Didacticiel avancé: vos premières animations avec POV-Ray!

1ère partie

Merci à Diamond Editions pour son aimable autorisation pour la mise en ligne de cet article, initialement publié dans Linux Magazine N°66

Saraja Olivier – olivier.saraja@linuxgraphic.org

En tant que lanceur de rayons, nous avons déjà eu l'occasion de constater que POV-ray était capable de rendus d'une grande qualité et d'une précision remarquable. Pour ces raisons, traditionnellement, les lanceurs de rayons sont surtout employés dans l'industrie de la conception d'images statiques. Mais avec la montée en puissance de calcul des processeurs modernes, et avec la démocratisation, même à fins personnelles et amateur, des fermes de calcul, la création d'animations complexes à l'aide de moteurs tels que POV-ray devient envisageable.

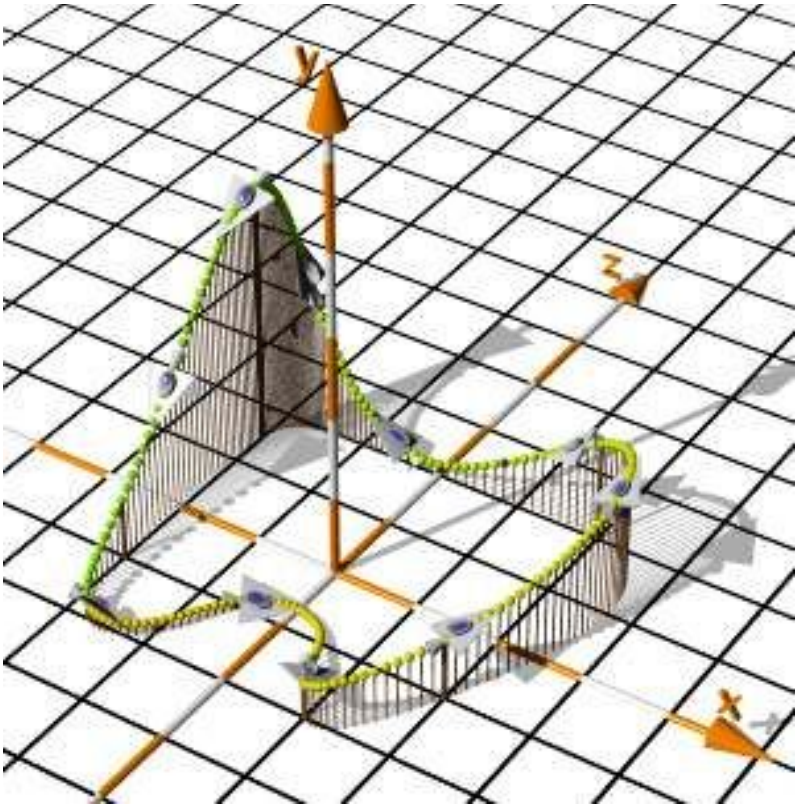

Figure 1: exemple d'animation complexe par Friedrich A. Lohmüller

1. Ce qu'il faut savoir avant de démarrer

S'il est souvent nécessaire avec POV-ray, ou pour le moins utile, de planifier la composition de votre scène, cela devient presque rigoureusement indispensable lorsque vous envisagez la création d'une animation. Il y en a effet certaines décisions qu'il vous faudra prendre très tôt, car elles auront un impact direct sur la qualité de l'animation. Du point de vue artistique, vous découperez certainement votre animation en une ou plusieurs séquences. Du point de vue technique, en revanche, chaque séquence sera définie de façon rigoureuse.

1.1 La durée

Il sera souvent pratique, dans le cadre des petites productions personnelles qui vous intéressent certainement, de définir la durée d'une séquence en secondes. Votre animation sera peut-être composée de plusieurs séquences, mais dans la mesure où l'assemblage de celles-ci se fera dans un logiciel tiers, nous pouvons nous contenter de nous intéresser à la création d'une séquence unique. Les animations de débutant sont souvent constituées d'une scène unique, nous nous en tiendrons donc là dans le cadre de cet article. Exemple: nous décidons que notre animation fera 15 secondes, pas plus.

1.2 Nombre d'images par seconde

Les films qui passent dans votre télévision sont en réalité une succession de plans fixes, qui diffèrent très peu l'un de l'autre. Diffusés à la vitesse de 25 images par seconde, votre cerveau accepte le résultat comme étant une illusion acceptable d'un mouvement continu à l'écran. Augmentez cette vitesse et vous ne ferez probablement pas la différence. Diminuez-là, en revanche, et vous constaterez d'autant plus de saccades dans les mouvements amples que la vitesse sera faible. Si la qualité du résultat souhaité est bien sûr à prendre à compte, l'usage futur de l'animation doit également être considéré. Pour un sketch, un petit film ou un dessin animé, vous souhaiterez certainement avoir une animation de qualité irréprochable, et vous choisirez certainement une vitesse de 25 images par seconde. Pour un logo tournant ou la création d'une GIF animée, vous estimerez qu'une vitesse de 12 images par seconde est souvent suffisant. Exemple: n'ayant pas trop de temps devant nous mais souhaitant malgré tout une animation de qualité satisfaisante, nous choisissons une vitesse intermédiaire de 19 images par seconde.

L'animation de notre exemple a donc une durée de 15 secondes, à un rythme de 19 images par seconde. Cela veut dire que POV-ray devra générer 15 fois 19 images qui, mises bout à bout, constitueront notre animation, soit 285 images au total. Nous commençons à appréhender les difficultés posées par tout animateur disposant d'une puissance de calcul allant de faible à modérée (un ordinateur personnel et/ou une petite ferme de rendu basée sur quelques machines qui ne sont plus de prime jeunesse): le temps de rendu d'une image individuelle. Celle-ci dépend de nombreux facteurs, parmi lesquels: la résolution de l'image; la profondeur de l'anti-crénelage; la complexité de la scène (temps de parsing, matériaux avec réflection et réfraction, nombreuses macro recalculant la géométrie ou la position d'objets en fonction du temps); les méthodes de rendu (radiosité, lancé de photons...); etc. Certaines peuvent évoluer en fonction des besoins de votre animation, mais d'autres doivent de préférence être figées dès la planification de l'animation.

1.3 Résolution des images

Tout dépend de l'usage que vous comptez faire de votre animation: diffusion en continu sur Internet, téléchargement, logo animé, pressage sur un CD-Rom ou sur un DVD-Rom pour les plus ambitieux d'entre vous. Si vous planifiez l'encodage de votre animation sous forme d'une vidéo mpeg, sachez que tant la hauteur que la largeur de votre image devra être un multiple de 16. Par exemple, un plan fixe d'un DVD vidéo classique est au format 720 x 576. Si vous ne savez pas trop ce que vous allez faire de votre future animation, autant respecter ce ratio. Sinon, vous pouvez tout aussi bien déterminer les résolutions qui vous conviennent le mieux. Des résolutions de 640 x 480 ou 320 x 256 viennent naturellement à l'esprit, même s'il ne s'agit pas des résolutions classiques de l'animation.

1.4 Anti-crénelage

Pour la création d'images statiques comme pour celle d'animations, l'anti-crénelage conduit à un résultat d'une part visuellement attractif, d'autre part plus photo-réaliste et donc plus facilement acceptable par l'esprit humain comme étant une animation de qualité.

1.5 Quelques conseils élémentaires

Effectuez toujours les prévisualisations de vos animations au format le plus petit possible, et n'effectuez le rendu de votre animation au format final que lorsque vous aurez débogué et affiné tous les paramètres de votre animation. Vous gagnerez un temps précieux! Par exemple, vous pouvez très bien effectuer un premier rendu à une résolution de 352 x 288 sans anti-crénelage, radiosité et lancé de photons. Vous effectuerez ensuite quelques tests grandeur nature sur quelques plans fixes au maximum de qualité souhaité (par exemple avec l'anti-crénelage, mais aussi la radiosité et/ou la carte de photons) afin de vérifier le bon rendu de ceux-ci, avant de vous aventurer à effectuer le rendu de votre animation entière.

D'autre part, vous gagnerez certainement en impact visuel si vos animations font usage du flou focal. Outre le fait de basculer virtuellement du premier-plan à l'arrière-plan, vous aidez l'attention du spectateur à se focaliser sur les éléments que vous aurez choisis. Ainsi, celui-ci sera plus sensible aux efforts que vous avez mis dans la déformation de la balle en caoutchouc que dans la texture très pixelisée qui constitue le décor en arrière-plan.

Si vous débutez, ne cherchez pas à réaliser une animation qui durera plusieurs minutes, voire plusieurs heures. Contentez-vous d'une succession de plans n'excédant pas plus d'une dizaine de secondes, et vous arriverez déjà à des résultats très satisfaisants!

2. La variable clock et les fichiers ini

En bon français, `clock` veut dire horloge. De la à imaginer qu'il s'agit de la variable qui va cadencer votre animation, il n'y a qu'un pas que nous n'aurons pas d'hésitation à franchir. L'animation commencera à une valeur t0=0 et terminera à une valeur tn=n. A chaque incrément temporel donné, la valeur de `clock` croîtra légèrement, affectant tous les variables de votre scène qui auront `clock` comme argument, selon la formule mathématique qui y sera associée. Par exemple, à t0=clock=0, `translate<0, 2*clock+0.25,0>` matérialisera votre objet à 2*0+0.25=0.25 unités d'altitude; à t1=clock=1, l'objet sera à une altitude de 2*1+0.25=2.25 unités.

Ceci étant compris, vous possédez d'ores et déjà les bases fondamentales de l'animation avec POV-ray. Il ne vous reste plus qu'à savoir comment déterminer les valeurs de départ et d'arrivée de la variable clock, ainsi que le nombre d'images à rendre pour respecter le nombre d'images par seconde que vous souhaitez. C'est là qu'interviennent les fichiers `ini`. Habituellement, vous y stoquez les paramètres du rendu, à savoir la résolution de l'image, la profondeur de l'anti-crénelage et diverses options que vous souhaitez voir appliquer à votre rendu. Et bien, ce même fichier `ini` va vous permettre d'initialiser la variable `clock`, ainsi que de définir le nombre d'images rendues pour constituer votre animation.

Par exemple, `monfichier.ini` suivant:

```
Antialias=Off
Antialias_Threshold=0.1
Antialias_Depth=2

Input_File_Name=monfichier.pov

Initial_Frame=1
Final_Frame=30
Initial_Clock=0
Final_Clock=1

Cyclic_Animation=on
Pause_when_Done=off
```

Les trois premières lignes précisent le comportement de l'anti-crénelage et plus particulièrement, la première l'active, les deux suivantes étant des paramètres associés à son usage. Vous réserverez l'usage actif de cette option à vos rendus finaux, les opérations d'anti-crénelage ayant tendance à alourdir les temps de calculs plus que vous ne le souhaiteriez pour une simple prévisualisation de votre travail.

La ligne `Input_File` précise le fichier qui contient la scène à rendre; il n'est pas éncessaire qu'il y ait correspondance entre les deux noms, mais cela se révèle souvent utile si votre répertoire de travail contient plusieurs racine de fichiers, généralement une par séquence de votre animation.

Par exemple, `sequence04.pov` et `.ini`...

Le bloc suivant permet de rythmer votre animation. Par exemple, celle-ci atteindra une longueur de 30 images lorsque l'horloge `clock` aura été incrémentée jusqu'à la valeur 1. Il est fortement recommandé de laisser les valeurs initiale et finale de `clock` à respectivement 0 et 1. POV-ray commence alors le premier rendu avec un `clock = 0`, puis incrémente celui-ci pour le rendu suivant de `1/Final_Frame` de sorte qu'il est toujours possible de déduire la nouvelle valeur de `clock` en fonction de la précédente: `clock = clock + 1/Final_Frame`

Enfin, `Cyclic_Animation` vous permet de spécifier si votre animation est prévue pour tourner en boucle ou non. En effet, si vous spécifiez par exemple à un objet de tourner sur 360° sur un cycle d'horloge, il va commencer à une position donnée lors du premier rendu, terminer sur la même position lors du dernier rendu. Si vous jouez l'animation en boucle, vous constaterez un «gel» de l'animation lorsque l'objet sera revenu à sa position d'origine au cours d'une rotation complète, car au cours de deux images consécutives, l'objet aura son angle de départ. En spécifiant `Cyclic_Animation=on`, vous donnez carte blanche à POV-ray pour qu'il ajuste l'incrément de l'horloge de sorte qu'arrivé à la dernière image de l'animation, l'objet à tourné de 360° moins ce petit incrément qui vous donnera l'illusion d'une rotation en boucle parfaitement continue.

A noter que dans cet exemple, si vous encodez votre animation au rythme de 25 images par seconde, celle-ci durera un tout petit peu plus de 1 seconde; à 15 images par seconde elle durera exactement 2 secondes et enfin à 10 images par seconde, exactement 3 secondes.

3. Quelques cas pratiques pour bien assimiler

Nous ne ferons certes pas d'animation de personnages avec ce que nous verrons ici. Pas de dessin animé, ni de grand projet trop ambitieux, donc. En revanche, nous allons apprendre comment illustrer agréablement un projet scolaire, une page web ou des petites choses dans le genre. A chaque fois, nous commencerons par la planification de l'animation, c'est à dire l'écriture du fichier `ini` correspondant. Une fois ceci fait, nous étudierons la scène à animer et surtout l'usage réservé à la variable `clock`.

3.1 Cas 1: un logo animé

Supposons que vous souhaitiez construire un logo animé pour votre page personnelle, afin de lui donner un peu de vie propre. Pour ce faire, décidons de créer un icône pour inviter l'internaute à vous envoyer ses commentaires, et pour rendre les choses au plus explicites, décidons que l'icône sera le texte « e-m@il » qui tournoiera rapidement sur lui-même. Commençons par créer un fichier `ini`, que nous nommerons `cas01.ini`. Le fichier scène qui sera rendu s'appellera, pour plus de simplicité et pour garder une bonne correspondance entre les fichiers `ini` et `pov`, `cas01.pov`. L'avenir de ce texte tournoyant est de finir en GIF animé sur votre page, aussi déciderons-nous qu'une vitesse de 3 images par seconde satisfaira nos besoins. En revanche, nous ne souhaitons pas que le texte accomplisse une rotation entière trop rapidement, et nous choisissons 4 secondes comme un temps satisfaisant. Notre animation sera donc finalement composée de 4*3=12 images au total. Enfin, comme il est prévu que ce logo tourne à l'écran *ad nauseam*, nous montrerons notre intérêt pour l'option d'animation cyclique. Voici notre `cas01.ini`:

```
Antialias=Off
Antialias_Threshold=0.1
Antialias_Depth=2
Input_File_Name=cas01.pov
Initial_Frame=1
Final_Frame=12
Initial_Clock=0
Final_Clock=1
Cyclic_Animation=on
Pause_when_Done=off
```

Il ne nous reste plus qu'à construire notre scène. Nous allons commencer par poser une caméra minimaliste et une lumière tout ce qui a de plus simple. Nous allons ensuite ajouter un objet de type texte, et enfin l'animer au travers de ses paramètres de transformation et bien sûr de la fonction `clock`. Ce que nous souhaitons, c'est que le texte tourne autour de l'axe vertical. Nous voulons qu'au début de l'animation (`Initial_Clock=0`) l'objet ait subi une rotation de 0° et qu'en

fin d'animation (`Final_Clock=1`) il ait subi une rotation de 360°. La relation entre l'angle de rotation et la variable `clock` est dans ce cas très simple à deviner: `angle = clock*360`, ce qui nous donne bien 0° pour `clock` = 0 et 360° pour `clock` = 1.

```
//notre lumière
light_source {
<0, 5, -5>, rgb <1, 1, 1>
}

//notre caméra
camera {
location <0, 0, -5>
look_at <0, 0, 0>
}

//notre texte tournant
text {
ttf "/chemin/vers/votre/police/bragga.TTF"
"e-m@il"
0.25, <0, 0>
pigment {
color rgb <0, 0.75, 0>
}
finish {
specular 1
}
scale 1
translate x*(-2)
rotate <0, clock*360, 0>
}
```

Figure 2: le résultat de l'animation de notre cas numéro un: un logo animé pour votre page web

Vous noterez que sur la dernière vignette, le texte n'est pas redevenu « à plat » aux yeux de la caméra. Cela est du au fait que nous avons spécifier `Cyclic_Animation=on` dans notre fichier `ini`, et que POV-ray a modifié l'incrément de la valeur `clock` entre deux images pour que la dernière image soit à nouveau suivie de la première dans le cadre d'une animation en boucle.

3.2 Cas 2: le bras du pendule

Nous allons rester dans le cadre des animations cycliques pour ce deuxième cas. Nous allons ici chercher à reproduire le mouvement dit de pendule des vieilles horloges de nos grands-parents (ou plutôt devrais-je dire arrière-grand-parent, en fonction de la génération, pas si évident, à laquelle je m'adresse). Le fichier `ini` ne sera pas très différent du cas précédent, sauf que nous allons changer un peu la longueur de l'animation. Nous conserverons une vitesse d'animation de 3 images par seconde, ce qui correspond assez bien à une illustration animée au format GIF. En revanche, cette fois, nous allons accorder deux secondes au pendule pour aller de sa position de repos à l'un de ses extremums, soit huit secondes pour aller de sa position de départ à sa position extrême sur la droite de l'écran (2 secondes), pour revenir à sa position centrale (2 secondes), pour rejoindre sa position extrême sur la gauche de l'écran (2 secondes) et enfin revenir à sa position de repos au centre de l'écran (encore 2 secondes). Cela nous fera donc 3*8=24 images pour une animation durant, au final, huit secondes, ce qui est un peu plus long que le cas précédent. L'animation étant prévue pour être jouée en boucle, nous penserons également à activer `Cyclic_Animation=on`.

```
Antialias=Off
```

```
Antialias_Threshold=0.1
Antialias_Depth=2
Input_File_Name=cas02.pov
Initial_Frame=1
Final_Frame=24
Initial_Clock=0
Final_Clock=1
Cyclic_Animation=on
Pause_when_Done=off
```

Nous allons à nouveau reprendre une source de lumière et une caméra très basique. L'horloge va être figurée par une boîte, son cadran par un disque, et le pendule par un cylindre et une sphère unie. Le mouvement d'oscillation va une fois de plus passer par l'usage de la variable `clock` au sein d'une opération `rotate`. Toutefois, nous ferons usage de formules trigonométriques afin de maîtriser l'oscillation; le mouvement commençant avec le pendule vertical, nous dirons que la déviation angulaire est nulle. Cela nous oriente donc vers l'usage d'une fonction sinus(x) plutôt que cosinus(x); en effet, lorsque x=0, sin(0)=0 et cos(0)=1. De plus, la valeur retournée par une fonction sinus est comprise entre -1 et +1. Cela se traduit donc par une déviation de +/- 1° seulement, ce qui n'est pas un résultat très visuel. Nous allons donc multiplier la fonction sinus par une valeur qui va augmenter l'amplitude du mouvement. Appelons tout simplement cette variable `Amplitude`. Enfin, il est bon de savoir que les fonctions trigonométriques fonctionnent avec des angles en radians. Pour passer d'un angle en degrés à un angle en radians, nous multiplierons la valeur en degrés par 2π. Ainsi, plutôt que d'utiliser `sin(clock)` nous utiliserons `sin(clock*2*pi)` dans notre formule. La valeur de la rotation de notre pendule deviens donc: `Amplitude*sin(clock*2*pi)`. A noter que la rotation, une fois de plus, se fait par rapport à l'origine du système. Heureusement pour nous, nous concevrons notre scène de façon à avoir la base de notre pendule au centre exact de notre scène (c'est à dire en décalant la boîte et le cadran légèrement vers le haut, car ils apparaissent normalement centrés sur l'origine.

```
//notre lumière
light_source {
<0, 5, -5>, rgb <1, 1, 1>
}

//notre caméra
camera {
location <0, 0, -5>
look_at <0, 0, 0>
}

//notre horloge
box {
<-0.5, -0.5, -0.5>, <0.5, 0.5, 0.5>
pigment {
color rgb <0.3, 1, 0.3>
}
translate y*0.5
}

//notre cadran
disc {
<0, 0.5, -0.51>,<0, 0, -1>, 0.35,0
pigment {
color rgb <1, 1, 1>
}
}

//notre pendule
union {
//le bras du pendule
cylinder {
<0, 0, 0>, <0, -2, 0>, 0.025
pigment {
color rgb <0.3, 1, 0.3>
}
}
//le poids du pendule
sphere {
<0, -2, 0>, 0.2
```

```
pigment {
color rgb <1, 1, 1>
}
}
#declare Amplitude=20;
rotate <0,0,Amplitude*sin( clock*2*pi)>
}
```

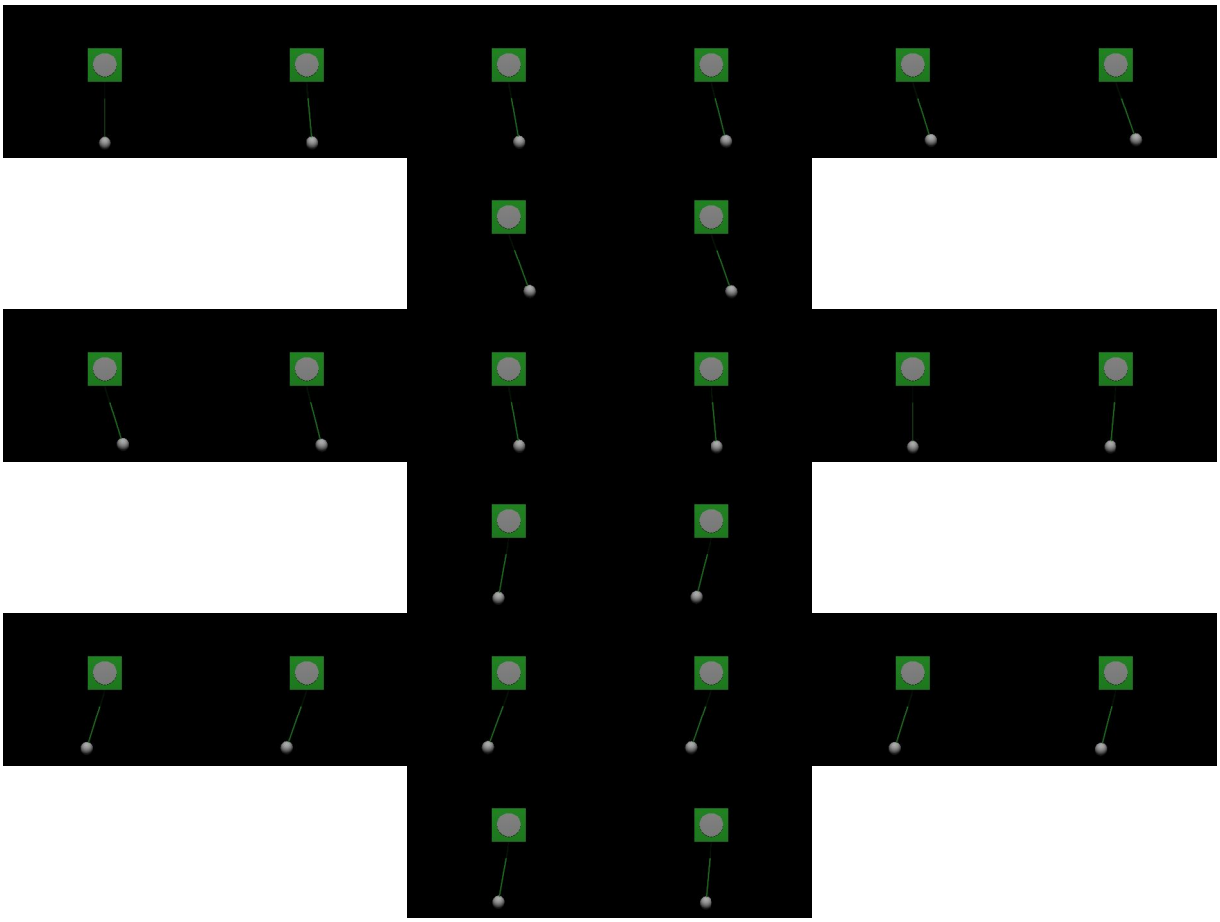

Figure 3: le résultat de l'animation de notre cas numéro deux: le mouvement oscillatoire d'une pendule

A nouveau, nous notons que la dernière image ne reflète pas la position du pendule à l'état du repos; POV-ray a une fois de plus réglé ses incréments de sorte que la première image puisse s'enchaîner naturellement et sans à coup après la dernière image rendu de l'animation.

3.3 Cas 03: animation des lumières, des textures et déplacement de la caméra

Hola! Cela à l'air de drôlement se compliquer. Rassurez-vous, il n'en est rien. Pour le cas pratique qui suit, il convient de se rappeler que toute variable d'une scène POV-ray est paramétrable. Cela veut dire que vous pouvez « piloter » n'importe quelle valeur dans le temps, à condition de maîtriser les outils mathématiques qui vont bien.

Une fois de plus, nous allons rester très simples et abordables, mais la richesse et le réalisme d'une animation résident souvent dans le temps passé par l'animateur à fignoler les paramètres de sa scène. Cette fois ci, nous allons partir dans l'idée de simuler une lumière qui tournoie en changeant d'intensité au-dessus d'un cube qui change de couleur, avec la caméra qui s'approche lentement de la scène. A noter, pour cette dernière, que la composante `look_at` ne changera pas, de sorte que quel que soit le mouvement que nous pourrions imprimer à la caméra, celle-ci regardera toujours le centre de la scène.

Pour notre fichier `cas03.ini`, nous tablerons sur une animation à 5 images par seconde (ce qui devrait nous permettre d'apprécier plus facilement les variations de lumière et de texture de la scène) sur une durée de 5 secondes. En revanche, cette fois, nous désactiverons l'option

d'animation en boucle. Nous devrions donc obtenir 5*5=25 images.

```
Antialias=Off
Antialias_Threshold=0.1
Antialias_Depth=2
Input_File_Name=cas03.pov
Initial_Frame=1
Final_Frame=25
Initial_Clock=0
Final_Clock=1
Cyclic_Animation=off
Pause_when_Done=off
```

Notre décor sera constitué de quelques primitives, dont le seul objectif est de produire des ombres intéressantes. Nous avons choisi une couleur très neutre, un gris clair, afin que le changement de couleur de la sphère soit le plus visible possible. La variable `clock` ayant été définie comme variant de 0 à 1, nous décidons qu'au cours de l'animation, la sphère verra sa couleur varier sur ses trois composantes: la composante rouge variera de 1 à 0 (`1-clock`), sa composante verte de 0 à 1 (`0+clock`) et enfin sa composante bleue de 1 à 0.5 (`1-clock/2`).

La lumière est en fait un spot dont l'origine est à <0, 5, -2>. Nous allons lui ajouter un paramètre `rotate`, qui fera tourner le spot autour de l'origine, et les composantes `<0,5,-2>` matérialisent alors le rayon de giration du spot. Nous utiliserons exactement la même fonction de rotation qu'au cours du cas numéro un: `rotate <0, clock*360, 0>`.

Enfin, notre dernier objet animé est notre caméra. Nous connaissons le point de départ que nous lui souhaitons (`location <10, 10, -10>`), ainsi que le point d'arrivée (`location <5, 5, -5>`). La seule difficulté réside donc à trouver la fonction de `clock` qui conduira aux composantes voulues lorsque `clock=0` et lorsque `clock=1`. Nous souhaitons que la translation soit linéaire, donc nous poserons `valeur=a*clock+b` (une très classique équation de droite). Traitons ensemble la première composante; nous savons donc que l'équation est de la forme `valeur=a*clock+b`. Lorsque `clock=0`, nous souhaitons `valeur=10`; écrivons le: `10=a*0+b`. On en déduit que `b=10`. Nous avons donc maintenant `valeur=a*clock+10`. Lorsque `clock=1`, nous souhaitons `valeur=5`, ce qui se traduit par `5=a*1+10`. On en déduit donc `a=-5`. Notre première composante serait donc `-5*clock+10`. En appliquant le même cheminement aux deux autres composantes, nous obtenons finalement `location <-5*clock+10, -5*clock+10, 5*clock-10>`.

```
//notre décor
union {
box {
<-0.5, -0.5, -0.5>, <0.5, 0.5, 0.5>
scale 2
rotate <0, 0, 0>
translate <-2.5, 1, 2.5>
}
cylinder {
<0, 1, 0>, <0, 0, 0>, 1
scale 1
rotate <0, 0, 0>
translate x*3
}
cylinder {
<0, 3, 0>, <0, 0, 0>, 1
scale 1
rotate <0, 0, 0>
translate <-4, 0, -0.5>
}
box {
<-3.9, 0, -4.8>, <1, 0.5, -2.9>
scale 1
rotate <0, 0, 0>
translate <0, 0, 0>
}
plane {
<0, 1, 0>, 0
scale 1
rotate <0, 0, 0>
translate <0, 0, 0>
}
pigment {
```

```
color rgb <0.862745, 0.862745, 0.862745>
}
}

//notre sujet qui change de couleur et de taille
sphere {
<0, 1.5, 0>, 1
pigment {
color rgb <1-clock, 0+clock, 1-clock/2>
}
scale 1
rotate <0, 0, 0>
translate <0, 0, 0>
}

//notre source de lumière qui tourne autour du sujet
light_source {
<0, 5, -2>, rgb <1, 1, 1>
spotlight
radius 35
falloff 45
point_at <0, 0, 0>
fade_distance 10
fade_power 1
rotate <0, clock*360, 0>
}

//notre caméra qui s'approche lentement
camera {
location <-5*clock+10, -5*clock+10, 5*clock-10>
look_at <0, 0, 0>
}
```

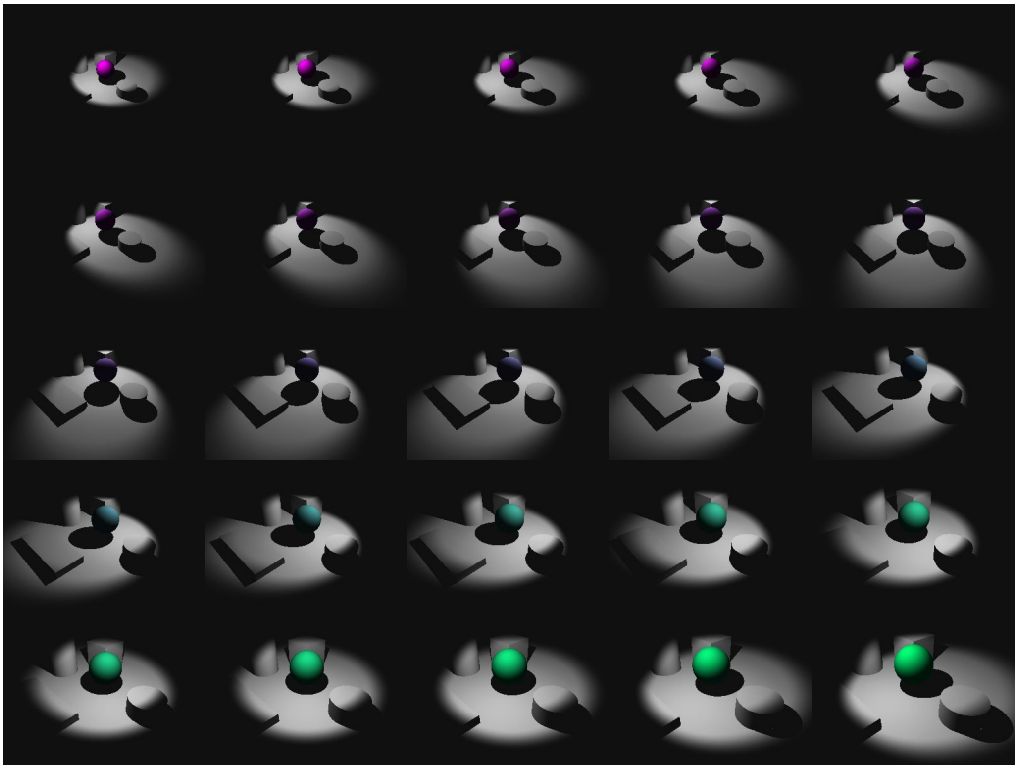

Figure 4: le résultat de l'animation de notre cas numéro trois: l'animation de lumières, de texture et de caméra

4. Le montage des animations

Si POV-ray est capable de créer des images fixes et de les nommer de sorte qu'elles se suivent logiquement, alphanumériquement parlant, il n'est pas (encore?) capable de produire une animation directement lisible sous forme de mpeg, avi ou simplement GIF animé. Pour assurer le

montage de votre animation, il vous faudra donc passer par des logiciels tiers. Voyons un peu ce qui est disponible, et brièvement comment s'en servir. En particulier, nous traiterons deux formats très courants: le GIF animé, le format vidéo AVI.

4.1 Un GIF animé avec The Gimp

Nous allons voir comment créer un GIF animé à partir de la série d'images de notre premier cas pratique: le logo animé. Pour cela, nous allons utiliser The Gimp (et nous en couvrirons pas ici son installation ou son utilisation; nous supposons que vous possédez au moins les bases fondamentales de ce très bon logiciel de manipulation d'images). Lancez celui-ci et cliquez sur **Fichier > Ouvrir...** pour sélectionner `cas0101.png` (ou la première image chronologique de votre animation). Si le gestionnaire de calques n'apparaît pas, pressez CTRL+L pour le faire apparaître. Cliquez avec le bouton droit sur le calque nommé « **Fond** » et choisissez **Editer les attributs du calque**. Une boîte de dialogue apparaît, changez le nom en quelque chose de plus explicite, comme **Frame 1**. Ajoutez maintenant un nouveau calque, en acceptant les attributs par défaut (les dimensions, en particulier, sont les mêmes que celles de votre première image) à l'exception du nom que vous remplacerez par **Frame 2**.

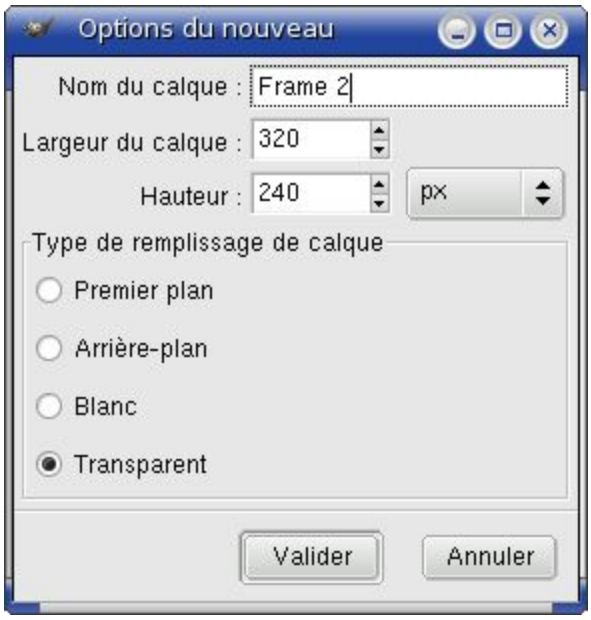

Figure 5: création d'un nouveau calque sous Gimp

Cliquez à nouveau sur **Fichier > Ouvrir...** pour charger `cas0102.png`. Cette image est la nouvelle image active. Faîtes un CTRL+A pour tout sélectionner, un CTRL+C pour copier la sélection, puis fermer votre image `cas0102.png`. La boîte de dialogue des calques doit normalement se mettre à jour pour montrer le nouveau calque **Frame 2**, vide. Cliquez sur votre image `cas0101.png` et faîtes un CTRL+V pour coller votre image `cas0102.png`; le résultat apparaît sous la forme d'une sélection flottante. Cliquez sur la petite ancre pour attribuer définitivement l'image collée au calque précédent. Ouvrez maintenant `cas0103.png` et recommencez les mêmes manipulations que précédemment, jusqu'à être arrivé à votre dernière image `cas0112.png` dans votre calque intitulé **Frame 12**.

Maintenant, sélectionnez votre **Frame 1** dans le gestionnaire des calques, et faîtes un **Fichier > Enregistrer sous...** en renommant votre fichier `cas01.gif`. Une boîte de dialogue vous permettant de gérer l'exportation apparaît. Prenez soin de cocher l'option vous permettant de sauvegarder en tant que GIF animée:

Figure 6: l'exportation au format de la GIF animée

Vous pouvez maintenant cliquer sur le bouton **Exporter**. Une nouvelle boîte de dialogue apparaît. Par défaut, GIMP vous propose d'enregistrer avec l'option **Boucle infinie**, c'est bien notre intention, alors cochez cette case. Enfin, il vous est laissé la possibilité de régler maintenant le **délai entre les images** de votre animation. Pour le cas 01, nous avions prévu une animation de 4 secondes, au rythme de 3 images par seconde, ce qui veut dire que nous avons une image tous les 0.333 secondes, soit encore toutes les 333 millisecondes.

Figure 7: spécifions le délai d'affichage entre deux images consécutives, ici du 3 images par seconde

Saisissons cette valeur et cliquons sur **Valider**. Si nous jetons un oeil sur l'image finale dans un navigateur ou dans Konqueror, par exemple, nous verrons le résultat de nos manipulations: un joli logo tournant sur lui-même, prêt à être utilisé sur votre page web! A noter que le fichier GIF par défaut pèse 33,2 ko.

4.2 Une vidéo AVI avec Blender

Blender, Blender... Ce nom vous est familier. Ne s'agit-il pas de ce logiciel de création 3D tant vanté? Mais qu'a-t-il de si particulier, déjà? Et bien disons pour résumer qu'il s'agit d'une véritable suite de création d'images de synthèse, de la modélisation à l'animation en passant par... le montage vidéo! Et oui, du haut de ses 2.4 Mo de taille, Blender est également capable de vous servir à assembler vos images éparses en vidéo. Et comme cela se fait avec une facilité

déconcertante, pourquoi ne pas en abuser un peu, même pour des POV-types amateurs comme nous?

Commencez par lancer Blender. Une fois celui-démarré, basculez la fenêtre 3D en mode **Video Sequence Editor** par la combinaison de touches SHIT+F8.

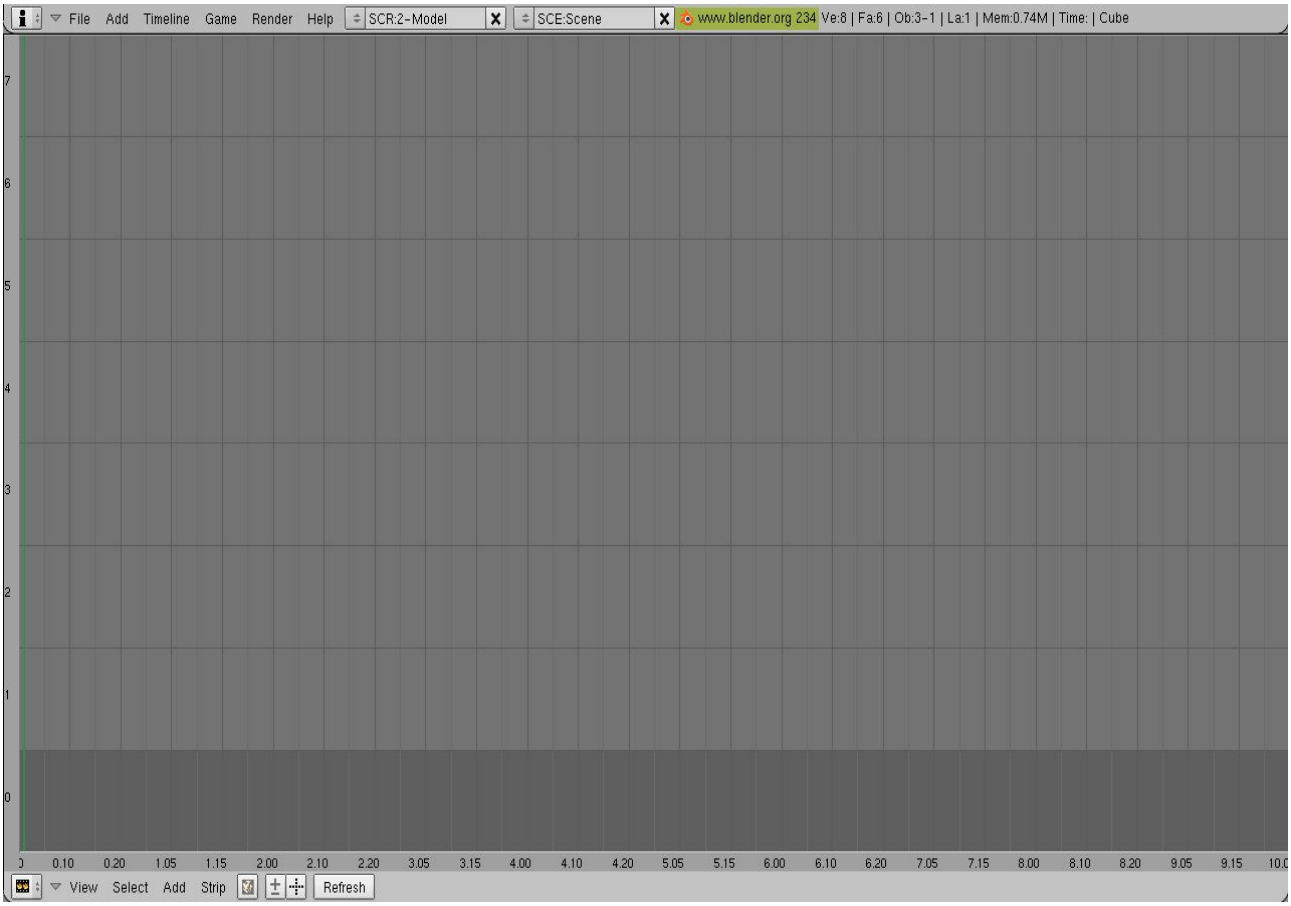

Figure 8: l'éditeur de séquences vidéo de Blender

Vous noterez au bas de la fenêtre principale (celle qui occupe les 3 / 4 de l'écran) un petit menu. C'est celui-ci qui va nous permettre de réaliser le montage vidéo. Choisissez **Add > Images** pour arriver à un gestionnaire de fichiers embarqué. Naviguez jusqu'au répertoire où vous avez le rendu de votre animation, et sélectionner chacun des fichiers avec un clic droit de la souris: les fichiers sélectionnés apparaissent surlignés en bleu: de `cas0301.png` à `cas0325.png`. Après avoir appuyé sur la touche Entrée, vous êtes de retour à l'éditeur de séquence. Vous constaterez que la suite d'images (les 25!) apparaissent sous la forme d'une bande flottante qui suit la souris. Blender attend de vous que vous ameniez la première image à la position 0 de l'axe des temps, en abscisse et que vous validiez la position à l'aide d'un clic gauche.

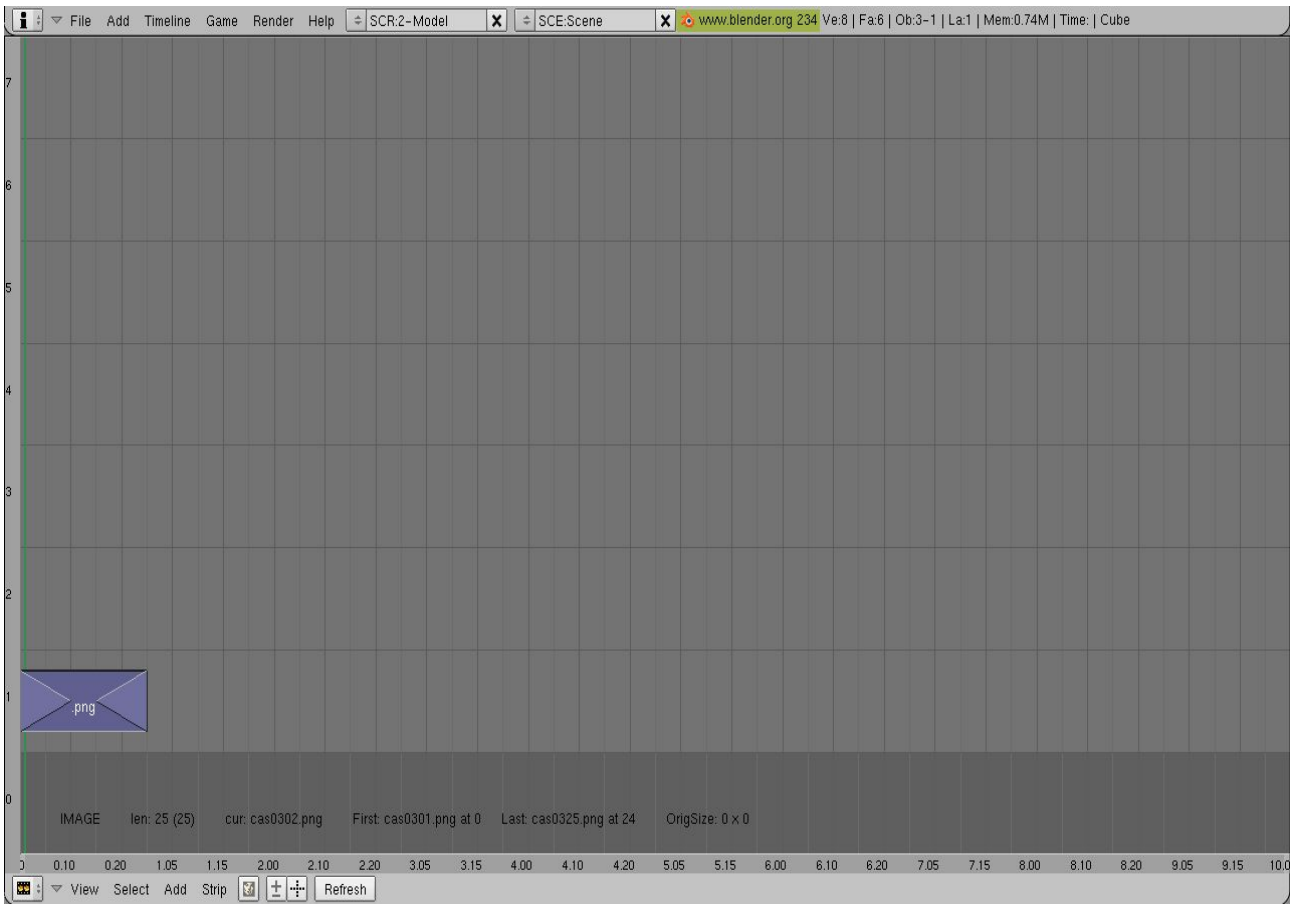

Figure 9: les 25 premières frames sont occupées par les images rendues avec POV-ray

Nous allons maintenant appeler l'écran de réglage des paramètres de rendu (qui nous permettront de paramétrer notre futur vidéo selon nos souhaits) en appuyant sur le bouton F10. La partie inférieure de l'écran (celle située sous le **video sequence editor**) change pour afficher une bordée de paramètres qui peuvent paraître, au prime abord, impressionnants.

Figure 10: les paramètres de rendu qui permettent la reconstruction d'une vidéo autonome

Dans le panneau le plus à gauche intitulé **Output**, il y a trois chemins par défaut. Le premier est normalement `//tmp/`. Cliquez sur le petit symbole de répertoire sur la gauche du chemin pour choisir le répertoire de destination de votre future vidéo; chez moi ce sera `/home/olivier/tmp/`. Le deuxième panneau, intitulé **Render**, ne nous intéresse pas, car nous n'avons pas rendu nos images avec Blender mais avec une application externe, en l'occurence POV-ray. Nous sauterons directement au quatrième panneau, **Format**, pour choisir dans la deuxième moitié le type de fichier vidéo (nous choisirons **AVI Jpeg**), la qualité (nous conserverons le **Quality:90** par défaut) et la vitesse de l'animation (nous spécifierons **Frs/sec: 5** à la place du 25 par défaut, car notre animation qui fait 25 images a été calculée pour durée 5 secondes à 5 images par seconde). Revenons au troisième panneau intitulé **Anim** car c'est là que l'alchimie va se produire: activons le bouton **Do Sequence** et spécifions les frames de départ et de fin de noter animation: **Sta: 1** et **End:25** (au lieu du **End:250** par défaut; si vous laissez cette valeur alors que votre animation ne comporte que 5 images, les 225 images manquantes seront remplacées par des écrans noirs). Appuyons maintenant sur le gros bouton **ANIM**. Une petite fenêtre va apparaître et l'animation va se jouer sous vos yeux à vitesse grand V. Vous pouvez désormais quitter Blender, et utiliser votre

navigateur de fichiers pour remonter jusqu'à l'endroit où vous avez choisi de stocker votre animation. Un nouveau fichier y est apparu: `0001_0025.avi`. Le fichier résultant pèse 157,8 ko, ce qui aurait pu être réduit au détriment de la qualité ou du format des images. A noter que vous avez utilisé, pour alimenter l'éditeur de séquences, des images au format `png`, et que Blender a fait tous les travaux de conversion nécessaires.

5. Conclusions

Voici pour nos premiers pas d'animateurs avec POV-ray. Vous aurez compris qu'il faut avoir quelques restes de vos années collèges et lycées (en particulier 3ème et 2nde pour les maths analytiques et la trigonométrie) pour se sentir totalement à l'aise avec cette méthode d'animation. Sachez toutefois que pour les mouvements d'objet, il est possible de définir une courbe de type spline: `spline {natural curve ...}` en tant que « chemin » pour tout objet de votre scène; grâce à l'usage des boucles (voir GNU/Linux Magazine 64, ou en ligne sur **http://www.linuxgraphic.org**) vous pouvez ainsi faire coïncider la trajectoire d'un objet avec la spline que vous aurez préalablement décrite. Cette méthode d'animation sera traitée dans le prochain numéro, et nous en profiterons pour voir comment encoder une vidéo mpeg, qui manque encore à notre panel de compétences. Enfin, pour approfondir les thèmes du présent article, je ne saurais que trop vous conseiller la lecture de l'excellent didacticiel de Friedrich A. Lohmüller, dont vous trouverez l'URL dans les liens utiles, ci-après. Vous y découvrirez entre autres les méthodes d'animation pour les ressorts (que vous avez également appris à modéliser dans GNU/Linux Magazine 64, ou en ligne sur **http://www.linuxgraphic.org**) mais aussi pour les systèmes bielle-manivelle ou plus simplement les textures.

6. Liens

L'excellent didacticiel de Friedrich A. Lohmüller: Animations with POV-Ray:
http://www.f-lohmueller.de/pov_tut/animate/index.htm

La homepage de kpovmodeler: **http://www.kpovmodeler.org**

La homepage de povray (version courante: v3.6):
http://www.povray.org

La documentation officielle en français de povray:
http://users.skynet.be/bs936509/povfr/index.htm

Le site officiel de The Gimp: **http://www.gimp.org**

Le site officiel de Blender: **http://www.blender3d.org**