Brève introduction aux **logiciels de 3D**

Module SPI6

Jean-Marie Favreau
mai 2013

Brève introduction aux **logiciels de 3D**

Module SPI6

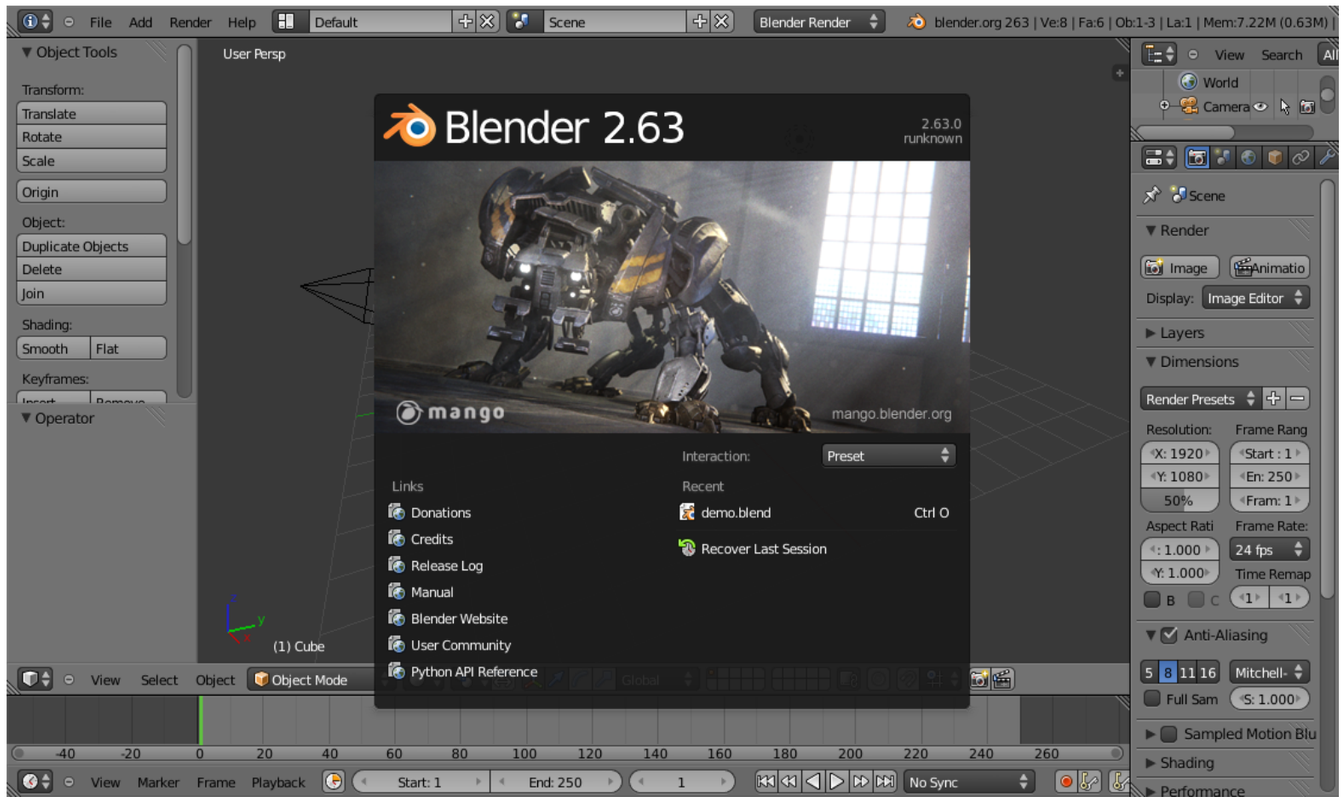
Jean-Marie Favreau

mai 2013

Les logiciels de **modélisation**, d'**animation** et de **rendu 3D**

Logiciels de 3D

Une brève introduction

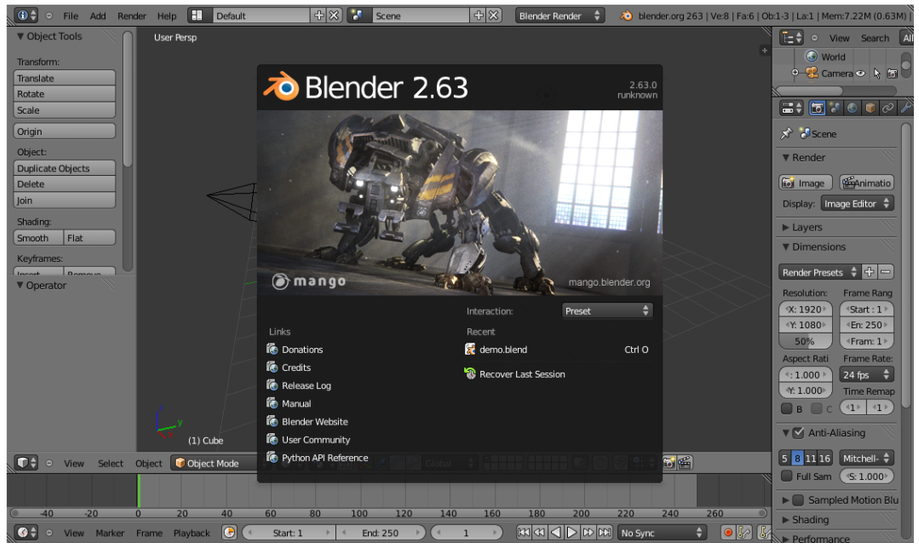


- L'exemple de Blender
- Qu'est-ce qu'une scène 3D ?
- Qu'est-ce qu'un objet 3D ?
- Principes d'affichage et de rendu
- Modélisation et animation
- Texture et matériaux
- Découverte de l'interface de Blender

Les logiciels de **modélisation**, d'**animation** et de **rendu 3D**

Blender

Un logiciel à l'histoire tumultueuse

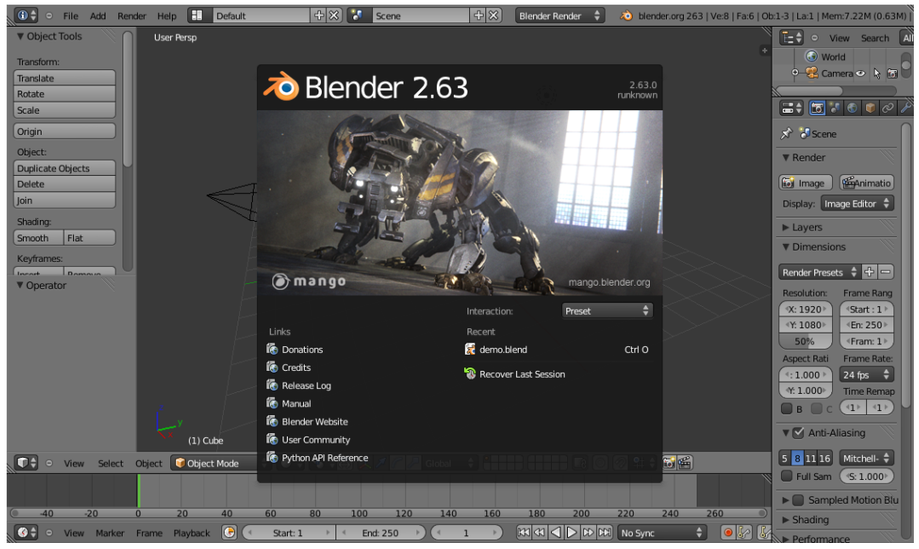


Chronologie :

- **1989 :** logiciel développé pour NeoGeo
- **1998 :** NaN distribue *blender* en partagiciel
- **2002 :** libération du code source (GPL v2 ou suivantes)
- début de la refonte du logiciel
- **2011 :** sortir de la version 2.5 (refonte du logiciel)
- **2012 :** réécriture de l'interface python

Blender

Quelques caractéristiques



Site internet :

- http://blender.org

Plateformes :

- Microsoft Windows, GNU/Linux, MacOS X, FreeBSD

Langage :

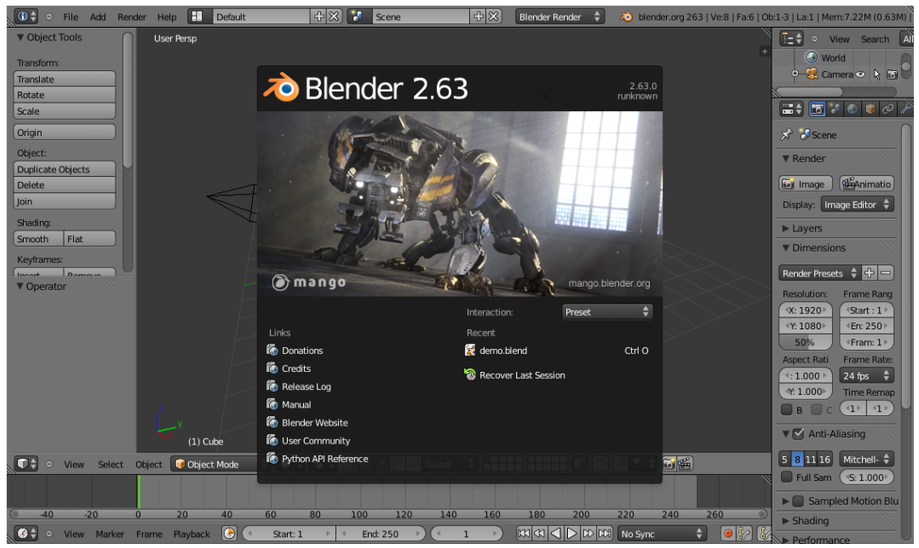
- C++ et python

Licence :

- GPL v2 ou suivantes

Blender

Une suite complète



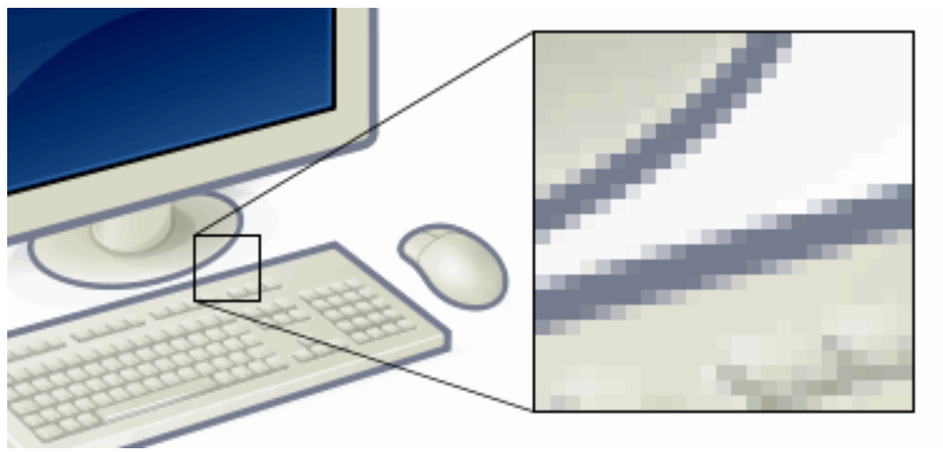
Fonctionnalités :

- Interface de modélisation
- Larges possibilités de textures, matériaux
- Animation
- Moteur de rendu intégré (+ interfaçage à yafray, POV-ray, ...)
- Moteur physique, particules
- Moteur de jeu (interface joueur, etc.)
- Interface de montage

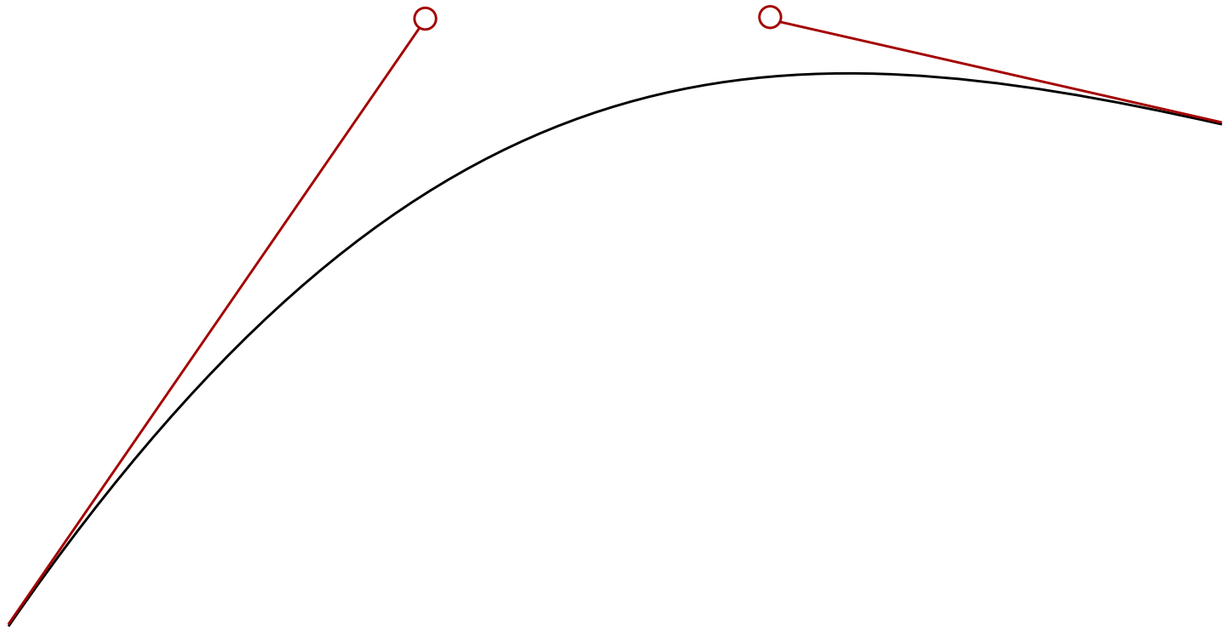
Modélisation

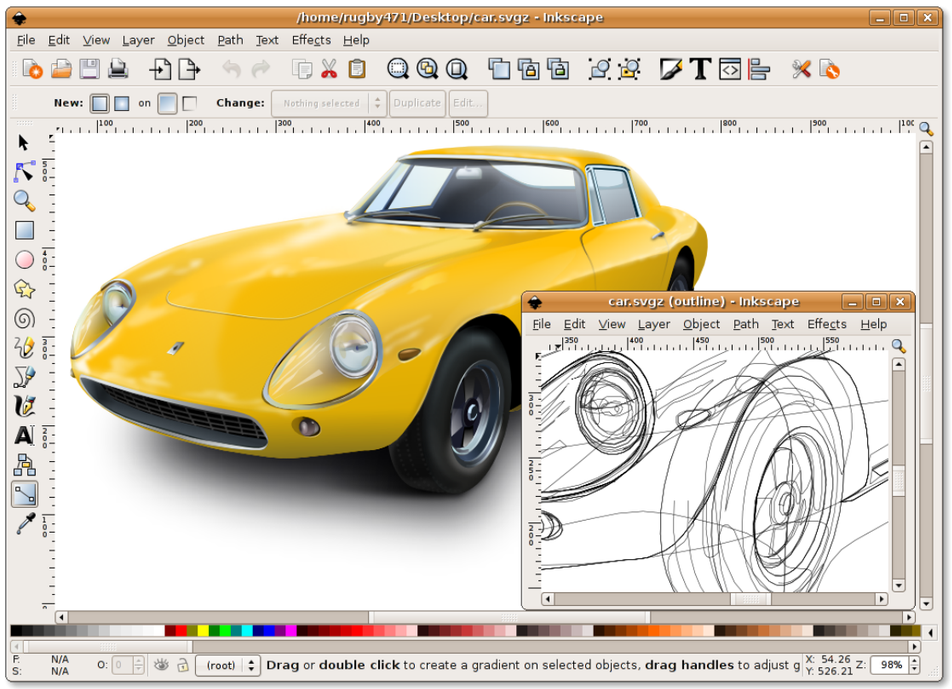
Qu'est-ce qu'une image en informatique ?

- Images **bitmap** (= raster, discrètes)



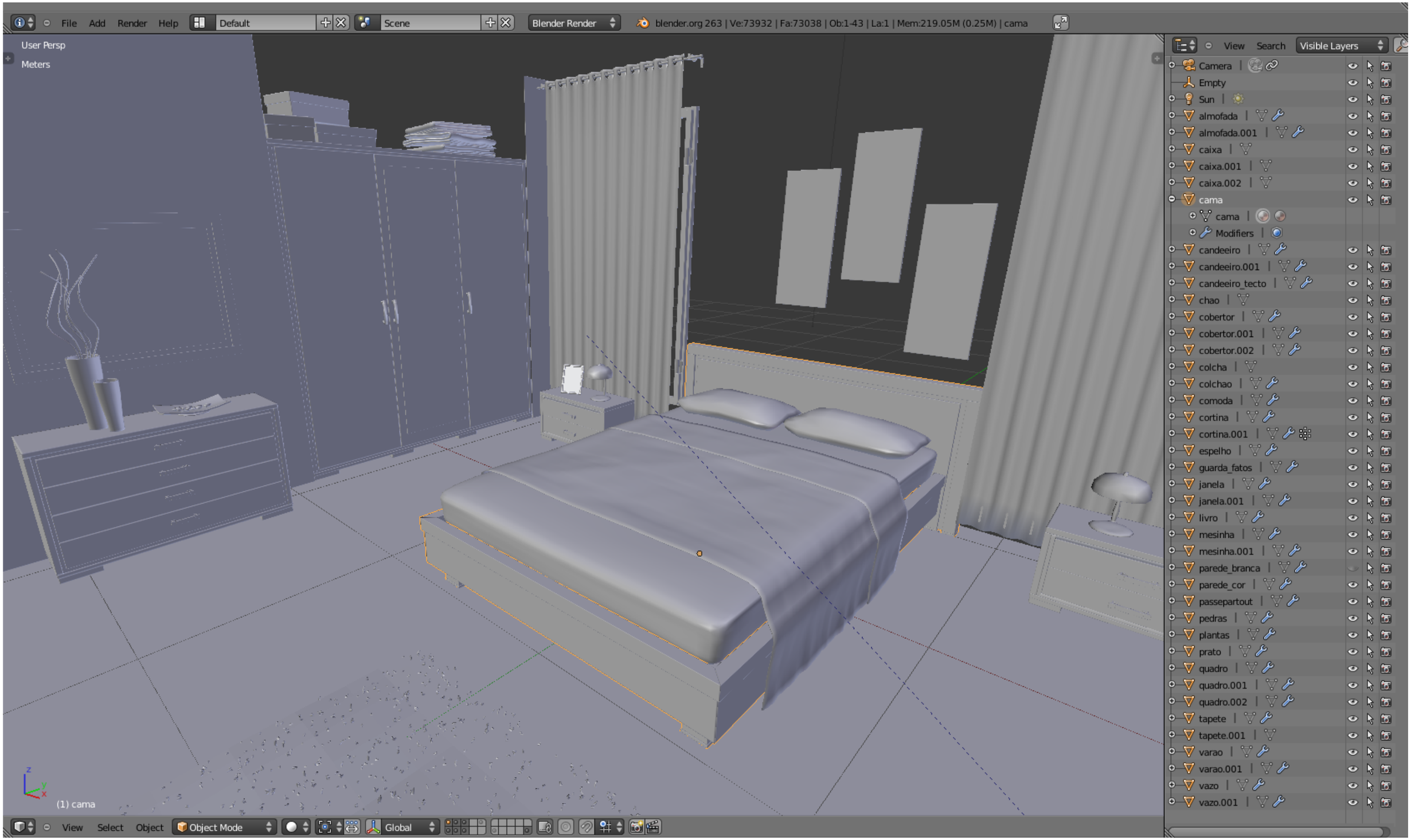
- Images **vectorielles**





Modélisation

Qu'est-ce qu'une scène 3D ?

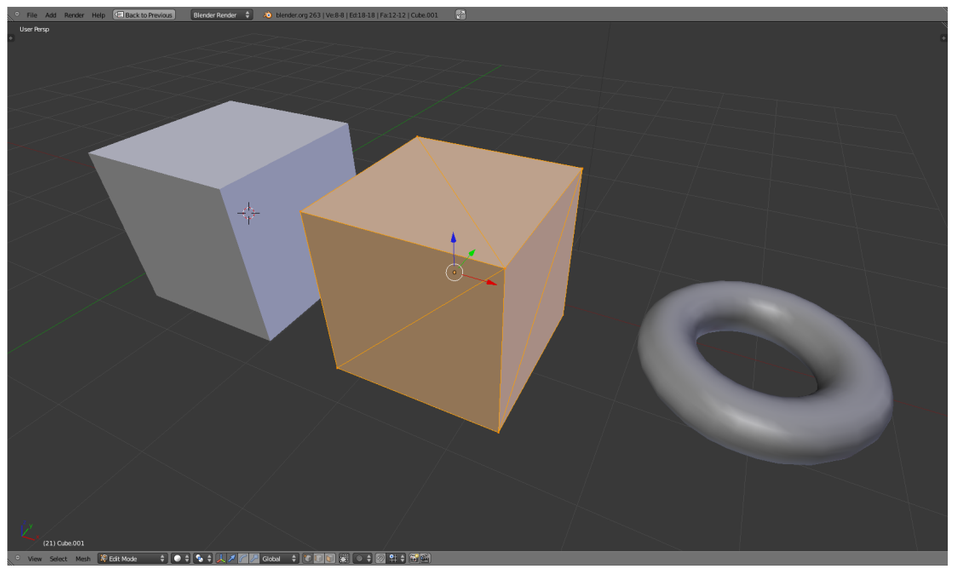


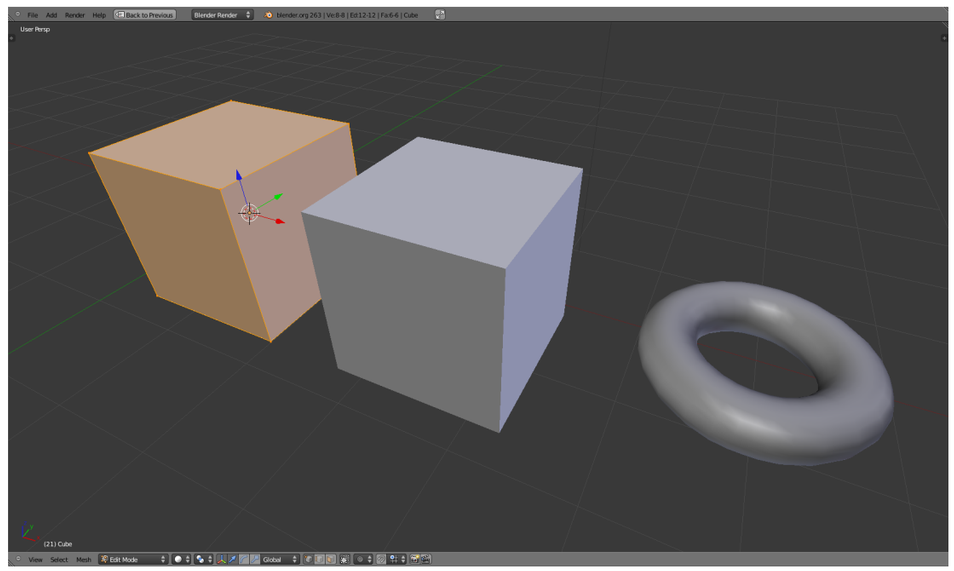
Modélisation

Qu'est-ce qu'un objet 3D ?

Surfaces

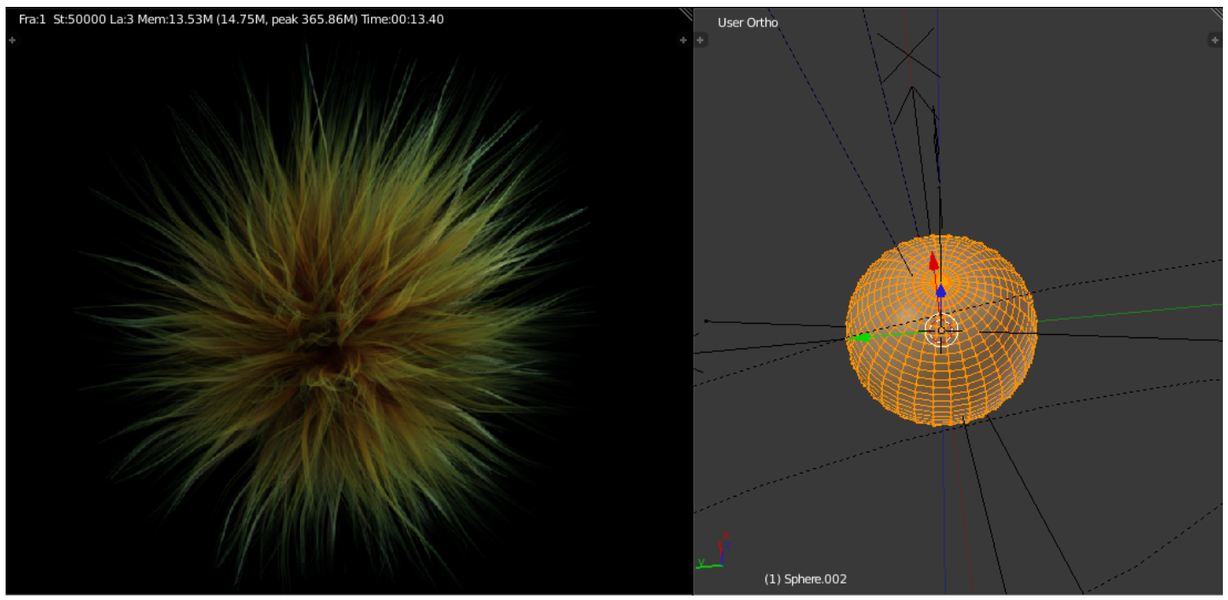
- Maillages

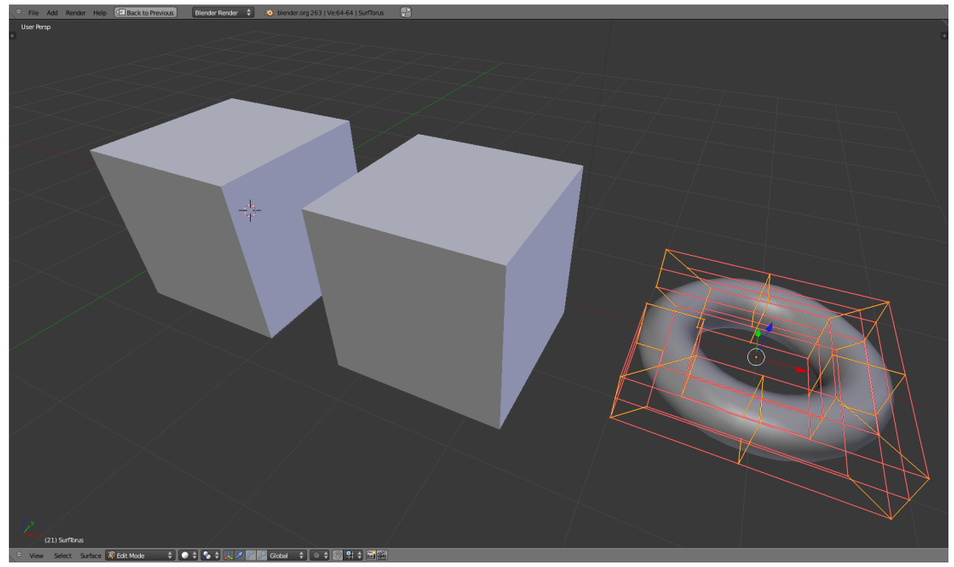




- Surfaces paramétriques

Particules





+ structures volumiques, ...

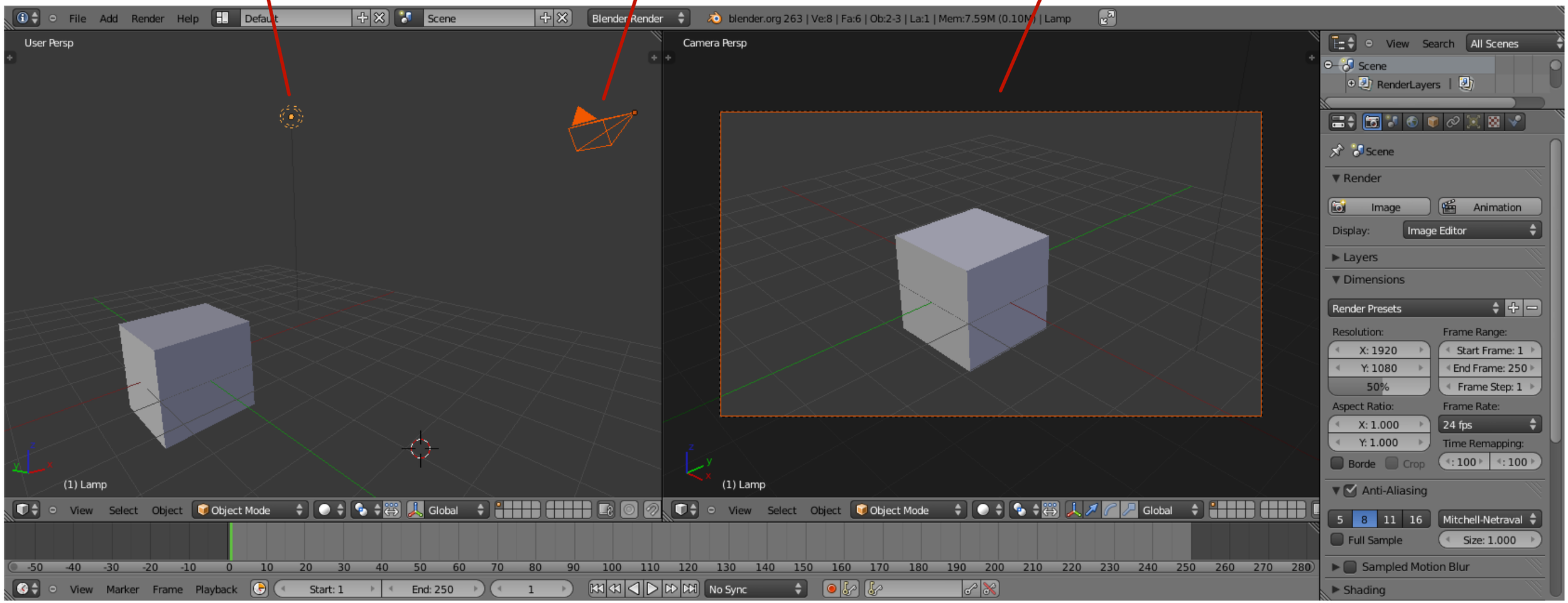
Rendu

Principe général

source(s) de lumière

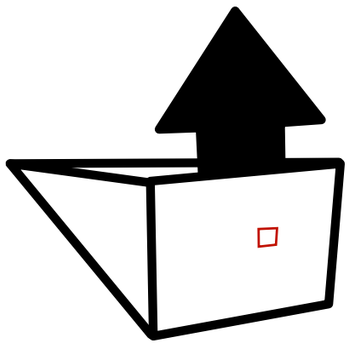
caméra

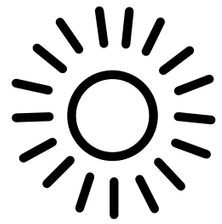
vue caméra

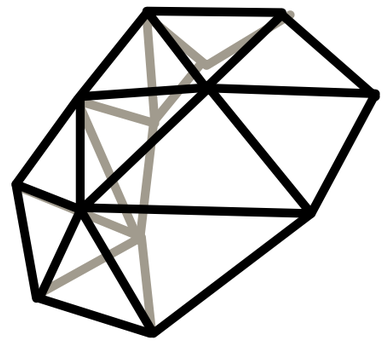


Rendu

Lancer de rayon



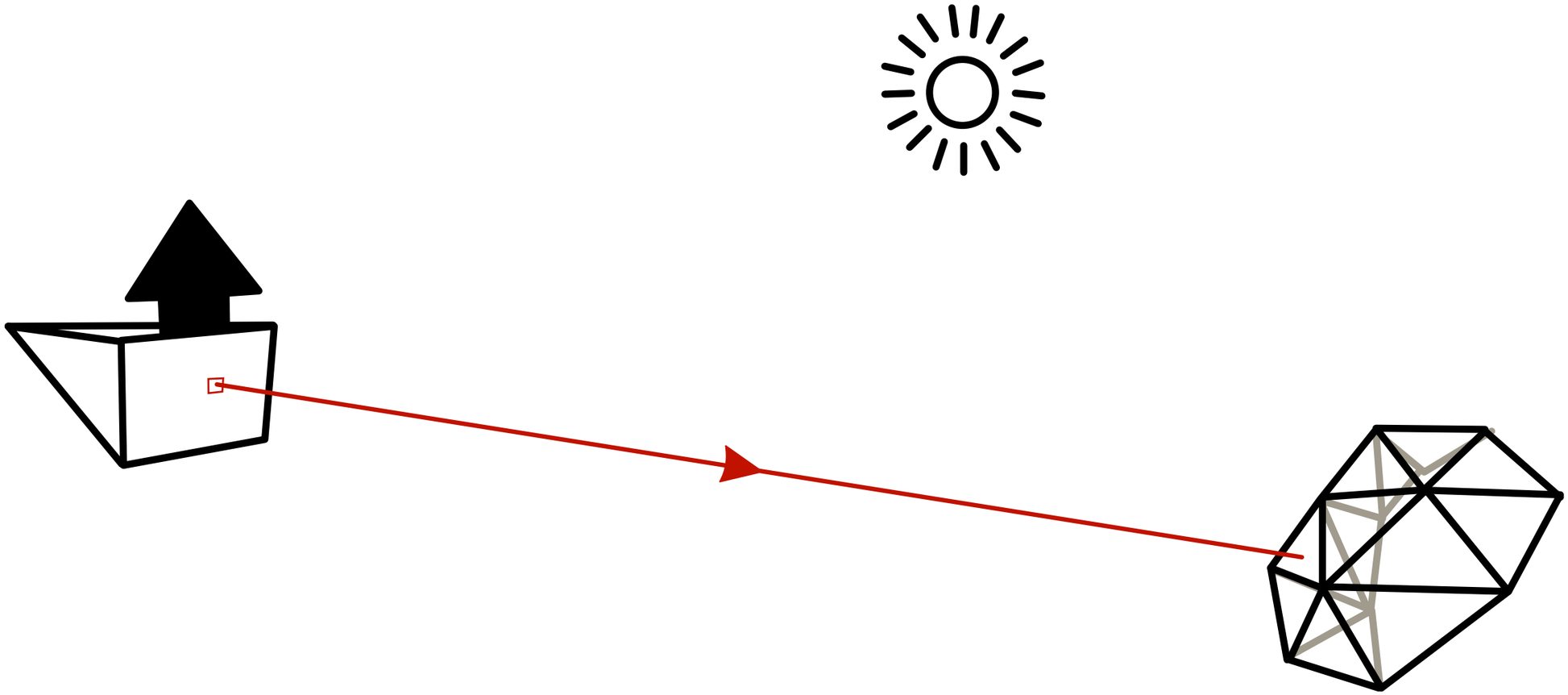




Pour chaque pixel de l'image résultat

Rendu

Lancer de rayon

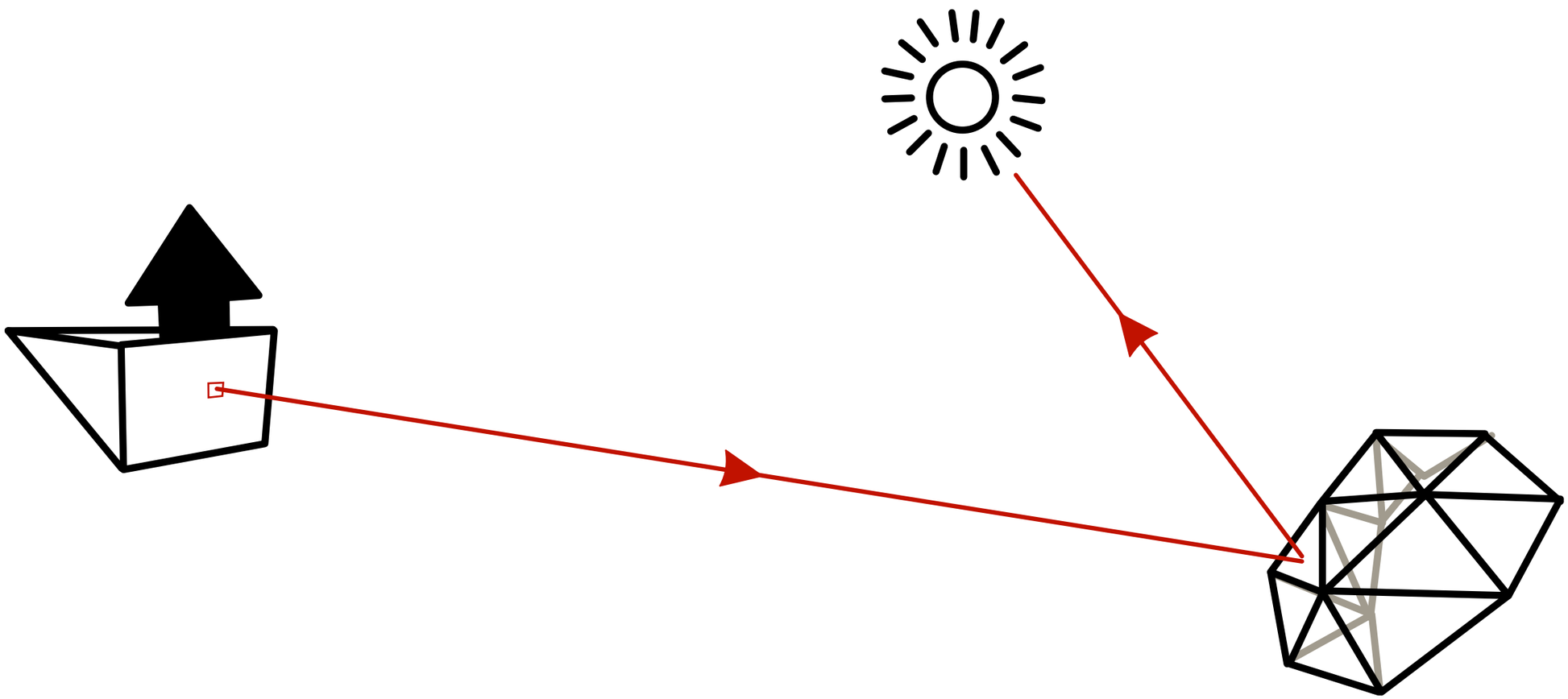


Pour chaque pixel de l'image résultat

- **lancer d'un rayon** et calcul d'**intersection**

Rendu

Lancer de rayon



Pour chaque pixel de l'image résultat

- **lancer d'un rayon** et calcul d'**intersection**
- estimation de l'éclairage par **rayon vers les sources**

Rendu

Radiosité (ou radiance)

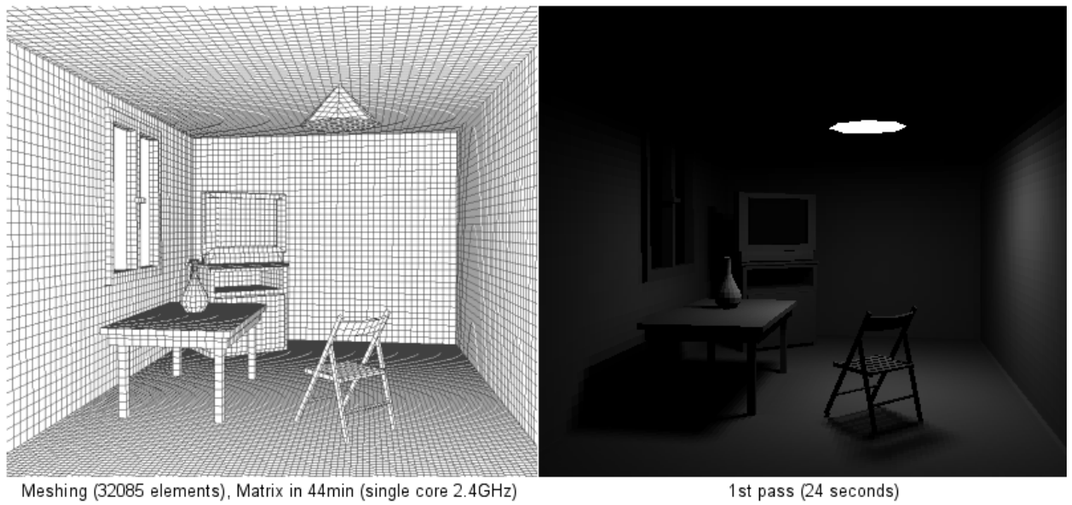
Principe : chaque objet émet de la lumière, directement ou indirectement

Calcul itératif :

Lancer d'un rayon depuis la source et calcul d'une trajectoire avec rebonds

Fonctionnement :

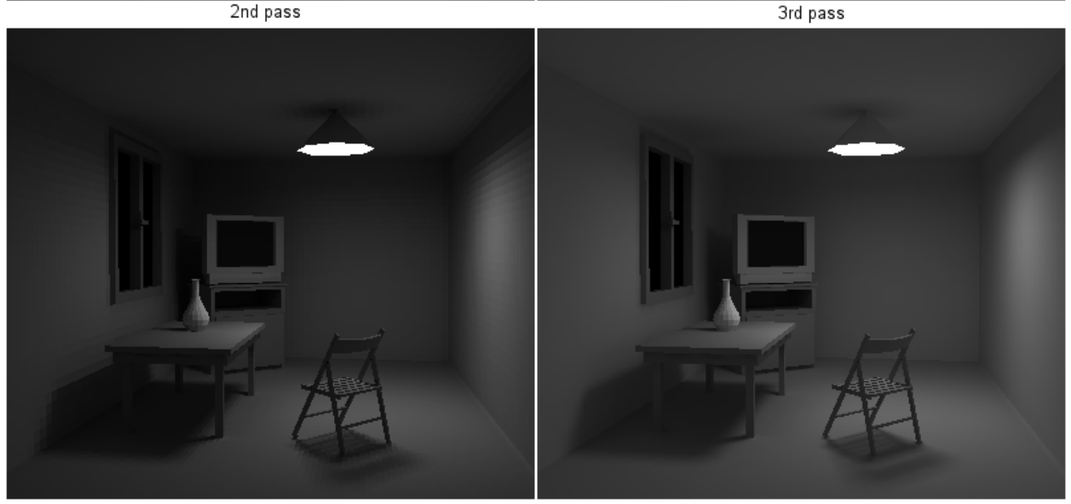
- Tirage **statistique** suivant distribution de la direction
- **Subdivision des tessels** au besoin


Meshing (32085 elements), Matrix in 44min (single core 2.4GHz)

1st pass (24 seconds)


2nd pass

3rd pass

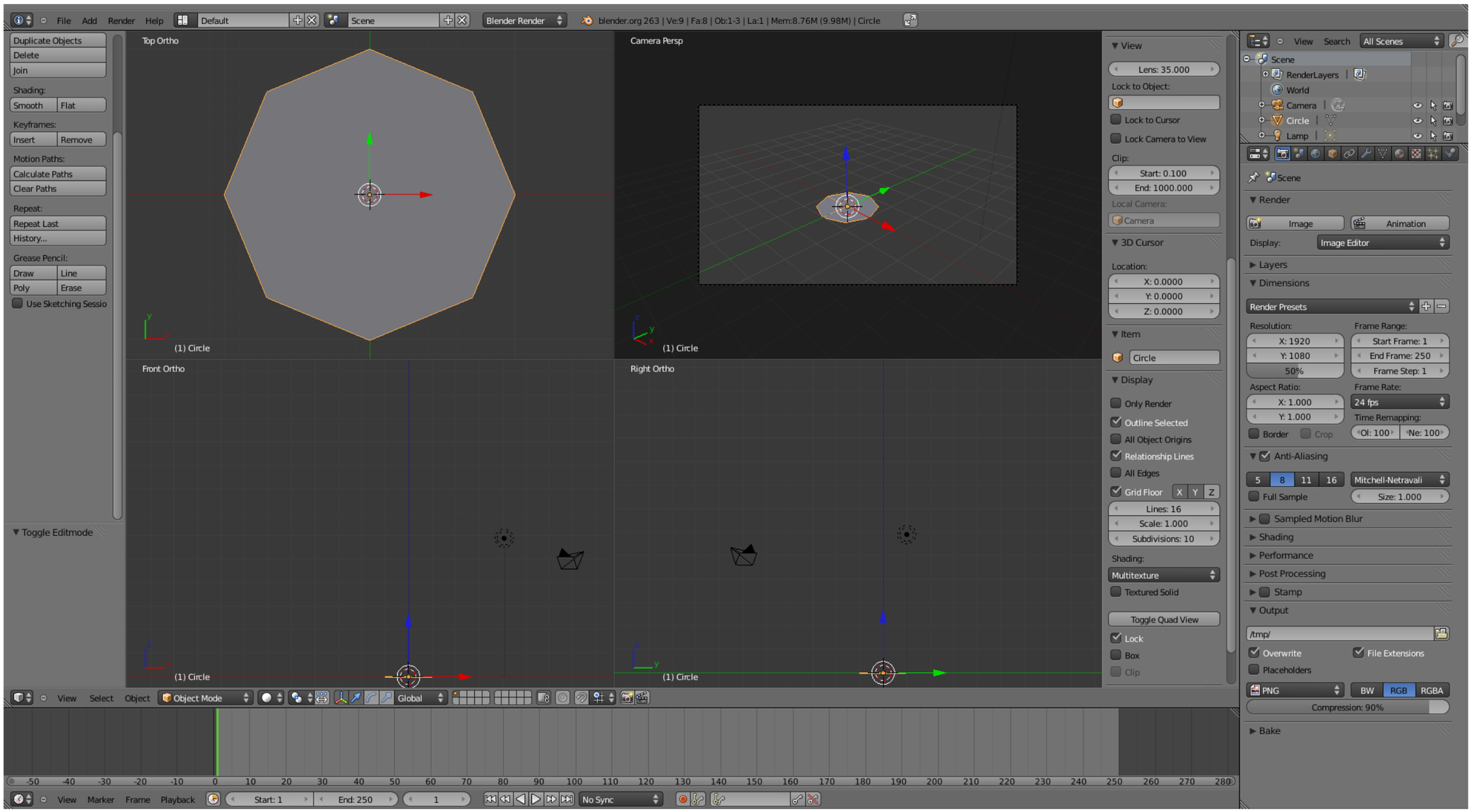

4th pass

20th pass (with bilinear filtering)

Modélisation

Principes de base

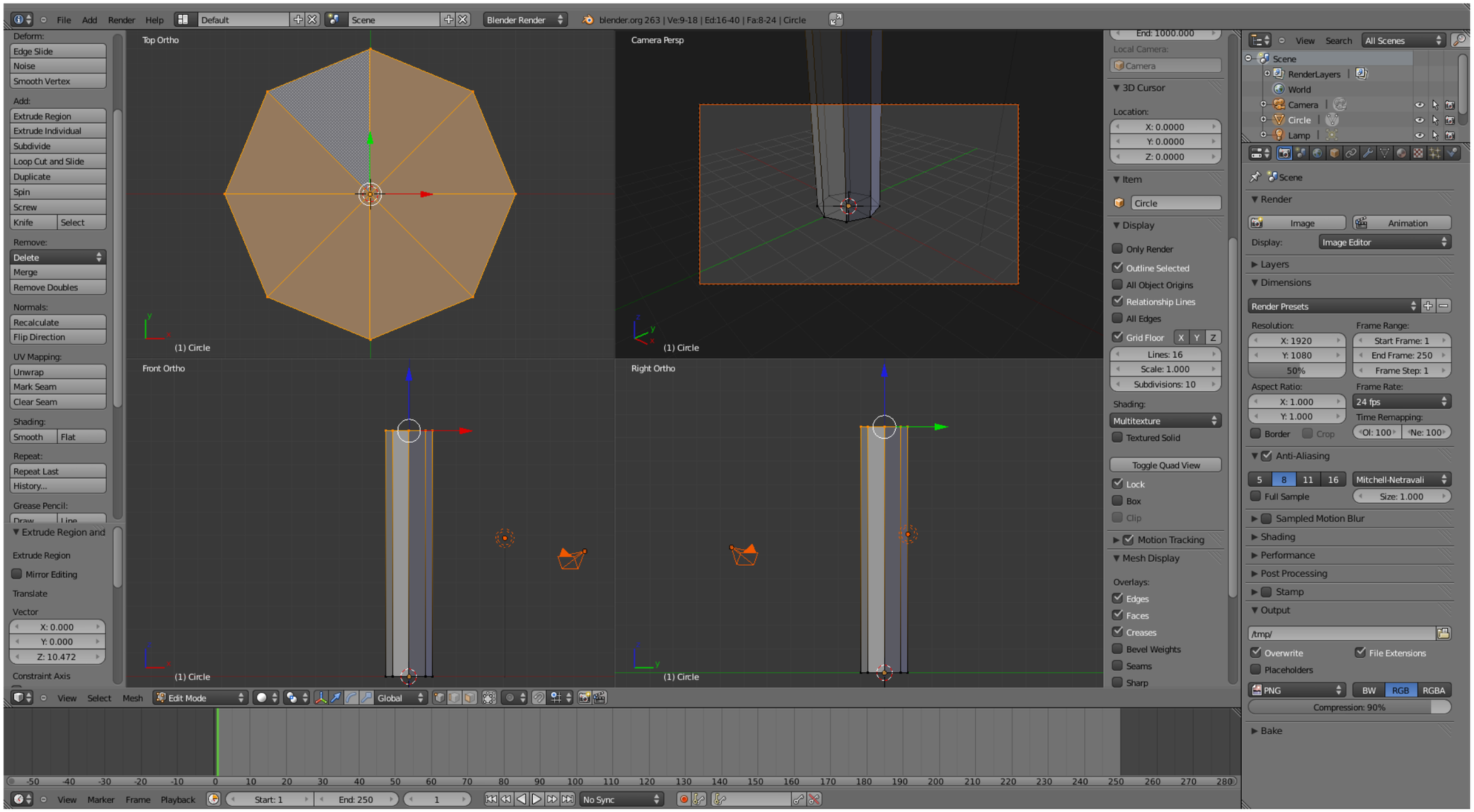
- Partir d'une **forme de base**
- **Extruder** pour former la silhouette globale
- **Raffiner** successivement et **modeler**



Modélisation

Principes de base

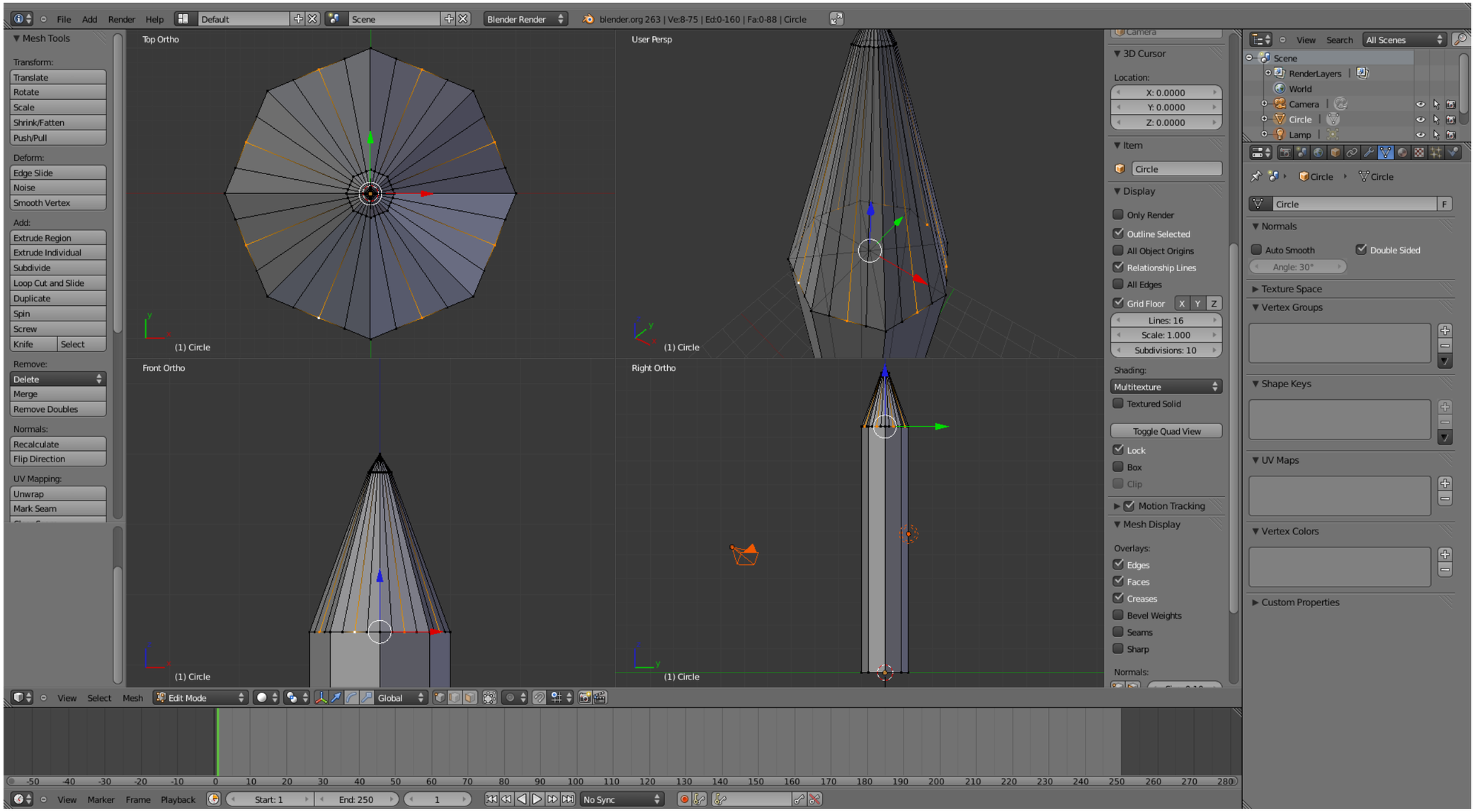
- Partir d'une **forme de base**
- **Extruder** pour former la silhouette globale
- **Raffiner** successivement et **modeler**



Modélisation

Principes de base

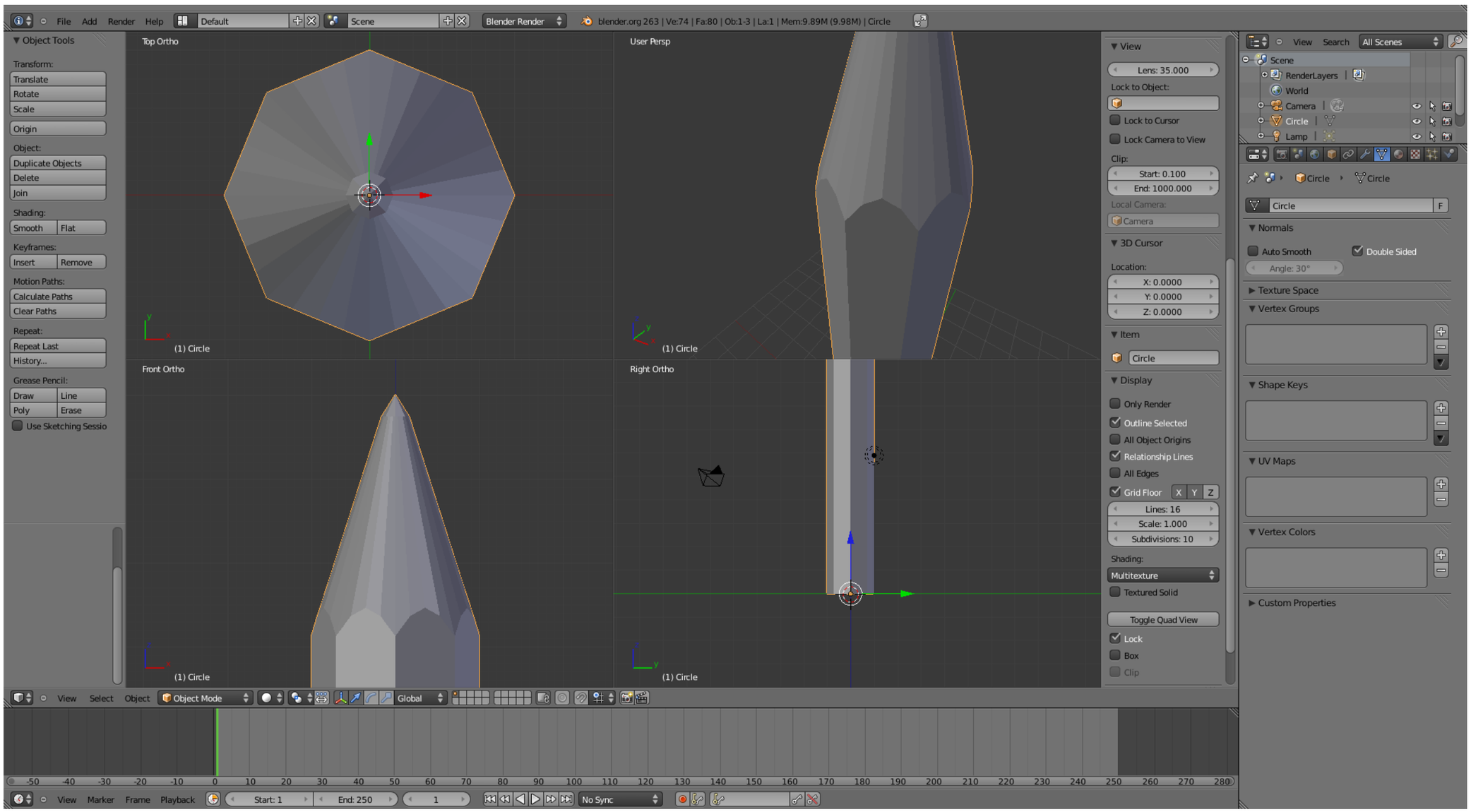
- Partir d'une **forme de base**
- **Extruder** pour former la silhouette globale
- **Raffiner** successivement et **modeler**



Modélisation

Principes de base

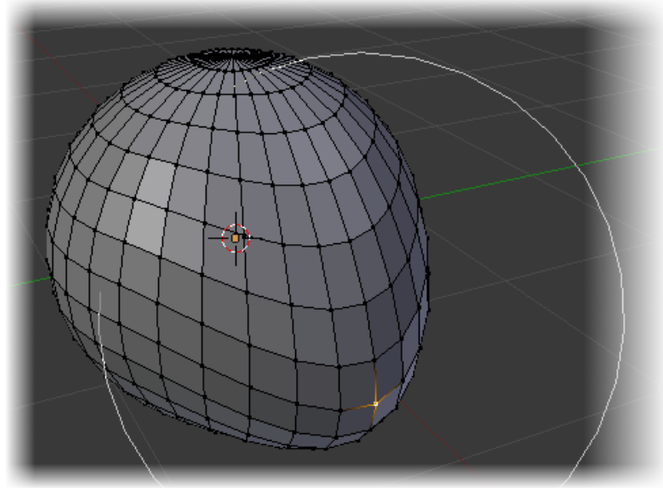
- Partir d'une **forme de base**
- **Extruder** pour former la silhouette globale
- **Raffiner** successivement et **modeler**



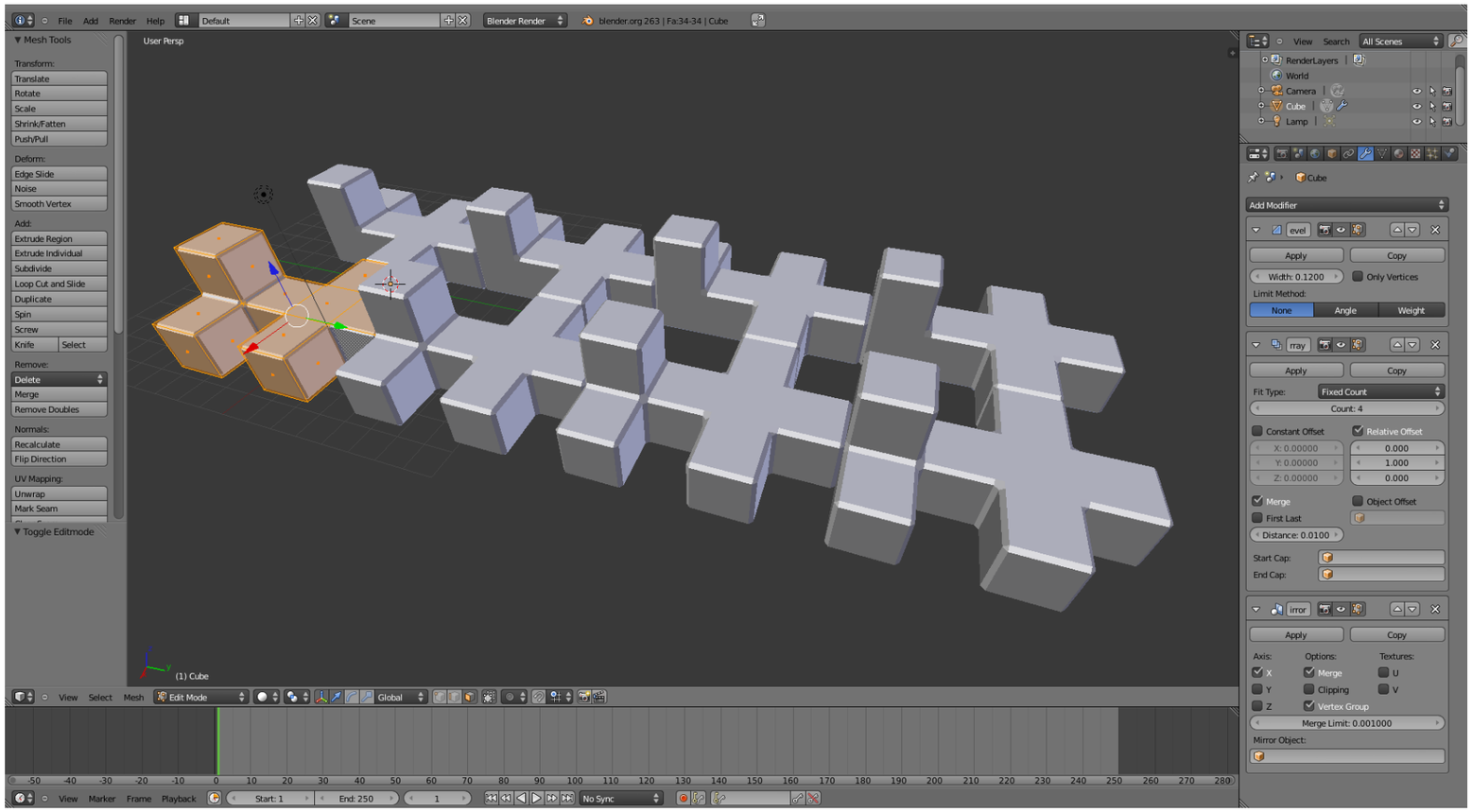
Modélisation

Outils d'automatisation

Utiliser les outils de sélection et de déformation



Utiliser les outils de répétition, de symétrie, de rotation...



Modélisation

Quelques principes

- Éviter les détails inutiles
- Chaque triangle doit être utile
- Penser à l'aspect animation

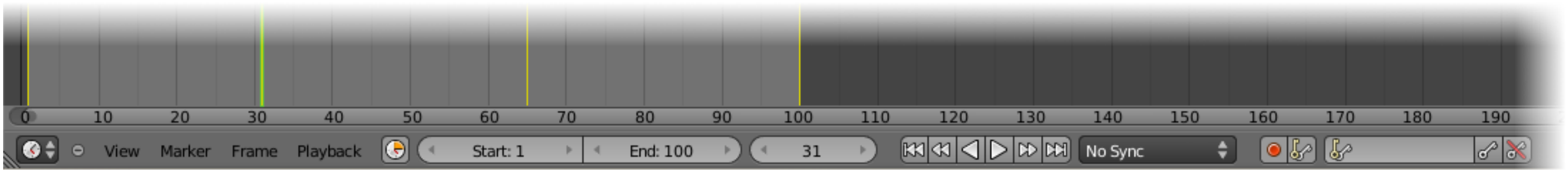
Animation

Principes élémentaires

Animation : 25 images par seconde

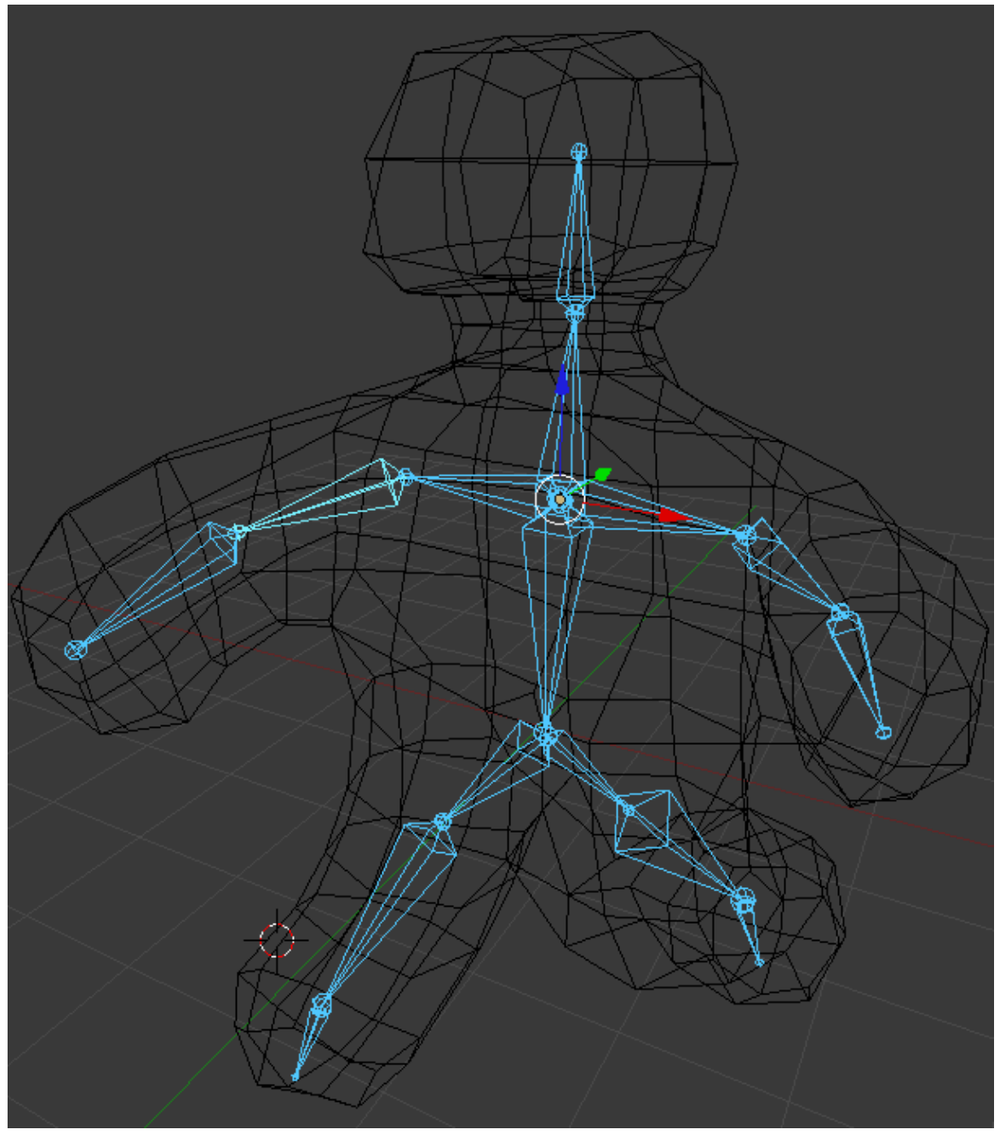
- Décrire la position et la forme des objets **à chaque frame**
- Possibilité d'**interpolation**
- Possibilité d'**automatisation**

Utilisation d'une *timeline* pour décrire les contraintes



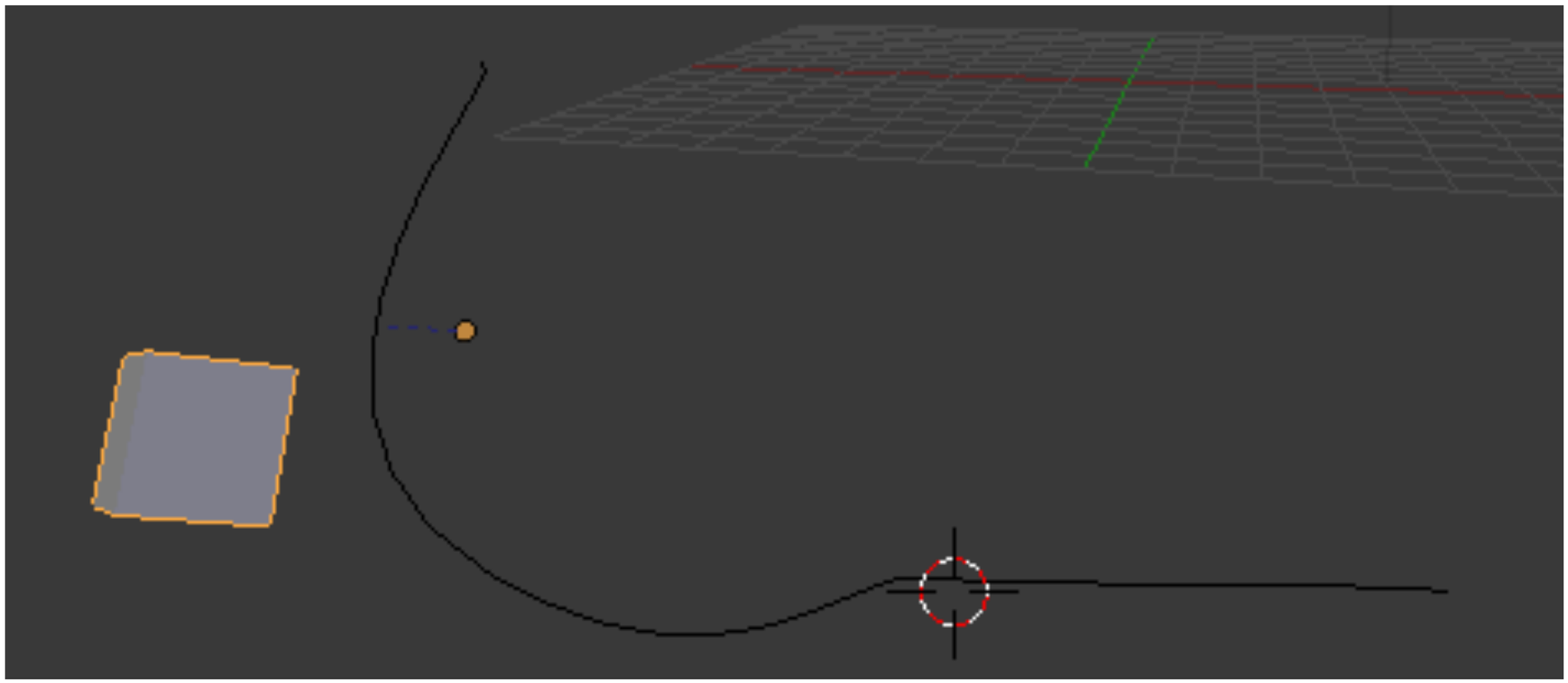
Armatures de déformation

- Groupe de sommets associés à chaque armature
- **Ajustement de la position** de chaque armature pour les poses clé
- Ajouts de **clés** (rotation, translation) à la *timeline*



Déplacement d'objet par courbe

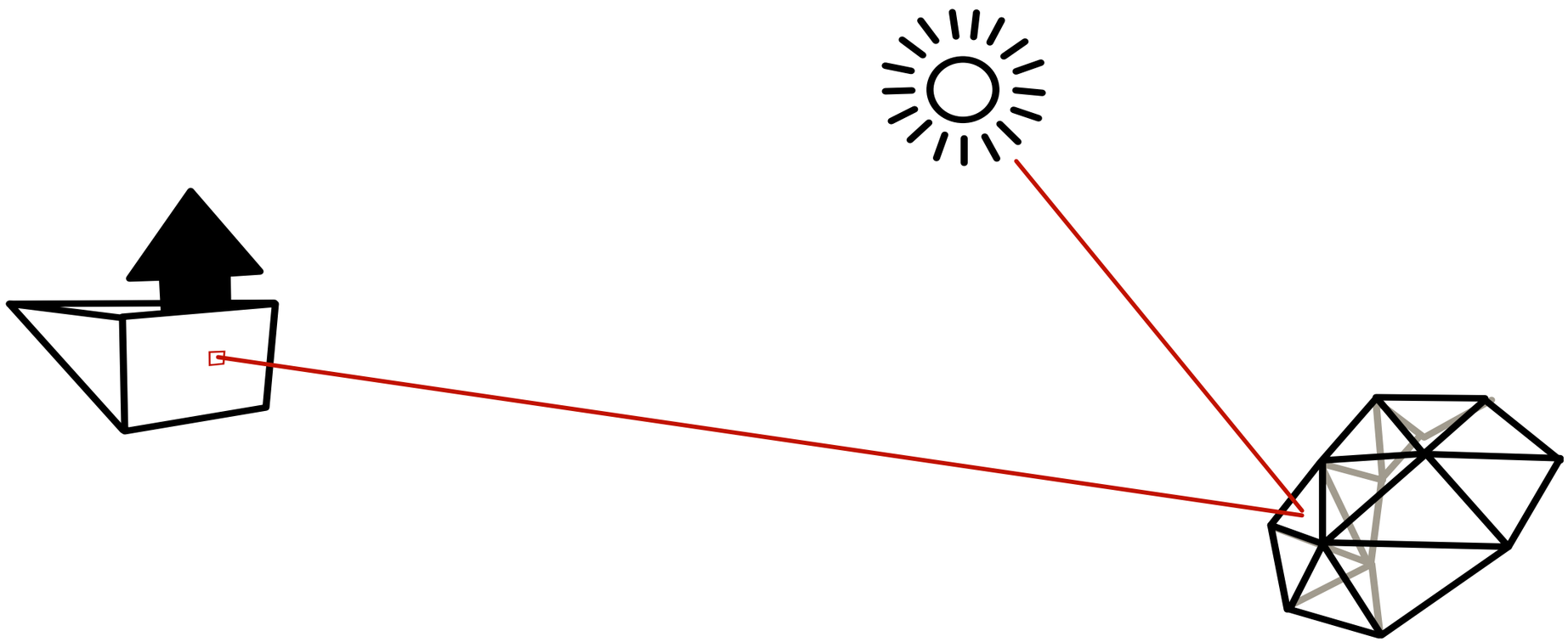
- Description de **trajectoires** par courbes type Bézier
- Ajouts de **clés** de position à la *timeline*



Texture

Principes élémentaires

Décrire le « rebond » de la lumière sur la surface

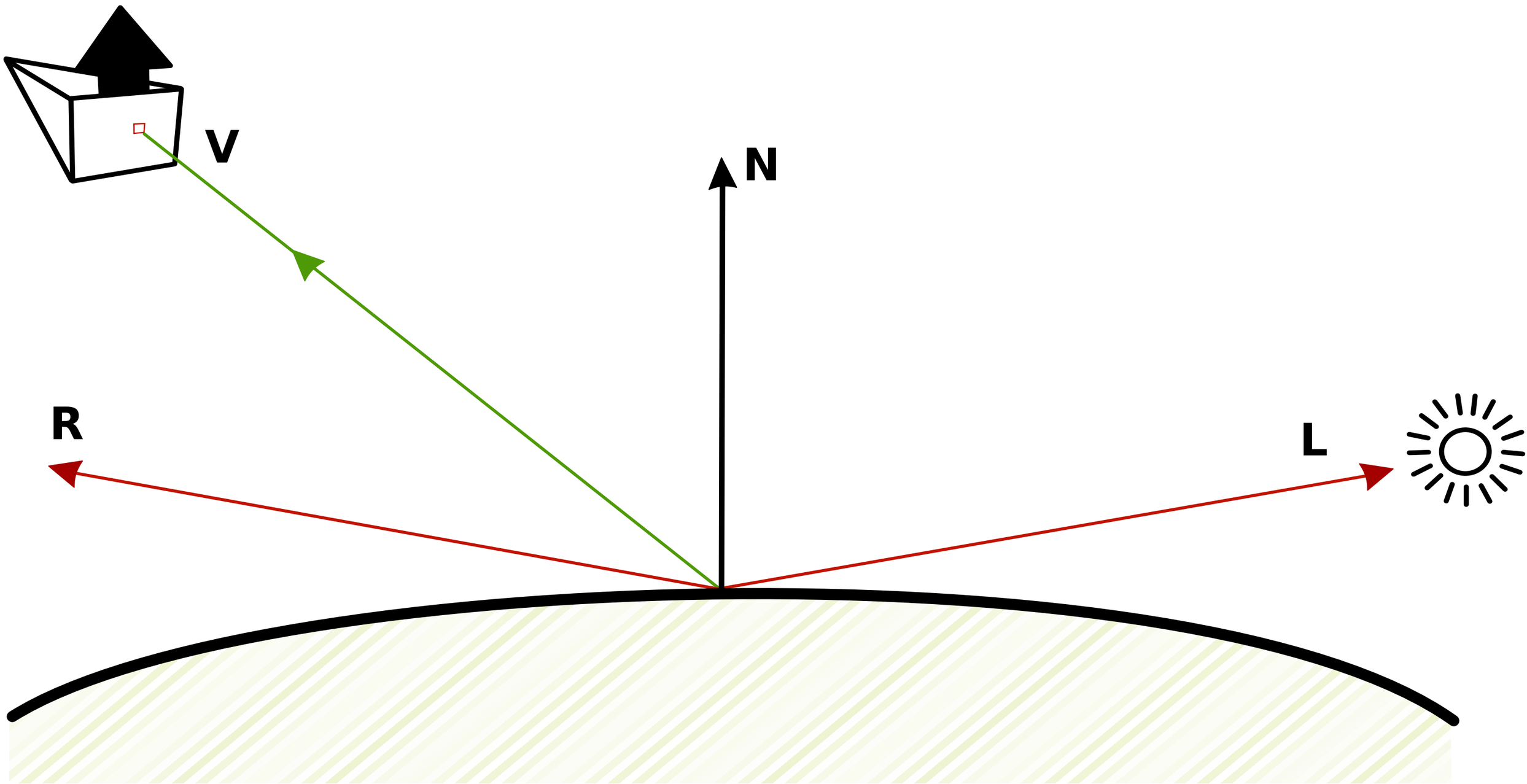


Associer à **chaque triangle** une **apparence**

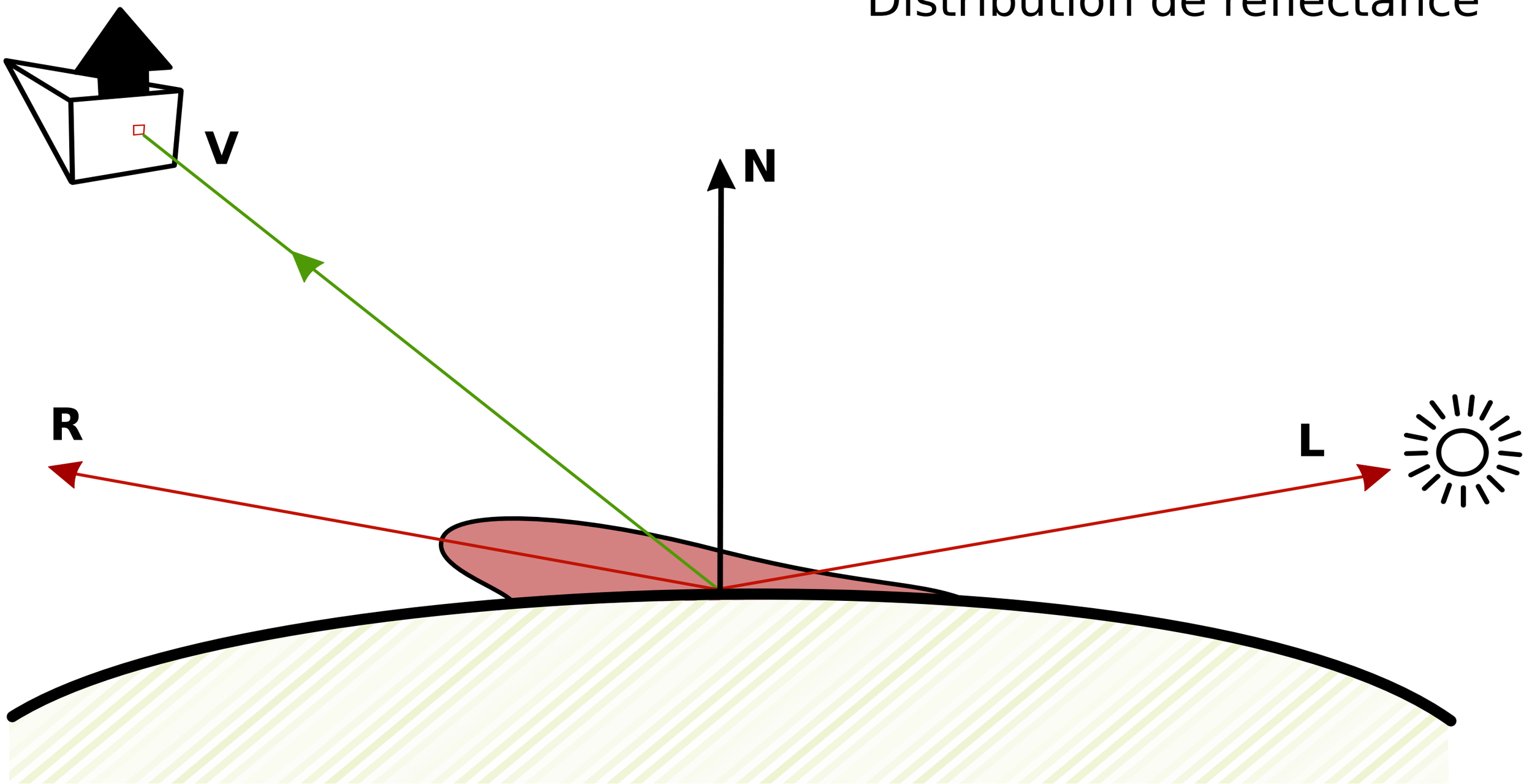
- une couleur
- un modèle de réflectance
- des motifs

Texture

Principes de l'illumination

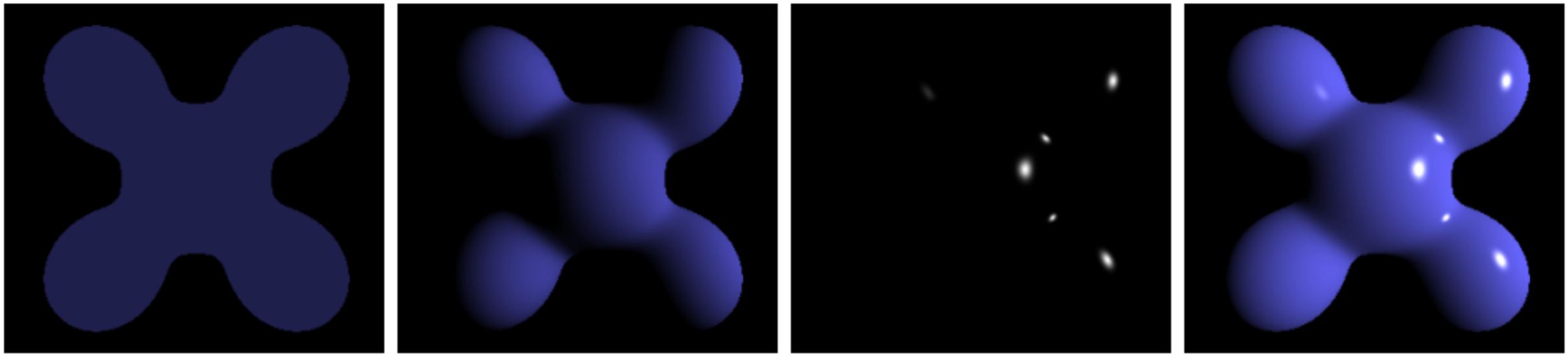


Distribution de réflectance



Texture

Modèle de Phong (modèle empirique d'illumination)



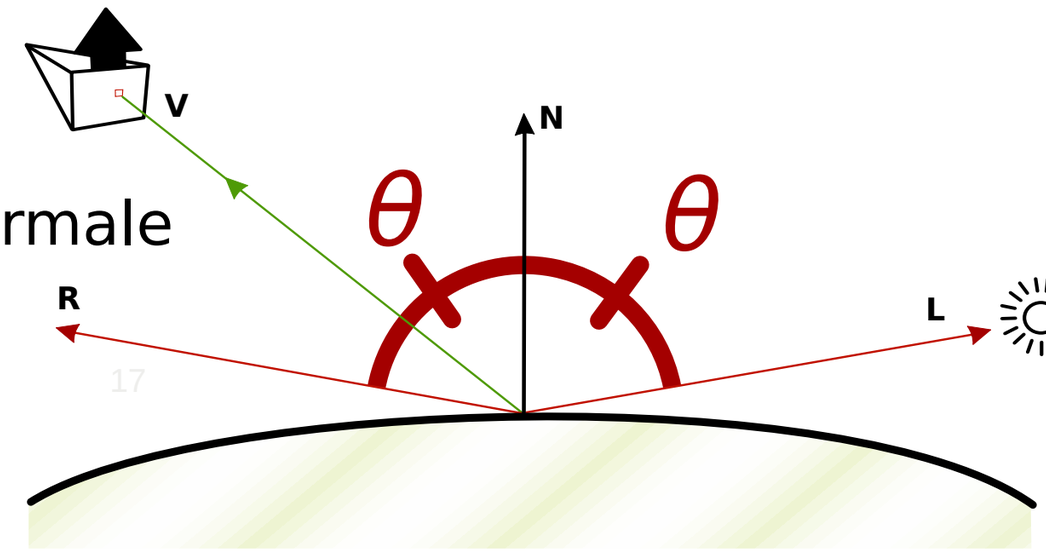
Ambient + Diffuse + Specular = Phong Reflection

Composante ambiante :

- **Ambiant :** lumières parasites (uniforme)

Composante incidente :

- **Diffuse :** lumière ne dépendant que de la normale
- **Specular :** effet miroir



$$I = i_a k_a + i_d k_d \cos\theta + i_s k_s \cos^{\alpha}\theta$$

$k_{\cdot} \in [0; 1]$: coefficients de la texture

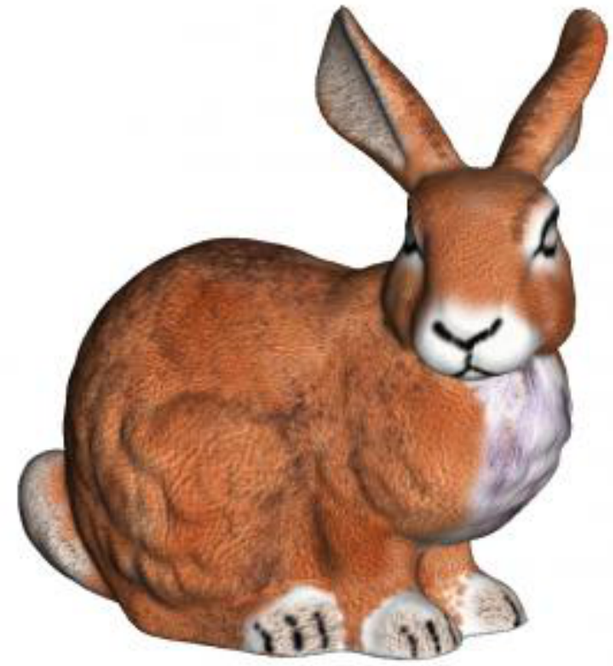
$\alpha >> 1$: constante de *brillance*

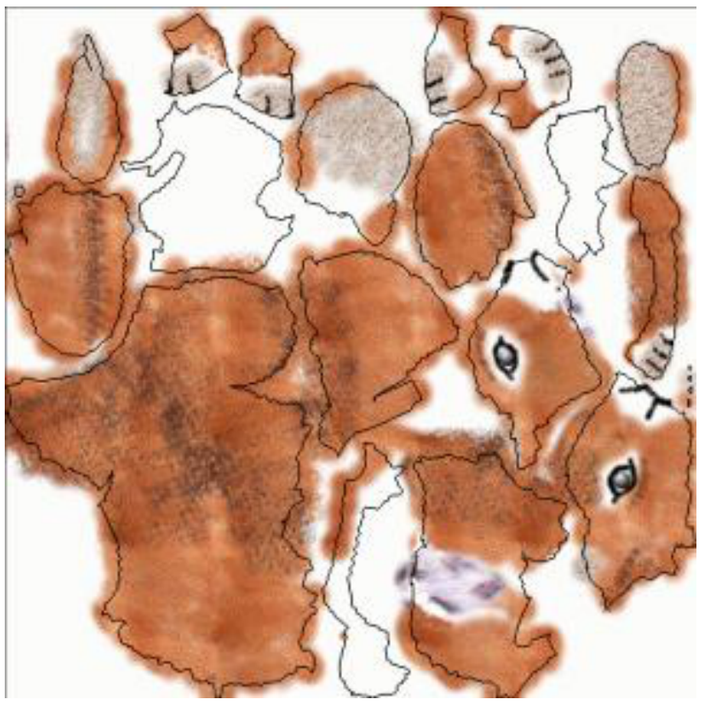
$i_{\cdot}$: couleur de la texture

Texture

Textures par l'image

Utilité de la paramétrisation

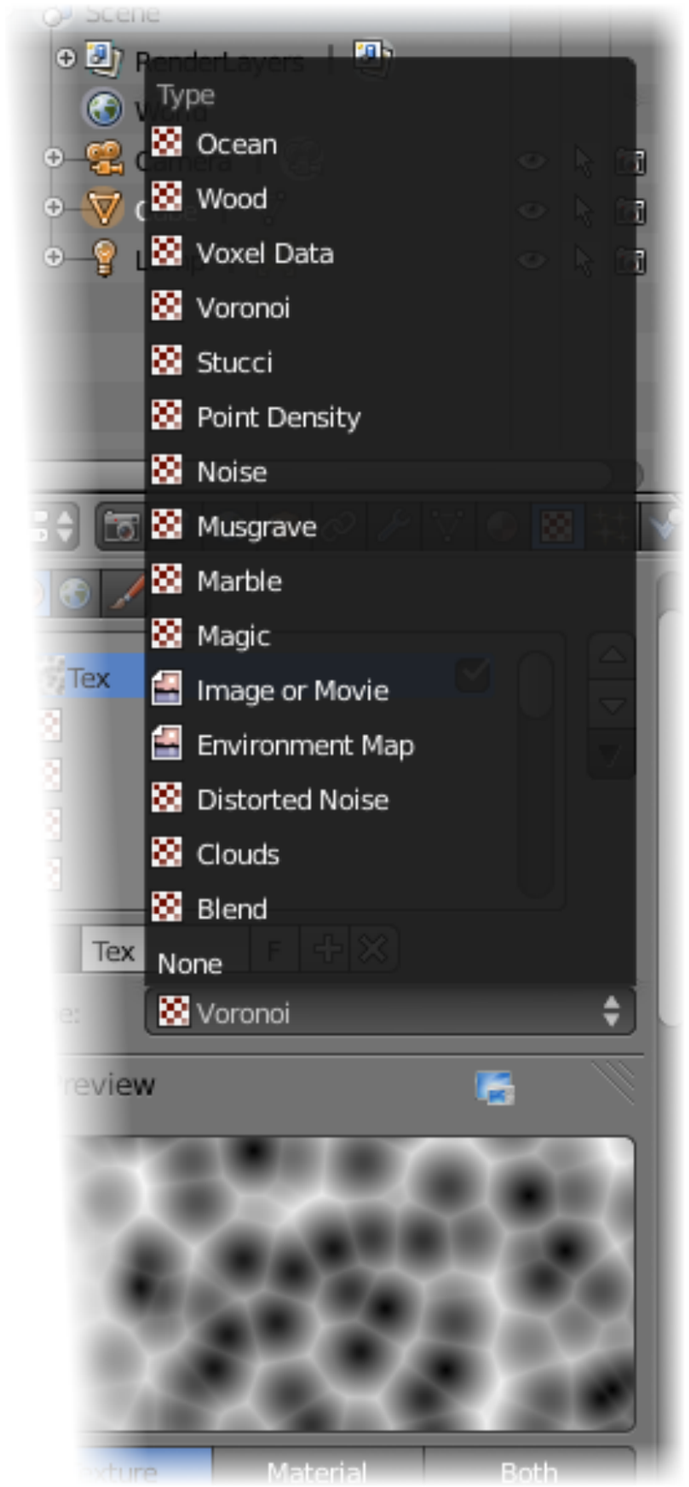




Texture

Génération automatique

Générer des **textures logicielles**
ou *textures procédurales*



Intérêts :

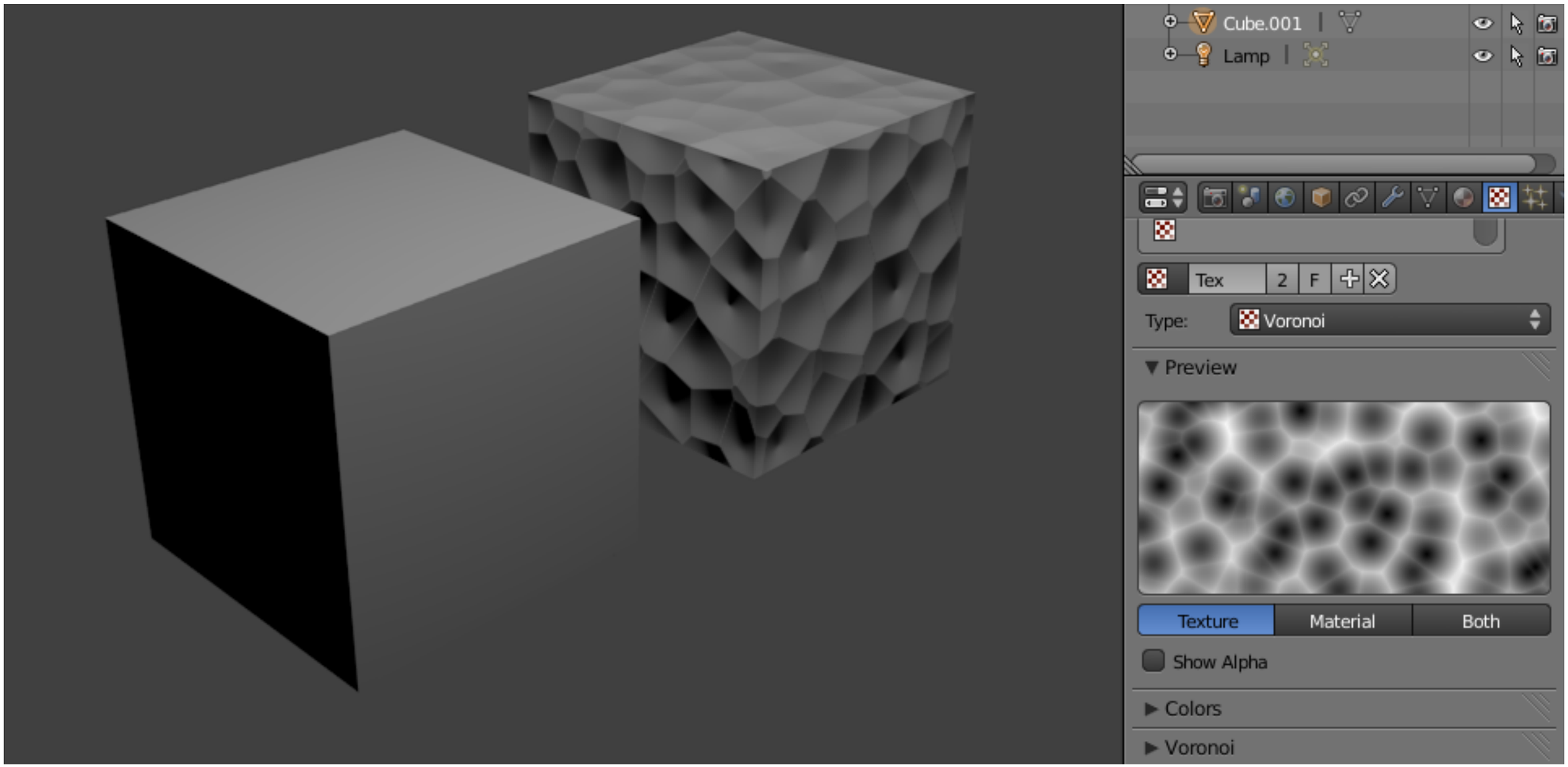
- gain de place
- infinité de possibilités
- adaptation au contexte
- Multi-échelle

Exemple : allegorithmic

Matériau

Donner du volume

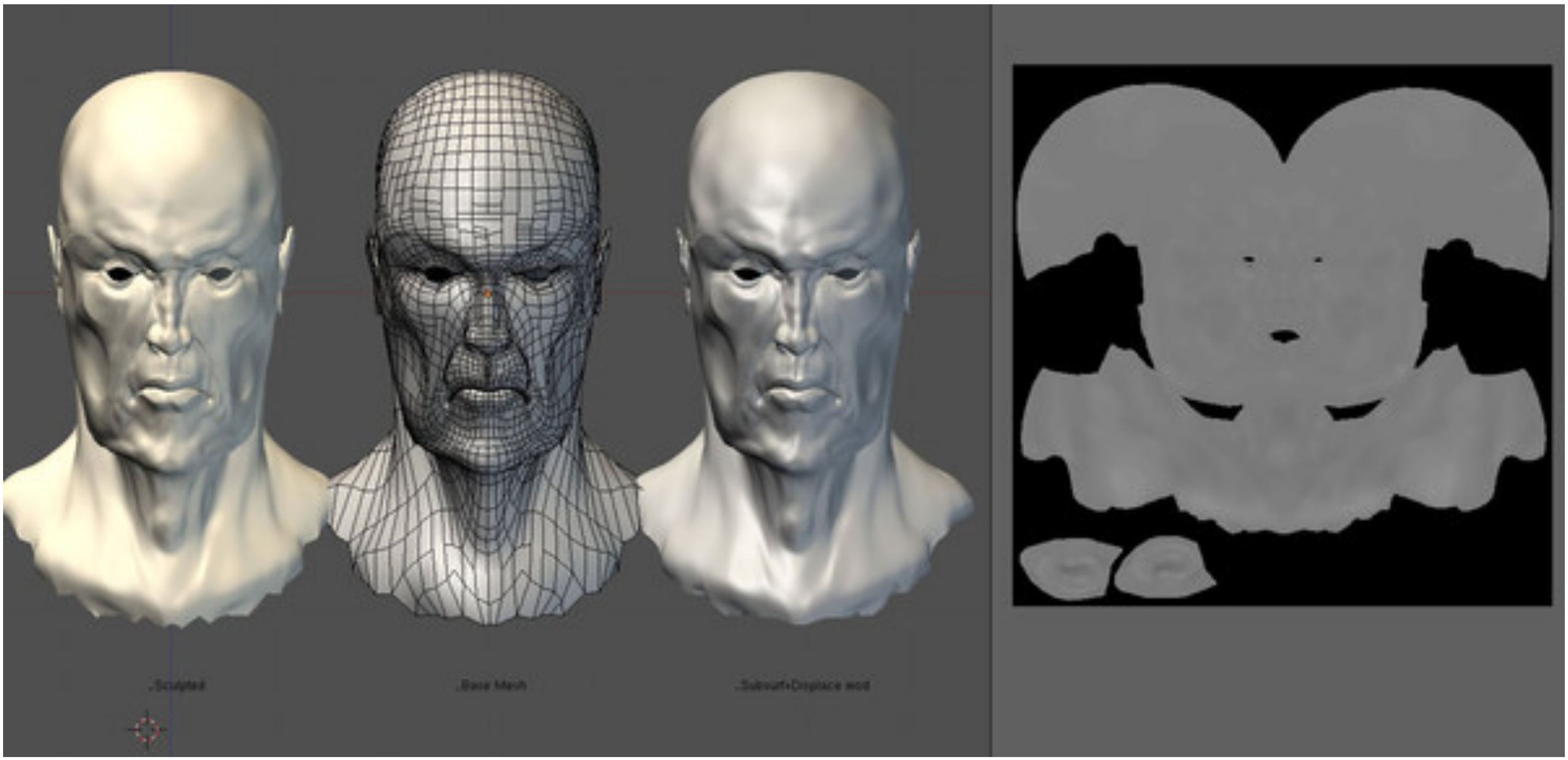
La texture décrit le relief (ou la normale)



Matériau

Donner du volume

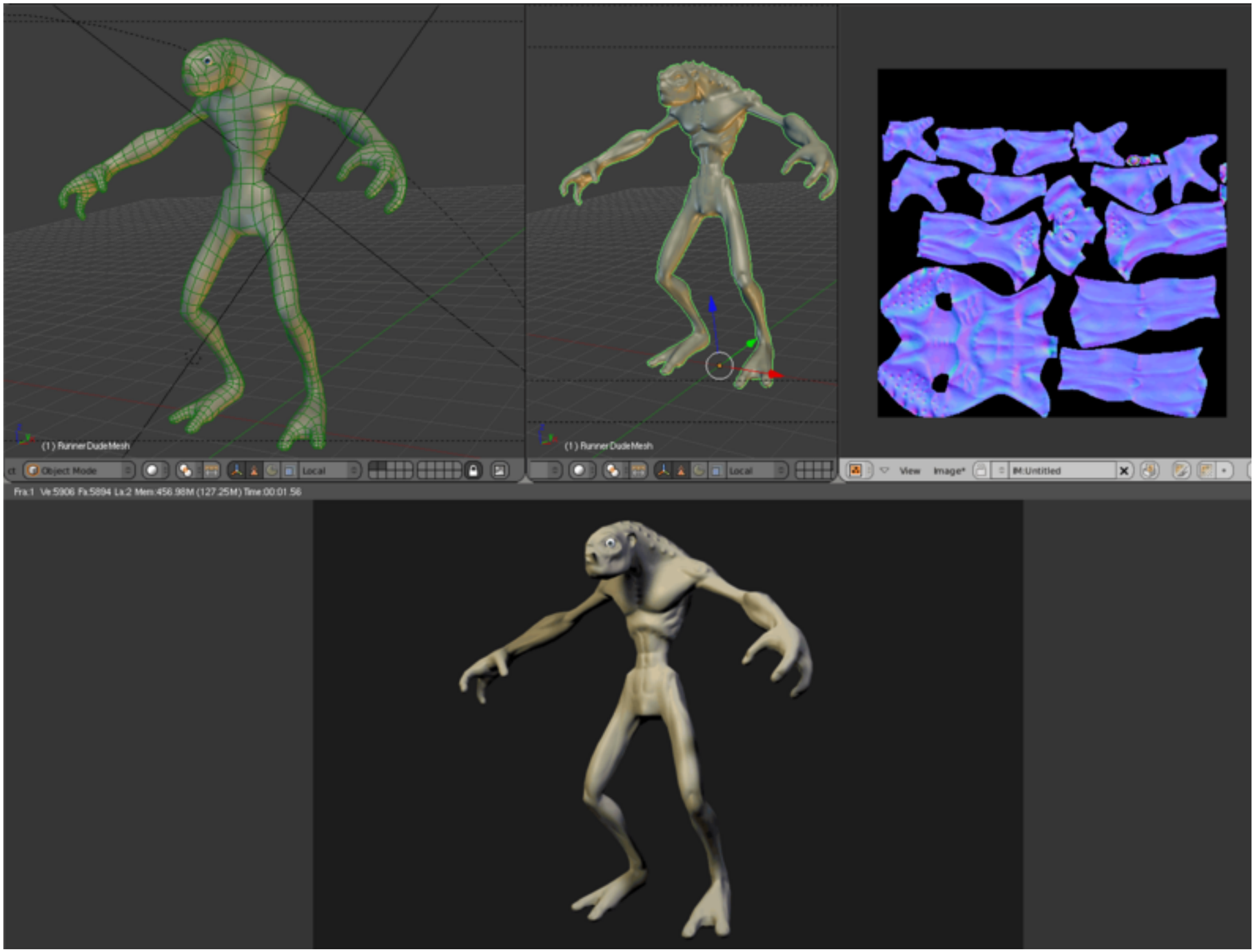
La texture décrit le relief (ou la normale)



Matériau

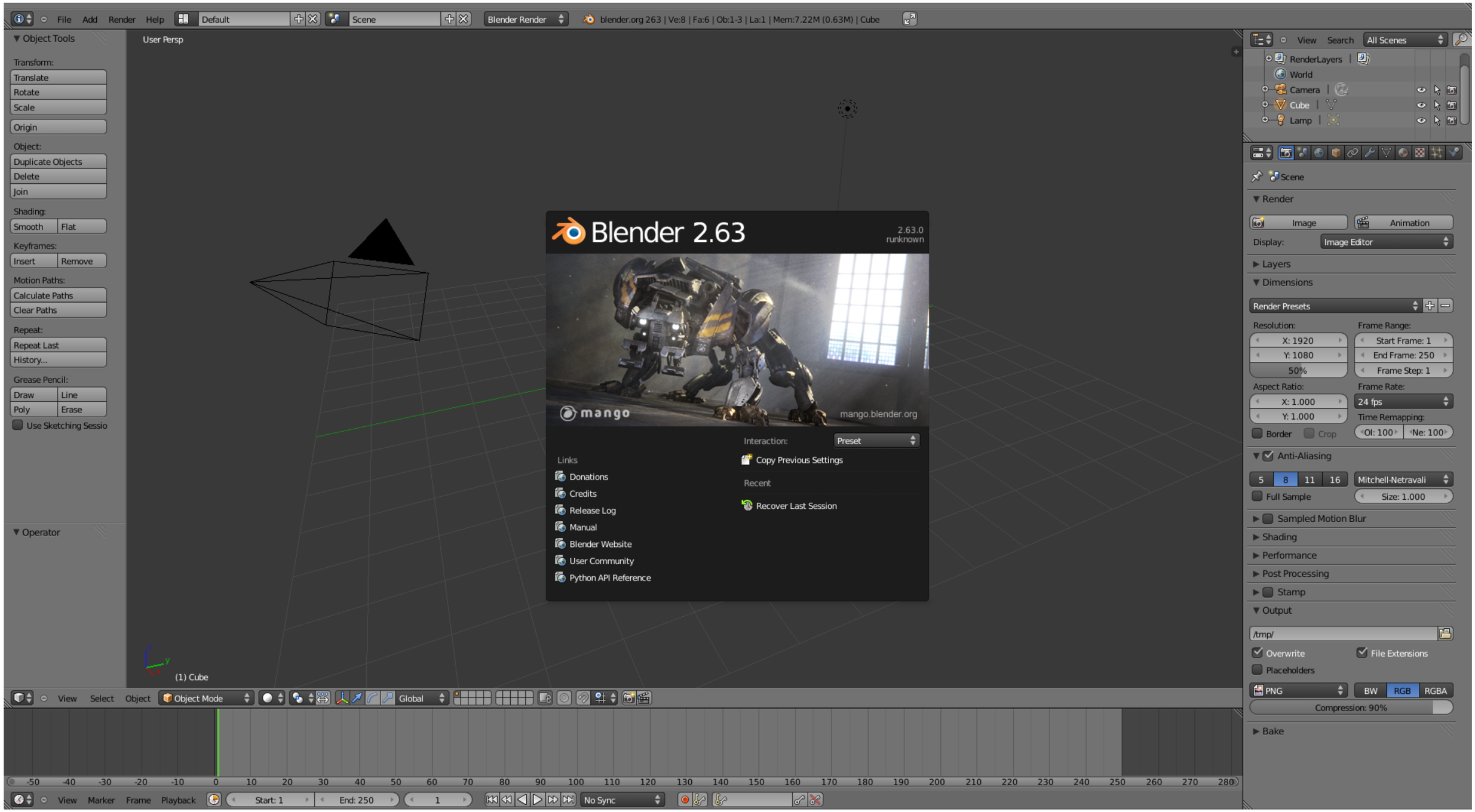
Donner du volume

La texture décrit le relief (ou la normale)



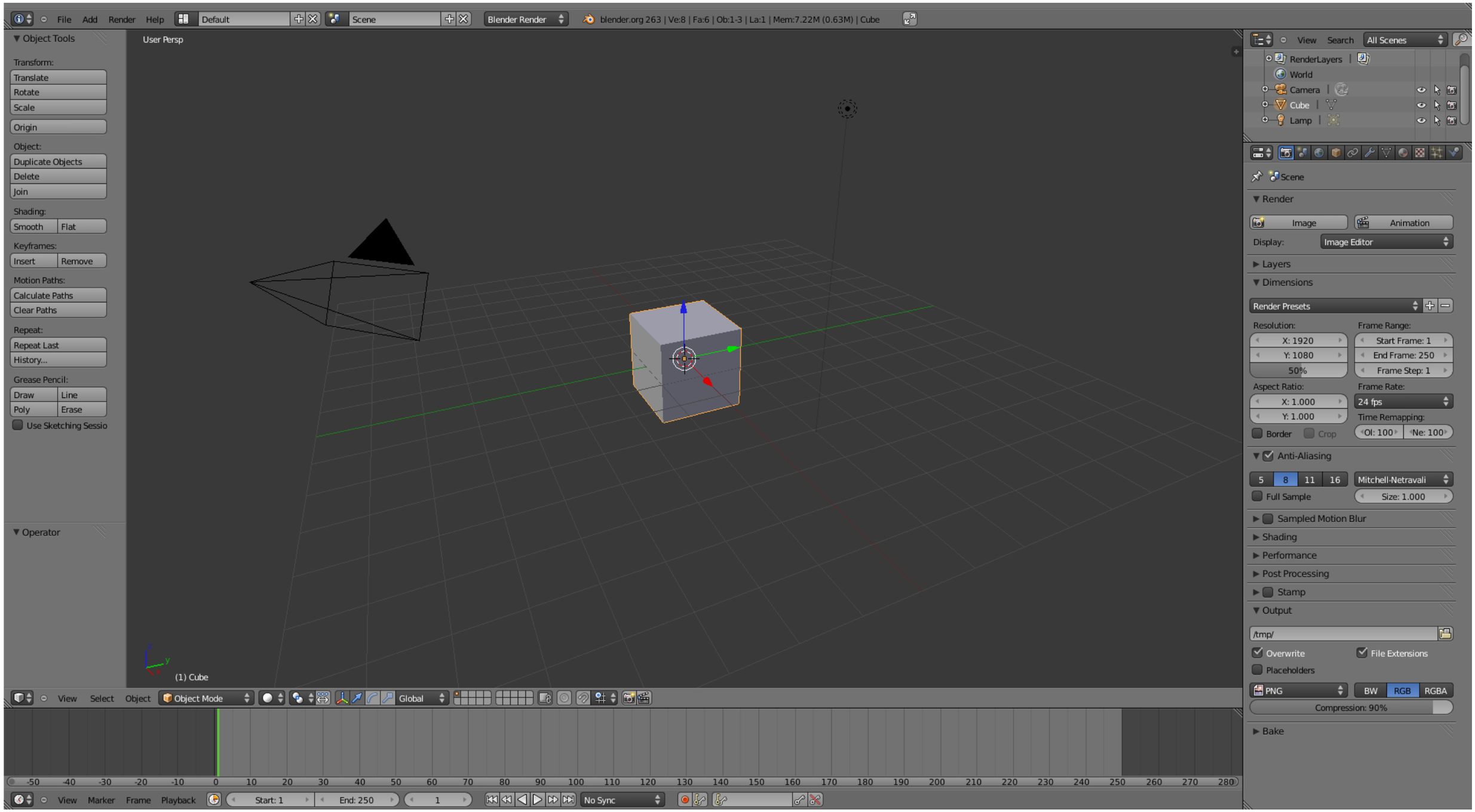
Blender

Découverte de l'interface



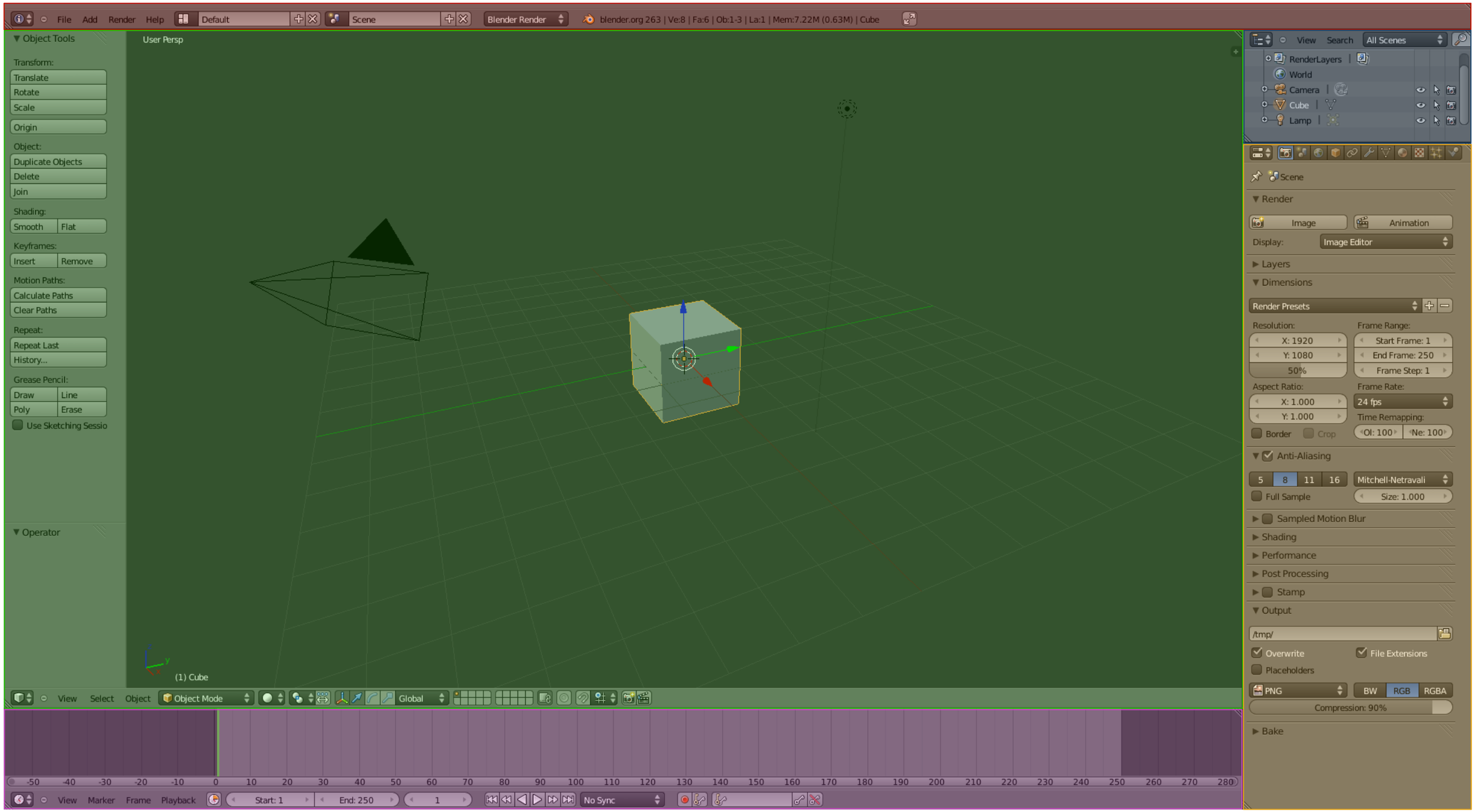
Blender

Découverte de l'interface



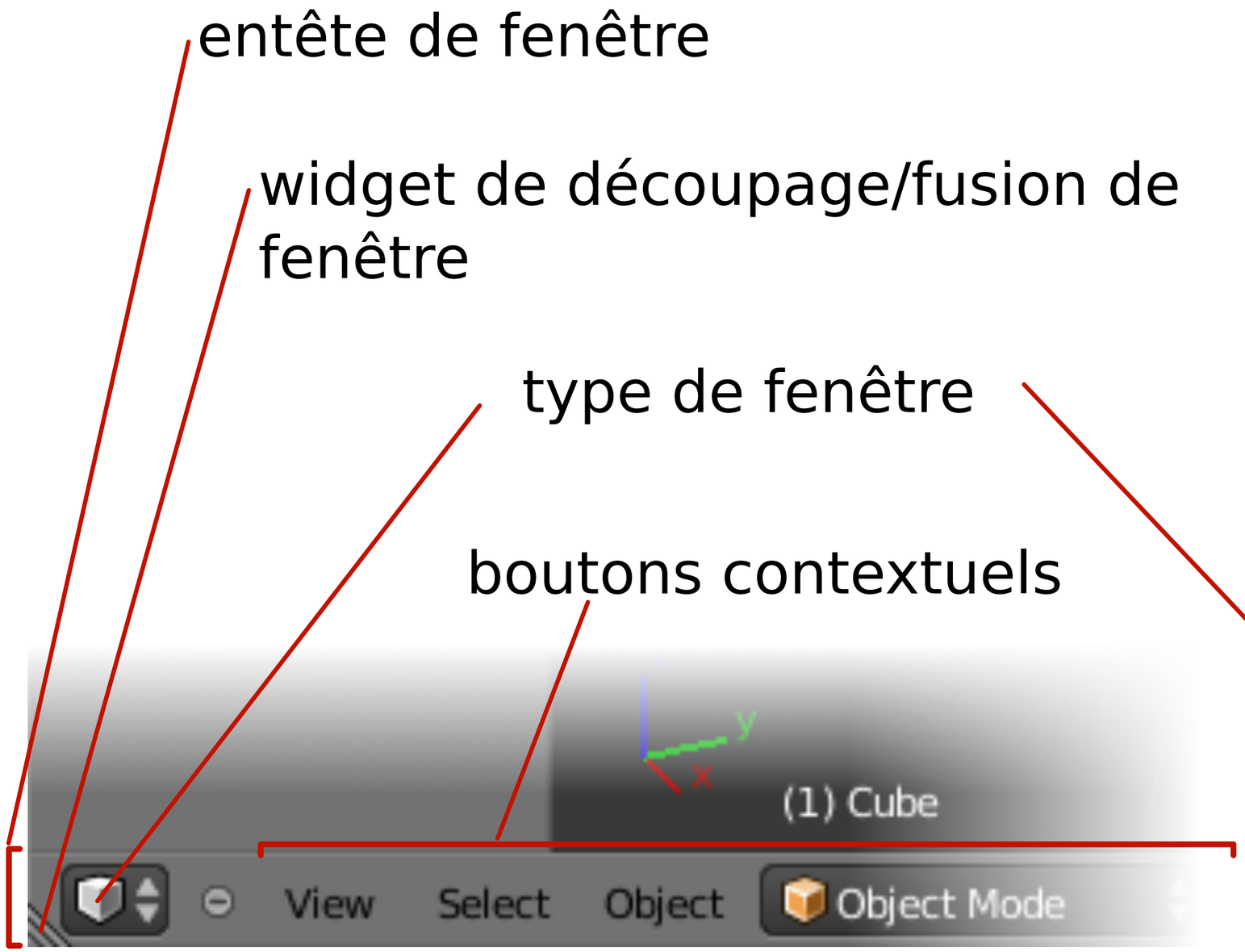
Blender

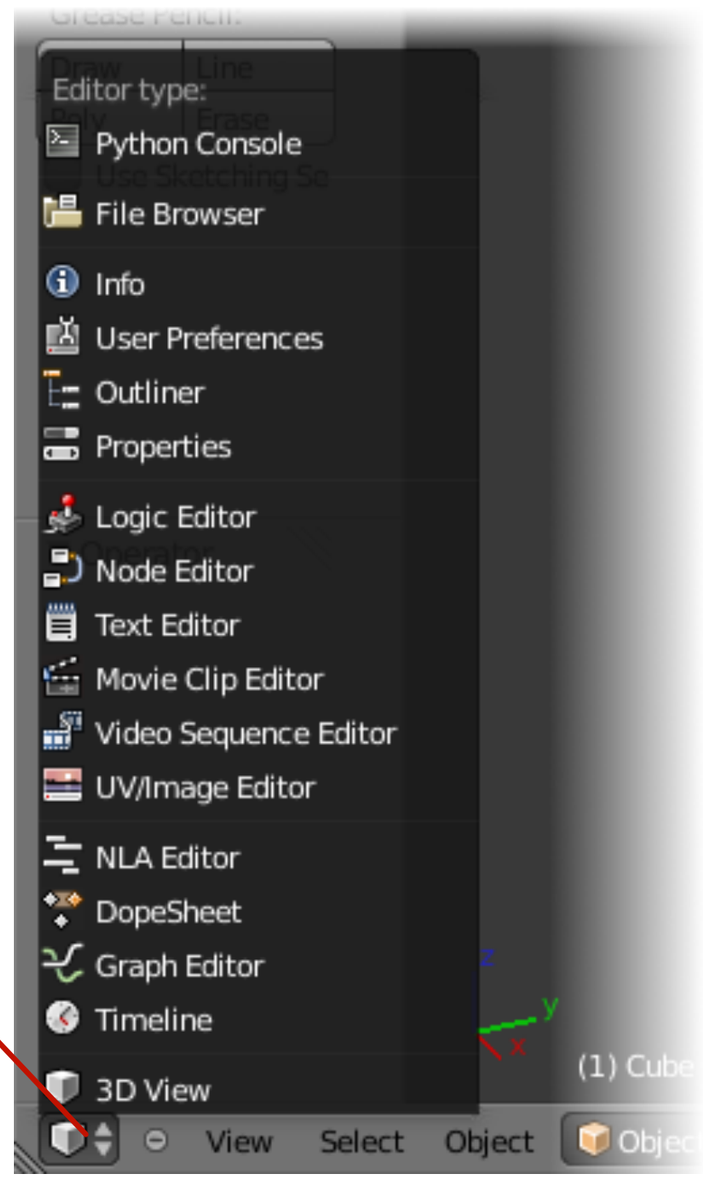
Découverte de l'interface

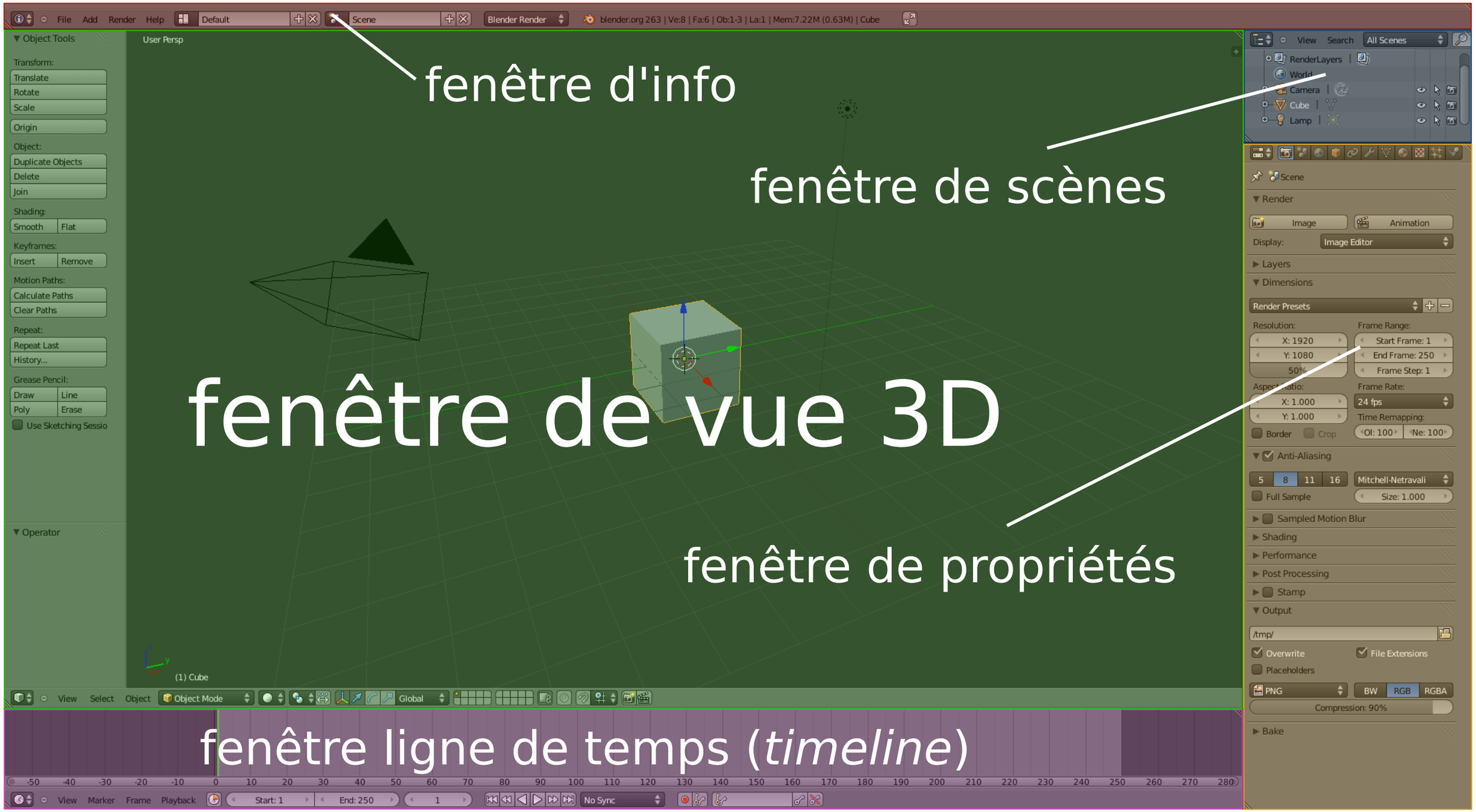


découpage en fenêtres

Détails d'une fenêtre







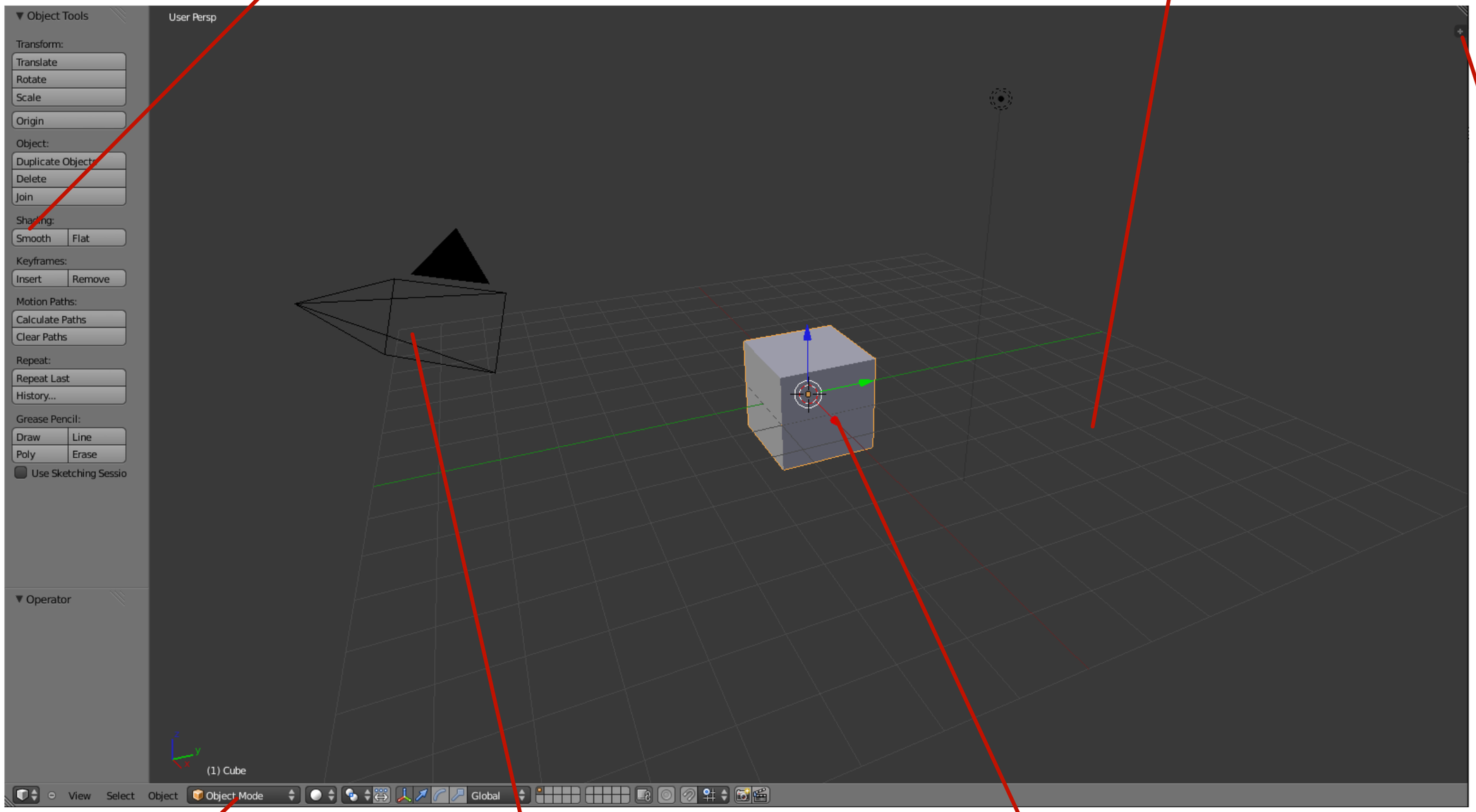
découpage en fenêtres

Blender

La fenêtre de vue 3D

Panneau latéral (outils)

Grille (plan de référence)



Bouton d'affichage du panneau latéral droit

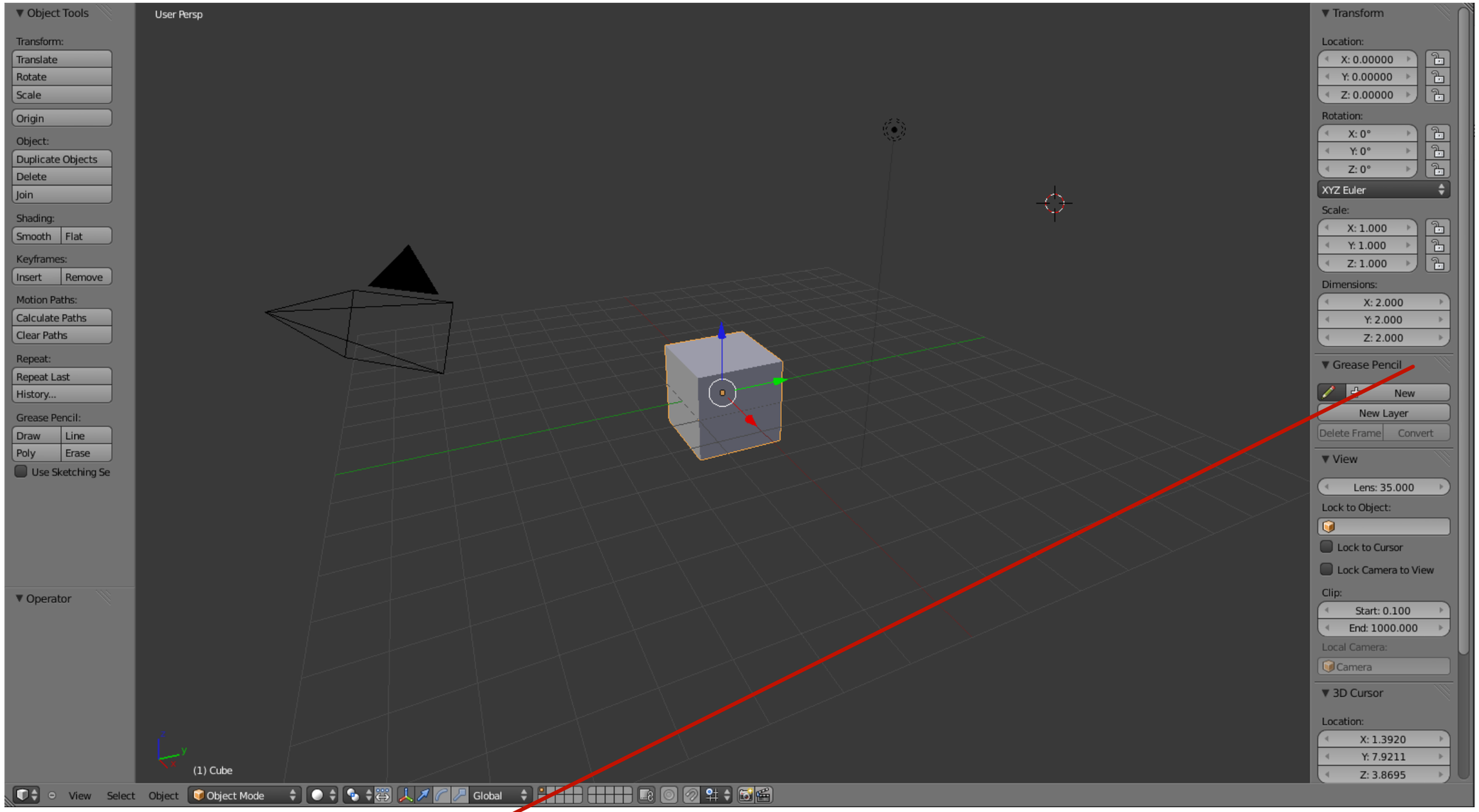
Caméra

Objet (cube)

Options contextuelles : affichage, mode de sélection, calques, etc.

Blender

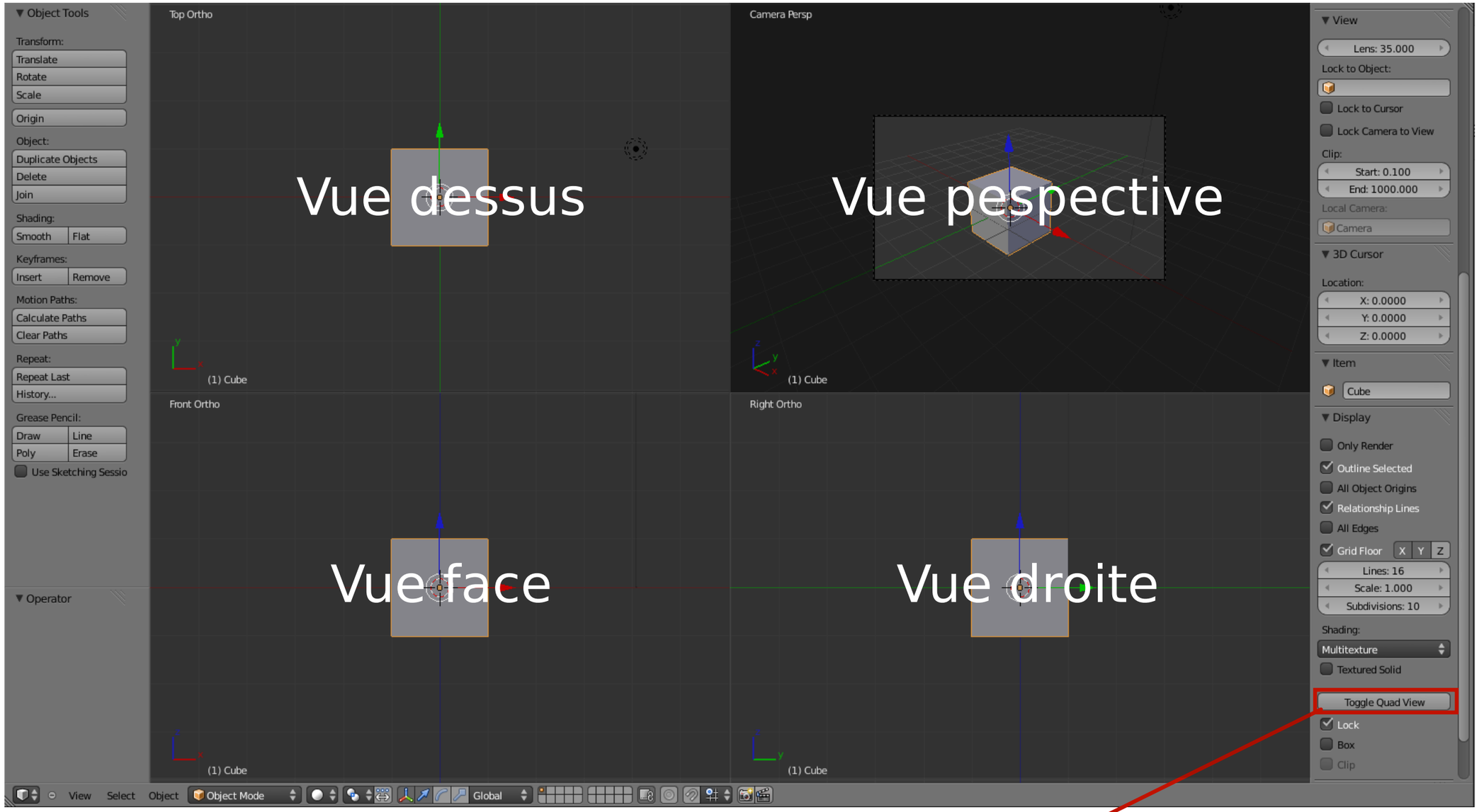
La fenêtre de vue 3D



Panneau latéral (propriétés de l'objet, de l'affichage, etc.)

Blender

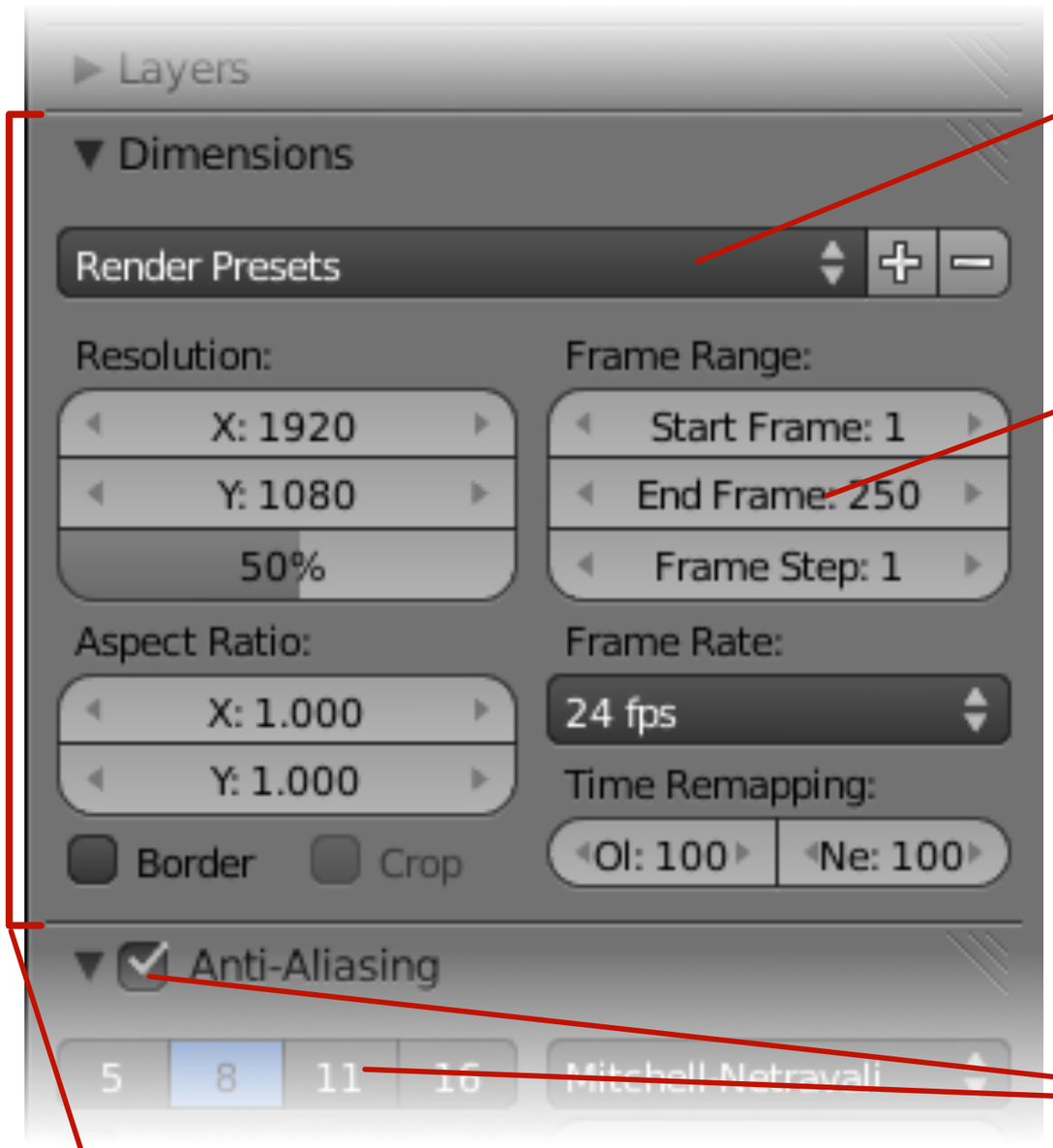
La fenêtre de vue 3D



Bouton d'activation de la vue en quatre zones

Blender

Les boutons et panels



bouton à choix multiple

bouton à valeur numérique

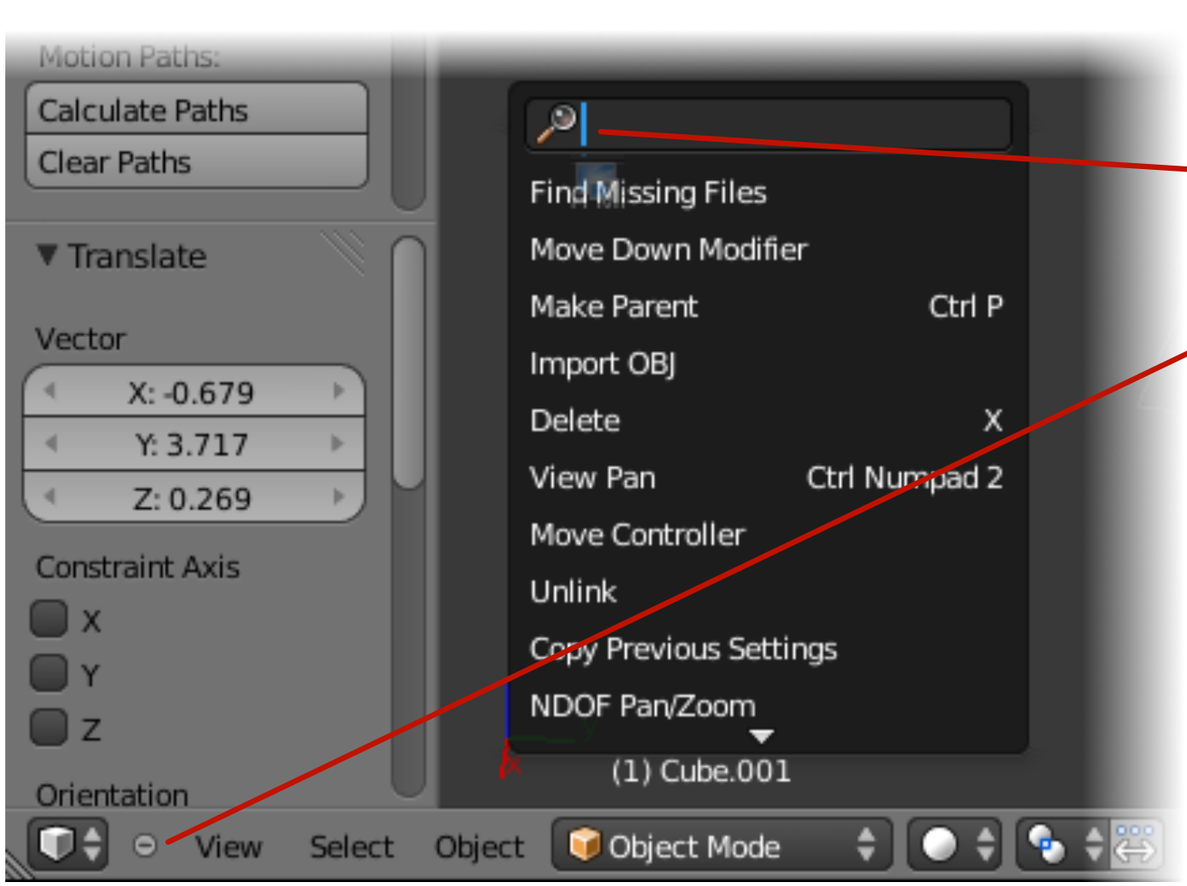
- clic gauche et glissé
- clic gauche sur flèches
- shift+clic gauche et saisie

boutons on/off

panel

Comportement **contextuel**

- suivant l'objet sélectionné
- suivant la position de la souris



Utilisation des **menus**

- contextuel (espace)
- entêtes des fenêtres

Utilisation des **raccourcis clavier**

Blender

La fenêtre 3D : sélection des éléments

Sélection :

- *clic droit* : sélectionner / désélectionner
- *shift* + *clic droit* : ajouter à la sélection
- *ctrl* + *clic droit* : sélection lasso
- « *b* » *:* sélection rectangle

Edition :

- *tabulation :* **mode édition** pour l'objet sélectionné

En mode édition :

- « *l* » : sélection de la **composante connexe** la plus proche
- sélection des **sommets**, **arêtes** ou **faces**
- sélection ou non des **objets cachés**

Voir aussi le menu **Select**

Blender

La fenêtre 3D : modification des éléments

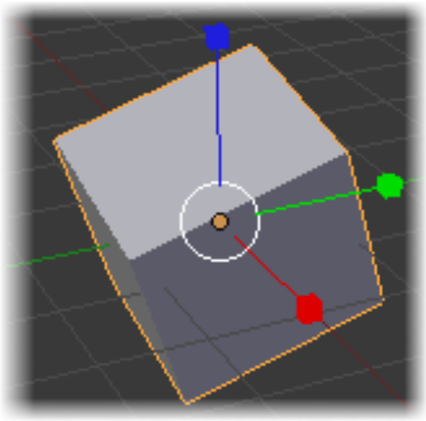
Sélection :

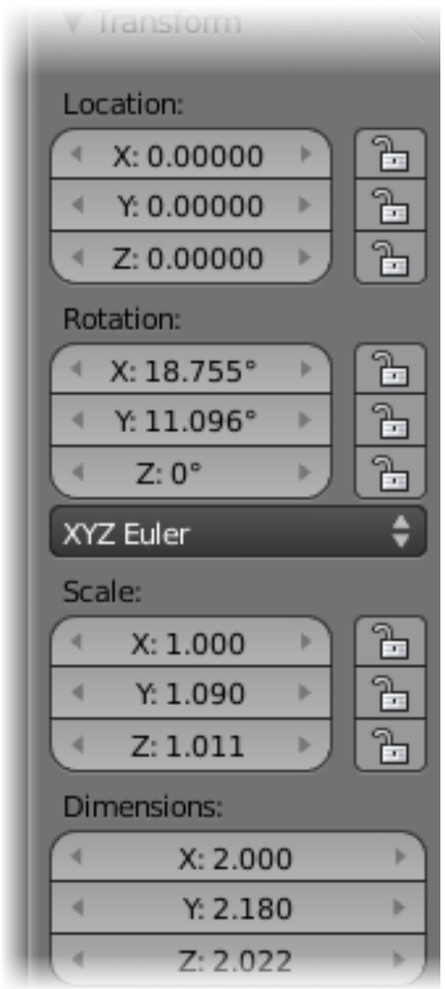
- « *x* » : **supprimer** la sélection
- « *e* » : **extrusion** de la sélection (mode édition)

Transformations :

- « *g* » : **déplacement** de la sélection
- « *r* » *:* **rotation** de la sélection
- « *s* » : **changement d'échelle** de la sélection

ou utilisation du manipulateur 3D, ou du panneau





Voir aussi le menu **Object** ou **Mesh**

Blender

La fenêtre 3D : changement de la vue

Changement de point de vue :

- *clic milieu* : **rotation** de la vue
- *shift* + *clic milieu* : **translation** de la vue
- « + » / « - » *ou molette souris* : zoom de la vue

Projection :

- « *5* » *(numpad)* : **projection** perspective ou orthogonale

Voir aussi le menu **View**

Blender

Modélisation, les bases

Ajout d'un objet :

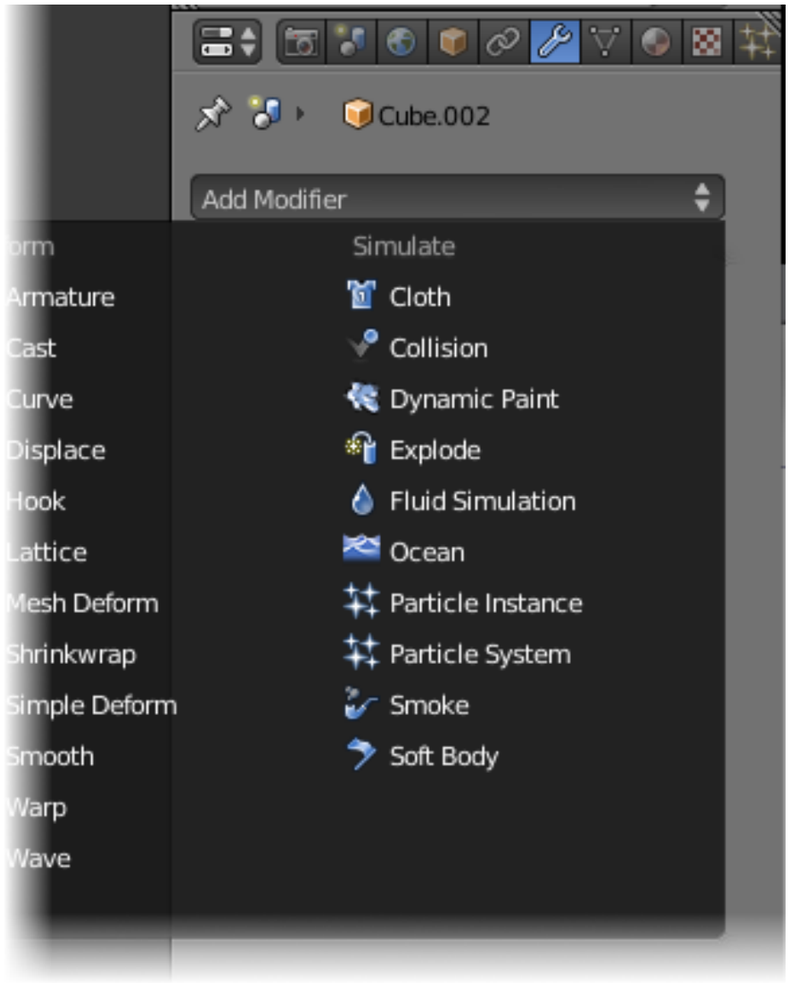
- Menu **Add**, puis **Mesh**

Modification d'un objet :

- Panneau latéral gauche

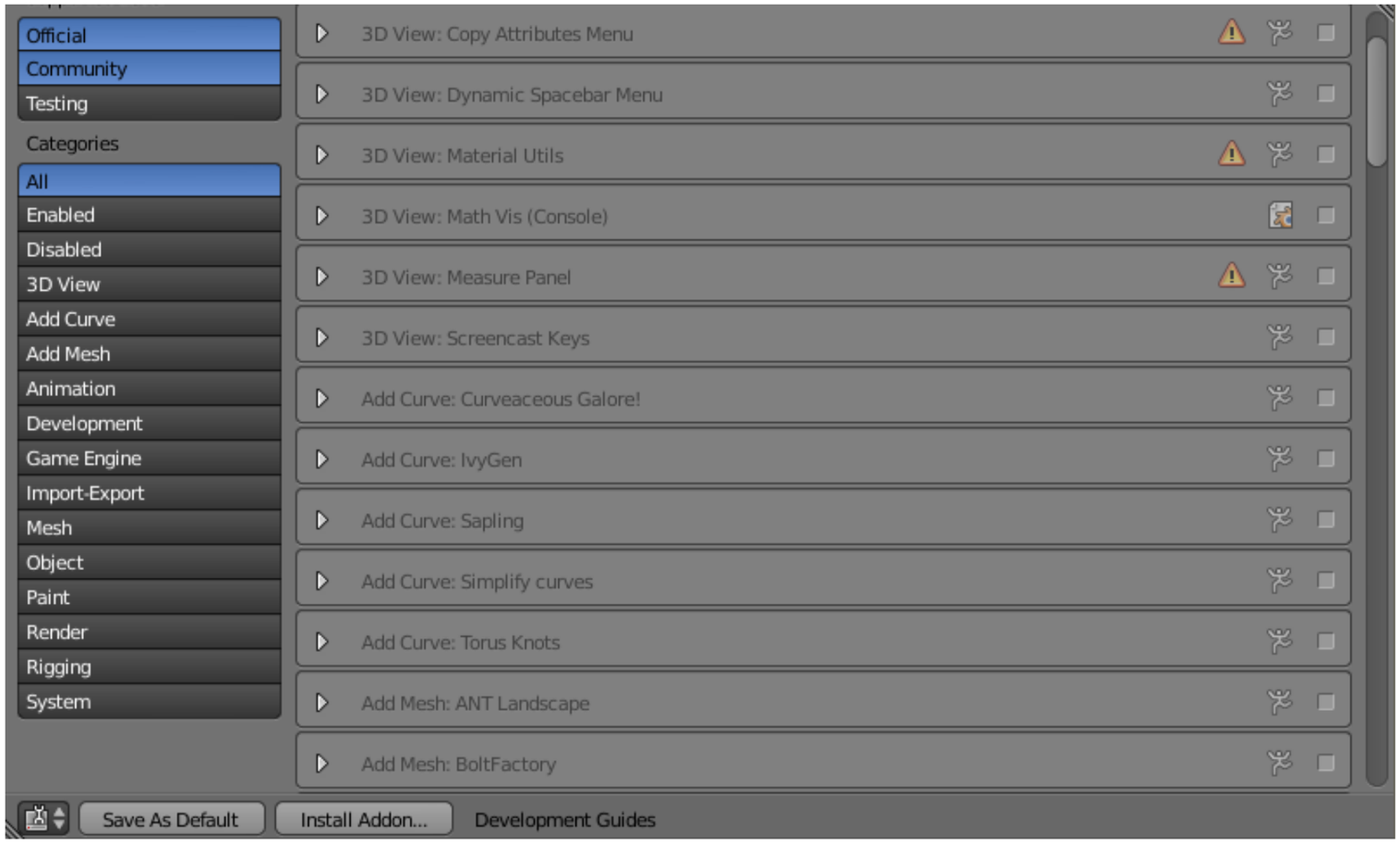
Modifieurs d'objet :

- Panneau **object modifiers**



De nombreuses extensions disponibles :

- écrites en langage python
- complètent les fonctionalités (import, export, animation, etc.)
- possibilité d'en créer facilement



Blender

Extensions python

```python
import bpy


def main(context):
    for ob in context.scene.objects:
        print(ob)


class SimpleOperator(bpy.types.Operator):
    '''Tooltip'''
    bl_idname = "object.simple_operator"
    bl_label = "Simple Object Operator"

    @classmethod
    def poll(cls, context):
        return context.active_object is not None

    def execute(self, context):
        main(context)
        return {'FINISHED'}


def register():
    bpy.utils.register_class(SimpleOperator)


def unregister():
    bpy.utils.unregister_class(SimpleOperator)


if __name__ == "__main__":
    register()

    # test call
    bpy.ops.object.simple_operator()
```

operator_simple.py | Run Script

Templates opérateurs :

- Mécanisme d'enregistrement
- Une classe par outil
- Méthodes prédéfinies
  - conditions d'utilisation
  - exécution
- Conventions de nommage

Calcul du genre d'une surface

- Vérification de la surface (est-ce bien une variété ?)
- Vérification du nombre de composantes connexes
- Affichage du genre

```
    def execute(self, context):
        main(context)
        mesh = bmesh.new()
        mesh.from_mesh(context.active_object.data)
        if not self.surface(mesh):
            self.report({'ERROR'}, "The selected mesh is not a surface")
            return {'CANCELLED'}
        if not self.one_cc(mesh):
            self.report({'ERROR'}, "The selected mesh have more than one
connected component")
            return {'CANCELLED'}
        self.report({'INFO'}, "Genus: " + str(self.getGenus(mesh)))
        return {'FINISHED'}
```

Blender

Exemple : calcul du genre d'une surface

```
    def surface(self, mesh):
        for v in mesh.verts:
            if not v.is_manifold:
                return False
        return True

    def browse_cc(self, mesh, v):
        if v.tag:
            return
        else:
            v.tag = True
        for e in v.link_edges:
            vv = e.other_vert(v)
            if not vv.tag:
                self.browse_cc(mesh, vv)

    def one_cc(self, mesh):
        for v in mesh.verts:
            v.tag = False
        self.browse_cc(mesh, mesh.verts[0])
        for v in mesh.verts:
            if not v.tag:
                return False
        return True
```

Blender

Exemple : calcul du genre d'une surface

```
    def getGenus(self, mesh):
        return int((2 - self.getEulerCharacteristic(mesh) - self.getNbBoundaries(mesh)) / 2)
```

```
    def getEulerCharacteristic(self, mesh):
        nbverts = len(mesh.verts)
        nbedges = len(mesh.edges)
        nbfaces = len(mesh.faces)
        return nbverts - nbedges + nbfaces
```

Blender

Exemple : calcul du genre d'une surface

```
    def browse_boundary(self, mesh, v):
        if v.tag:
            return
        else:
            v.tag = True
        for e in v.link_edges:
            if e.is_boundary:
                vv = e.other_vert(v)
                self.browse_boundary(mesh, vv)

    def getNbBoundaries(self, mesh):
        nbBoundaries = 0
        # set flags: True means inside, False means boundary
        for v in mesh.verts:
            v.tag = len(v.link_edges) == len(v.link_faces)
        # now compute connected components
        for v in mesh.verts:
            if not v.tag:
                self.browse_boundary(mesh, v)
                nbBoundaries = nbBoundaries + 1

        return nbBoundaries
```