# Les Ateliers de 3D Gamestudio

## Créez votre premier vol dans l'espace

par Nicholas CHIONILOS janvier 2001

Les dernières nouvelles, les démonstrations, les mises à jour et les outils, aussi bien que le Magazine des Utilisateurs, le Forum des Utilisateurs et le concours annuel sont disponibles à la page principale GameStudio http://www.3dgamestudio.com/

## Avant propos à propos de la traduction française de ce manuel

La traduction est un art difficile. D'autant plus qu'il m'a fallu apprendre le moteur au fur et à mesure de la traduction. Malgré tous mes efforts il est clair que tout n'est pas parfait. Je vous invite donc lorsque vous avez le moindre doute, un problème de compréhension voire un mauvais fonctionnement par rapport à ce que vous avez lu, à consulter le manuel en Anglais ou en Allemand, tous deux étant téléchargeables sur le site de Conitec et à me faire part de vos remarques. Et je réclame par avance votre indulgence.

Je vous souhaite le plus grand plaisir à lire ces lignes et à mettre en pratique ce que vous apprendrez.

Alain Brégeon
mailto:alainbregeon@hotmail.com

## Avant propos de la part de l'auteur

Cher lecteur
Bienvenue à l'atelier sur le vol dans l'espace! J'ai produit cet atelier pour aider les gens à créer leur propre jeu 3D dans l'espace avec 3D GameStudio. Cet atelier utilise des possibilités disponibles à partir de la version (4.22) ou supérieure.

Cet atelier, comme les autres ateliers avant cela, vise surtout les utilisateurs qui ont un peu d'expérience de 3DGameStudio. Je suppose que vous avez travaillé les différents tutoriaux et savez comment employer les outils (WED, MED et WDL).

Le scénario SPACESHP.WDL qui est fourni avec l'atelier a été conçu pour simuler un vaisseau spatial dans un environnement de gravité zéro. Il est fourni avec trois vues de caméra différentes, quatre drapeaux de contrôle différents et 5 paramètres différents qui peuvent tous être utilisés vous permettant beaucoup de flexibilité dans la personnalisation des caractéristiques de vol de votre vaisseau.

Ce texte complète la documentation qui va avec 3DGameStudio, et ne la remplace pas. Si quelque chose dans cet atelier est peu clair lisez s'il vous plait les manuels qui sont fournis avec 3DGameStudio. Je fais par avance des excuses pour des formulations que vous trouveriez peu claires, un code défectueux, des erreurs ou des omissions.

Pour conclure cet avant propos, je voudrais remercier plus particulièrement :
- Remi, pour m'avoir initié au concept des sphères de ciel
- Kyodai, pour sa généreuse donation du modèle de vaisseau utilisé dans cet atelier
- Vivi, pour sa 'peau' sympa sur le vaisseau
- JCL, pour sa donation de script de caméra de poursuite
- Doug, pour sa patience infinie et tous ses bons conseils.

J'espère que vous aurez beaucoup de plaisir avec cet atelier ! Vos questions ou vos commentaires peuvent être envoyés à Nicholas.Chionilos@Vertexconsulting.com et je ferai de mon mieux pour y répondre

Nick "WildCat" Chionilos.

# Sommaire :

# Avant de commencer :

## Obtenez la dernière version

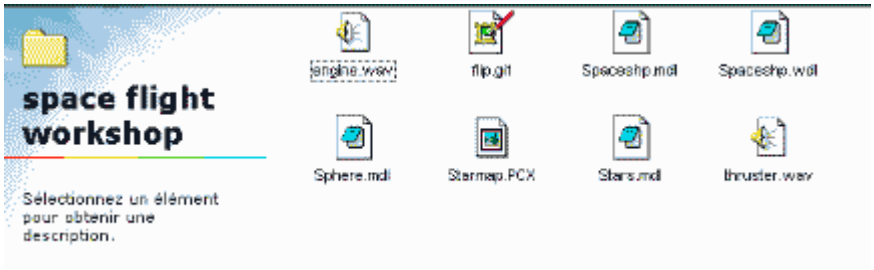
Avant de commencer, assurez-vous d'avoir la dernière version de 3DGameStudio (4.22 ou au-dessus) puisque nous allons nous servir de plusieurs des nouvelles particularités qui ont été ajoutées.

## Préparez votre espace de travail

Créez un dossier appelé " Space Flight Workshop" dans votre dossier GSTUDIO. C'est le répertoire ou vous stockerez tous les éléments du jeu.

La première chose que nous allons ajouter à notre dossier est le modèle de jeu et le sous dossier image. Si vous ne les avez pas déjà, allez à la page de téléchargement de Conitec (http: // www.conitec.net/a4update.htm) et récupérez le niveau de SPACE. Décompressez le contenu dans votre dossier. Votre dossier doit maintenant contenir au moins les fichiers suivants :

SPACESHP.WDL
SPACESHP.MDL
SPHERE.MDL
STARMAP.PCX
STARS.MDL
ENGINE.WAV
THRUSTER.WAV.



# Il est temps de commencer

## Vue d'ensemble :

La création d'un jeu spatial est un peu différente que de créer un jeu de tireur, un jeu de rôle ou même un simulateur de vol. La différence la plus grande est que le plus conventionnel des niveaux emploie une boîte de ciel stationnaire défini par WED pour englober le monde, tandis qu'un niveau spatial emploie un modèle de sphère d'étoiles créé dans MED qui se déplace comme le bateau se déplace pour englober le niveau.

Nous allons créer notre scène spatiale dans les étapes suivantes:
- Créer une sphère d'étoile pour définir à quoi votre ciel ressemble
- Créer un grand niveau dans WED
- Placer un vaisseau dans le centre du monde
- Y assigner l'action de PLAYER_SHIP

# Créer une sphère d'étoile

Une sphère d'étoile est simplement un très grand modèle de sphère qui créera la vue des étoiles que nous voyons quand nous sommes dans notre vaisseau spatial. Nous le faisons par les étapes suivantes :

- Créer un modèle de sphère
- Basculer les sommets sur les polygones pour tourner efficacement la sphère à l'intérieur
- Tailler la sphère à une taille appropriée
- Lui mettre une peau de scène spatiale

Tandis que la théorie pour créer une sphère d'étoile est très simple, il y a malgré tout certaines considérations pratiques à prendre en compte pour créer un regard professionnel de la carte d'étoile.

1) Plus vous emploierez de polygones dans votre sphère, plus vous verrez d'altération sur les bords de votre vue lorsque vous vous tournez dans votre jeu. Je recommande entre 500 et 1200 polygones.
2) Plus grande est votre sphère, plus les polygones qui la composent seront éloignés de la caméra et ainsi il y a moins de probabilité de voir les formes des polygones individuels.
3) Plus vos polygones individuels seront uniformes, moins il y aura de probabilités d'avoir des anomalies dans votre champ d'étoile

MED n'est pas nécessairement le meilleur produit pour commencer à construire une sphère à partir de zéro. La sphère primitive dans MED a trop peu de polygones et si vous augmentez le nombre de polygone avec l'outil de subdivision, vous obtenez des polygones de différentes formes.

Pour ces raisons, je recommande que soit vous importez une sphère d'un autre logiciel comme Milkshape ou plus simplement d'utiliser le SPHERE.MDL que j'ai inclus comme point de départ pour votre sphère d'étoile.

Assez parlé. Ouvrez votre éditeur MED et créons le ciel !

**Etape** 1: Démarrer avec une sphère dans MED.
Nous commencerons en chargeant la SPHERE.MDL inclus avec ce tutorial. Ce modèle a 720 polygones qui sont tous de la même forme et est donc un point de départ excellent pour notre sphère de ciel.

Ouvrez MED et sélectionnez File -> OPEN et sélectionnez SPHERE.MDL depuis votre répertoire “Space Flight Workshop”.
Votre écran doit ressembler à quelques choses comme cela :

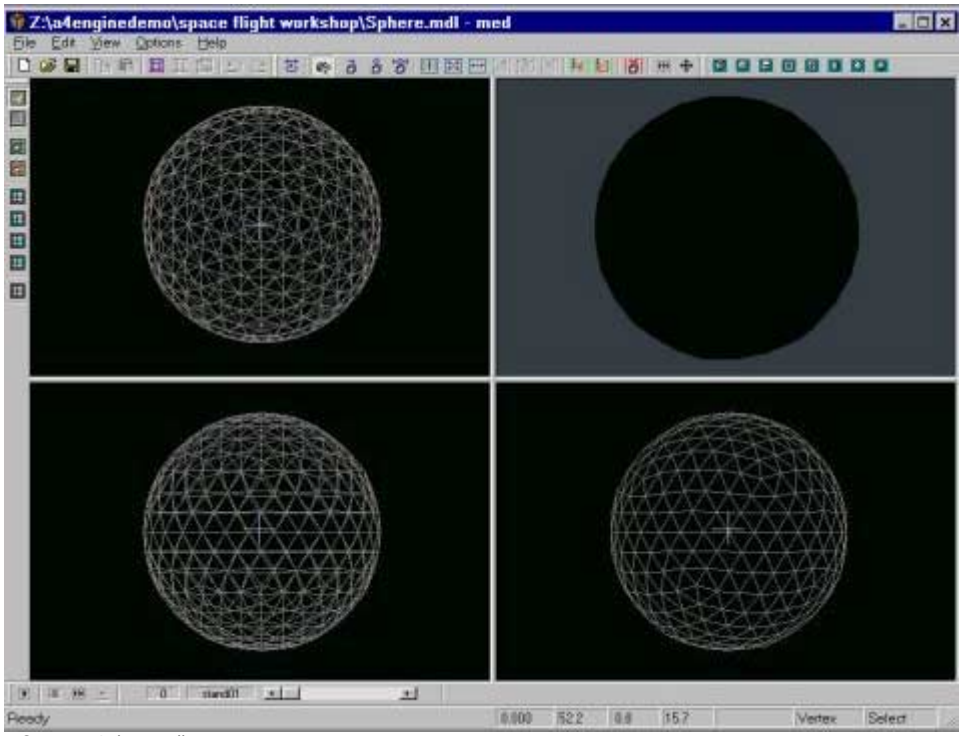
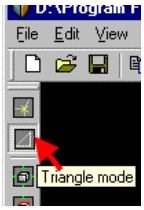



**FIGURE 1:** Sphere.mdl

! Avant de procéder, nous voulons être sur de bien travailler en mode triangle pour les sélections et les manipulations de notre modèle. Ceci peut être fait en cliquant sur cette icône




**Etape 2:** Menu Edit et "Select All"
Vous devez maintenant voir tous les polygones de la sphère mise en évidence en doré, comme dans la figure ci-dessous :

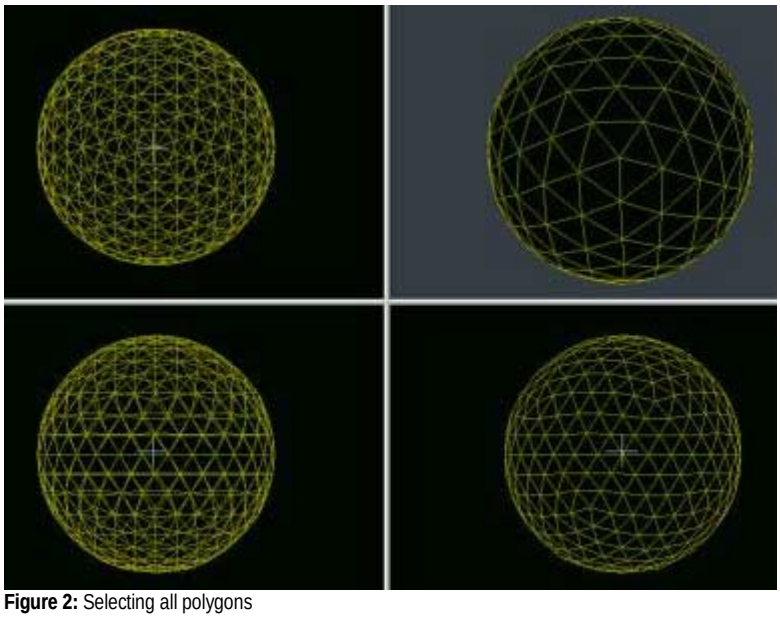
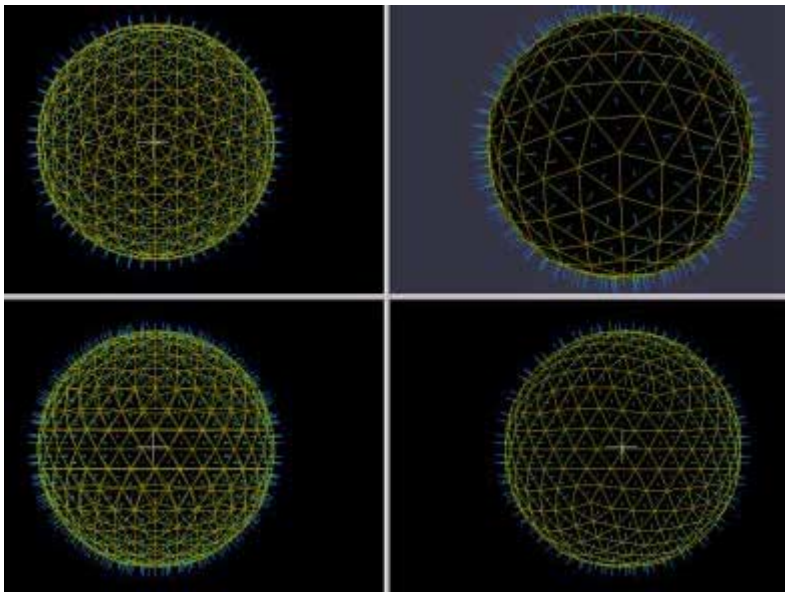


**Figure 2:** Selecting all polygons

**Etape 3:** Nous allons maintenant inverser les normales de nos polygones sélectionnés. Une normale est simplement la direction à laquelle le côté visible d'un polygone fait face. Maintenant visualisons les normales en choisissant :

Options -> Show Normals -> All.

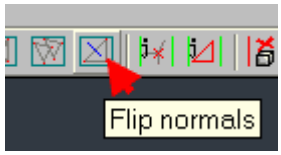
Votre résultat doit ressembler à cela :



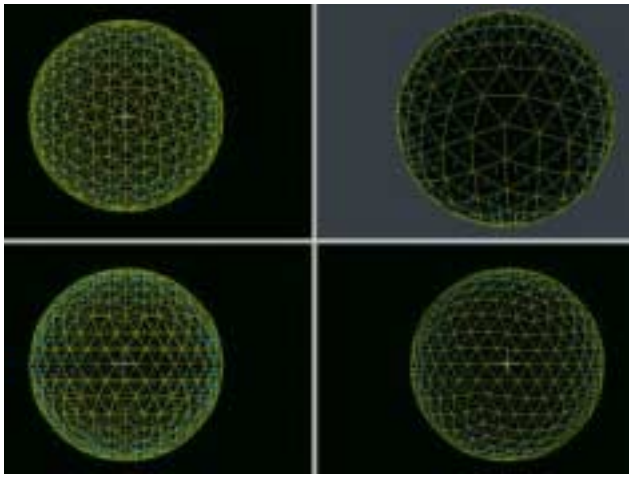
Notez que le modèle ressemble à un épineux à présent. C'est parce que chaque polygone a une petite ligne venant de son centre et se dirigeant loin du centre du modèle. Cela nous dit que tous les polygones sont actuellement désignés.

La raison pour que nous soyons si concernés par la direction que pointent ses normales est que la peau que nous appliquons à cette sphère sera seulement visible de la direction que les normales indiquent. Si les normales désignent l'extérieur, vous ne pourrez voir la peau de cet objet quand le regardant de l'extérieur. Cela aurait été excellent si nous avions voulu faire une planète que nous verrions seulement de l'extérieur.

Puisque notre vaisseau voyagera à l'intérieur de la sphère, nous devons inverser les normales pour qu'elles pointent vers le centre de la sphère. De cette façon nous serons capables de voir la peau qui est nos étoiles quand notre caméra est à l'intérieur du modèle. Faire cela en cliquant sur le bouton de "Flip Normals" sur la barre d'outil, comme indiqué ici :




Vous devez maintenant avoir un modèle de sphère avec les normales pointant vers l'intérieur comme ceci :



Vous pouvez maintenant retourner à Options -> Show Normals -> None. Cela vous redonnera votre vue habituelle.

**Etape 4:** Etirez le modèle pour l'agrandir. Je suggère environ 12 000 unités le long de chaque axe. Faites ceci en choisissant l'outil de Position dans la barre d'outil comme indiqué ici :


Dans la fenêtre gauche supérieure, clic droit, maintenez le bouton de souris enfoncé et traînez ensuite votre curseur vers le bas. Cela aura pour effet de tirer la caméra loin de votre sphère et donnera par conséquent un regard très petit. (Évaluez votre distance en mettant votre curseur à côté de le frontière droite de la vue supérieure. Assurez-vous que le déplacement Y est assez grand, autour de 12 000 comme indiqué dans la figure suivante :

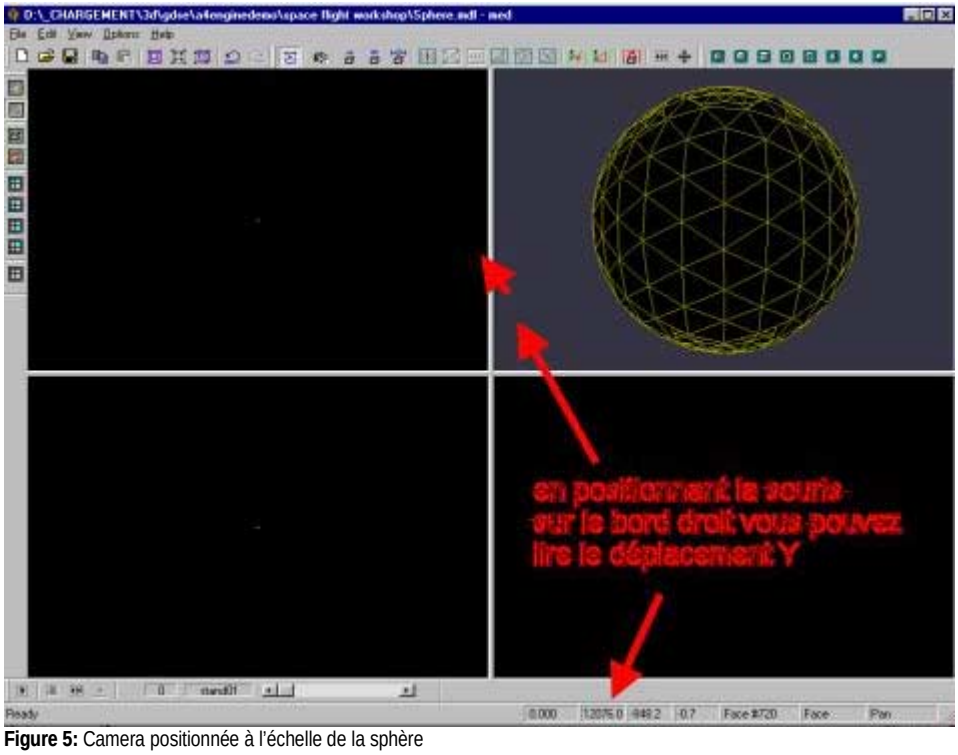
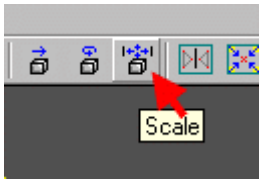



**Figure 5:** Camera positionnée à l'échelle de la sphère

Ensuite, choisissez l'outil d'échelle




La sphère entière doit toujours être sélectionnée, puis bouton gauche enfoncé et maintenu, glissez le curseur de souris à travers une des vues 2D jusqu'à ce qu'elle se remplisse.

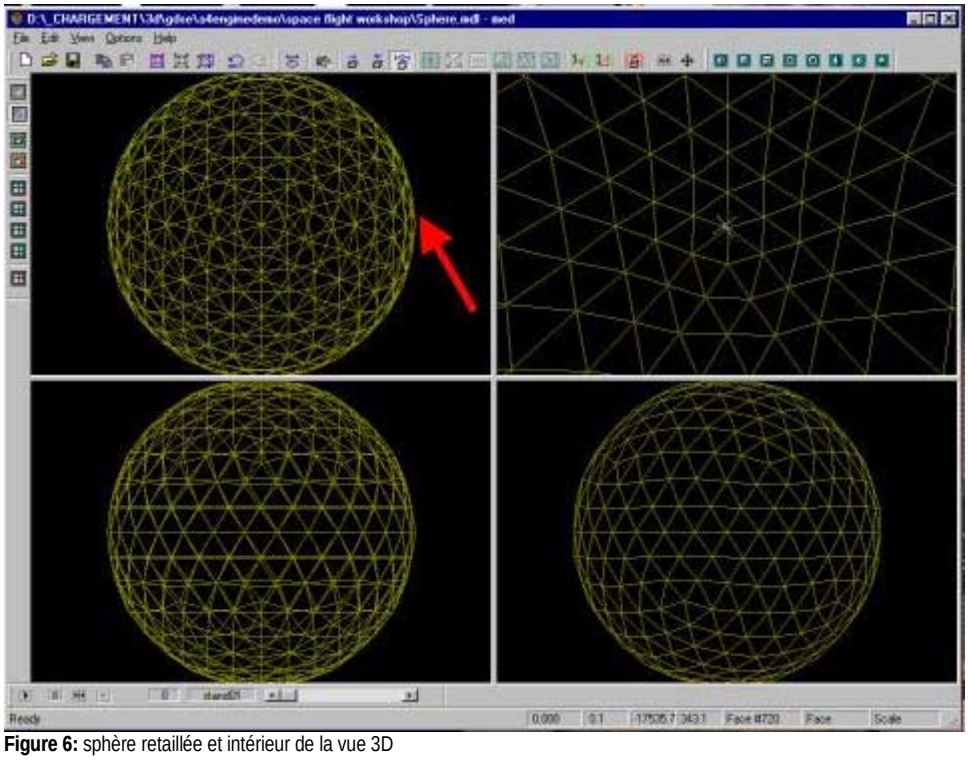


**Figure 6:** sphère retaillée et intérieur de la vue 3D

Remarquez que la vue 3D semble maintenant être à l'intérieur de la sphère! Remarquez aussi la flèche rouge mise à droite de la sphère dans la vue supérieure. Quand votre curseur est à la même position que cette flèche, l'indicateur de déplacement en bas de l'écran vous donnera une bonne évaluation de la taille de votre sphère. La taille spécifique n'est pas trop critique. Faites-la simplement relativement grande. Vous pouvez toujours retourner et l'ajuster plus tard après l'avoir essayée dans votre niveau.

**Etape 5**: Mettre une peau à la sphère!
Le pas final à la création de la sphère d'étoile doit lui créer une peau. Dans notre exemple, je vais employer le style de peau MDL pour économiser due temps. Selon la disposition de votre ciel, cela pourrait produire une altération considérable des étoiles à la couture entre l'avant et l'arrière. L'utilisation du style de peau MD2 règlerait cela. La sphère spatiale que j'ai incluse avec ce tutorial, "STARS.MDL" emploie le style de peau MD2. Pour en apprendre plus sur le style de peau MD2, je vous suggère de chercher "MD2 Peau" sur le forum utilisateur de Conitec ou de vérifier s'il n'y a pas un excellent tutorial sur le net via la page de liens de Conitec.

Pour créer notre peau style MDL:

1) Menu "VIEW -> Skins" pour faire apparaître l'éditeur de peau.
2) Dans l'éditeur de peau, cliquez sur Edit, puis "Resize Skin". Dans la boîte de dialogue, mettre la taille de votre fichier image que vous voulez utiliser pour votre peau. Dans notre exemple j'ai utilise 512x512. (Certaines vieilles carte vidéo comme les cartes Voodo 2 requièrent des images bitmap de 256 X 256. Essayez cette valeur si vous obtenez une erreur mémoire)
3) Cliquez sur Edit, puis "Create MDL Mapping". Sélectionnez "Front" et cliquez sur ok dans la boîte de dialogue. La carte de polygone de l'avant et de l'arrière de la sphère modèle doit être maintenant visible dans la vue de peau comme dans la figure suivante :

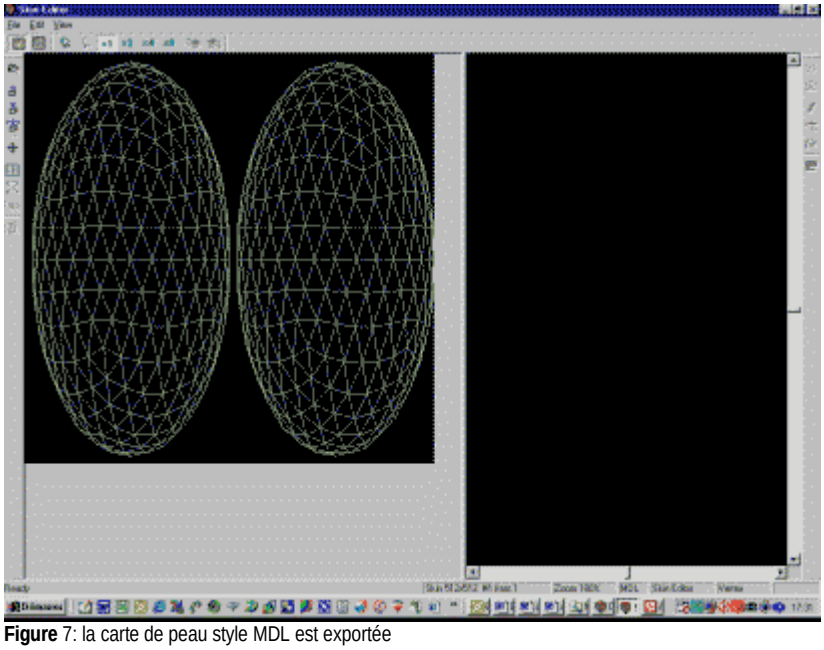
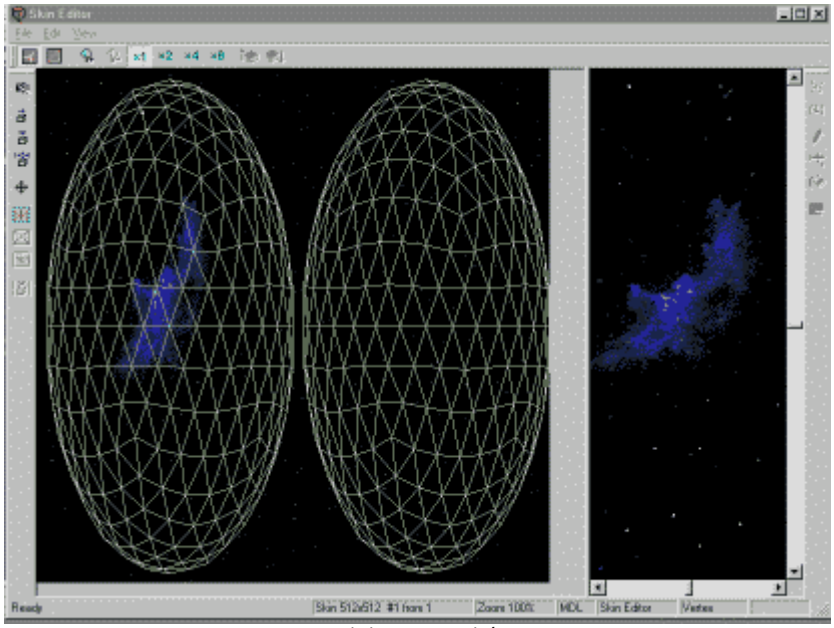


**Figure** 7: la carte de peau style MDL est exportée

4) Cliquez sur File, puis “Export” et ensuite “Current image to BMP’’ ou ‘’Current image to PCX’’ dépendant de votre préférence.
5) Sauvez votre carte de peau.
6) Ouvrez la carte de peau dans le programme de peinture de votre choix. Peignez-la tout en noir, avec une pluie d'étoiles et d'autres phénomènes célestes. (J'ai employé le fichier STARMAP.PCX, fichier qui est inclus avec ce tutorial.)
7) Sauvegardez la nouvelle carte de peau
8) Revenez dans l’éditeur de peau et cliquez sur File “Import”, puis “Skin image”. Utilisez l’image de peau que vous venez de peindre. Vous devriez maintenant avoir quelques choses qui ressemble à ça :



Notre peau de l’espace terminée

Notez que c'est une sphère d'étoile 'rapide et sale'. Les triangles de la carte ont évidemment des tailles différentes et les étoiles dans l'image plate ne sont pas calquées sur la sphère. Pour une vraiment bonne sphère nous aurions besoin de plus de temps pour créer une configuration cylindrique, comme pour le globe de la terre dans un atlas. Cependant, L'espace est sombre et couvre gracieusement les trous et les pièces de notre configuration. Fermez le rédacteur de peau et admirez votre travail en employant l'outil de position dans la 3D Vue!

Cliquez sur File-> “Save As” pour sauvegarder votre nouveau modèle d'étoile terminé dans dossier "Space Flight Workshop". Nommez-le MYSTARS.MDL

Ciel ! Vous avez juste créé… ***votre*** ciel !

Pour résumer :

1) Créer une sphère
2) Inverser les normales
3) L'agrandir
4) Lui mettre une peau!

# Création du niveau de l'espace

Le niveau de l'espace n'est en réalité rien de plus qu'une boîte de limitation invisible avec notre vaisseau spatial à l'intérieur. Notre vision des étoiles est liée à la position de notre vaisseau de l'espace. Pour cette raison, notre sphère d'étoile est créée par le scénario du vaisseau spatial, après que le synonyme du vaisseau spatial soit mis. De cette façon on s'occupera du ciel automatiquement une fois que nous commençons le niveau et il se déplacera comme les mouvements de vaisseau, pour donner l'impression d'un vaste espace vide.

Pour créer notre niveau spatial, ouvrez WED et choisissez Fichier -> New

Ajouter un WAD à votre niveau en sélectionnant Texture -> Texture Manager
Cliquez sur le bouton “Add WAD” et sélectionnez le WAD de votre choix puis
Cliquez sur le bouton "open". J'ai utilisé le fichier "STANDARD.WAD".

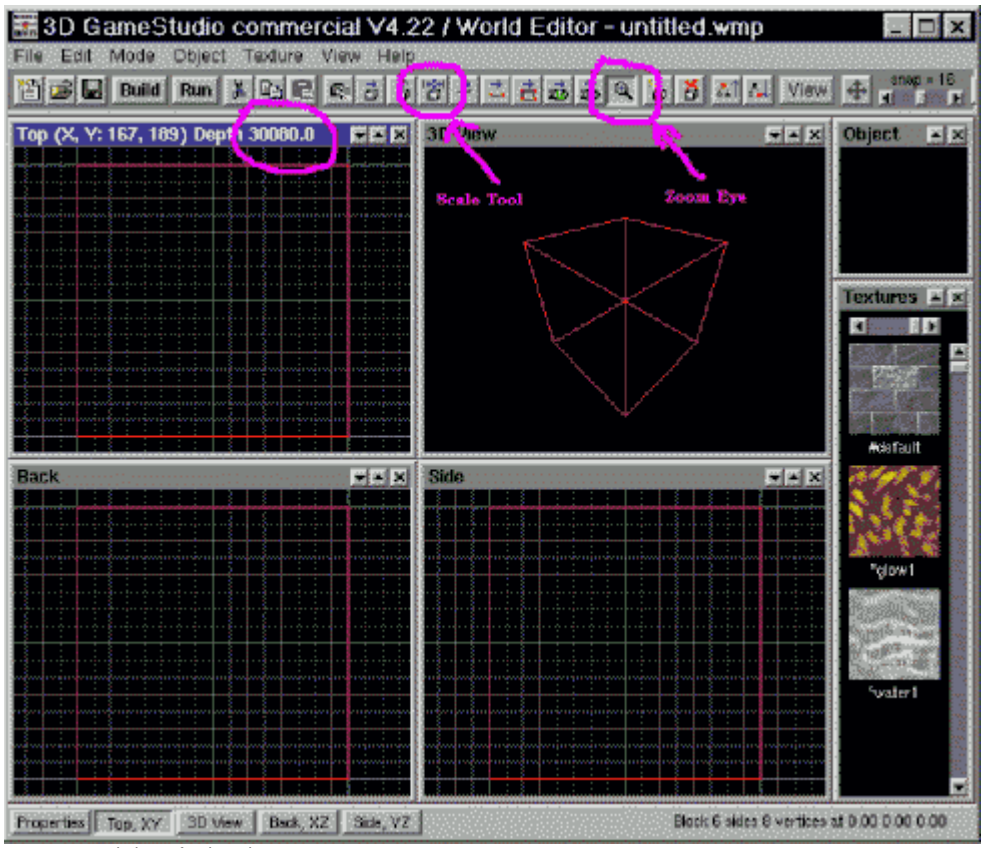
Fermez la fenêtre du gestionnaire de textures lorsque c'est fait.

Ensuite, nous créerons notre boîte de limitation. Faites cela en choisissant "Objet -> Add Primitive -> Cube (large)"

Agrandissez le cube, pour qu'il fasse approximativement 30,000 quants de côté.

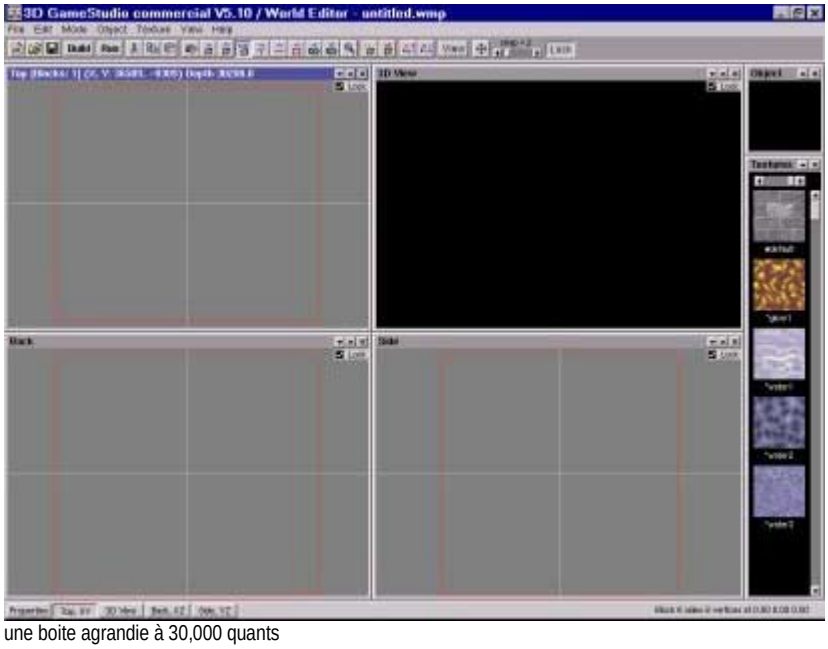
Pour ce faire augmentez la profondeur de vue pour chaque fenêtre en employant la touche "+".
Étendez la profondeur de chaque fenêtre à la taille que vous avez choisi(dans notre cas, 30080 comme vu dans figure qui suit).

Ensuite, choisissez l'outil “Zoom Eye” dans la barre d'outil principale, clic gauche enfoncé et tirez la souris vers le bas dans une des vues 2D. Cela aura l'effet d'un zoom arrière de caméra jusqu'à ce que nous puissions voir que l'échelle de notre niveau est correcte. ( J'ai fait un zoom arrière aussi loin que me le permettait le programme.) Vos cubes 2D apparaîtront comme un minuscule point à ce moment là.



Arrangement de la profondeur de vue

Maintenant, choisissez “Scale tool” et agrandissez votre cube dans chaque fenêtre jusqu'à ce qu'il soit à la taille désirée. Il doit ressembler à cela :



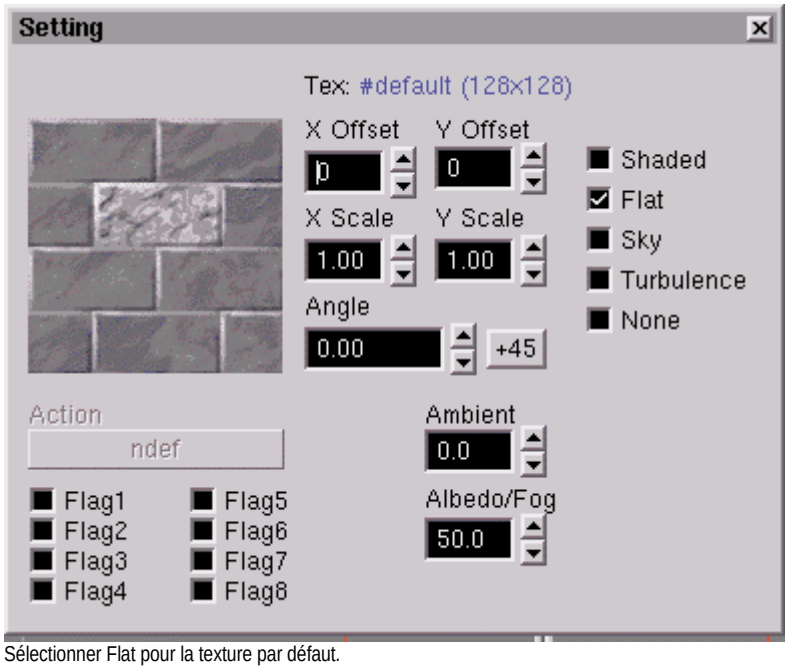
une boite agrandie à 30,000 quants

Maintenant creusez le cube avec la commande ALT + H. (ou par les menus Edit -> Hollow Block)

Notez que nous n'avons pas fait notre boîte de limitation à la taille de Max de 50,000 quants (100.000 quants sur la version A5). C'est parce que notre vaisseau sera toujours placé dans le centre de la sphère d'étoile et la sphère d'étoile s'étend d'environ 12,000 quants.

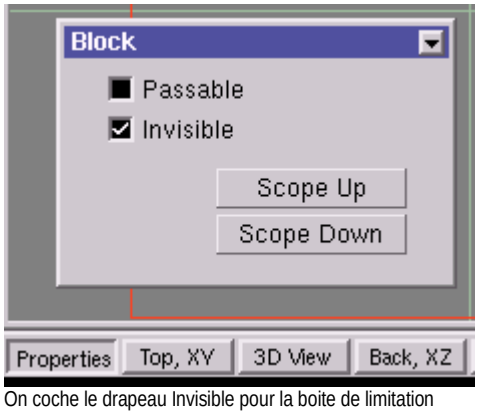
Si notre vaisseau est arrêté à 30,000 quants et que nos étoiles s'étendent en dehors d'encore 12,000 nous avons couvert 42,000 quants ... sans risque pour notre taille maximale de niveau. Vous devez vous assurer que les parties de votre sphère d'étoile ne dérivent pas au-delà de la limite magique de 50 000 quants ou vous pourriez avoir des problèmes.

Appliquez la texture par défaut à votre boîte de limitation par un clic droit sur la texture dans la partie droite puis clic gauche sur "apply" et assurez-vous que le drapeau 'flat' est coché avant de l'appliquer à votre boite



Sélectionner Flat pour la texture par défaut.

Maintenant, que vous avez texturé votre boîte, ouvrez la fenêtre de propriétés de la boîte par un clic sur le bouton de propriétés dans le coin inférieur gauche de l'écran. Assurez-vous qu'il n'y ait que le drapeau "Invisible" de mis comme sur cette figure :



On coche le drapeau Invisible pour la boite de limitation

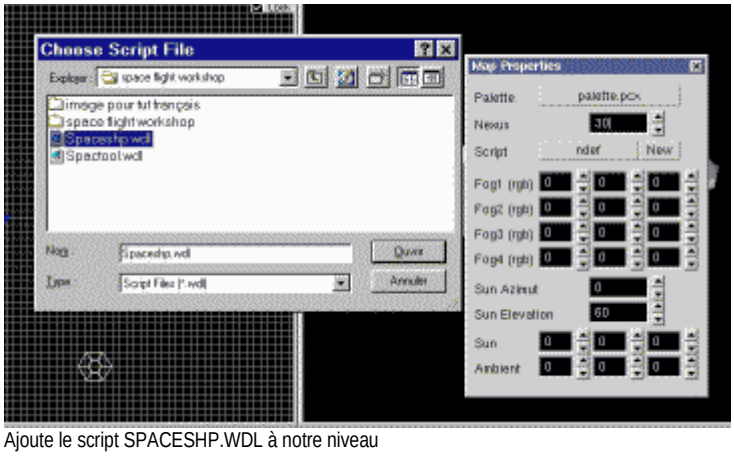
Félicitations ! Vous venez de créer votre boîte de limitation, les limites connus de l'espace !

Maintenant faites un zoom arrière et ajoutez un certain nombre de primitives à votre niveau en choisissant "'Objet -> Add Primitives" et mettez les textures que vous souhaitez.

La taille, le placement et les textures n'ont pas beaucoup d'importance, mais assurez-vous d'en mettre un certain nombre un peu partout. Il est particulièrement important d'avoir quelqu'un près de l'endroit où vous mettrez le vaisseau ainsi vous pourrez obtenir une sensation pour la façon dont se déplace le vaisseau

Un des problèmes avec l'espace est qu'il est la plupart du temps vide. Sans points de référence stationnaires, il est impossible à dire si vous vous déplacez ou pas, et il est très facile se perdre

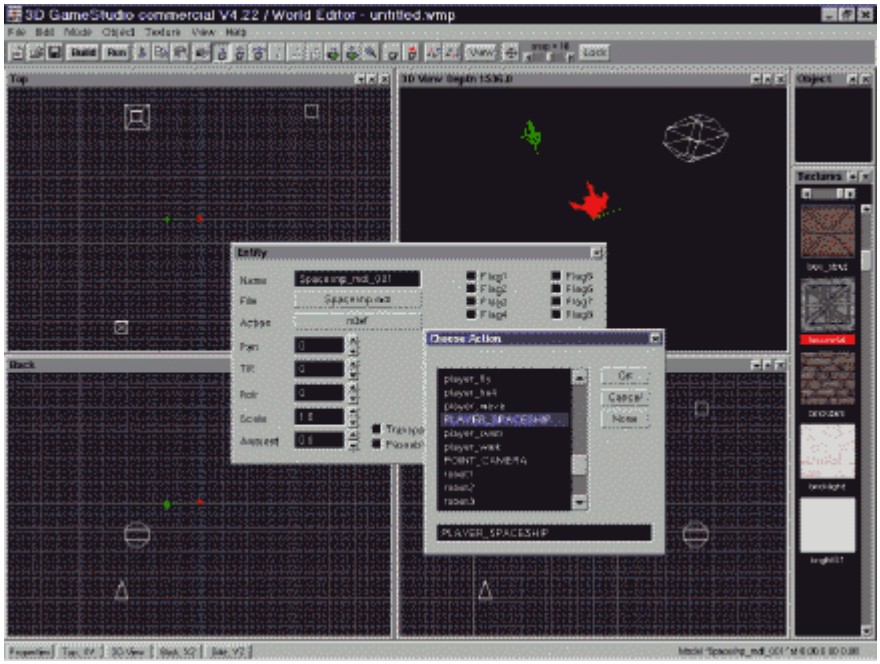
Maintenant, assignez le scénario SPACESHIP.WDL à ce niveau en allant à "File -> Map -> Properties" et assignez SPACESHIP.WDL à votre carte comme sur l'image qui suit :



Ajoute le script SPACESHP.WDL à notre niveau

Nous sommes au bout de nos peines à présent !

Ajoutez juste une position de début de caméra en allant à "Objet -> Add position". Placez la caméra quelque part légèrement de côté, mais pointant vers l'origine comme vous le voyez sur la figure suivante.




La dernière chose que nous devons faire pour achever notre premier niveau spatial est d'ajouter le vaisseau! Choisissez "Objet -> load entité" et utilisez les fenêtres de fichier pour trouver et ajouter le modèle SPACESHP.MDL à notre niveau. Placez-le sur l'origine et assurer vous que votre position de caméra pointe sur lui.

Ensuite, clic droit sur le vaisseau montre sa boîte de propriétés. Cliquez sur le bouton action et parcourez ensuite la liste jusqu'à ce que vous puissiez choisir "PLAYER_SPACESHIP". Cliquez OK.

Félicitations! Vous venez de finir de construire votre premier niveau spatial !

Sauvegardez le niveau dans votre répertoire "Space Flight Workshop" et appelez le “BigSpace.wmp”.

Compilez le niveau avec la Level Map de coché et exécutez le ensuite.

Si vous avez fait tout correctement, vous devriez voir votre vaisseau spatial flottant dans Espace !

Notre niveau spatial terminé !

Amusez-vous à vous déplacer dans votre niveau. Les contrôles sont très simples :
- Utilisez les flèches Gauche et Droite pour diriger votre vaisseau
- Utilisez les flèches Haut et Bas pour contrôler l'inclinaison du vaisseau
- Utilisez la barre Espace pour allumer les moteurs

# Personnalisation de votre vol spatial

Une fois que vous êtes familier avec votre vaisseau et que vous savez comment le manipuler, sauvegardez votre scénario "**SPACESHP.WDL"** et essayez ensuite de faire quelques changements aux paramètres et aux variables de l'action PLAYER_SPACESHIP de votre scénario SPACESHP.WDL.

Ouvrez le scénario SPACESHP.WDL dans votre éditeur de texte préféré et modifiez les variables inscrites ci-dessous pour les adapter à vos propres besoins. Pensez à sauvegarder vos changements et ensuite exécutez votre niveau pour voir vos changements en action.

Vous pouvez changer les variables suivantes dans le scénario :

**CAMERA_TYPE:**
Est mis initialement à 1, en mode "Caméra de Poursuite", où le vaisseau semble stationnaire et le monde semble être en mouvement. Ce scénario a été donné par JCL et sera amélioré dans l'avenir. Avec la valeur 2 vous verrez la vue de la cabine et avec la valeur 3 vous pouvez voir votre vaisseau d'une caméra fixée à l'arrière.

**MY.AUTO_SPIN_STOP:**
Est mis initialement à 1, qui ralentira automatiquement votre rotation tant que vous n'avez pas activé les tuyères. Lorsqu'il est à 0, le vaisseau tournera tant que vous n'appliquerez pas manuellement les tuyères dans la direction opposée.

**MY.AUTO_DECEL:**
Est mis initialement à 1, qui permettra au vaisseau de s'arrêter. Si à 0, le vaisseau continuera à dériver selon son vecteur actuel.

**MY.LIMIT_TURN_SPEED:**
Est mis initialement à 1, qui limitera la vitesse de rotation en fonction de la valeur de la variable MY.MAX_SPIN_SPEED. Si à 0, il n'y a aucune limitation d'effet de rotation.

**MY.LIMIT_TOP_SPEED:**
Imposera une limitation de vitesse à chaque vecteur de la poussée basée sur la valeur de MY.MAX_SHIP_SPEED. Si à 0, il n'y a aucune limitation de vitesse.

**MY.ENGINE_THRUST:**
Définit la puissance effective "des Moteurs Principaux". De plus grandes valeurs font plus de poussée. La valeur par défaut est 5.

**MY.SPIN_RATE:**
Définit la puissance effective des tuyères de rotation. De plus grandes valeurs font tourner le vaisseau plus rapidement et le rendent également plus dur à contrôler ! La valeur par défaut est 15.

**MY. DECEL_RATE:**
Définit la puissance effective de friction sur le vaisseau dans l'espace. Je sais, cela n'est pas très réaliste, c'est pourquoi vous pouvez l'éteindre en mettant le drapeau MY.AUTO_DECEL à 0, mais avec une valeur autre que 0, le vaisseau est beaucoup plus facile à manipuler. La valeur par défaut est 04.

**MY. MAX_SPIN_SPEED:**
Définit la vitesse supérieure effective des tuyères de rotation. De plus grandes valeurs permettent une vitesse supérieure plus rapide pour la rotation du vaisseau. Vous pouvez éteindre cette option en mettant le drapeau MY.LIMIT_TURN_SPEED à 0. Le La valeur par défaut est 5.

**MY. MAX_SHIP_SPEED:**

Définit la vitesse supérieure efficace pour chaque vecteur du mouvement du vaisseau. De plus grandes valeurs permettent au vaisseau de réaliser des vitesses supérieures plus rapides. Vous pouvez tourner cette option en mettant le drapeau MY.LIMIT_TOP_SPEED à 0. La valeur par défaut est 20.

# Pour conclure !

Cela conclut l'atelier de vol Spatial. J'espère que vous l'avez aimé! Cet atelier jette les bases pour les jeux qui se passent dans l'univers cosmique. Il est construit sur un modèle de physique assez réaliste et il peut être une source d'amusement , mais il y a aussi beaucoup de choses à améliorer. Bien que ce soit un tremplin excellent pour faire un jeu de type de simulateur spatial, vous pourriez vouloir un peu moins de réalisme et un peu plus de contrôle pour faire un jeu d'espace de type arcade! J'aimerais voir vos inventions.

Voici quelques idées qui pourraient être ajoutées pour donner un peu d'épice à votre activité extraterrestre: ajouter un panneau pour un cockpit (au cas où vous opèreriez dans cette vue !), ajouter des armes, ajouter des commandes pour contrôler le roulis de votre vaisseau, faire une carte d'étoile plus jolies, ajouter des fonctionnalités plus agréables (à commencer par les planètes!) construit dans WED. Ordres d'atterrissage, d'amarrage et décollage...

Aussi, si vous inventez un profil de vol notamment intéressant pour un vaisseau, postez-le sur le Forum afin que d'autres puissent en profiter aussi !

Une dernière chose... Chargez le "Vertex Space Tool" à partir de la page de téléchargement de Conitec. C'est une interface type panneau qui vous permet de changer tous les drapeaux et réglages à la volée !

Le ciel n'a plus de limite!

- Nick "WildCat" Chionilos

Pour les âmes courageuses qui veulent jeter un coup d'œil sous le capot et voir ce qui fait avancer notre vaisseau spatial, j'ai inclus l'Annexe A pour des commentaires détaillés du scénario ce qui suppose une assez solide compréhension de WDL et à quelques endroits, de connaître un peu les maths. Sinon vous devez me croire sur parole !

# APPENDIX A: Details on the SPACESHP.WDL script

```
#                                   Bienvenu au Calibre Spatial !
#
#           Les contrôles sont très simples :
#            Utilisez les flèches Gauche et Droite pour diriger votre vaisseau
#           Utilisez les flèches Haut et Bas pour contrôler l'inclinaison du vaisseau
#           Utilisez la barre Espace pour allumer les moteurs
#
#            NOTEZ: les champs modifiables par l'utilisateur sont en Bleu.
#                        les commentaires sont en rouge.
////////////////////////////////////////////////////////////////////////////
//            A4 main wdl
////////////////////////////////////////////////////////////////////////////
// Le mot clé  PATH donne les répertoires ou les fichiers du jeu peuvent être trouvés
// relatif au répertoire du niveau
PATH "models"; // Chemin du sous répertoire des modèles s'il existe
PATH "sounds"; // Chemin du sous répertoire des sons s'il existe
PATH "bmaps"; // Chemin du sous répertoire des dessins s'il existe
PATH "images"; // Chemin du sous répertoire des images s'il existe
PATH "..\\template"; Chemin du sous répertoire template de WDL

////////////////////////////////////////////////////////////////////////////
// Le mot clé INCLUDE est utilise pour inclure des fichiers WDL,
// comme ceux du sous répertoire TEMPLATE avec les actions prédéfinies
!////////////////////////////////////////////////////////////////////////////
INCLUDE <movement.wdl>;
INCLUDE <messages.wdl>;
INCLUDE <particle.wdl>;
INCLUDE <doors.wdl>;
INCLUDE <actors.wdl>;
INCLUDE <weapons.wdl>;
INCLUDE <war.wdl>;
INCLUDE <menu.wdl>;

// Met la résolution de démarrage pour le système
IFDEF LORES;
VAR VIDEO_MODE = 4;   // 320x240
IFELSE;
VAR VIDEO_MODE = 6;   // 640x480
ENDIF;

VAR VIDEO_DEPTH = 16;              // D3D, résolution 16 bits
VAR FPS_MAX = 40;        // 40 fps max (taux d'affichage maximum)


// La fonction MAIN est appelée au démarrage du jeu et fait toute sorte de chose

FUNCTION MAIN()
{
            LOAD_LEVEL <BIGSPACE.WMB>;
            LOAD_STATUS();            // rétabli les variables globales
            WAIT 16;
}.

////////////////////////////////////////////////////////////////////////////
// le scénario de l'espace commence ici
////////////////////////////////////////////////////////////////////////////

// Sons:
            SOUND MAIN_ENGINES,<ENGINE.WAV>;
            SOUND THRUSTERS,<THRUSTER.WAV>;
            VAR THRUST_HANDLE = 0;          // Pointeurs pour le son vous indiquant
            VAR ENGINE_HANDLE = 0;          // si le son est joué
```

```
// Synonymes:
        SYNONYM PLAYER_SHIP {TYPE ENTITY;}
        SYNONYM STAR_SPHERE {TYPE ENTITY;}


// L'ACCEL_VECTOR exprime dans quelle direction le vaisseau est accéléré
// par la poussée. Parce que le vaisseau a été  construit avec son avant pointant
// le long de l'axe de X, la poussée de stimulation agira toujours sur la composante
// "X" d'ACCEL_VECTOR. Pour plus de clarté,  je l'ai débaptisé et appelé
//ENGINE_THRUST ('poussée de machine').

        DEFINE ACCEL_VECTOR,SKILL1;
        DEFINE ENGINE_THRUST,SKILL1;


// Pendant qu'ACCEL_VECTOR détermine la poussée relative à l'orientation du vaisseau,
// Le DRIFT_VECTOR applique la vitesse existante relative à l'orientation du monde.

        DEFINE DRIFT_VECTOR,SKILL4;
        DEFINE DRIFT_VECTOR_X,SKILL4;
        DEFINE DRIFT_VECTOR_Y,SKILL5;
        DEFINE DRIFT_VECTOR_Z,SKILL6;

        DEFINE SHIP_ANGLES,SKILL7;   // vitesse de gîte, d'inclinaison et de rotation du vaisseau
        DEFINE PAN_SPEED,SKILL7;     // la vitesse à laquelle vous prenez du gîte (tourner autour de l'axe Z)
        DEFINE TILT_SPEED,SKILL8;    // la vitesse à laquelle vous vous inclinez (tourner autour de l'axe Y)
        DEFINE ROLL_SPEED,SKILL9;    // la vitesse à laquelle vous tournez (tourner autour de l'axe X)
        DEFINE SPIN_RATE,SKILL10;    // l'accélération de votre mouvement de rotation


// ***     Note: Bien que le coefficient de friction dans l'espace soit pratiquement proche
//         de zéro, le vaisseau sera néanmoins beaucoup plus maniable
//         si vous donner une petite valeur dans DECEL_RATE.
//          Représentez-vous un pilote automatique qui entreprend les actions appropriées
//         lorsque le vaisseau n'est pas en contrôle actif.
//         Ces champs sont employés quand leur drapeau correspondant est sur 1 (on).

//  Comment la friction rapide peut vous ralentir !
        DEFINE DECEL_RATE,SKILL11;

// La restriction de vitesse de rotation si le drapeau LIMIT_TURN_SPEED est mis à 1 (on)
        DEFINE MAX_SPIN_SPEED,SKILL12;

// restriction de la vitesse du vaisseau si LIMIT_TOP_SPEED est mis à 1 (on)
        DEFINE MAX_SHIP_SPEED,SKILL13;

//*** Définit les Drapeaux pour les caractéristiques de vol du vaisseau spatial
        DEFINE AUTO_SPIN_STOP,FLAG1; // ce drapeau doit toujours être à 0 pour un
        DEFINE AUTO_DECEL,FLAG2; // Maximum de réalisme, mais peut être.
        DEFINE LIMIT_TURN_SPEED,FLAG3; // utilisé pour faire un vaisseau
        DEFINE LIMIT_TOP_SPEED,FLAG4; // plus maniable.

//DEFINE STAR_SKY,<STARS.MDL>; // Modèle de sphère par défaut du tutorial
DEFINE STAR_SKY,<MYSTARS.MDL>; // Nouveau modèle de sphère créé par l'utilisateur


// ********** les variables actives commencent ici ******************
VAR SHIP_ANGS[3];            // Utilisé pour les calculs d'angles pan et tilt
VAR ENGINE_WORK_SPEED[3];   // Utilisé pour les calculs de vitesse et de dérive


// Camera Variables
  VAR CAMERA_TYPE;
  VAR CAMERA_SPOT[3];
  VAR WORKANG[3];
```

```
/*
La sphère spatiale se déplace simplement avec le vaisseau d'espace du joueur. En changeant
les coordonnées X, Y, and Z de la sphère directement à travers sa variable POS, au lieu
d'employer la commande MOVE, nous éliminons tout contrôle de collision, et c'est ce que
nous voulons. La sphère d'étoile doit être capable de passer la boîte de limitation du niveau.
Le vaisseau, lui, ne doit pas.
*/

ACTION SPACE_SPHERE
{
	STAR_SPHERE = ME;

	WHILE (1) // maintient constamment le joueur au centre de l'univers
	{
		VEC_SET(MY.POS,PLAYER_SHIP.POS);
		WAIT 1;
	}
}


ACTION PLAYER_SPACESHIP
{
// Met le  mode de camera
	CAMERA_TYPE = 1; //**** 1 = caméra de poursuite, 2 = Cockpit, 3 = stationnaire ****

// Met les synonymes pour un usage ultérieur et zéro à toutes les vitesses du vaisseau
	PLAYER_SHIP = ME;
	MY.NARROW = ON; // utilise la plus petite coque de collision
	VEC_SET(MY.ACCEL_VECTOR,NULLVECTOR);	// aucune vitesse au départ
	VEC_SET(MY.DRIFT_VECTOR,NULLVECTOR);	// aucun élan au départ
	CREATE STAR_SKY,MY.POS,SPACE_SPHERE;	// initialise les étoiles de fond

//*********************************************************
// Tous les drapeaux de vol spéciaux sont activés pour la mise en marche. Pour les couper, mettez les sur 0.
	MY.AUTO_SPIN_STOP = 1;		// fait que le vaisseau s'arrête de tourner par lui-même
	MY.AUTO_DECEL = 1;	// fait que le vaisseau ralentit en employant MY.DECEL_RATE
	MY.LIMIT_TURN_SPEED = 1;	// limite la vitesse de rotation en fonction de MY.MAX_SPIN_SPEED
	MY.LIMIT_TOP_SPEED = 1;		// la vitesse plafond du vaisseau, dépendant de MY.MAX_SHIP_SPEED

// Les caractéristiques de vol du vaisseau lors de la mise en marche:
	MY.ENGINE_THRUST = .5;		// la force des moteurs principaux
	MY.SPIN_RATE = .15;		// la force des moteurs de poussée
	MY.DECEL_RATE = .04;		// le degré de la réduction de vitesse de la dérive.
	MY.MAX_SPIN_SPEED = 5;		// la vitesse Maxima pour le changement de direction du vaisseau.
	MY.MAX_SHIP_SPEED = 20;		// la vitesse plafond absolue du vaisseau

.//*********************************************************

	WHILE (1) {

# *****c'est ici que se joue la logique de la poussée:
#
#	 si vous allumez un moteur de poussée et que vous utilisez LIMIT_TURN_SPEEDS alors
#	assurez-vous d'être encore sous la vitesse limite. Fa^tes cela avec les touches de direction
#	 Gauche (CUL), Droit (CUR), Haut (CUU) et Bas (CUD)

	IF (KEY_CUL == 1)		//ici on teste la flèche de gauche
		{IF ((MY.LIMIT_TURN_SPEED == 0) ||
			(MY.LIMIT_TURN_SPEED == 1) &&
			(MY.PAN_SPEED < MY.MAX_SPIN_SPEED))
		{MY.PAN_SPEED += MY.SPIN_RATE * TIME;
			IF (THRUST_HANDLE == 0)			// si aucun son n'est joué
				{PLAY_LOOP THRUSTERS,15; // joue le son du moteur
				THRUST_HANDLE = RESULT;}}}

	IF (KEY_CUR == 1)		//ici on teste la flèche de droite
		{IF ((MY.LIMIT_TURN_SPEED == 0) ||
```

```
                    (MY.LIMIT_TURN_SPEED == 1) &&
                    (MY.PAN_SPEED > -MY.MAX_SPIN_SPEED))
                {MY.PAN_SPEED -= MY.SPIN_RATE * TIME;
                    IF (THRUST_HANDLE == 0)                 //si aucun son n'est joué
                        {PLAY_LOOP THRUSTERS,15; // joue le son du moteur
                        THRUST_HANDLE = RESULT;}}}


        IF (KEY_CUU == 1)       // ici on teste la flèche qui monte
                {IF ((MY.LIMIT_TURN_SPEED == 0) ||
                    (MY.LIMIT_TURN_SPEED == 1) &&
                    (MY.TILT_SPEED < MY.MAX_SPIN_SPEED))
                {MY.TILT_SPEED += MY.SPIN_RATE * TIME;
                    IF (THRUST_HANDLE == 0)                 // si aucun son n'est joué
                        {PLAY_LOOP THRUSTERS,15; // joue le son du moteur
                        THRUST_HANDLE = RESULT;}}}


        IF (KEY_CUD == 1)       // ici on teste la flèche qui descend
                {IF ((MY.LIMIT_TURN_SPEED == 0) ||
                    (MY.LIMIT_TURN_SPEED == 1) &&
                    (MY.TILT_SPEED > -MY.MAX_SPIN_SPEED))
                {MY.TILT_SPEED -= MY.SPIN_RATE * TIME;
                    IF (THRUST_HANDLE == 0)                 // si aucun son n'est joué
                        {PLAY_LOOP THRUSTERS,15; // joue le son du moteur
                        THRUST_HANDLE = RESULT;}}}


        IF ((THRUST_HANDLE != 0) &&     // si aucun son n'est joué...
                (KEY_CUU == 0) &&
                (KEY_CUD == 0) &&               // ...et...
                (KEY_CUL == 0) &&
                (KEY_CUR == 0))                 // aucune touche de direction de pressée
                    {STOP_SOUND THRUST_HANDLE ;// arrête le son
                    THRUST_HANDLE = 0;}


//******* Au cas où Auto_Spin_Stop est utilisé, la rotation du vaisseau ralentit, si ce n'est pas réglé activement.
        IF (MY.AUTO_SPIN_STOP == 1)
                {STOP_ROTATION();}


//********On allume les moteurs principaux ici:.
        IF (KEY_SPACE == 1)     // si la barre espace est appuyée
                {IF (ENGINE_HANDLE == 0) // Si aucun son n'est joué alors joue
                    {PLAY_LOOP MAIN_ENGINES,25;
                        ENGINE_HANDLE = RESULT;}


//          1) Met ENGINE_WORK_SPEED à la poussée actuelle
//          2) Vec_Rotate convertit la poussée en orientation du vaisseau
//                  (Qui concerne seulement l'axe de X) et le convertit
//                  en composantes X, Y et Z reltifs à l'orientation du monde..
//
// Note: les étapes 1 et 2 pouvaient être faits tout aussi bien en utilisant la trigonométrie,pour convertir
//          en orientation du monde via les lignes suivantes:
//           X: ENGINE_WORK_SPEED[0] += (MY.ENGINE_THRUST * COS(MY.TILT)
//                                      * COS(MY.PAN));
//          Y: ENGINE_WORK_SPEED[1] += (MY.ENGINE_THRUST * COS(MY.TILT)
//                                      * SIN(MY.PAN));
//          Z: ENGINE_WORK_SPEED[2] += (MY.ENGINE_THRUST * SIN(MY.TILT));
//
//          L'application de VEC_ROTATE paraît simplement un peu agréable à l'œil.


                VEC_SET(ENGINE_WORK_SPEED,MY.ENGINE_THRUST);
                VEC_ROTATE(ENGINE_WORK_SPEED,MY.PAN);
```

```
//          S'il n'y a aucune limitation de vitesse, alors activer la poussée, s'il y a une,
//                    limitation de vitesse, appliquer seulement la composante de poussée
//                    qui fait qu'on ne dépasse pas la valeur max.
          IF (MY.LIMIT_TOP_SPEED == 0)
                    {VEC_ADD(MY.DRIFT_VECTOR,ENGINE_WORK_SPEED);}
          ELSE
                    {IF (ABS(ENGINE_WORK_SPEED.X + MY.DRIFT_VECTOR_X) < MY.MAX_SHIP_SPEED)
                    {MY.DRIFT_VECTOR_X += ENGINE_WORK_SPEED.X;}

                    IF (ABS(ENGINE_WORK_SPEED.Y + MY.DRIFT_VECTOR_Y) < MY.MAX_SHIP_SPEED)
                    {MY.DRIFT_VECTOR_Y += ENGINE_WORK_SPEED.Y;}

                    IF (ABS(ENGINE_WORK_SPEED.Z + MY.DRIFT_VECTOR_Z) < MY.MAX_SHIP_SPEED)
                    {MY.DRIFT_VECTOR_Z += ENGINE_WORK_SPEED.Z;}}}
          ELSE
                    {STOP_SOUND ENGINE_HANDLE ;          // arrête de jouer le son
                    ENGINE_HANDLE = 0;}


//**** Met tout en mouvement
          MY.ROLL_SPEED = 0; // nous n'utilisons pas la rotation comme un contrôle actif
          ROTATE ME,MY.SHIP_ANGLESS,NULLVECTOR; // fait tourner le vaisseau en fonction du rapport pan/tilt

// Si vous n'allumez pas activement les moteurs et que  Auto_decel est actif, ralentir le vaisseau
          IF ((KEY_SPACE != 1) && (MY.AUTO_DECEL == 1))
                    {CALL DECEL_SHIP;}

// calcule une distance à partir de la vitesse
          vec_set(temp,MY.DRIFT_VECTOR);
          vec_scale(temp,time);
          MOVE(ME,NULLVECTOR,temp); // Alors déplace le vaisseau selon la dérive

          POINT_CAMERA(); // Ajuste la caméra
          WAIT 1;}
}


FUNCTION STOP_ROTATION()
{
// **** si vous tournez toujours, appliquez une rotation dans la direction opposée pour ralentir
//.
          IF (PLAYER_SHIP.PAN_SPEED == 0)
                    {GOTO CHECKTILT;}
          ELSE
          {IF ((KEY_CUL == 0) && (KEY_CUR == 0))
                    {IF (PLAYER_SHIP.PAN_SPEED > 0)
                              {PLAYER_SHIP.PAN_SPEED -= PLAYER_SHIP.SPIN_RATE * TIME;}
                    ELSE
                              {PLAYER_SHIP.PAN_SPEED += PLAYER_SHIP.SPIN_RATE * TIME;}}
                    IF (ABS(PLAYER_SHIP.PAN_SPEED) < .02)
                              {PLAYER_SHIP.PAN_SPEED = 0;}}
CHECKTILT:

          IF (PLAYER_SHIP.TILT_SPEED == 0)
                    {RETURN;}
          ELSE
                    {IF ((KEY_CUU == 0) && (KEY_CUD == 0))
                              {IF (PLAYER_SHIP.TILT_SPEED > 0)
                                        {PLAYER_SHIP.TILT_SPEED -= PLAYER_SHIP.SPIN_RATE * TIME;}
                    ELSE
                              {PLAYER_SHIP.TILT_SPEED += PLAYER_SHIP.SPIN_RATE * TIME;}}
                    IF (ABS(PLAYER_SHIP.TILT_SPEED) < .02)
                              {PLAYER_SHIP.TILT_SPEED = 0;}}
}
}
```

```
# DECEL_SHIP travaille de cette façon:
# *** Si vous avez de l'élan:
#	1) Mettre ENGINE_WORK_SPEED pour avoir le taux de décélération comme sa composante "X" de poussée
#		(La valeur est négative parce que nous avons besoin d'une contre-poussée pour ralentir,
#		 ne pas pousser pour accélérer)
#	2) Calcule les angles pan et tilt angles qui correspondent à la dérive existante en utilisant Vec_to_angle
#		.
#	3) Utilise Vec_Rotate pour converit la poussée de la perspective du vaisseau en perspective du monde réelle
#.
#	 4) Utilise la commande Vec_Scale du temps pour corriger la vitesse en fonction du taux d'affichage
#	5) Ajoute le nouveau vecteur de décelleration au vecteur de dérive existante pour le ralentissement
#	6) Si vous allez de toute façon vraiment trés lentement ( < .02), met votre vitesse à 0
# *** Si vous êtes arrêtés: ne fait rien et retour

ACTION DECEL_SHIP
{
	IF (VEC_LENGTH(PLAYER_SHIP.DRIFT_VECTOR) != 0)	// Si vous avez de l'élan
		{VEC_SET(ENGINE_WORK_SPEED,NULLVECTOR);
		ENGINE_WORK_SPEED.X = -PLAYER_SHIP.DECEL_RATE;
		VEC_TO_ANGLE(SHIP_ANGS.PAN,PLAYER_SHIP.DRIFT_VECTOR);
		VEC_ROTATE(ENGINE_WORK_SPEED,SHIP_ANGS.PAN);
		VEC_ADD(MY.DRIFT_VECTOR,ENGINE_WORK_SPEED);}
	ELSE
		{RETURN;}  // Si DRIFT_VECTOR est Zero, nous sommes prêts


# *** Si DRIFT_VECTOR est trés près de 0, mettez le à zéro. Cela réduira le nombre
#	de fois ou nous devons passer dans la boucle avant qu'une entité n'ait plus d'élan.
#	Quand une entité ne dérive pas, nous n'avons pas besoin de mathématiques pour cela, et ainsi
#	nous réduisons le travail de l'ordinateur. Cela peut également aider à augmenter le taux d'affichage si
#	 plusieurs entités utilisent cette routine en même temps.

	IF (VEC_LENGTH(PLAYER_SHIP.DRIFT_VECTOR) < .02)
		{VEC_SET(PLAYER_SHIP.DRIFT_VECTOR,NULLVECTOR);}
}


VAR C1[3];.
VAR C2[3];
VAR TEMPA[3];
VAR CAMERA_WORK;


ACTION POINT_CAMERA
{
	IF (CAMERA_TYPE == 2)
		{GOTO COCKPIT;}
	IF (CAMERA_TYPE == 3)
		{GOTO STATIONARY;}

// Ici commence le code de la caméra de poursuite:
//	(NOTE: Cette portion de code a été donnée par JCL.
//	Il a été livré  brut de fonderie et je ne sais pas assez
//	 comment il travaille pour vous apporter des explications. Si vous avez donc des questions
//	concernant cette partie de code, adressez les lui.)

	CAMERA.GENIUS = NULL; // Camera n'ignore personne
	VEC_SET(WORKANG.PAN,PLAYER_SHIP.PAN);
	CAMERA_SPOT.X = -200; // déplacement relatif de la caméra
	CAMERA_SPOT.Y = 0; // vers le vaisseau lorsqu'on est en mode poursuite
	CAMERA_SPOT.Z = 80;

	VEC_SET(CAMERA.X,CAMERA_SPOT.X);
	VEC_ROTATE(CAMERA.X,PLAYER_SHIP.PAN);
	VEC_ADD(CAMERA.X,PLAYER_SHIP.X);
```

```
// Mets les angles PAN et TILT de la caméra
	VEC_SET (TEMPA.X,PLAYER_SHIP.X);
	VEC_SUB (TEMPA.X,CAMERA.X);
	VEC_TO_ANGLE(CAMERA.PAN,TEMPA);

// Mets l'angle ROLL de la caméra
	C1.X = TEMPA.Y;
	C1.Y = -TEMPA.X;
	C1.Z = 0;

	C2.X = -CAMERA_SPOT.Y;
	C2.Y = CAMERA_SPOT.X;
	C2.Z = 0;

	VEC_ROTATE(C2,PLAYER_SHIP.PAN);

// Obtient la différence angulaire entre les 2 vecteurs utilisant le point produit
	TEMPA = C1.X*C2.X +C1.Y*C2.Y;
	tempa /= VEC_LENGTH(C1);
	tempa /= VEC_LENGTH(C2);

	IF (C2.Z < 0)
		{CAMERA.ROLL = ACOS(TEMPA);}
	ELSE
		{CAMERA.ROLL = -ACOS(TEMPA);}
	RETURN;

// La caméra du Cockpit démarre ici
COCKPIT:
	CAMERA.GENIUS = PLAYER_SHIP; // la caméra ne voit pas les parties du vaisseau
	VEC_SET(CAMERA.X,PLAYER_SHIP.X); // la caméra est là où le vaisseau est
	VEC_SET(CAMERA.PAN,PLAYER_SHIP.PAN); // la caméra regarde dans la même direction que le vaisseau
	.RETURN;

//La caméra stationnaire démarre ici
STATIONARY:
	CAMERA.GENIUS = NULL; // La caméra n'ignore personne
	VEC_SET(CAMERA_WORK,PLAYER_SHIP.X); // Pointe la camera vers le joueur
	VEC_SUB(CAMERA_WORK,CAMERA.X);
	VEC_TO_ANGLE(CAMERA.PAN,CAMERA_WORK); // maintenant la caméra regarde le joueur
	RETURN;
}
////////////// Fin du Scénario de l'espace ici ///////////////////////
```