Qu'est-ce que OpenSCAD?

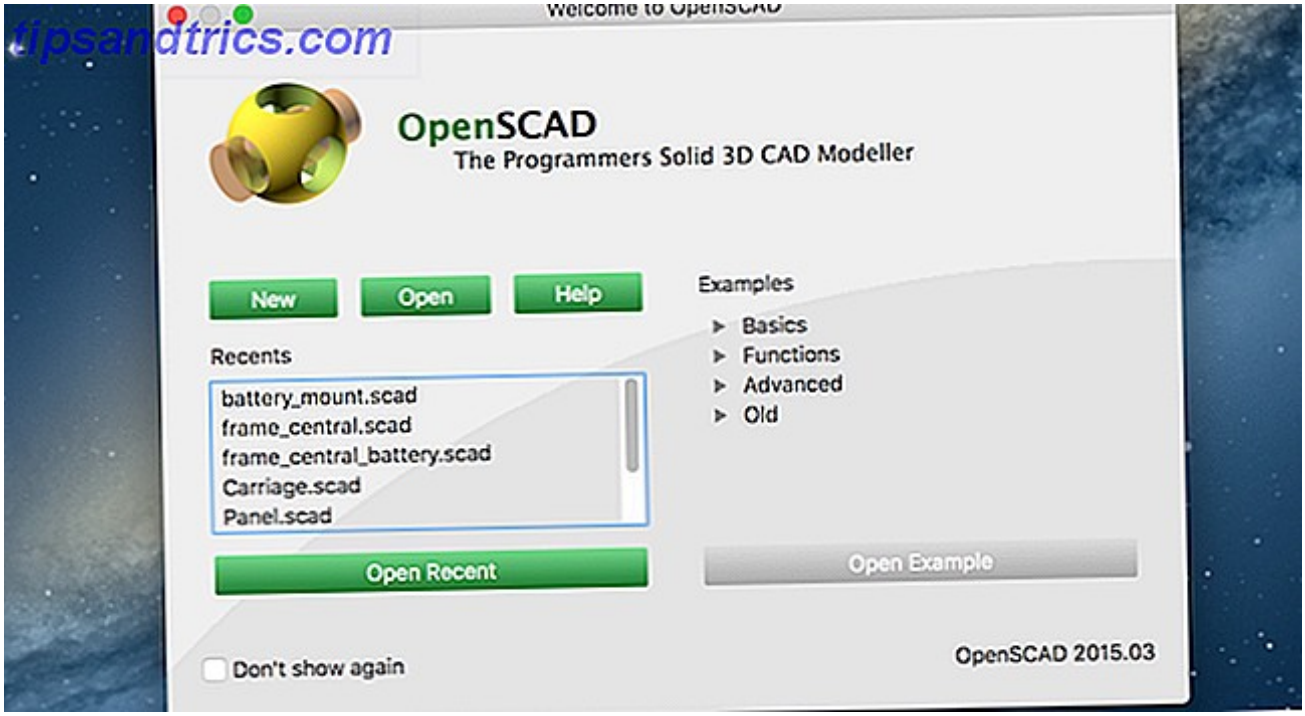
OpenSCAD est un modélisateur **Solid Computer Aided Design** gratuit. Il est disponible pour Windows, Mac et Linux. Ce qui le rend différent de beaucoup d'autres programmes, c'est que vous concevez des pièces en utilisant du code au lieu d'une souris. Cela facilite grandement les calculs mathématiques, le stockage des variables dans les variables, le redimensionnement des pièces, etc.

Vous devez tenir compte de certains facteurs lors de l'impression de modèles 3D, mais la plupart d'entre eux s'appliquent aux modèles CAO d'impression 3D en général, et pas seulement aux modèles OpenSCAD. Si vous voulez en savoir plus sur l'impression 3D, consultez notre guide Ultimate Beginner's Ultimate débutant Guide de l'impression 3D Ultimate Guide débutant à l'impression 3D L'impression 3D était censé être la nouvelle «révolution industrielle». Il n'a pas encore pris le contrôle du monde, mais je suis ici pour vous parler de tout ce que vous devez savoir pour commencer. Lire la suite . Si vous cherchez un modélisateur plus interactif, lisez le guide pour créer des objets dans Sketchup Design & Build 3D Bâtiments et objets virtuels avec Google SketchUp Conception et construction Bâtiments et objets virtuels 3D avec Google SketchUp Google SketchUp est le programme de modélisation de backbone pour Google BuildingMaker, qui permet aux concepteurs graphiques de soumettre des conceptions de bâtiments à Google pour les ajouter aux images officielles de Google Earth. Lire la suite .

Mise en place

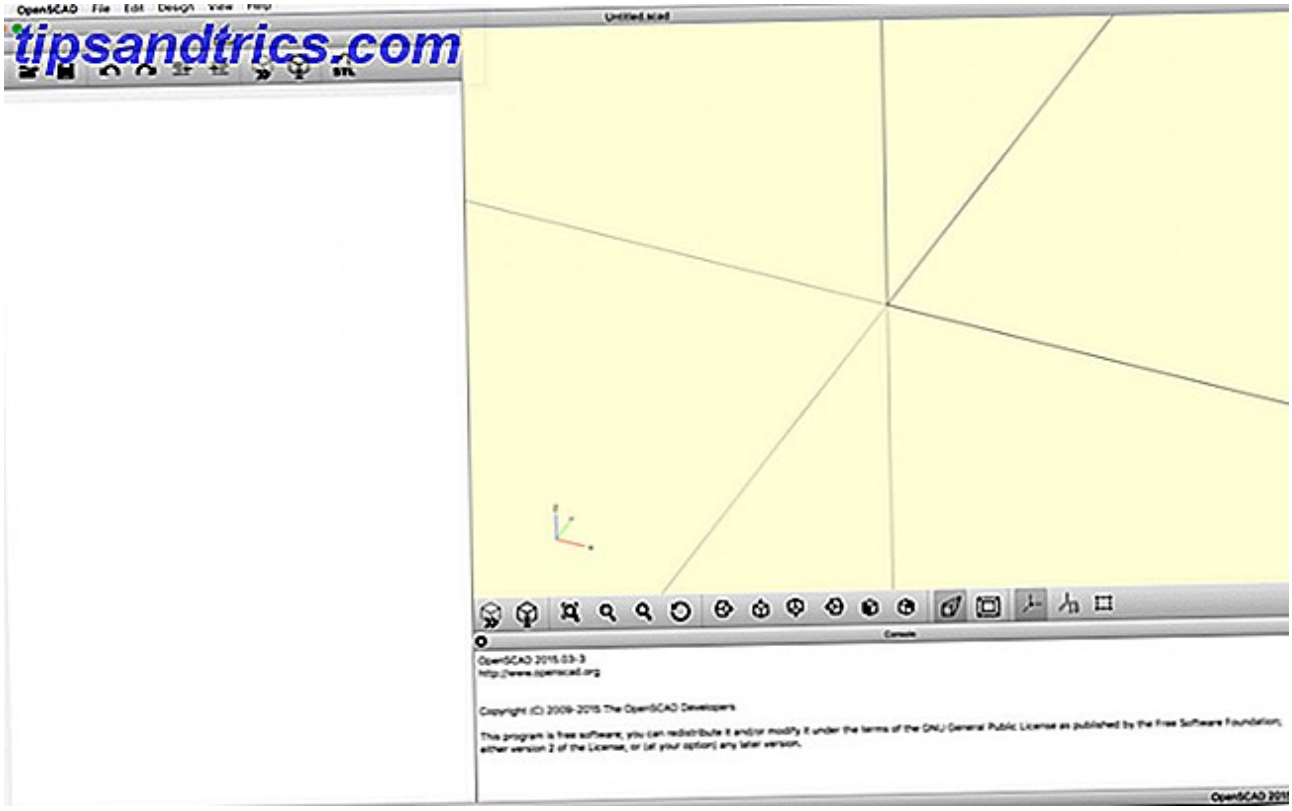
Tout d'abord, rendez-vous sur la page des téléchargements et trouvez une version d'OpenSCAD adaptée à votre système d'exploitation. J'utilise Mac OS, mais ces principes OpenSCAD s'appliquent à tous les systèmes.

Une fois installé, allez-y et ouvrez-le. Vous serez présenté avec ce menu de démarrage:

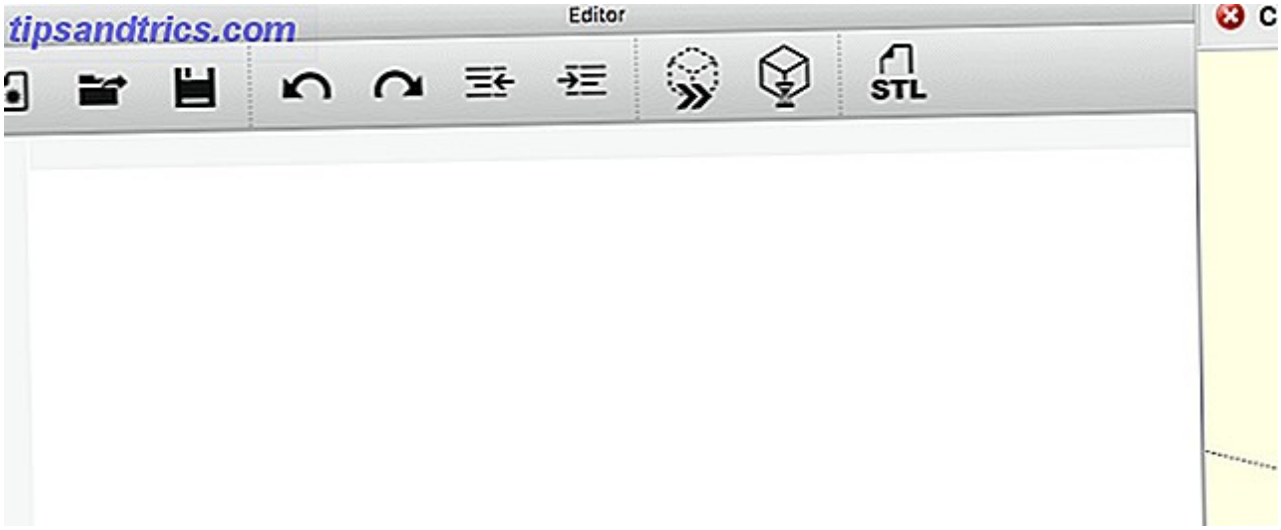


Cela vous montre les fichiers que vous avez ouverts en dernier et vous donne l'option de charger quelques exemples. N'hésitez pas à regarder autour de certains des exemples, mais j'ai trouvé que cela rendait les choses plus confuses au début. Pour ce tutoriel, créez un nouveau fichier en cliquant sur le **nouveau** bouton.

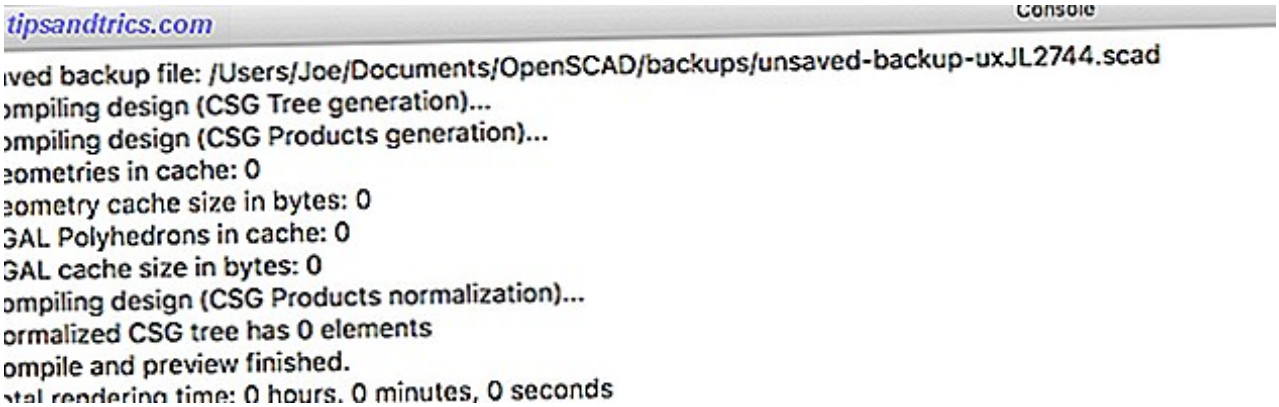
Une fois ouvert, vous serez présenté avec cette interface à nu:



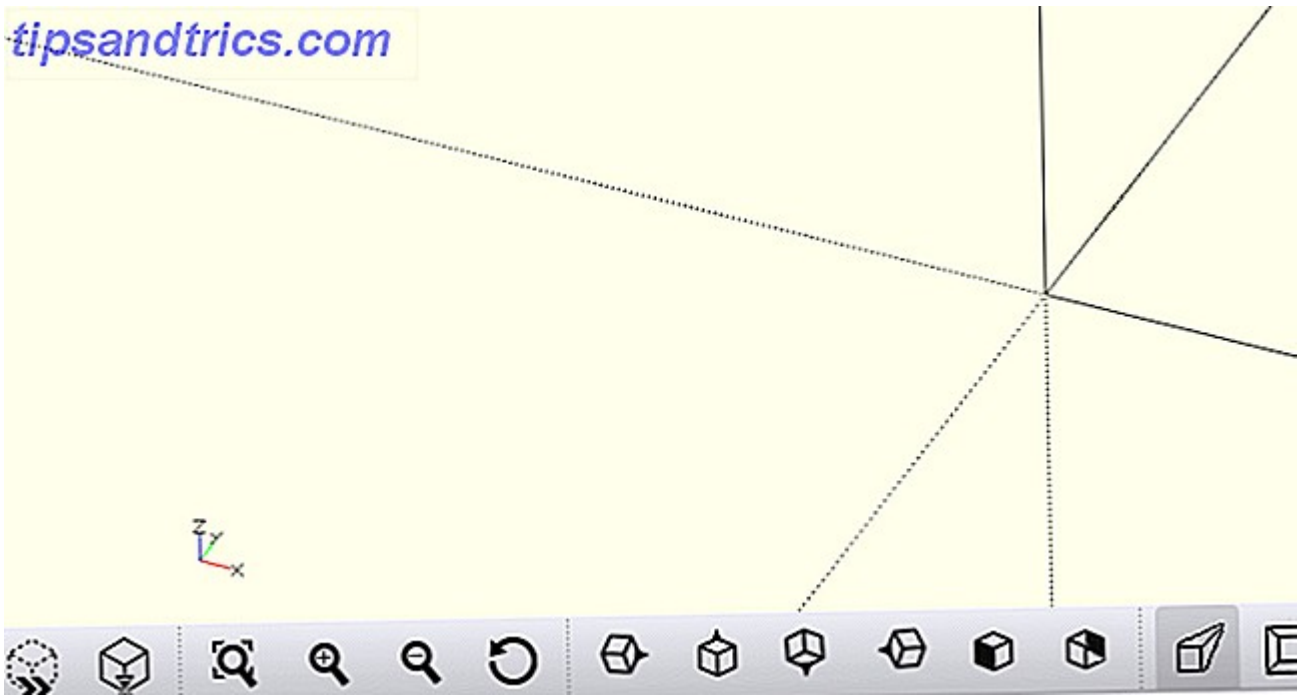
Ceci est divisé en trois zones principales. Sur la gauche est votre **éditeur** et votre menu. C'est ici que vous allez écrire votre code. Il n'y aura pas encore de code, car vous créez un nouveau fichier. En haut, il existe des boutons de menu pour effectuer des tâches de base, telles que charger, enregistrer, annuler, etc.



Le coin inférieur droit est la **console** . Cela vous montrera toutes les erreurs dans la construction du modèle.



La dernière section est l' **interface principale** en haut à droite. Ici vous pouvez interagir avec votre modèle, mais vous ne pourrez pas le modifier ici (vous allez écrire du code pour le faire).



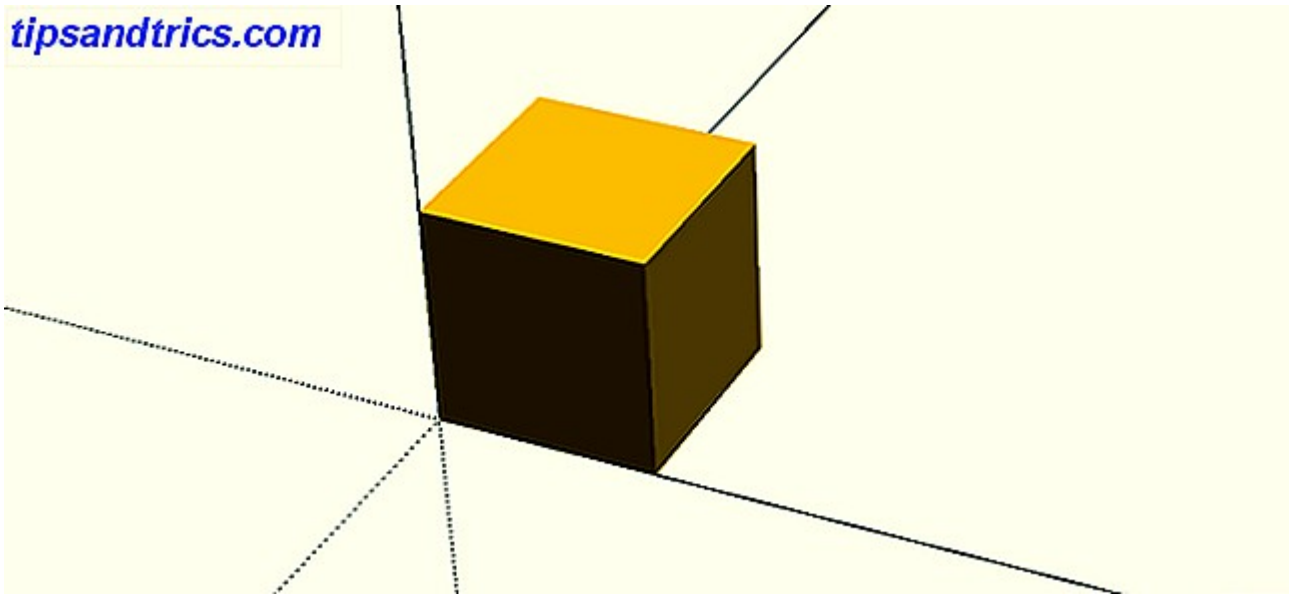
Il y a plusieurs boutons en bas de cette interface principale. Ceux-ci vous permettent principalement de voir votre design de différentes manières.

Allez-y et enregistrez un nouveau fichier en appuyant sur le **bouton Enregistrer** dans le **menu de** l' **éditeur** ou en allant dans **Fichier** > **Enregistrer** .

Les bases

La façon dont OpenSCAD fonctionne la plupart du temps est l'addition et la soustraction de formes simples. Vous pouvez construire des modèles très complexes de cette façon, alors allons-y tout de suite.

Voici la première forme, une boîte simple:



Et voici le code pour produire cela:

```
cube(); // create a cube
```

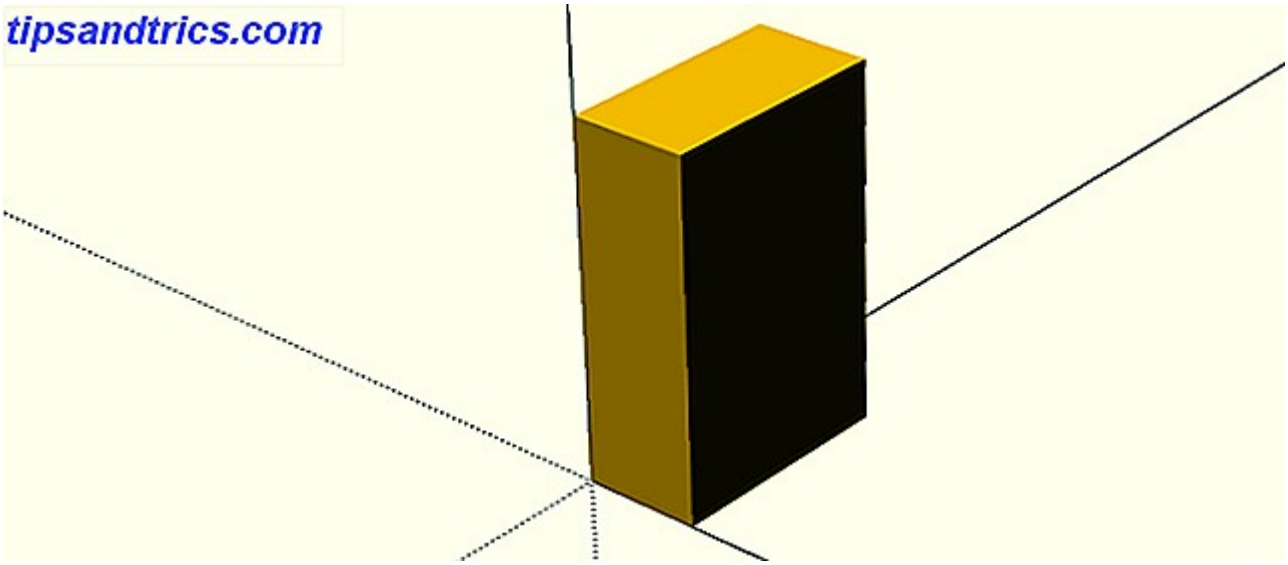
Pour que votre code s'exécute et que vous construisiez le modèle, vous devez le prévisualiser. OpenSCAD le fera par défaut à chaque fois que vous enregistrez, ou vous pouvez appuyer sur **F5** pour forcer une actualisation. Expérimentez en vous déplaçant dans l'espace 3D en maintenant les boutons gauche ou droit de la souris enfoncés.

Maintenant, cela produit un joli cube, mais ce n'est pas très utile sans aucune dimension. OpenSCAD ne fonctionne pas dans un système de mesure particulier, à la place, les unités sont toutes les unes par rapport aux autres. Vous pouvez créer une boîte 20 x 10, et c'est à n'importe quel autre programme (comme votre trancheuse d'impression 3D) d'interpréter ces valeurs, que ce soit métrique ou impérial. En fait, il offre une grande flexibilité.

Ajoutons quelques dimensions à votre cube. Vous faites cela passer des paramètres à la méthode du **cube** :

```
cube(size = [10, 20, 30]); // rectangle
```

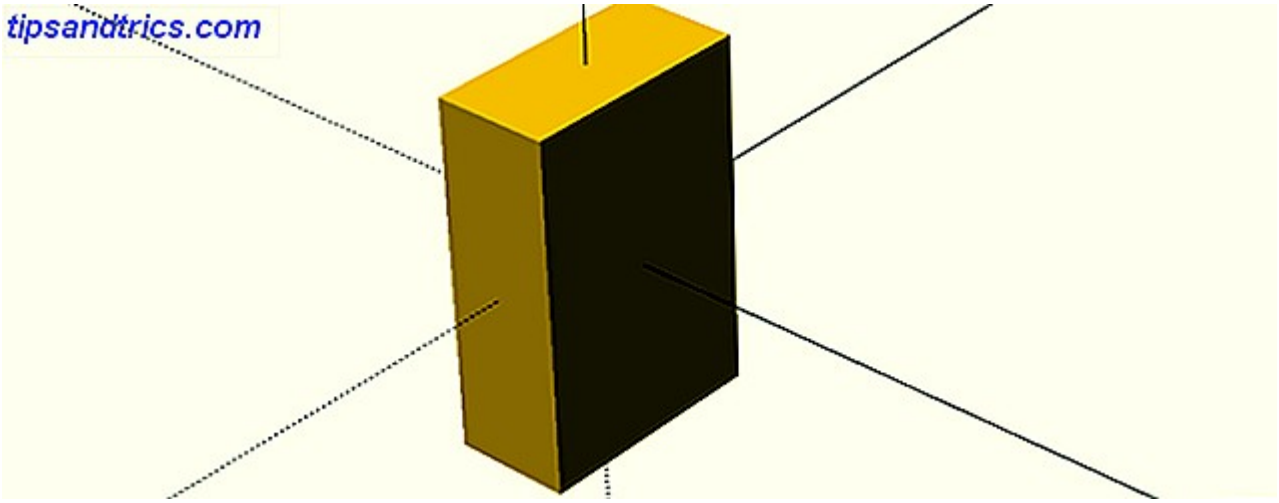
Les valeurs **10**, **20** et **30** représentent la taille du cube dans les axes **X**, **Y** et **Z.** Remarquez comment cela a produit un rectangle beaucoup plus grand:



Par défaut, OpenSCAD dessine les composants en bas à gauche. Vous pouvez ajuster cela en définissant le paramètre **center** sur **true** . Voici le code pour faire cela au rectangle:

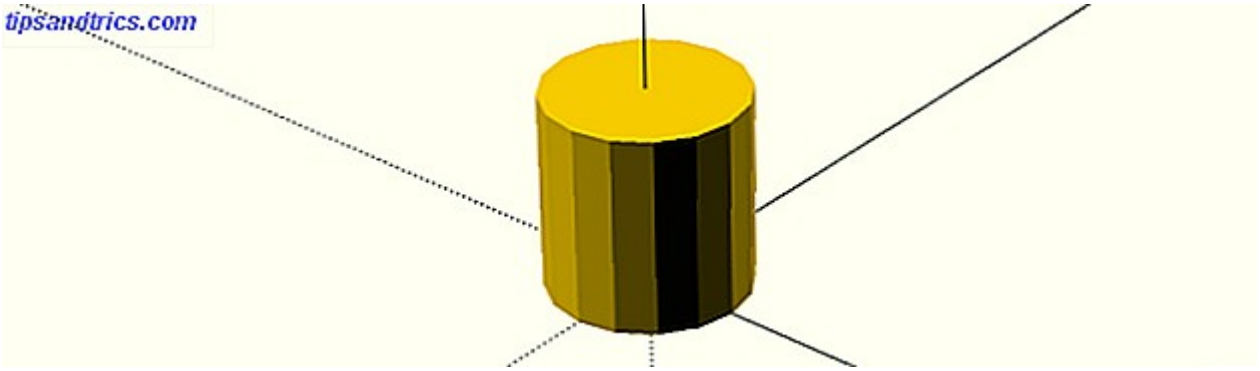
```
cube(size = [10, 20, 30], center = true); // rectangle centered
```

Et voici à quoi ça ressemble:



Centrer des objets fonctionne bien pour des formes simples, mais cela complique les choses pour les objets non symétriques. Vous devrez décider quelle méthode vous convient le mieux.

Passons à une forme plus complexe, voici un **cylindre** :



Voici le code pour le créer:

```
cylinder(d = 10, h = 10, center = true); // cylinder
```

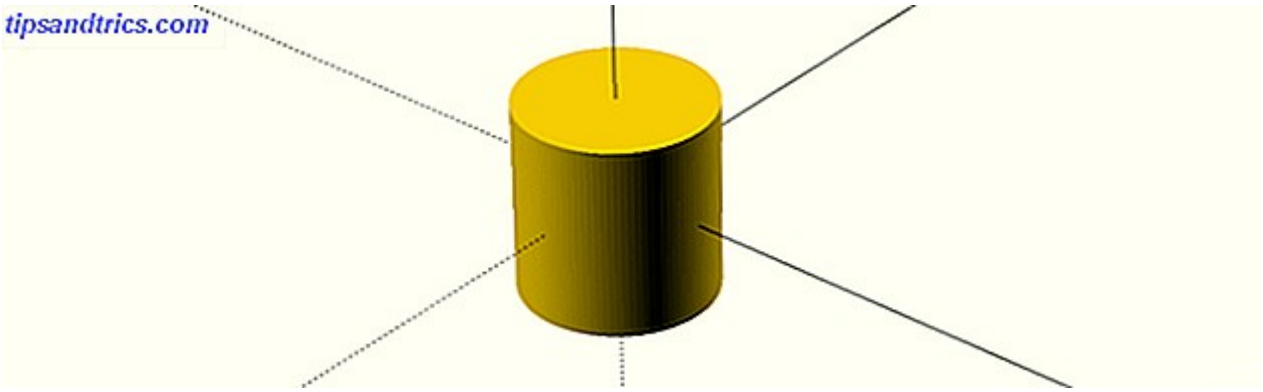
Contrairement aux **cubes**, les **cylindres** sont automatiquement dessinés au centre des axes X et Y. Le paramètre **d** représente le **diamètre** (vous pouvez passer dans le rayon à la place si vous préférez). Le paramètre **h** est la hauteur. Quelque chose ne va pas ici. Ce cylindre a l'air plutôt "blocky". Vous devez augmenter le nombre de faces dessinées sur la circonférence. C'est facile à faire - ajoutez le paramètre suivant à votre code de cylindre.

```
$fn = 100
```

Donc la définition du cylindre devient:

```
cylinder(d = 10, h = 10, center = true, $fn = 100);
```

Voici à quoi cela ressemble:

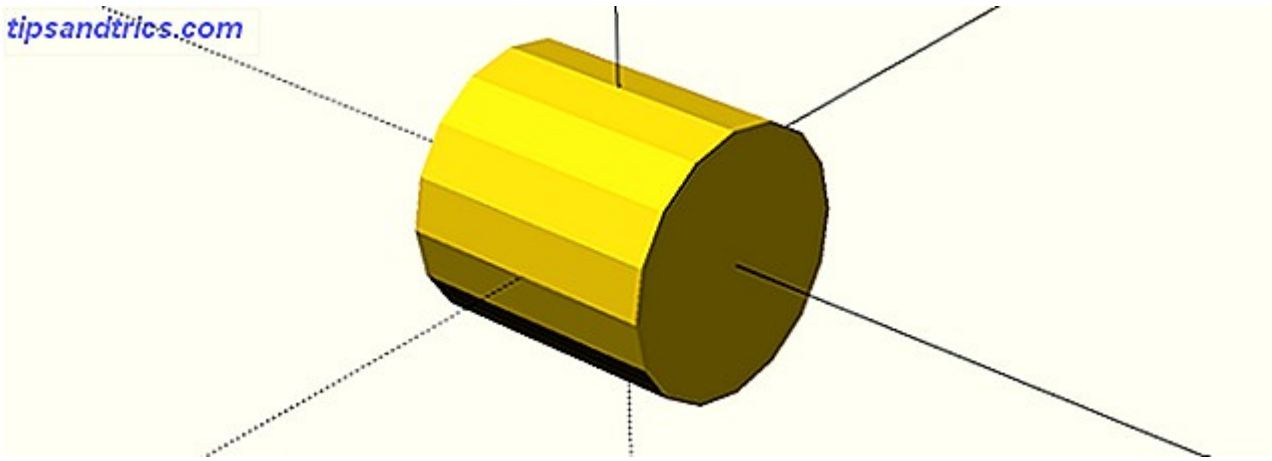


Cela augmente le nombre de faces nécessaires pour faire des cercles - 100 est un bon point de départ. Gardez à l'esprit que cela augmentera considérablement les temps de rendu, en particulier sur les modèles complexes, il est donc généralement préférable de laisser cela de côté jusqu'à ce que vous ayez fini de concevoir.

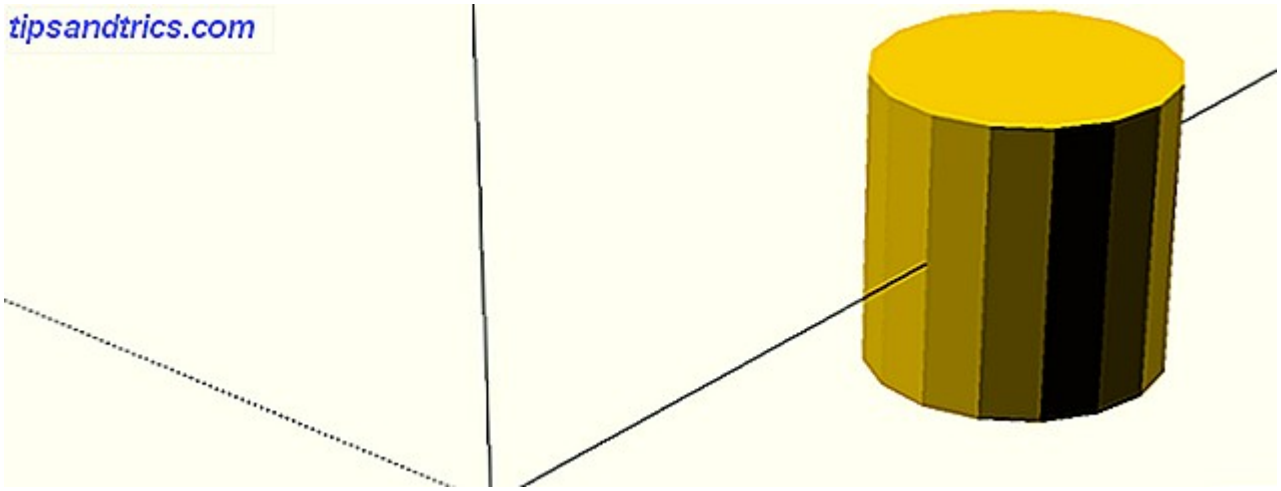
Il est facile d'appliquer des transformations sur des formes. Vous devez appeler des méthodes spéciales avant de créer vos formes. Voici comment faire pivoter le cylindre en utilisant la méthode de **rotation** :

```
rotate(a = [0, 90, 0]) cylinder(d = 10, h = 10, center = true); //
rotated cylinder
```

Les valeurs transmises au paramètre a représentent l'angle de rotation des axes X, Y et Z. Voici le résultat:



Une autre fonction très utile est la **traduction** . Cela vous permet de déplacer des objets dans l'espace 3D. Encore une fois, vous devrez transmettre la quantité de mouvement pour chaque axe. Voici le résultat:

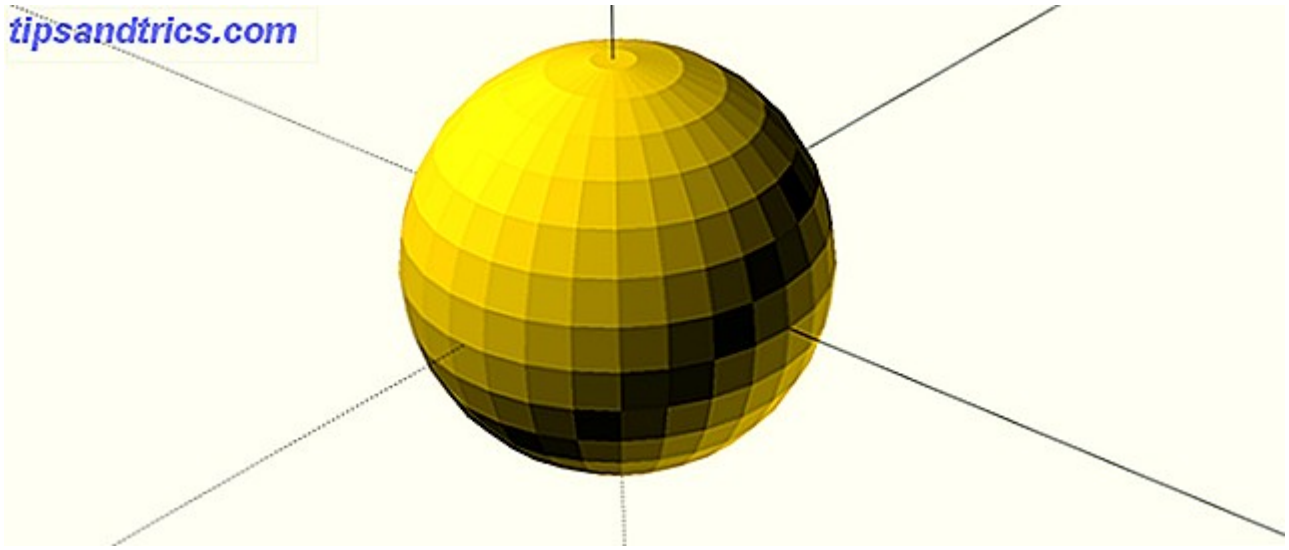


Voici le code:

```
translate(v = [0, 25, 0]) cylinder(d = 10, h = 10, center = true); //
translated cylinder
```

Comprendre la méthode de **traduction** est l'une des choses les plus importantes que vous puissiez faire. C'est nécessaire pour concevoir les conceptions les plus complexes.

Enfin, une autre forme utile est une **sphère** :



Voici le code:

```
sphere(d = 100);
```

Tout comme le cylindre, vous pouvez lisser ceci en utilisant le code **$ fn** ci-dessus.

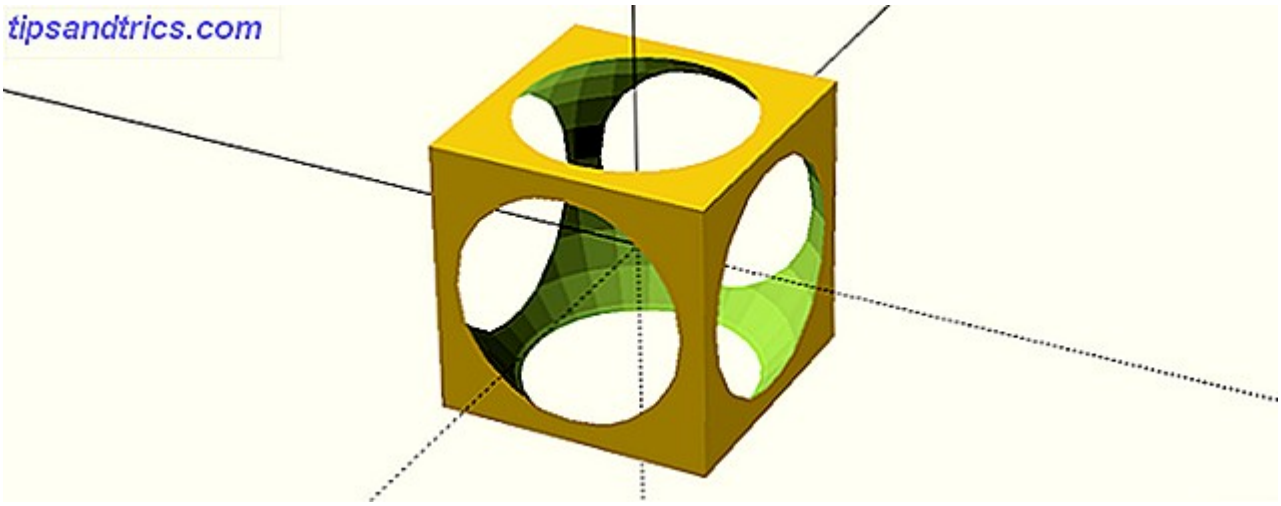
Codage avancé

Maintenant que vous connaissez les bases, regardons quelques compétences plus avancées. Lors de la conception d'une pièce, il est utile de réfléchir à la façon dont elle pourrait être composée de formes et d'objets plus petits. Vous n'avez pas à faire cela, et vous pouvez «faire les choses» au fur et à mesure, mais cela aide à avoir un plan approximatif - même si c'est seulement dans votre tête.

Créons une forme avancée: un cube avec un intérieur de sphère évidé. Créez un **cube** et une **sphère** avec le **centre** défini sur true. Soustrayez l'un de l'autre en utilisant la méthode de **différence** :

```
difference() { // subtraction
cube(size = [50, 50, 50], center = true); // outer cube
sphere(d = 65, center = true); // inner sphere }
```

Voici le résultat:



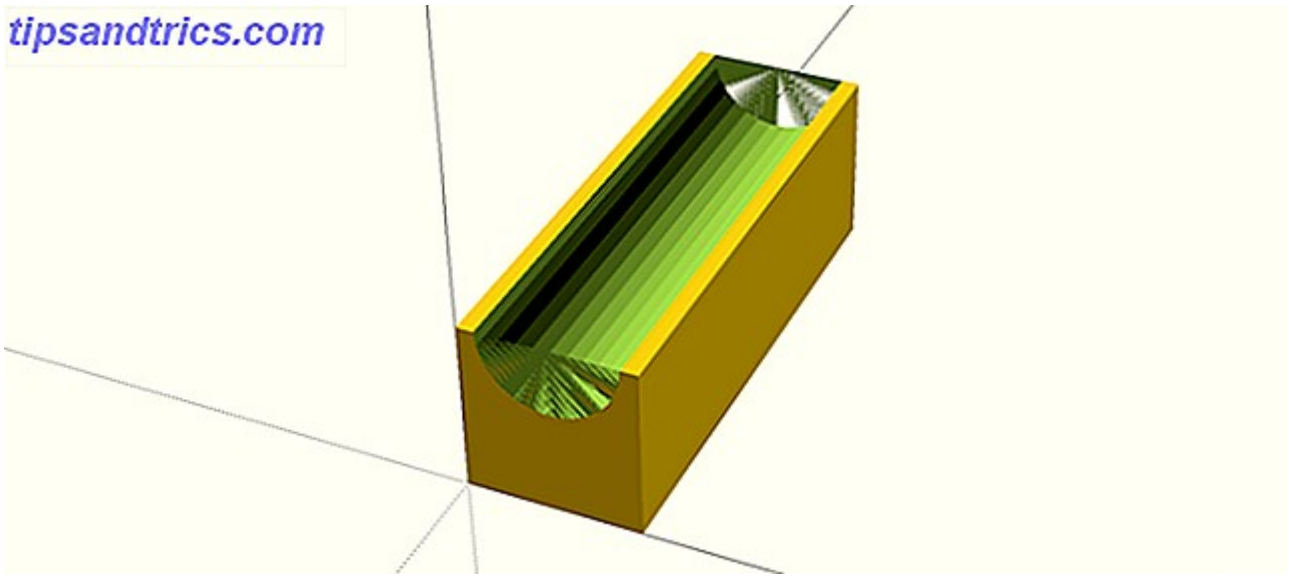
Expérimentez avec le diamètre (paramètre **d**) de la sphère et voir ce qui se passe.

Dans OpenSCAD, il existe généralement plusieurs façons d'accomplir la même tâche. Si vous vouliez un groove dans un cube, vous pourriez en soustraire un autre ou en ajouter deux autres au-dessus. Peu importe la façon dont les choses sont faites, mais selon la complexité de la pièce, il peut être plus facile de faire certaines manipulations en premier.

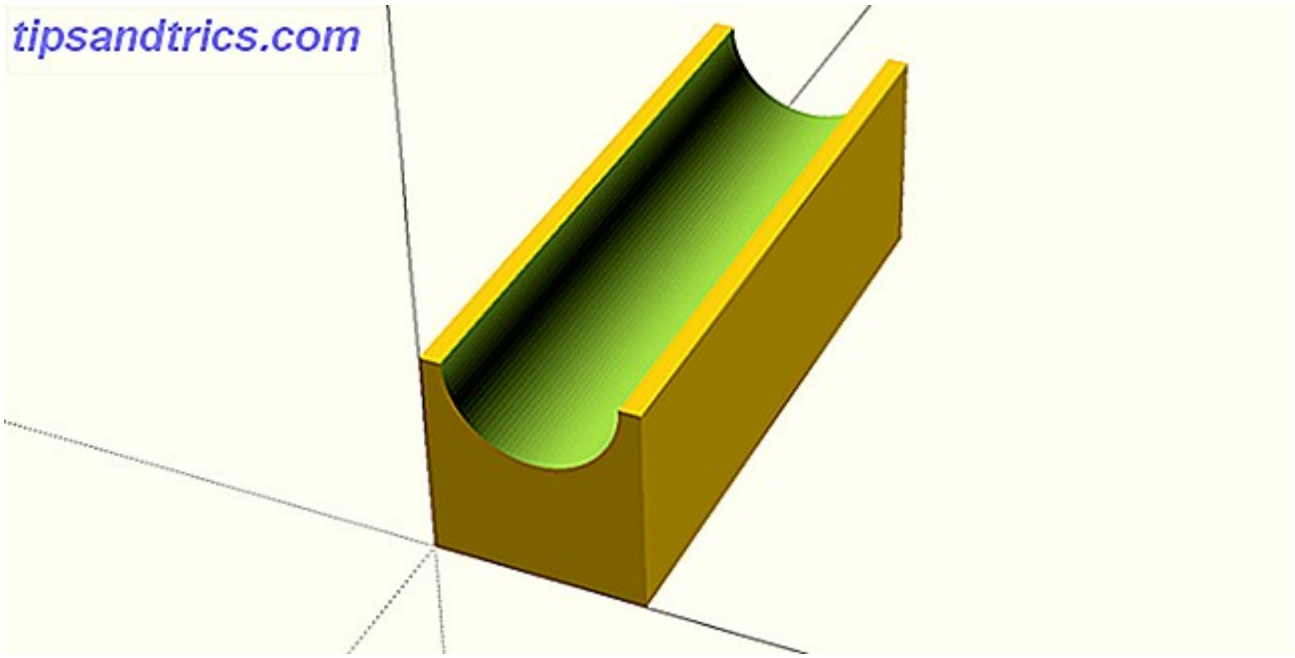
Voici comment créer un canal dans un cube. Au lieu d'utiliser un autre cube, l'utilisation d'un cylindre crée un canal arrondi. Notez comment la méthode de **différence** est utilisée une fois de plus, et comment les méthodes de **translation** et de **rotation** sont utilisées pour manipuler les formes. L'utilisation de la méthode de **rotation** rend souvent les transformations difficiles, alors jouez avec les paramètres jusqu'à ce que vous obteniez le résultat souhaité. Voici le code:

```
difference() { // subtraction
cube(size = [50, 150, 50]); //
outer cube
translate(v = [25, 150, 50])
rotate(a = [90, 0, 0]) cylinder(d = 40, h = 150); // cylinder channel }
```

Voici à quoi cela ressemble:



Vous vous demandez peut-être ce que sont les trucs verts. C'est ici parce que le modèle 3D est juste un aperçu en ce moment. Pour résoudre ce problème, appuyez sur **F6** pour rendre le modèle entièrement. Cela peut prendre un certain temps, selon la complexité. L'aperçu (**F5**) est généralement assez bon tout en travaillant. Voici à quoi ressemble le rendu final (avec **$ fn** défini sur 100):



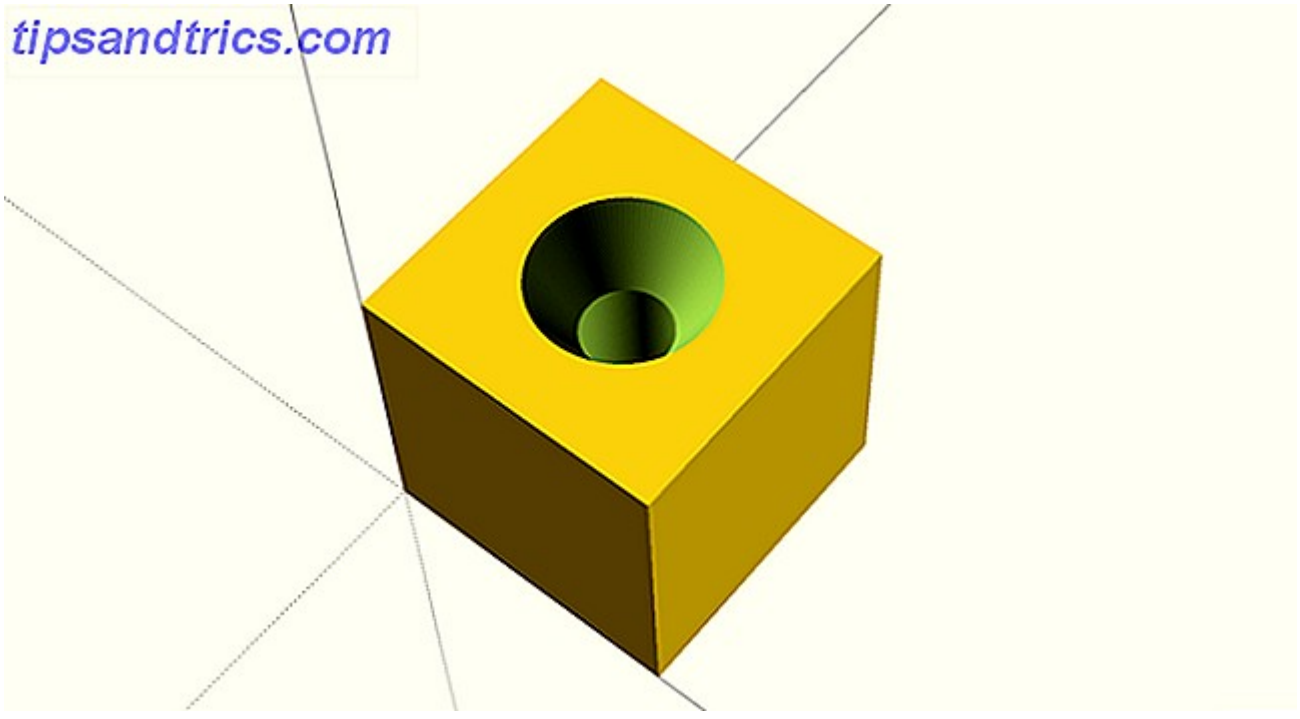
Voici un autre exemple avancé. Dire que vous vouliez un montage quelque chose en utilisant un boulon. Créer un trou est assez simple en utilisant un **cylindre**, mais que se passerait-il si vous vouliez que la tête de boulon soit encastrée pour les boulons à tête fraisée? Vous pourriez simplement créer un grand cylindre pour que la tête du boulon

puisse s'asseoir, mais cela ne serait pas très joli. La solution est un chanfrein, que vous pouvez créer avec la méthode du **cylindre** . L'astuce ici est de spécifier deux diamètres - **d1** et **d2** . Faites ces différentes tailles, et OpenSCAD fera le reste.

Comme je suis britannique, j'utiliserai des dimensions métriques ici, pour un boulon à tête fraisée M5. Vous pouvez facilement ajuster ceci pour s'adapter à toutes les fixations que vous souhaitez utiliser. Voici le code:

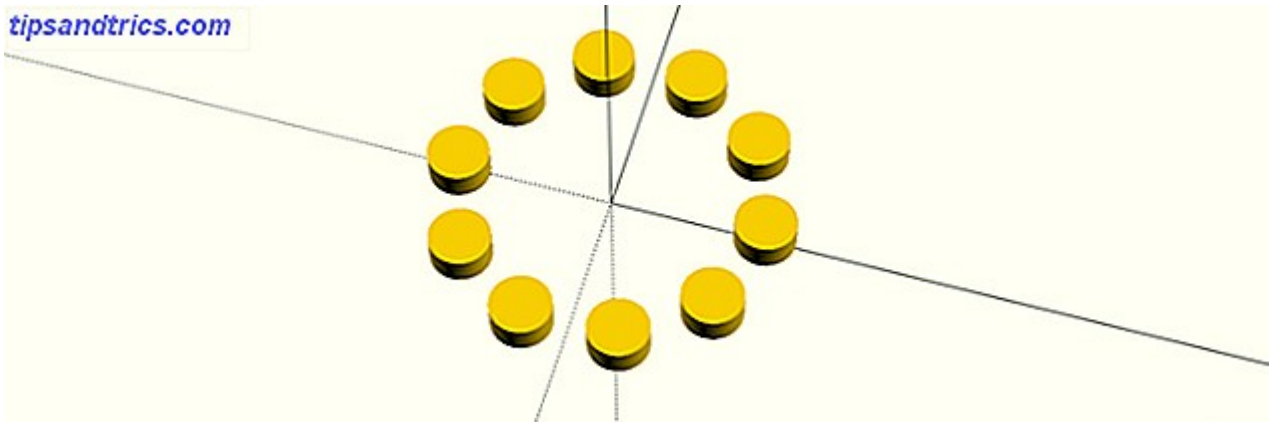
```
$fn = 100; // bolt settings m5_
clearance_diameter = 5.5;
m5_head_clearance_diameter = 11;
m5_head_depth = 5;
difference() { // subtract
cube(20, 20, 20); bolt_hole(10, 10, 20);
bolt_bevel(10, 10, 15); }


module bolt_hole(x, y, height) {
/* M5 hole at 90 deg. */
translate(v = [x, y, 0]) cylinder(d = m5_clearance_diameter, h = height);
}
module bolt_bevel(x, y, z) { // M5 bevel translate(v = [x, y, z])
cylinder(d2 = m5_head_clearance_diameter, d1 = m5_clearance_diameter,
h = m5_head_depth); }
```

Remarquez comment les dimensions des boulons sont stockées dans des variables? Cela facilite grandement le codage et la maintenance. Une méthode que vous n'avez peut-être pas encore rencontrée est le **module** . Cela vous permet de définir un bloc de code à exécuter quand vous le souhaitez. En réalité, c'est une *fonction* . Vous devriez utiliser des **modules** et des **variables** pour toute forme complexe, car ils facilitent la lecture et accélèrent les modifications. Voici à quoi ressemble le chanfrein:



Regardons un dernier exemple. Supposons que vous vouliez produire une série de trous autour d'un cercle. Vous pouvez les mesurer, les traduire et les faire pivoter

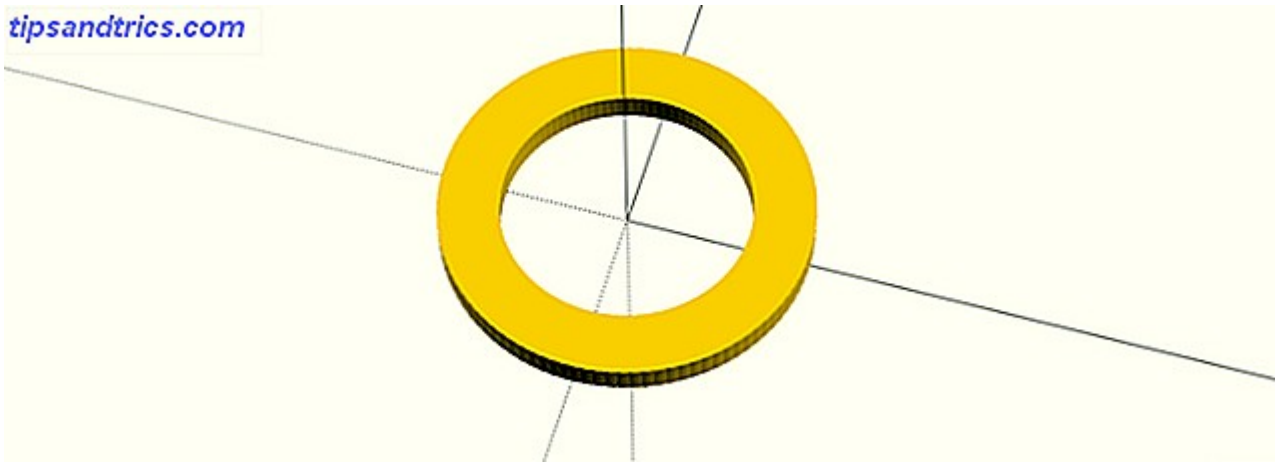
manuellement, mais même avec des modules, cela serait fastidieux. Voici le résultat final, 10 cylindres même répartis autour d'un cercle:



Voici le code:

```
$fn = 100;
number_of_holes = 10;
for(i = [1 : 360 / number_of_holes : 360]) {
// number_of_holes defines number of times this code runs
make_cylinder(i); }

module make_cylinder(i) { // make cylinder and even distribute
rotate([0, 0, i]) translate([10, 0, 0]) cylinder(h = 2, r = 2); }
```

Ce code est plus simple que prévu. Une boucle **for** est utilisée pour appeler le module **make_cylinder** dix fois. Comme il y a 360 degrés dans un cercle et 360/10 = 36, chaque cylindre doit être tourné par incréments de 36 degrés. Chaque itération de cette boucle incrémente la variable **i** de 36. Cette boucle appelle le module **make_cylinder**, qui dessine simplement un cylindre et le positionne en fonction des degrés qui lui sont transmis par la boucle. Vous pouvez dessiner plus ou moins de cylindres en modifiant la variable **number_of_holes** - bien que vous souhaitiez ajuster l'espacement si vous le faites. Voici à quoi ressemblent 100 cylindres, ils se chevauchent légèrement:



Exporter

Maintenant que vous savez coder dans OpenScad, une dernière étape est nécessaire avant de pouvoir imprimer vos modèles en 3D. Vous devez exporter votre conception

d'OpenSCAD dans le format **STL** standard utilisé par la plupart des imprimantes 3D. Heureusement, il y a un bouton d'exportation vers STL: **Menu de l'éditeur** > En **haut à droite** :

C'est tout pour aujourd'hui. Vous devriez maintenant avoir une excellente connaissance pratique d'OpenSCAD - toutes les choses complexes se construisent sur ces fondations, et de nombreuses formes complexes sont vraiment beaucoup de formes simples combinées.

Pour un défi, pourquoi ne pas regarder certains de nos projets d'impression 3D, et essayer de recréer les pièces dans OpenSCAD:

- Les meilleures imprimables 3D pour RPG sur table Les meilleurs imprimables 3D pour RPG sur table Si vous préférez une expérience immersive pour les jeux de rôle, voici une nouvelle façon de faire: utiliser l'impression 3D pour créer des pièces de terrain physiques et miniatures. Lire la suite
- Boutons de raccourci personnalisés Faites vos propres boutons de raccourci personnalisés avec un Arduino Faites vos propres boutons de raccourci personnalisés avec un Arduino L'humble Arduino peut faire beaucoup de choses, mais saviez-vous qu'il peut émuler un clavier USB? Vous pouvez combiner de longs raccourcis clavier en une seule touche de raccourci personnalisée, avec ce circuit simple. Lire la suite
- D20 électronique Roll dans le style avec ce DIY électronique D20 Die Roll dans le style avec ce Die électronique DIY D20 Vous voulez quelque chose d'un peu unique à votre prochain meetup de jeu? Découvrez ce D20 électronique DIY, avec des graphismes personnalisés pour les coups critiques et les échecs. Lire la suite
- Les jeux que vous pouvez imprimer en 3D 6 Les jeux les plus cool que vous pouvez imprimer en 3D 6 Les jeux les plus cool que vous pouvez imprimer en 3D Tout le monde connaît les imprimantes 3D, mais ce que vous ne savez pas, c'est toute ta famille. Nous parlons de jeux de table imprimés en 3D. Lire la suite

As-tu appris de nouveaux trucs aujourd'hui? Quelle est votre fonctionnalité OpenSCAD préférée? Passerez-vous bientôt à un autre outil de CAO? Faites-nous savoir dans les commentaires ci-dessous!

Source : fr.tipsandtrics.com