer
- OnRender (OnProcess sur la version Extreme)

L'événement OnTimer permet de procéder à tous les changements de la scène. Incrémentation de compteur, changement d'état, etc. L'événement OnRender, quant à lui, sert à **afficher** le rendu. Cette différenciation permet d'éclaircir son code et de séparer les traitements. A l'utilisation, cela se révèle fort pratique, car à l'époque de DelphiX, l'événement OnTimer du TDXTimer devenait très vite encombré et difficilement déchiffrable.

### 3.4 - TVTDb



Ce composant est d'une simplicité et d'une efficacité redoutables.

C'est un conteneur de ressources. A ceci près qu'elles sont chargées depuis un fichier de type .vtdb.

Des utilitaires très bien faits sont fournis dans le pack Asphyre Pro pour créer un tel fichier.

On peut enregistrer deux types de données: Des images et des polices.

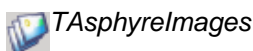
- Les images: les formats PNG, JPEG, TGA, BMP, DIB sont reconnus. A l'aide de l'outil d'import d'images, on peut paramétrer la taille du pattern, la taille de la texture, les channels utilisés, etc.
- Les polices: En utilisant l'utilitaire **Font Tool**, on peut créer une ressource à partir d'une police existante avec les mêmes paramètres que les images

A noter un gros avantage pour les polices : vous pouvez créer une ressource à partir de n'importe quelle police sans vous soucier du déploiement de votre programme : les polices choisies sont "intégrées" dans le fichier vtdb.

La seule contrainte est de fournir le fichier vtdb qui contient toutes les données...

Remarque: la version Extreme prévoit de protéger les données contenues dans un **TVTDb** par un mot de passe ce qui interdira toute copie de vos ressources.


### 3.5 - TAsphyreImages



Ce composant sert à utiliser des images chargées depuis un **TVTDb** ou depuis un fichier.




Il contient des **TAsphyreImage** qui ont chacune des propriétés pratiques pour l'affichage en texture ou directement sur le **TAsphyreCanvas**.

### 3.6 - TAsphyreFonts

 *TAsphyreFonts*

De la même façon que le **TAsphyreImages**, le **TAsphyreFonts** est un composant créé pour récupérer et utiliser les polices définies dans le **TVTDb**.

### 3.7 - TAsphyreKeyboard, TAsphyreMouse et TAsphyreJoysticks

 *TAsphyreKeyboard* *TAsphyreMouse* *TAsphyreJoysticks*

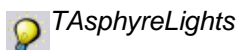
Ces 3 composants permettent de gérer les entrées du clavier, de la souris et des joysticks en utilisant DirectInput.

## 4 - Les composants 3D

Voici une présentation sommaire de quelques composants permettant d'afficher de la 3D.

A noter toutefois que la plupart de ces composants ne sont que des conteneurs d'objets.

### 4.1 - TAsphyreLights



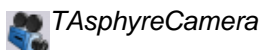
Ce composant sert à contenir les lumières d'une scène 3D.

Lors de la création de la scène, il suffit alors de les ajouter une par une (**TAsphyreLight**) et de les paramétrer.

Leur nombre maximum est autant que la mémoire vive peut en contenir mais en réalité, ce nombre est limité par la carte vidéo. Dans la plupart des cas, on peut donc en mettre 8.

Chaque lumière peut entièrement être paramétrée. On peut définir son type (omni, directionnelle, etc.), sa couleur diffuse, spéculaire, etc. Ainsi que sa position et, éventuellement, sa cible si la lumière est du type directionnelle.

### 4.2 - TAsphyreCamera



Comme son nom l'indique, ce composant représente la caméra de la scène.

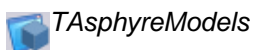
Evidemment, on peut ajouter autant de caméra que l'on veut. Mais à la différence des **TAsphyreLights**, **TAsphyreMaterials**, etc. ce composant n'est pas un conteneur : il représente directement la caméra.

Pour afficher un objet vu d'une caméra, il suffit de faire:

```
AsphyreModels[0].Draw(CubeMtx, AsphyreCamera);
```

Sachant que `AsphyreModels` est un **TAsphyreModels**, que son index 0 contient un modèle 3D, que `AsphyreCamera` est un **TAsphyreCamera** et que `CubeMtx` est la matrice de transformation (translation et rotation) du modèle 3D dans l'index 0.

### 4.3 - TAsphyreModels



Ce composant sert à contenir les modèles 3D utilisés dans la scène...

Remarque : il est possible d'importer des fichiers de type .3ds (*3D Studio MAX* et *G-Max*) et Ascii (.ase). Ce qui permet donc d'utiliser les modeleurs 3D les plus courants sur le marché... Pour cela, il est fourni un outil: **ModelConv.exe** qui remplit cette fonction.

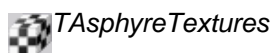
## 4.5 - TAsphyreMaterials



Ce composant contient tous les matériaux utilisés dans la scène.

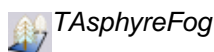
Les matériaux sont utilisés sur les modèles 3D que l'on affiche.

## 4.6 - TAsphyreTextures



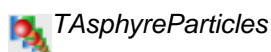
Ce composant contient toutes les textures utilisées dans la scène. Ces textures peuvent être aussi bien utilisées en 3D (en les associant avec des matériaux d'un **TAsphyreMaterials**) ou en 2D (en les dessinant directement sur un **TAsphyreCanvas**).

## 4.7 - TAsphyreFog



Ce composant permet d'afficher du brouillard volumétrique.

## 4.8 - TAsphyreParticles



Ce composant permet d'afficher des systèmes de particules.

## 5 - Conclusion

Ceci n'est qu'un très bref aperçu des possibilités qu'offrent les composants Asphyre. Et en aucun cas, cet article n'est exhaustif sur les composants et leurs propriétés.

Il sert uniquement à vous faire découvrir ce formidable travail qui permet -enfin- d'avoir correctement accès à DirectX sous Delphi.

Pour finir, je dirais que la version Extreme que je suis moi-même en train de tester est encore plus efficace.

Le code est revu de fond en comble et les composants sont beaucoup plus cohérents.

Il ne faudra plus appeler quelques fonctions pour afficher le contenu du **TAsphyreCanvas**. Une seule suffira pour l'ensemble.

Beaucoup d'autres changements sont au programme et malheureusement, beaucoup de nom de propriétés, de méthodes et d'événements seront modifiés pour des raisons d'homogénéité. Donc, les projet développés avec **Asphyre Pro** ne seront pas directement compatibles avec **Asphyre Extreme**. Toutefois, il suffit simplement de remettre le code à la bonne place. Et de ne rien oublier!

Bref je ne saurais que vous conseiller de vous faire la main sur la version Pro tout en attendant patiemment la sortie de la version Extreme, qui promet.

## 6 - Téléchargement et liens utiles

**Télécharger Asphyre Pro 1.1.0**

**Forum de discussion (en anglais) animé par Yuriy Kotsarenko (lifepower)**

**Forum Afterwarp section ressources**

## 7 - Remerciements

Je tenais à remercier spécialement le créateur d'Asphyre Pro et Extreme : Yuriy Kotsarenko qui est, non seulement d'une grande gentillesse mais aussi très disponible et j'espère que vous apprécierez, comme moi, son fabuleux travail.

Je voudrais aussi remercier **Giovanny Temgoua** pour son aide et ses encouragements pour écrire cet article ainsi que **Bestiol** et **Laurent Dardenne** pour leurs conseils avisés et leur corrections. Enfin, un dernier remerciement à **Nono40** pour son éditeur XML qui m'a rendu un fier service.



[Retour à l'accueil](#)

