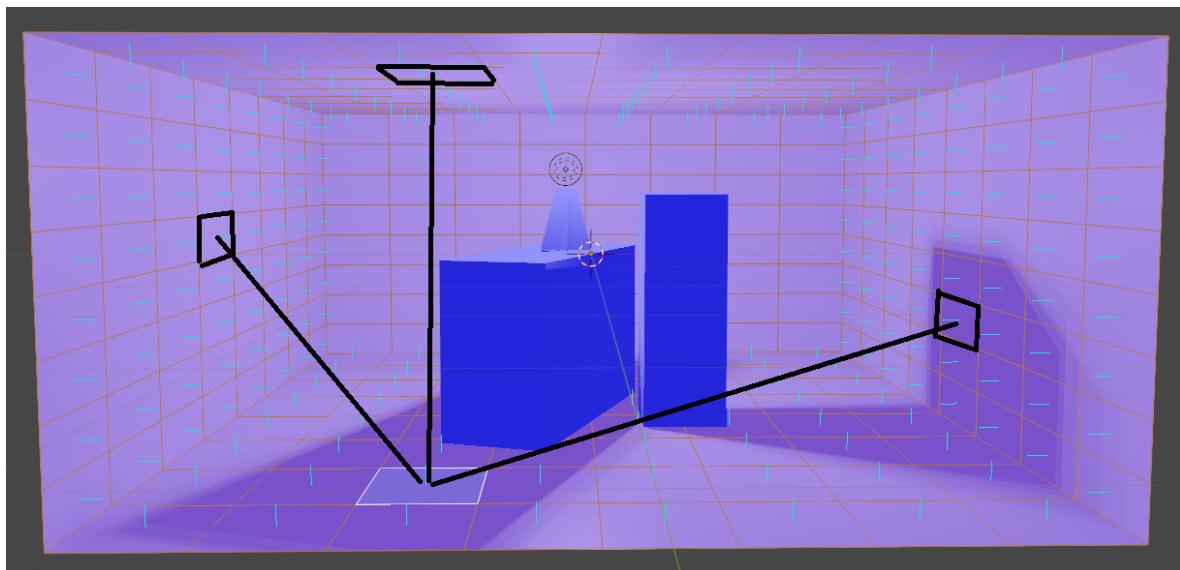


CCI 36 - Projeto 2 - Iluminação global

Prof. Carlos Forster - IEC

Radiosidade

O algoritmo de radiosidade é um algoritmo de iluminação global, ou seja, ele computa as interreflexões entre os objetos de uma cena, podendo tratar de uma forma generalizada os caminhos complexos da luz.



Cada faceta emite uma certa quantidade de luz, sendo uma parte emissão do próprio objeto (se for fonte de luz, ou zero caso contrário) e outra parte da reflexão da energia obtida das demais facetas da cena.

Isso pode ser computado separadamente para cada canal de cor (R, G, B). A equação da radiosidade é dada a seguir, onde B é a energia radiante emitida por cada face e será a nossa incógnita. ρ representa o quanto da luz incidente é refletida para cada componente de cor (ou seja, é a cor natural do objeto, já conhecida). E representa o quanto a faceta emite naturalmente, no caso em que é uma fonte de luz, janela, lâmpada, tela etc.

$$B_R = E_R + \rho_R I_R$$

$$B_G = E_G + \rho_G I_G$$

$$B_B = E_B + \rho_B I_B$$

Por exemplo, um objeto vermelho pode ter $\rho_R = 0.9$, $\rho_G = \rho_B = 0$, $E = 0$.

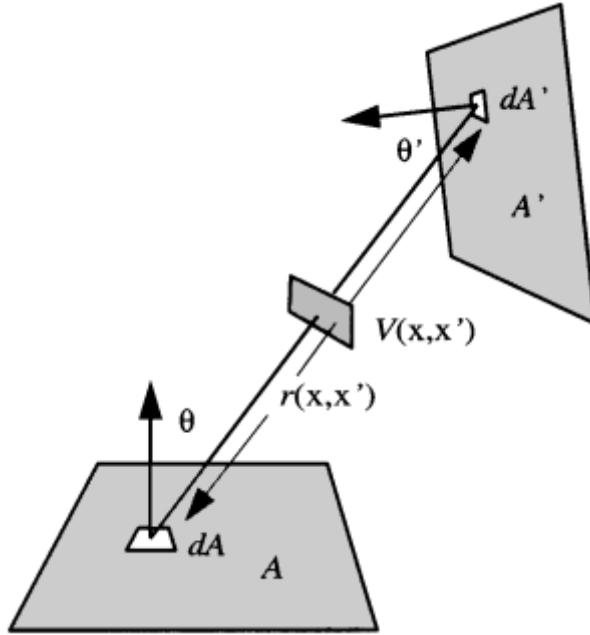
Uma fonte de luz branca poderia ter: $E_R = E_G = E_B = 200$, $\rho = 0$.

Fator de forma:

O fator de forma relaciona duas facetas pelo quanta energia é transmitida de uma para a outra (e vice-versa, pois é simétrica).

Efeitos:

- foreshortening (inclinação da superfície reduz a interação)
- ângulo sólido (radiância proporcional ao ângulo sólido, diminui com quadrado da distância)

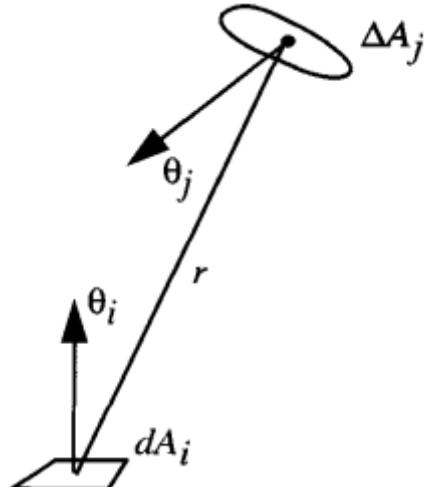


Diferencial do fator de forma:

$$dF_{dA \rightarrow dA'} = \frac{\cos \theta}{\pi} d\omega' = \frac{\cos \theta \cos \theta'}{\pi r(x, x')^2} dA'$$

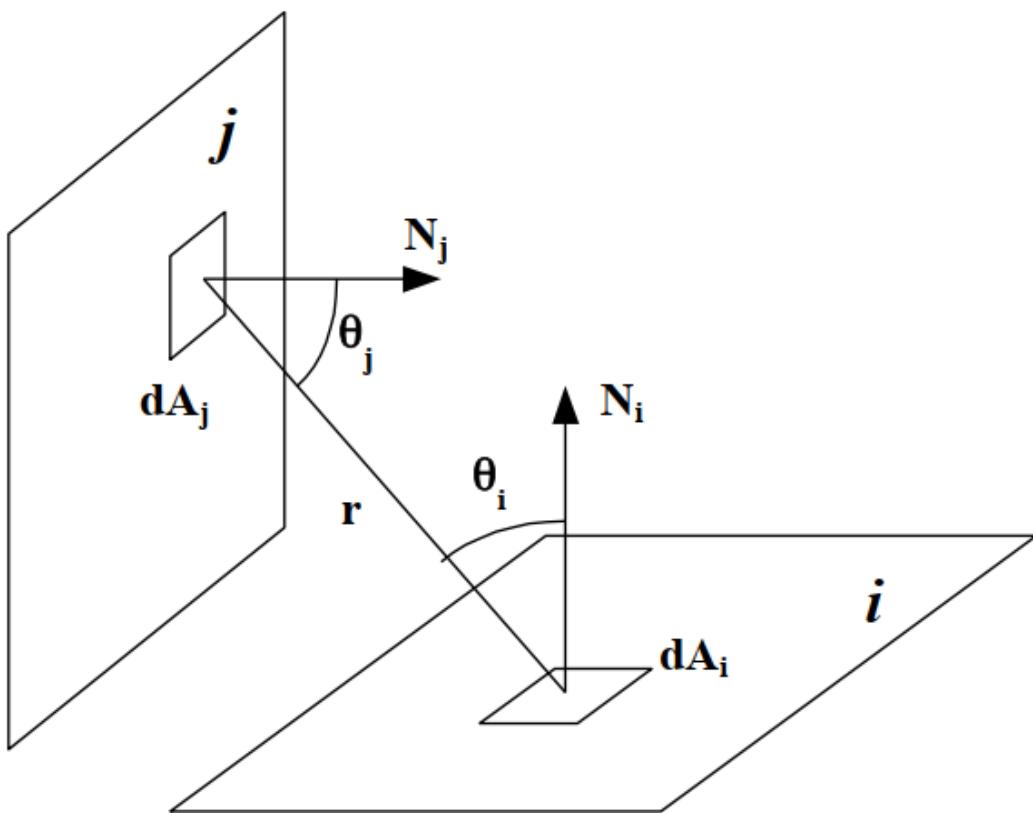
Incluindo a visibilidade "V", que é 0 se há obstrução no segmento de reta ligando os dois pontos das facetas ou 1 se o raio de luz passa livremente.

$$F_{dA \rightarrow dA'} = G(x, x') dA' = \frac{\cos \theta \cos \theta'}{\pi r^2} V(x, x') dA'$$



$$F_{dA_i \rightarrow \Delta A_j} = \frac{\Delta A_j \cos \theta_i \cos \theta_j}{\pi r^2 + \Delta A_j}$$

$$F_{i-j} = \frac{1}{A_i} \iint_{A_i A_j} \frac{\cos \theta_i \cos \theta_j}{\pi r^2} dA_j dA_i$$



Para computar o fator de forma, que pode ser feito de forma aproximada, utilizam-se a distância entre pontos da face, os ângulos entre o segmento ligando os pontos e as normais de cada face e as áreas das faces. O fator é zero se o segmento intercepta uma terceira face (visibilidade impedida por outro objeto, por exemplo).

O fator também é zero se as faces não se "vêem". Para isso, testar o ângulo do raio incidente vindo do centroide de uma face com a direção normal da outra. Se o ângulo for menor que 90, as faces não se vêem (uma está de costas para a outra).

Montando a equação da Radiosidade

A equação da radiosidade, que computa as interreflexões, agora incluindo os fatores de forma:

$$B_i = E_i + \rho_i \sum_j F_{i-j} B_j$$

$$\begin{bmatrix} 1 - \rho_1 F_{1-1} & -\rho_1 F_{1-2} & \cdots & -\rho_1 F_{1-n} \\ -\rho_2 F_{2-1} & 1 - \rho_2 F_{2-2} & \cdots & -\rho_2 F_{2-n} \\ \cdots & \cdots & \cdots & \cdots \\ -\rho_n F_{n-1} & -\rho_n F_{n-2} & \cdots & 1 - \rho_n F_{n-n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \cdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \cdots \\ E_n \end{bmatrix}$$

Veja que é um sistema linear bem grande. Usual é resolver com um processo de Gauss-Seidel, mas pode-se simplesmente utilizar um pacote para resolver a equação.

Veja que E_i só vai ser diferente de zero se a face i for uma fonte de luz.

Os valores de B correspondem à iluminação final (cores) das faces a serem desenhadas na imagem.

Tarefa

Parte 1 - montar cena no Blender

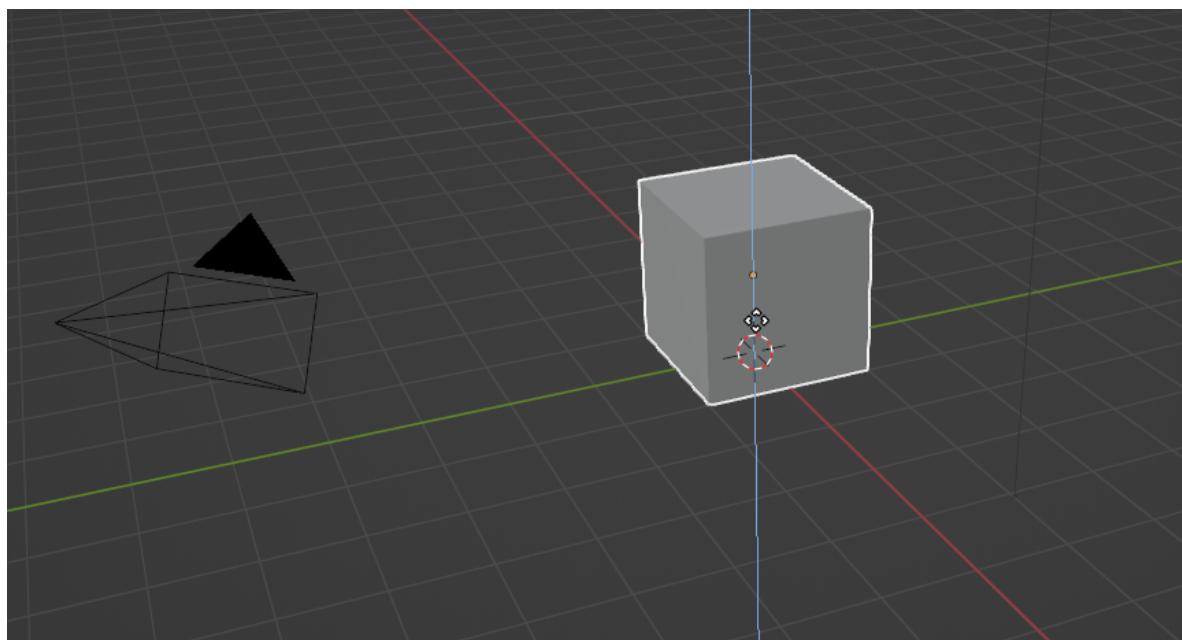
Montar uma cena no Blender para testar o algoritmo de radiosidade.

Primeiramente, colocar alguns objetos primitivos, evitando exagerar o número de faces.

Estou fazendo no Blender 3.2.1:

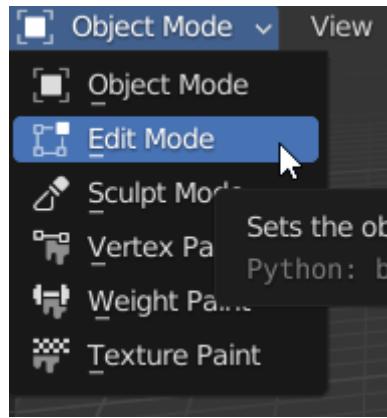


Com a tecla "G" e depois "Z" posicionar o cubo default no eixo "z". Digite "1" e "enter" para subir uma unidade no eixo z. Se quiser entrar os valores manualmente digite "N" para o painel de números.

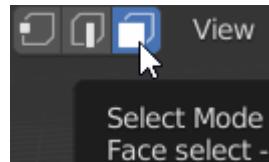


Com "S" para fazer a escala, digite "3 enter" para aumentar o cubo.

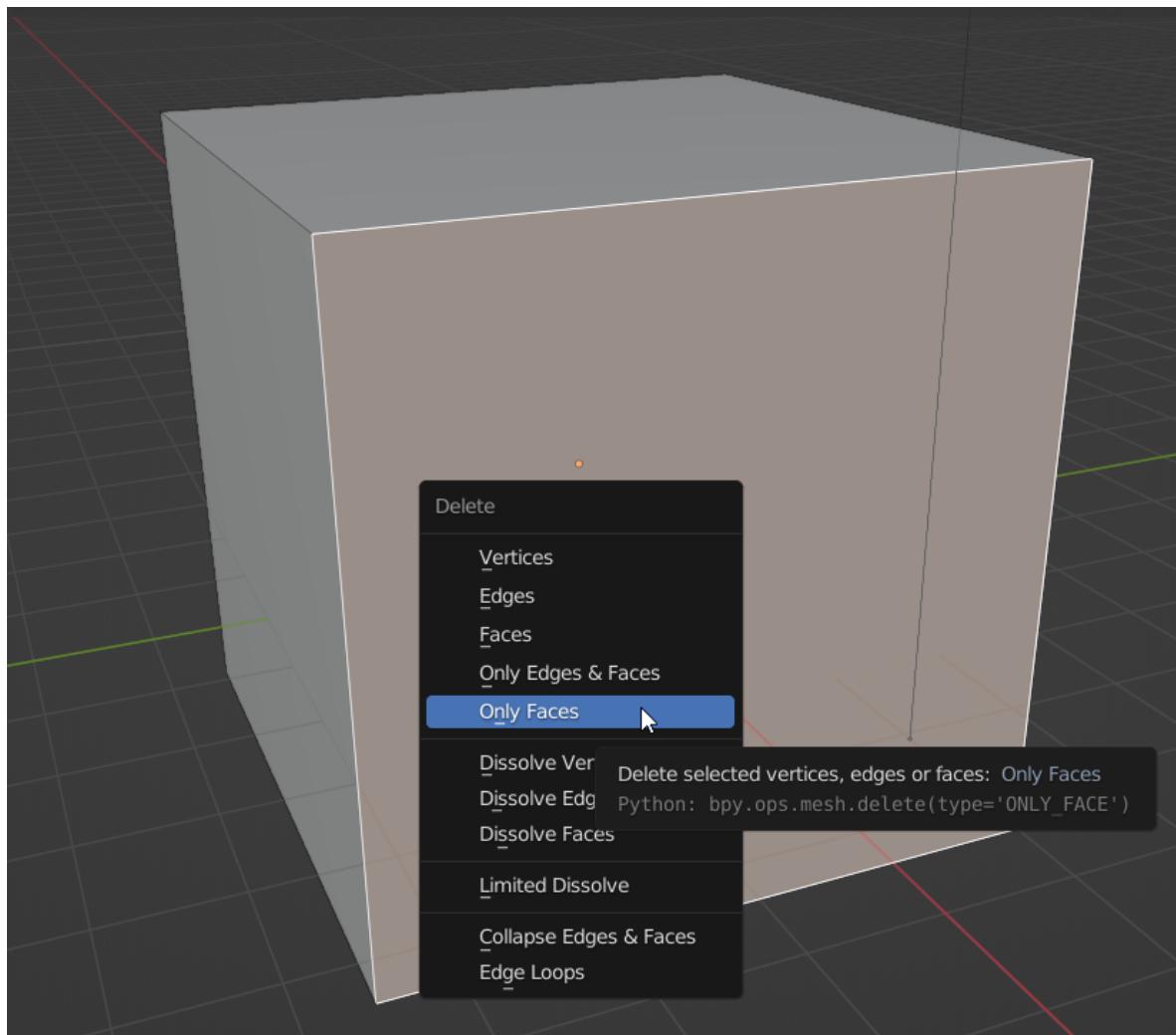
Com o cubo selecionado, entre em modo de edição.



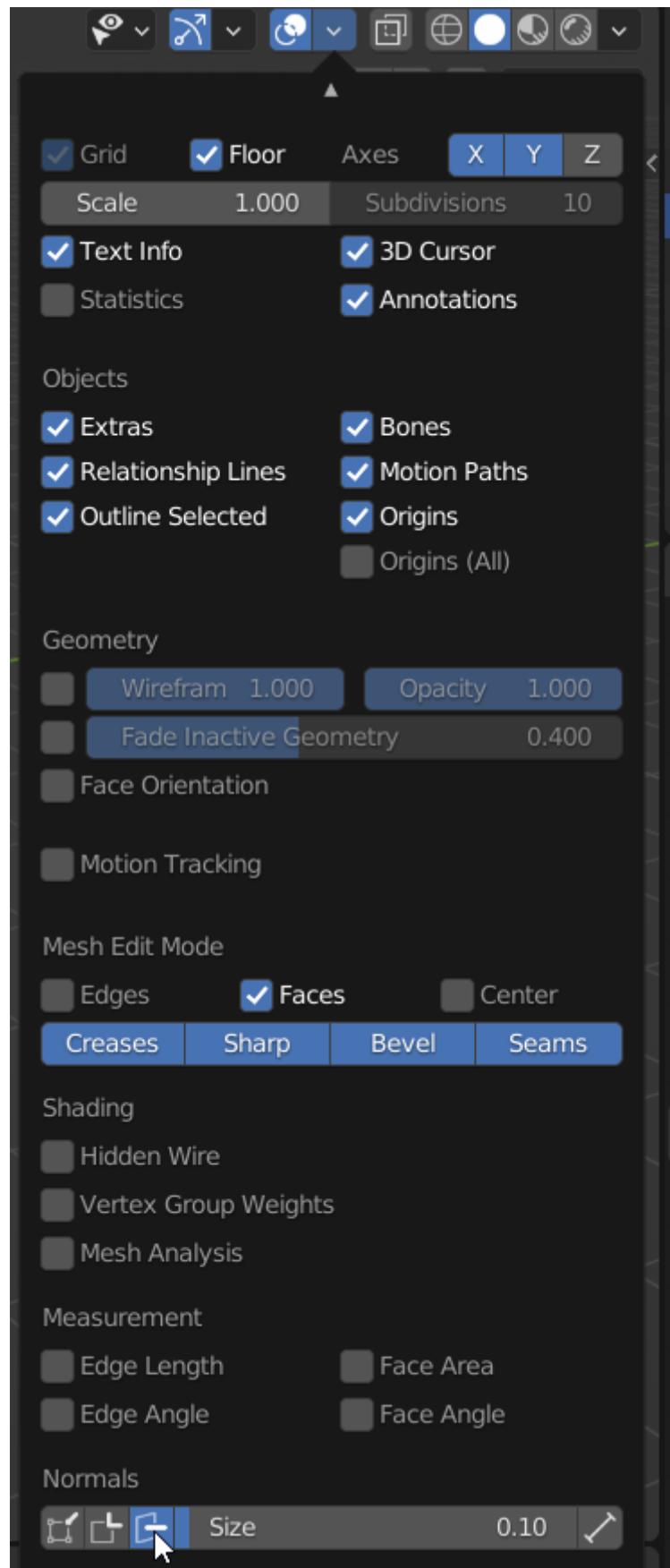
Pressione "3" para selecionar faces



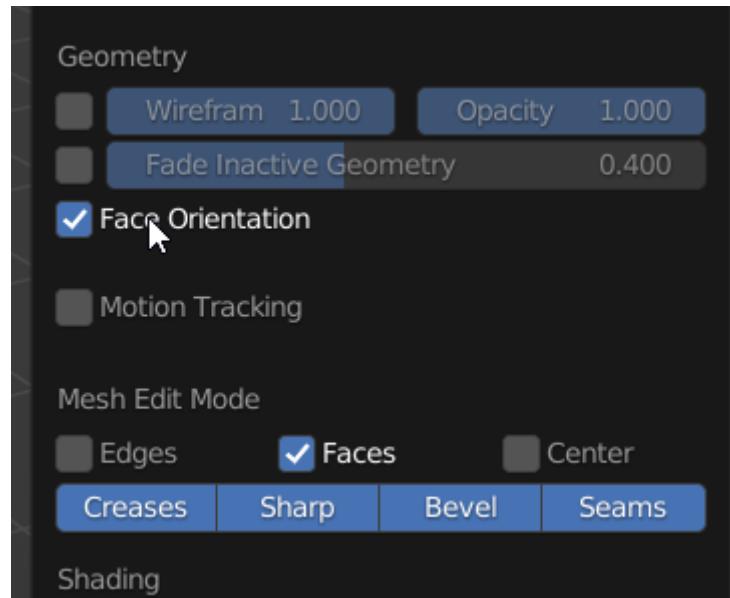
Selecione uma face frontal, pressione "del" e selecione "only faces".



No menu de viewport, marque o ícone : "display normals"

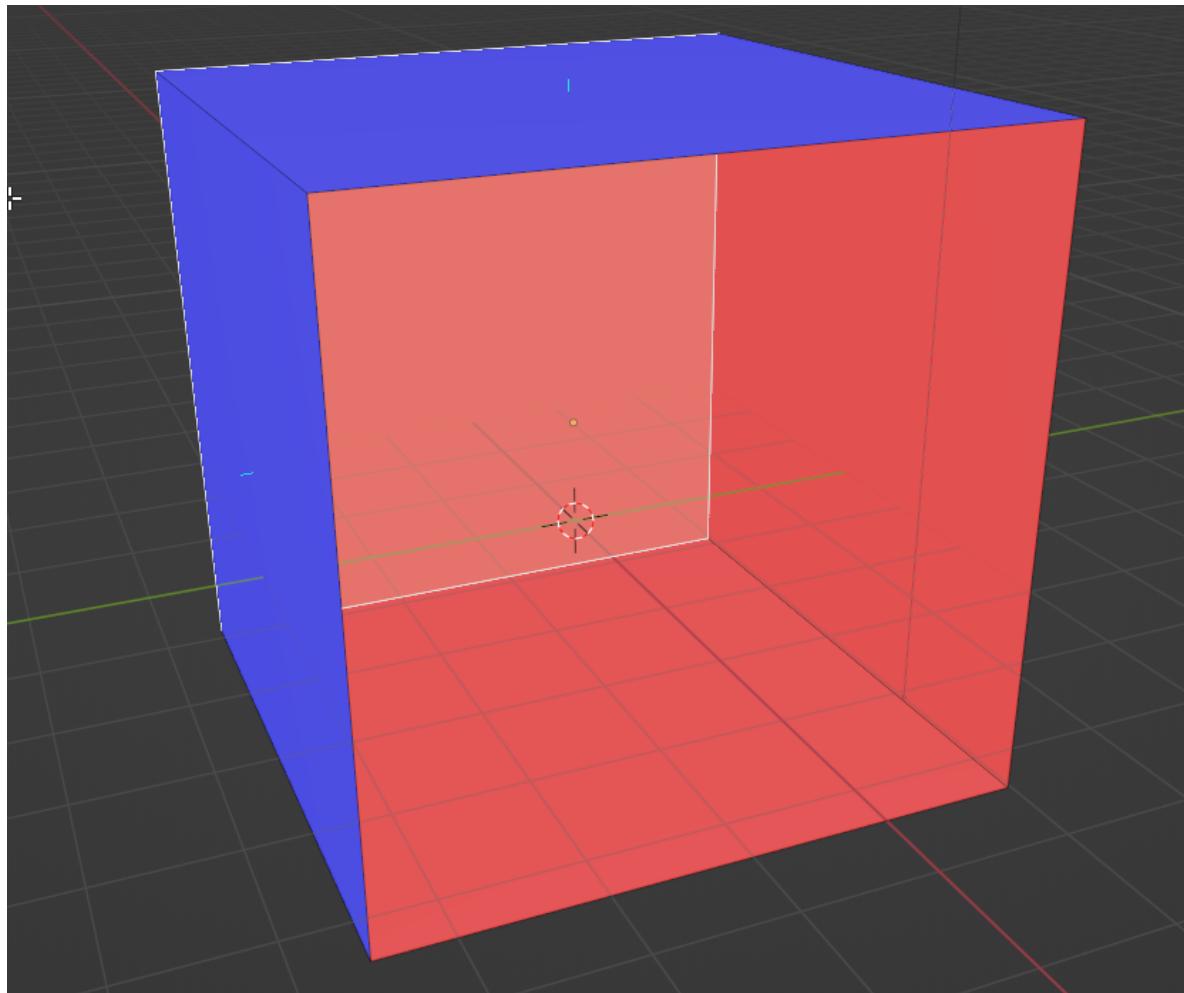


Marque também "face orientation"

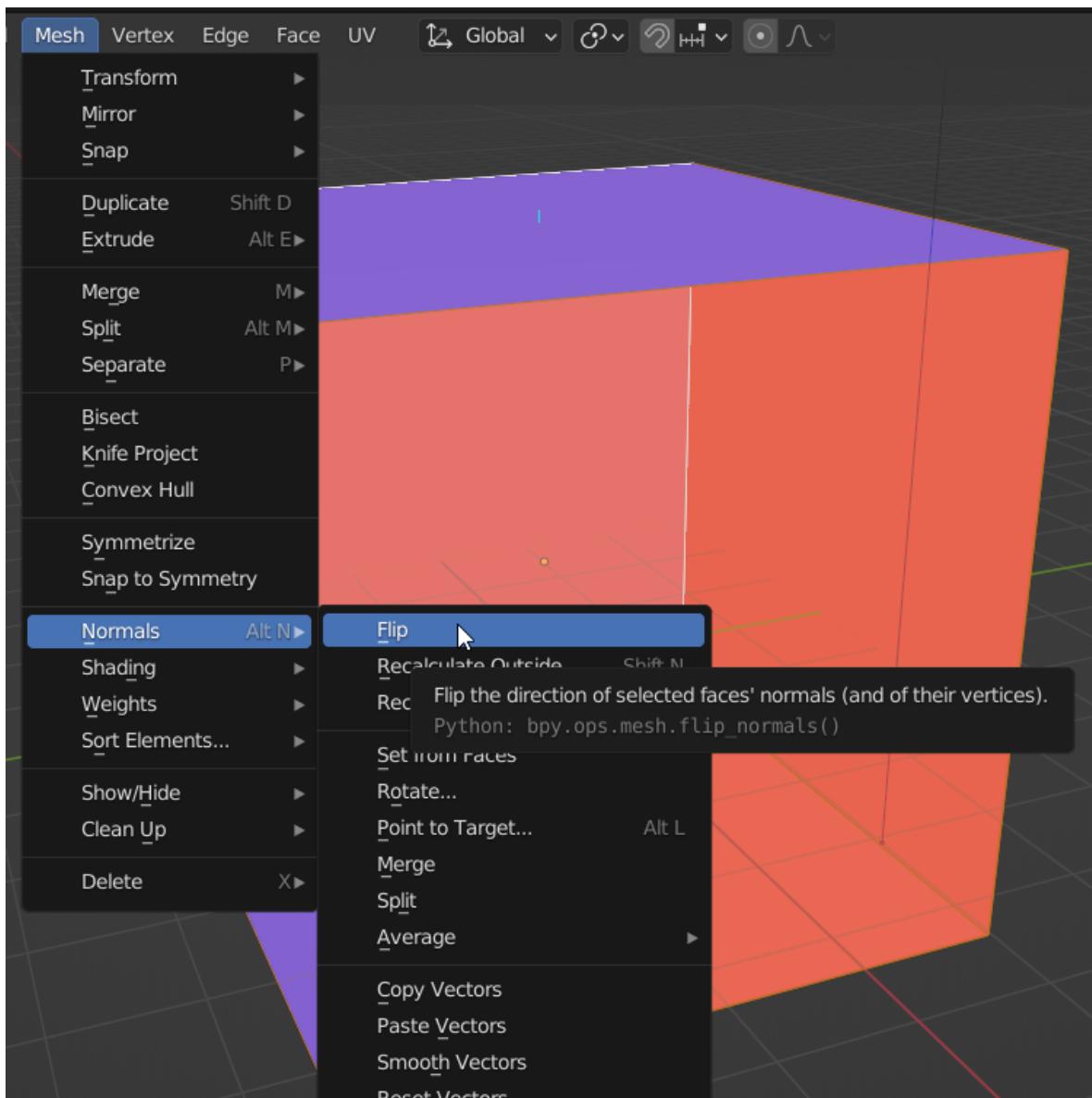


Você estará vendo o interior do cubo em vermelho e o exterior em azul. As normais apontando para fora do cubo.

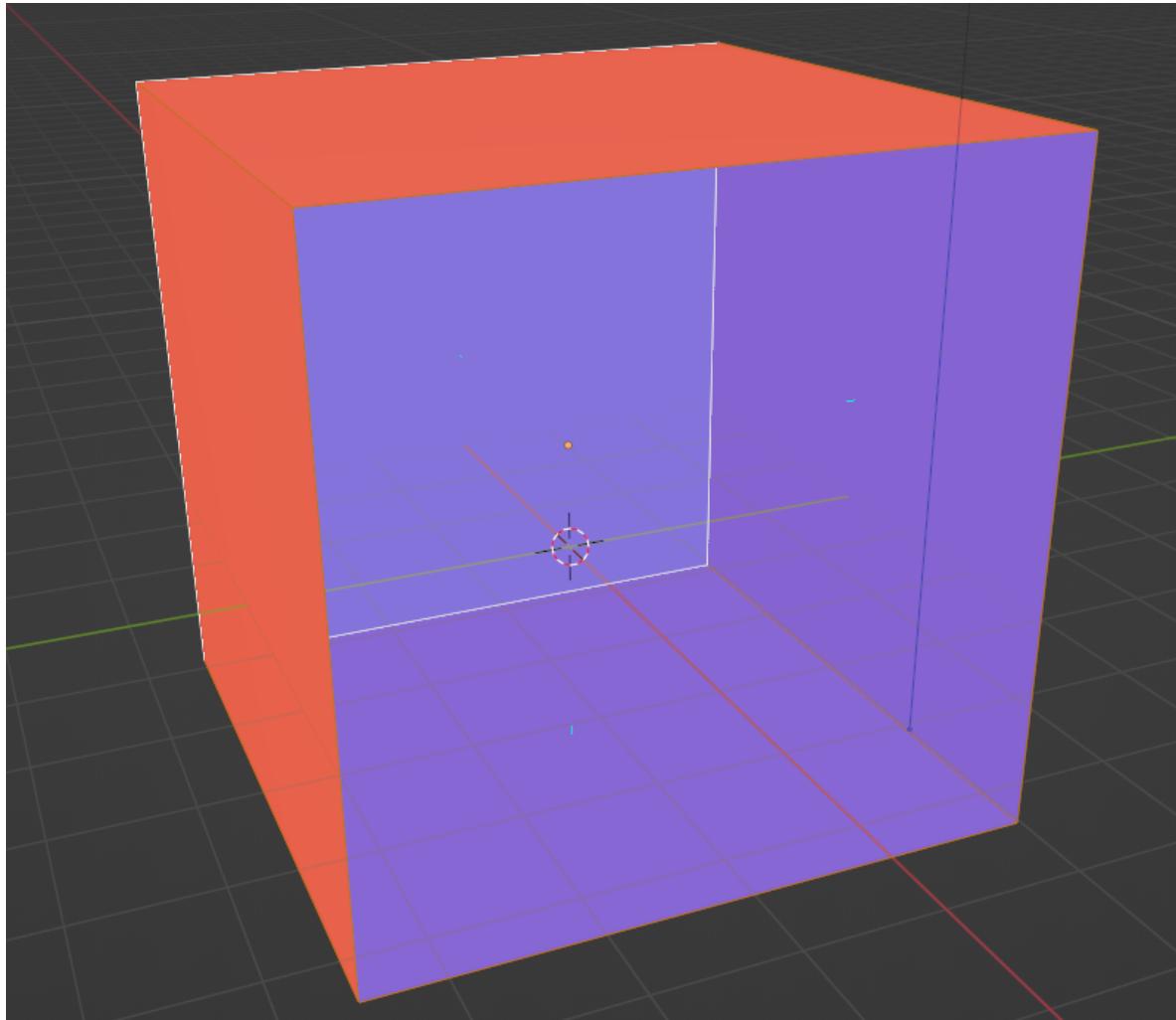
Queremos inverter porque nossa câmera estará vendo o cubo por dentro.



Tecle "A" para selecionar tudo, escolha no menu "Mesh" a opção "Normals/Flip"

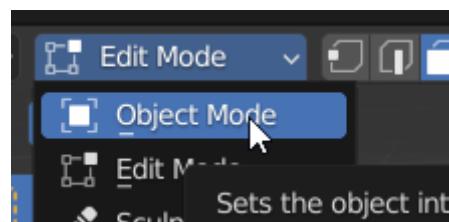


Feito. As normais apontam para dentro.



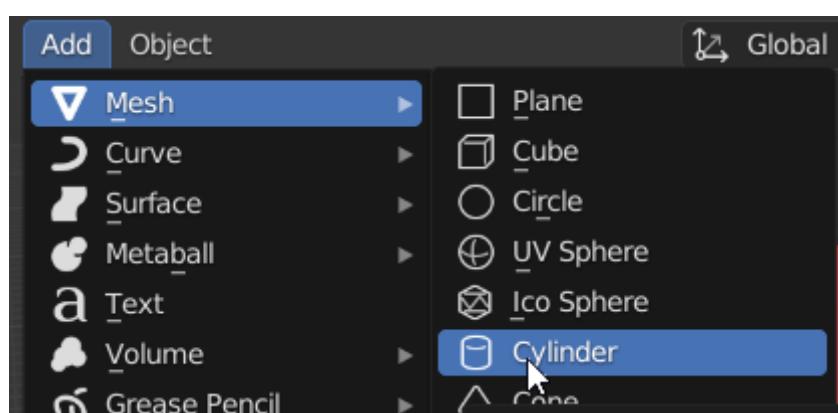
Vamos colocar outros objetos agora, dentro do cubo

Mudar para "object mode". A tecla "Tab" pode fazer isso direto.

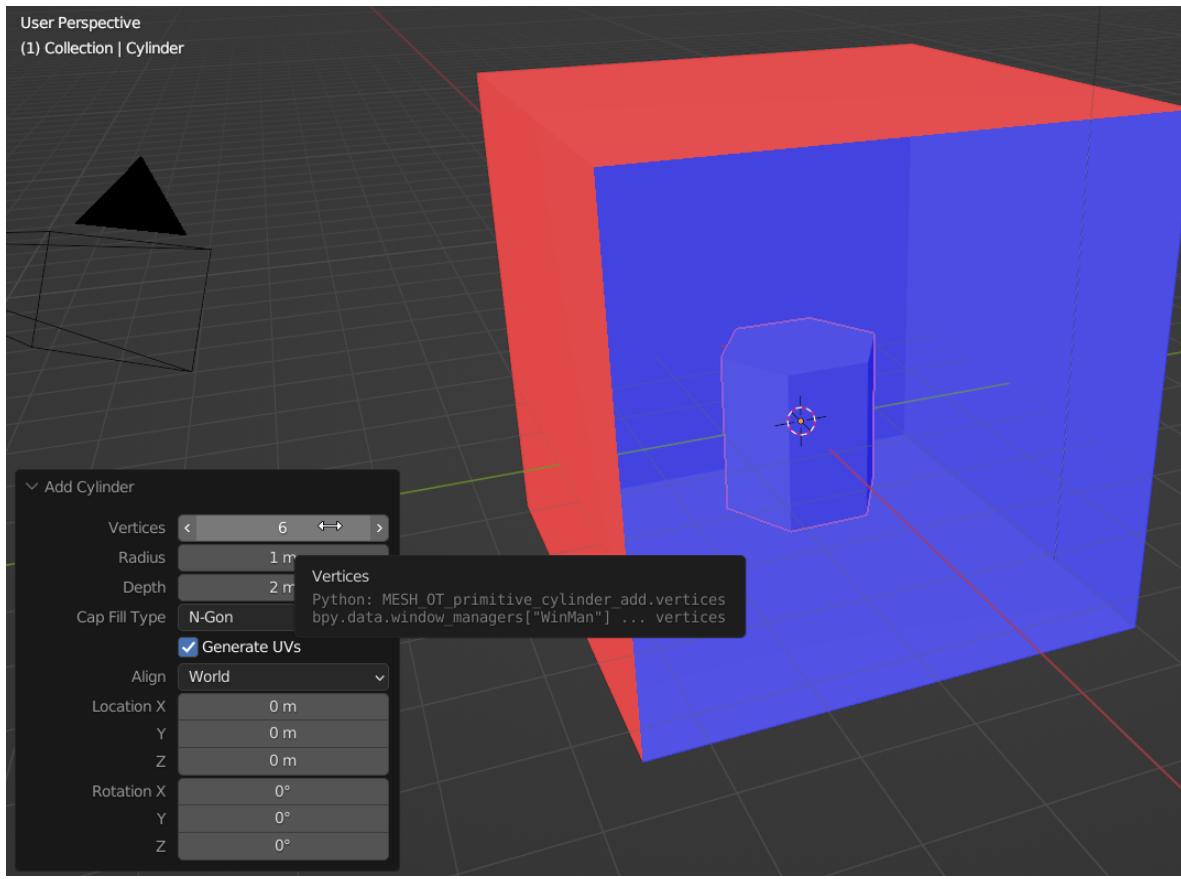


Colocar um cilindro, mas com poucas faces:

"Add - Mesh - Cylinder"

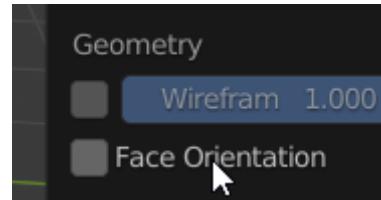


No quadro "Add Cylinder" que aparece, selecione "vertices" e coloque 6 por exemplo.

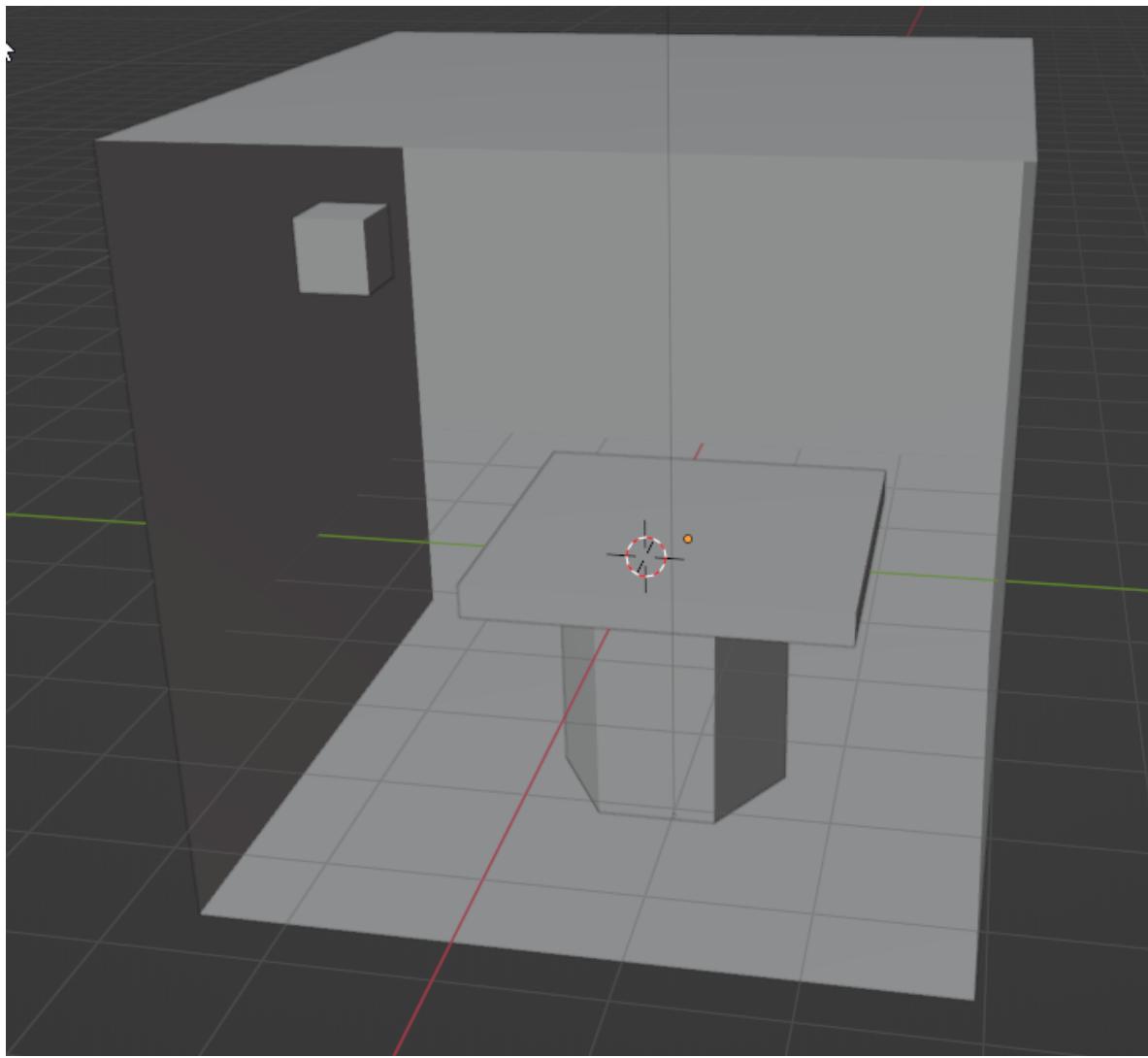


Ajustando a vista com o teclado numérico, o controle de vista ou o botão central do mouse. Utilizando "G" para translação, "S" para escala e "R" para rotação, vá posicionando objetos da maneira que preferir.

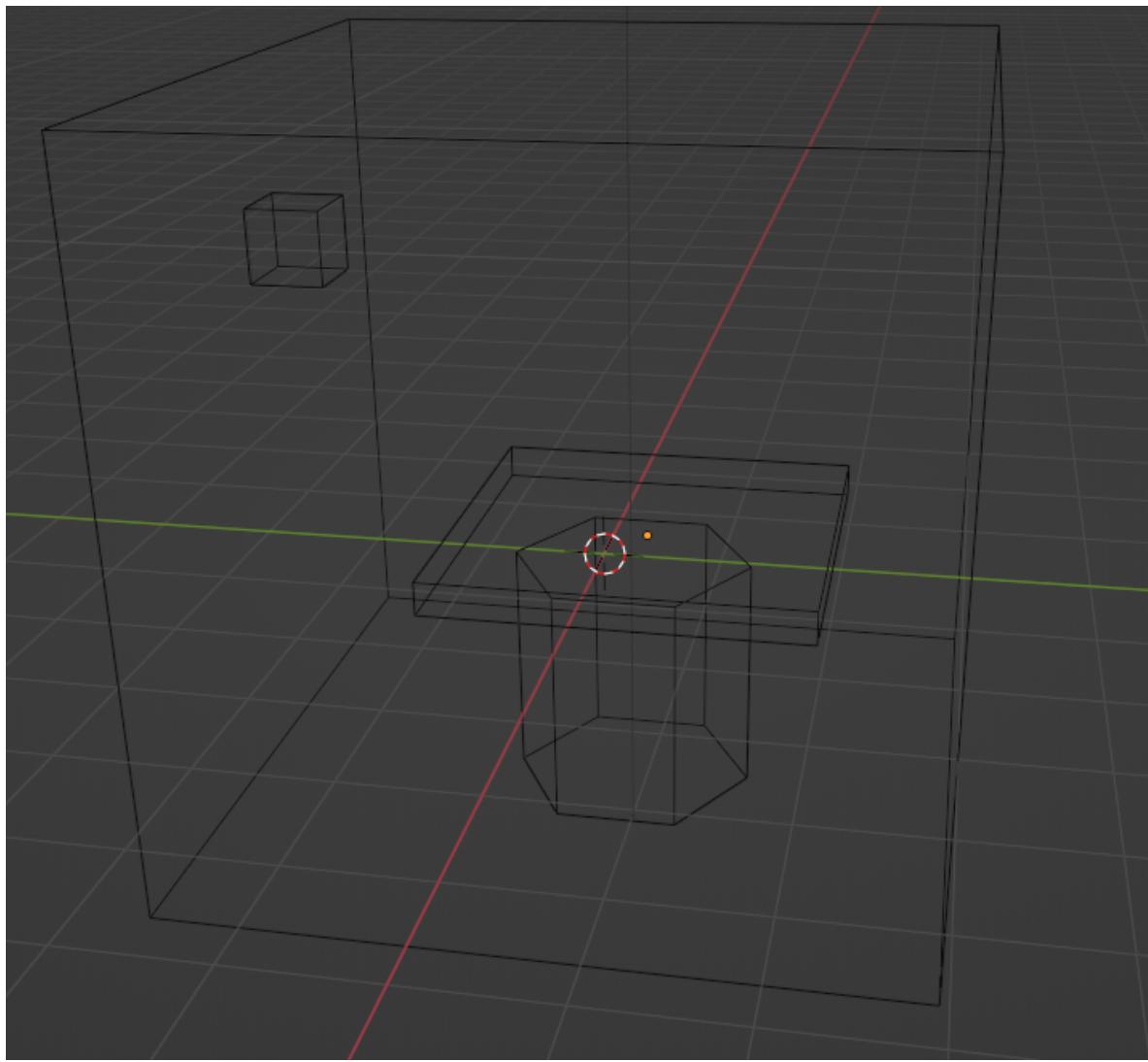
Remover o "face orientation" porque já nos serviu.



Coloque um objeto para ser uma fonte de luz. Coloquei um cubo no alto.

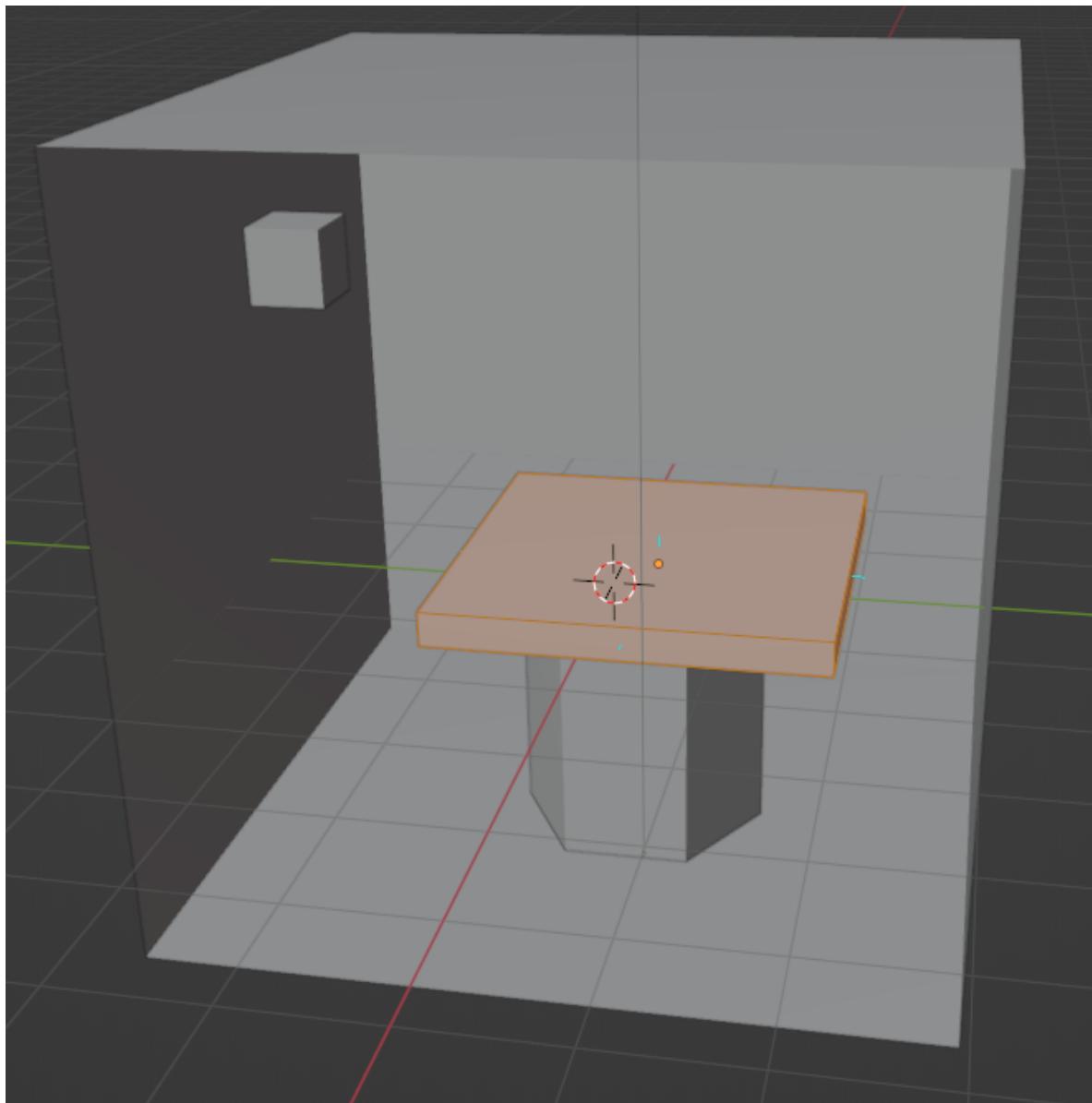


O "shift-Z" permite visualizar wireframe e selecionar objetos que estão por trás.

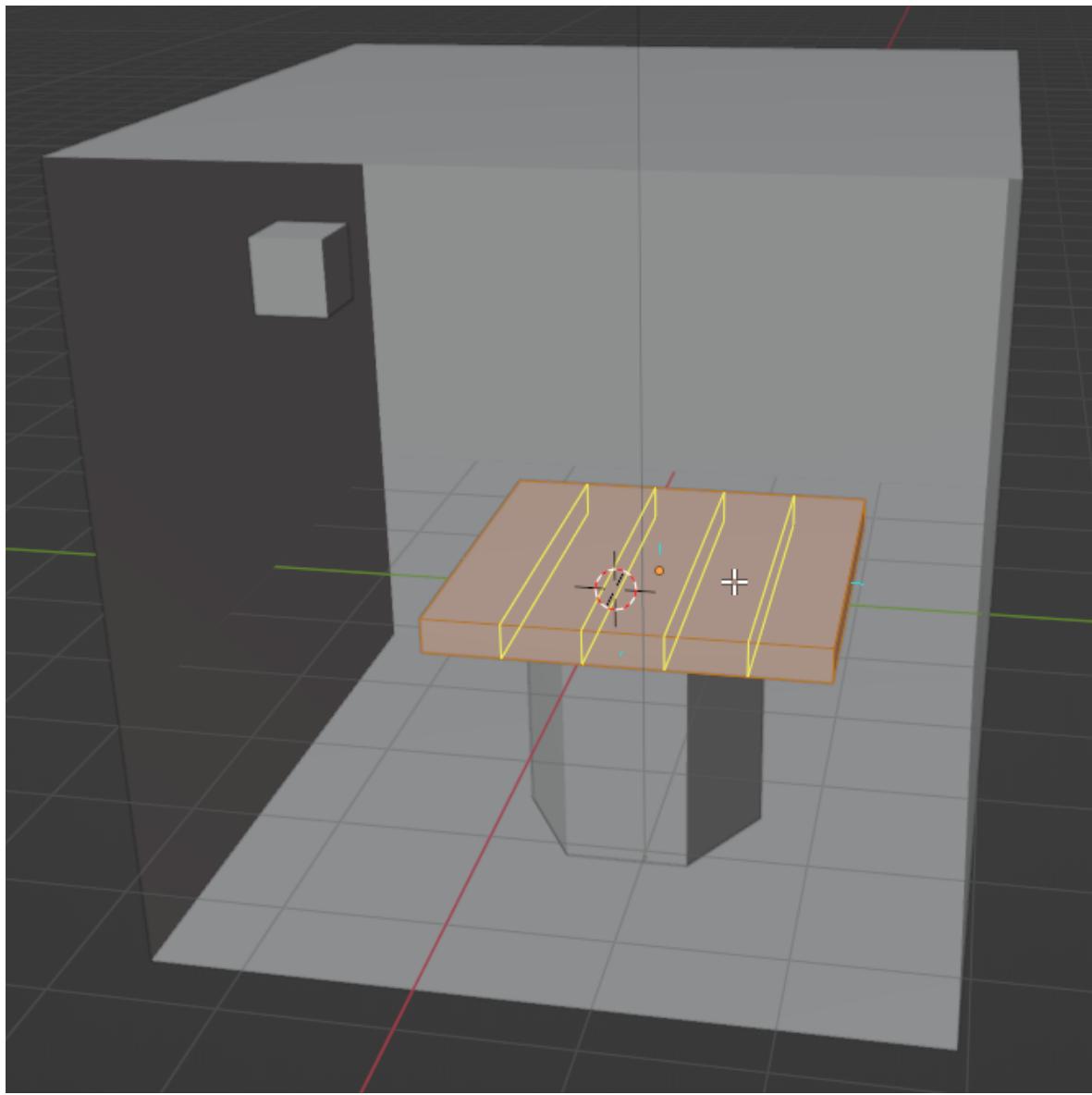


Vamos subdividir as malhas para calcular a Radiosidade

Selecione o objeto e vá para modo edição.



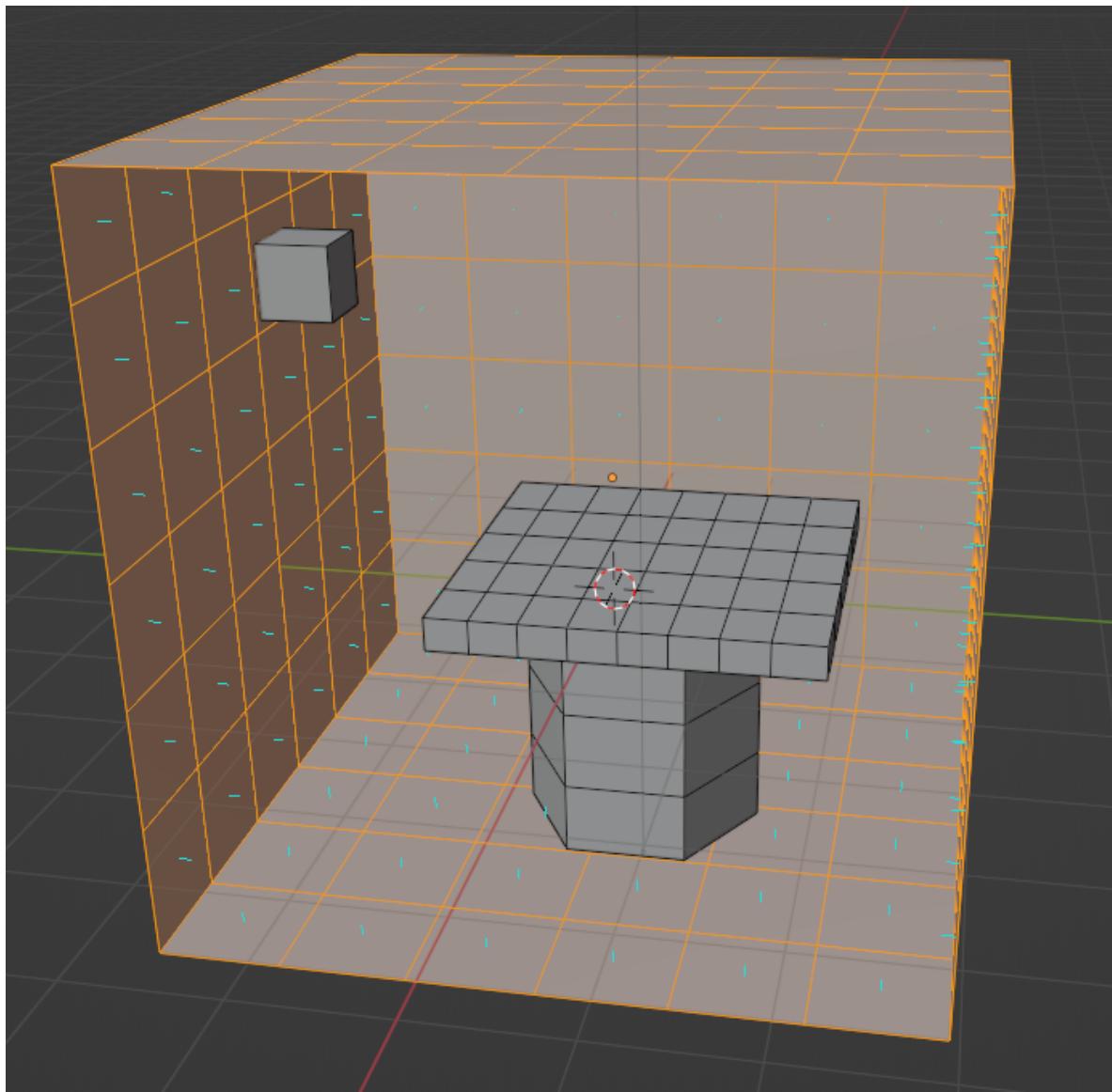
Pressione "ctrl-R" posicione o mouse sobre o objeto e mova a roda do mouse para selecionar o número de subdivisões.



Clique e deslize o mouse para posicionar as divisões (ou seja, não mexa o mouse para elas ficarem onde estão). Clique novamente.

Vai aparecer um painel "Loop Cut and Slide" em que pode ajustar numericamente as divisões feitas.

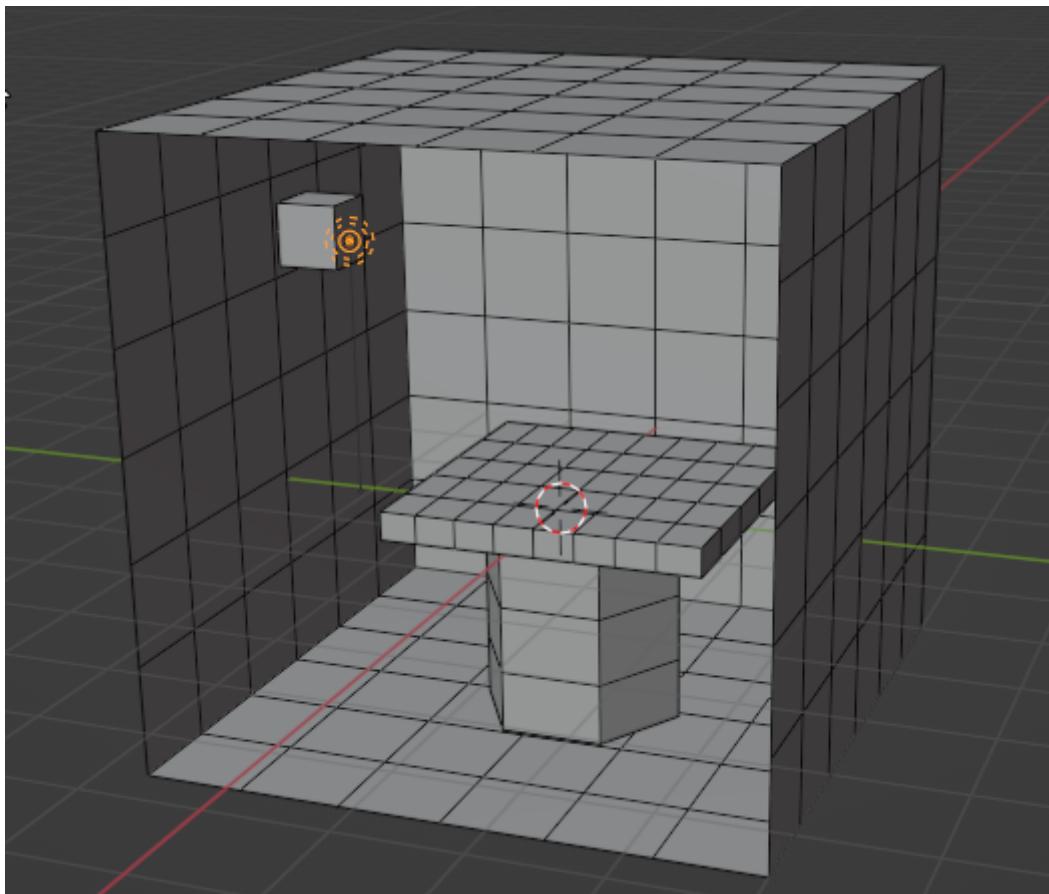
Volte ao modo objeto, selecione outro e volte ao modo edição para fazer outras divisões.



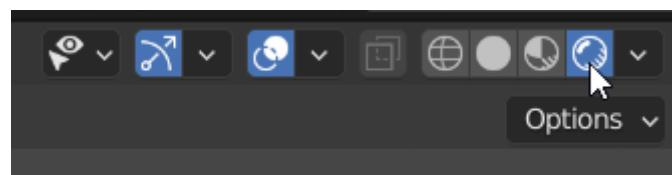
Aqui eu coloquei também a opção de visualizar com "wireframe" junto.

Vamos trazer aquela fonte de luz "perdida" para perto do cubo que será a fonte de luz.

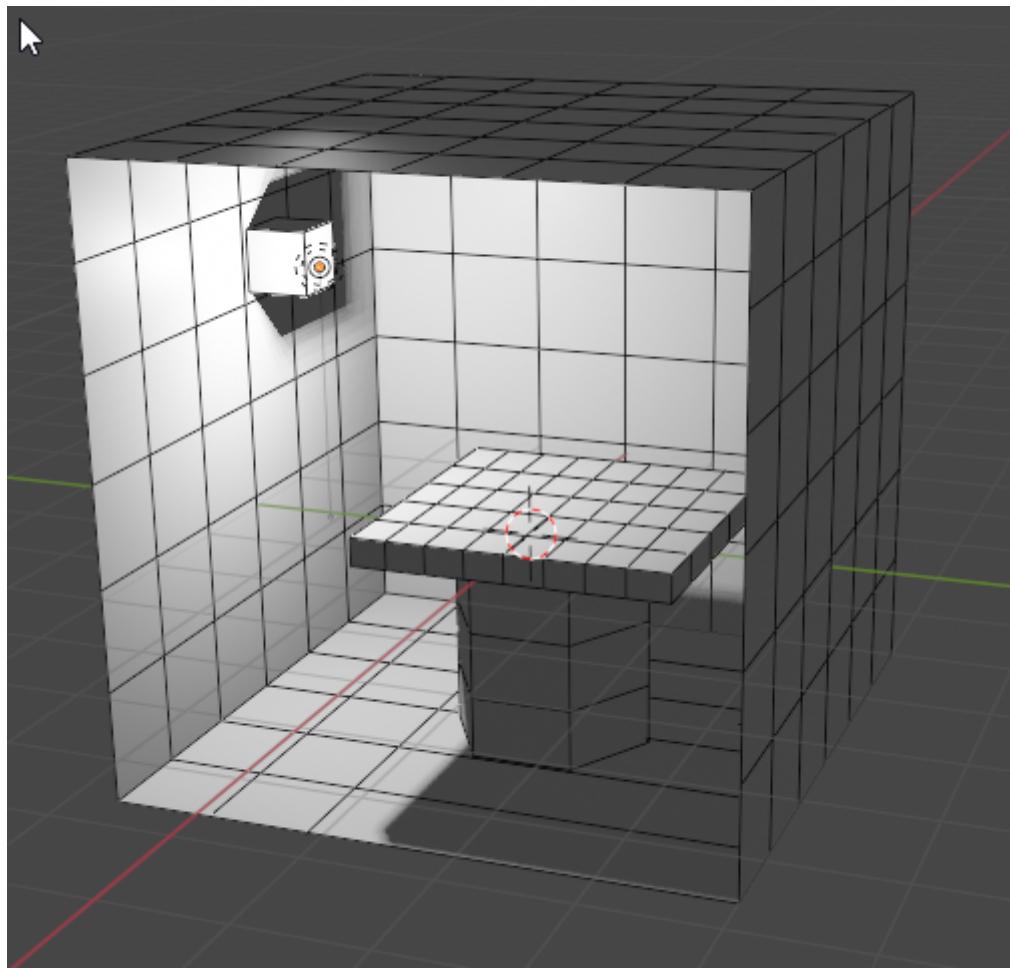
Com a roda do mouse diminua o zoom, selecione e com "G" vai posicionando próxima ao cubo.



Coloque em modo de visualização renderizada.



Temos agora uma noção do sombreamento.



Feitas essas subdivisões, já temos as facetas. Agora precisamos definir as cores dos objetos.

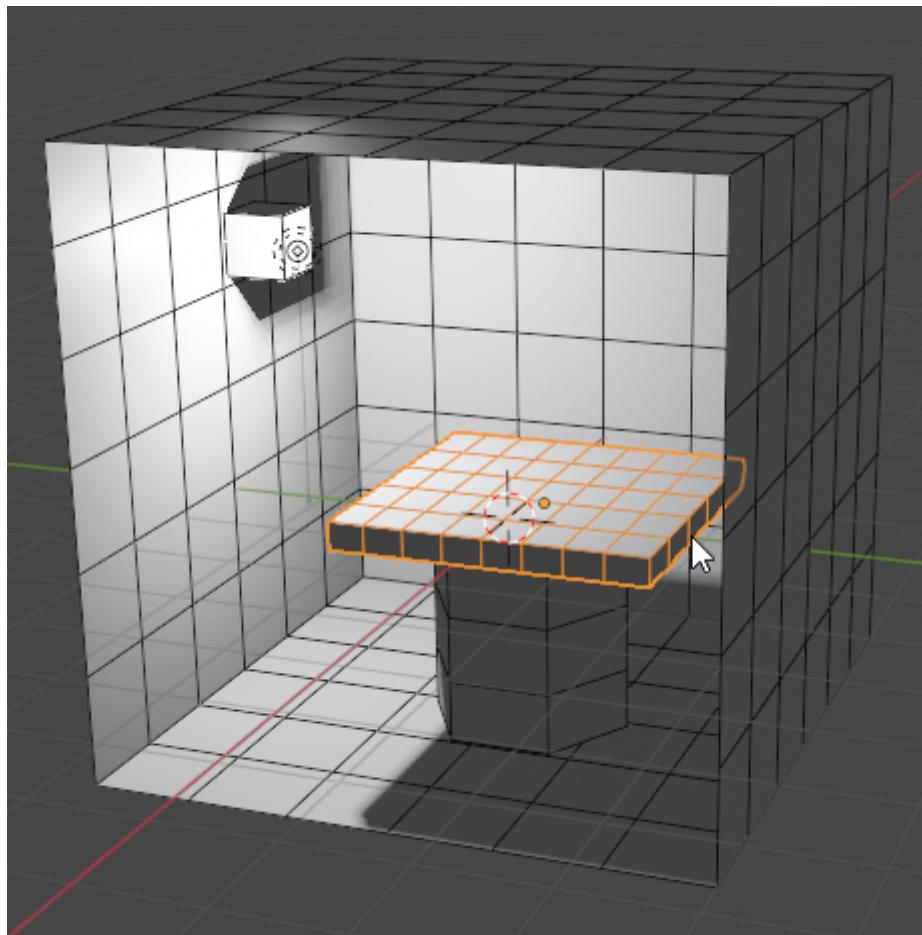
No Blender precisamos criar uma "Color Layer" em cada objeto.

Aqui é um ponto meio instável, propício a bugs, pois nem toda funcionalidade é implementada pelos exportadores de arquivo. Então vou mostrar o que deu certo.

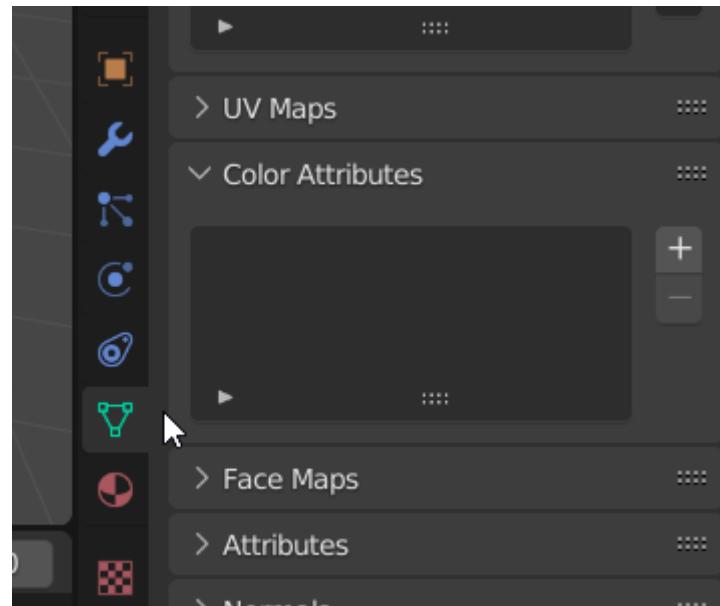
Coloque em modo objeto.

Selecione um objeto.

Selecionei o tampo da mesa. Quero todos os pixels marrons.



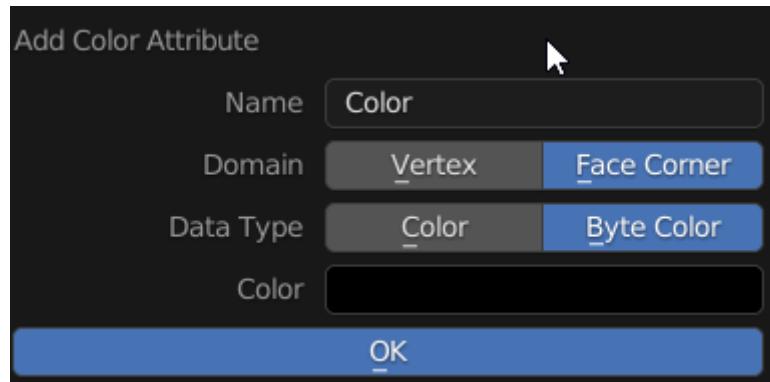
Selecione a aba "geometria"



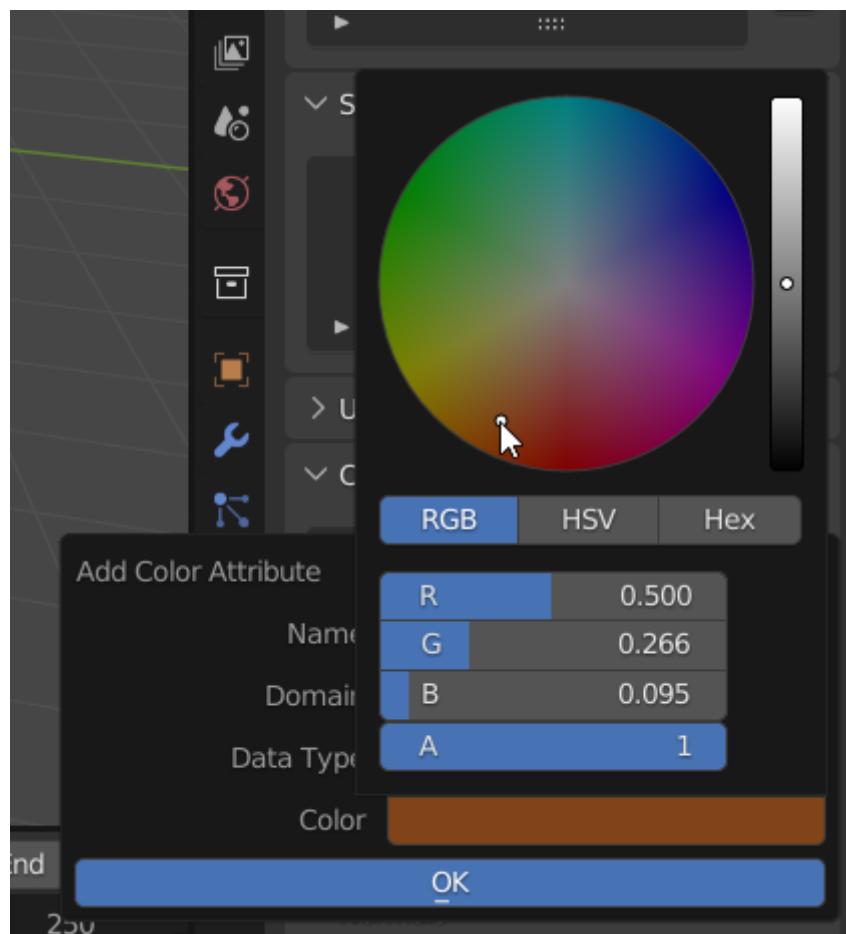
Abra o painel "Color Attributes" e aperte o "+" do painel (não do teclado).

Aparecerá o painel "Add Color Attribute".

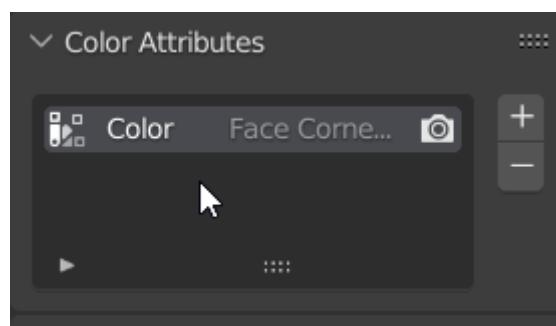
Selecione obrigatoriamente "Face Corner" e "Byte Color". Nenhuma outra opção vai funcionar na versão atual!!



Clique em Color e escolha a cor do objeto, ou entre manualmente valores RGB.



A cor escolhida vai ser a constante " ρ ", coeficiente da reflexão da superfície.



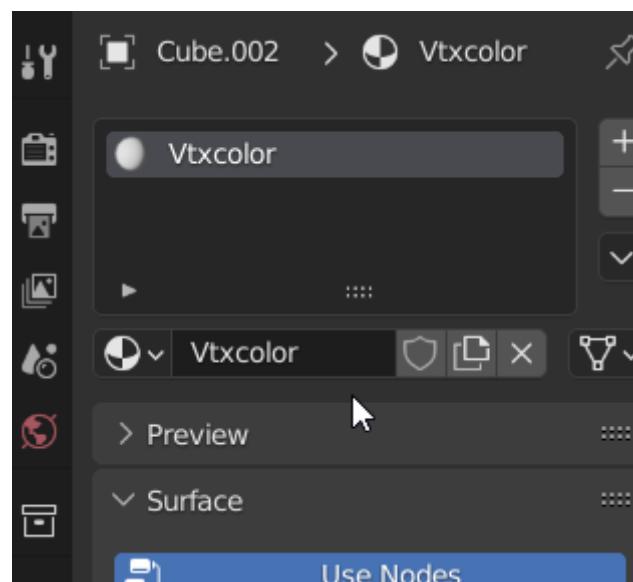
Pronto. Está criado o color layer desse objeto.

Esta operação deve ser repetida para os outros objetos, agora ou depois.

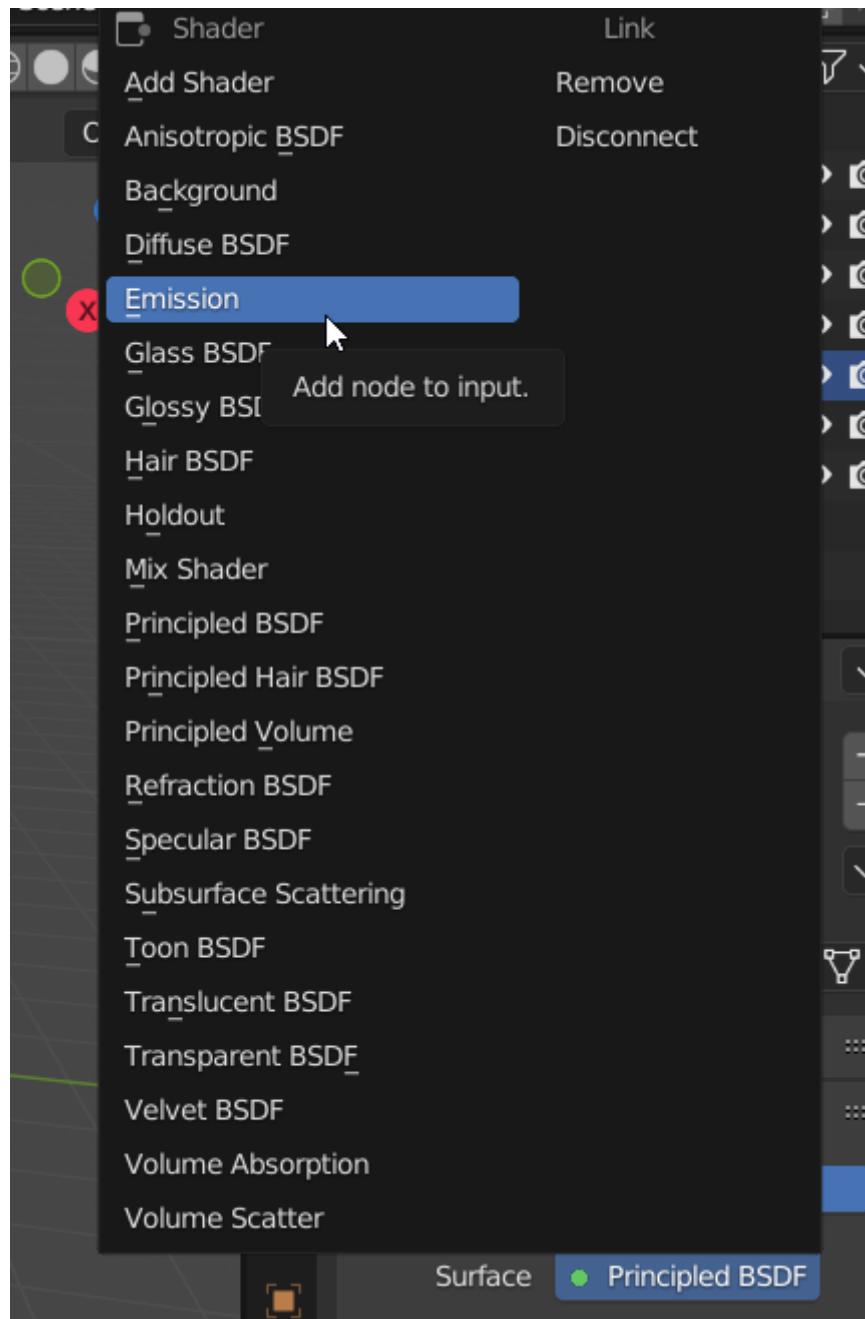
Selecione a aba "material"



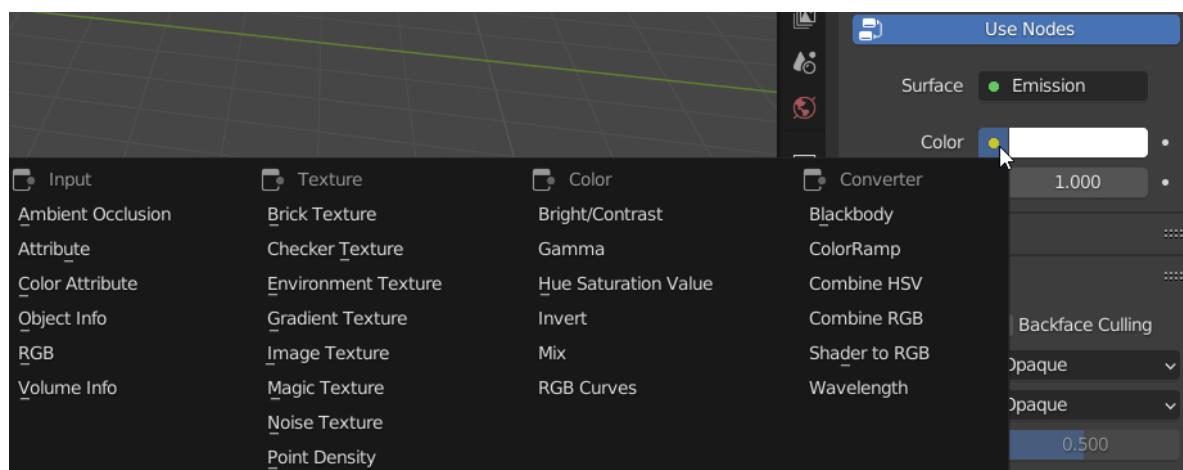
Escolha "novo" e renomeie para "vtxcolor" por exemplo



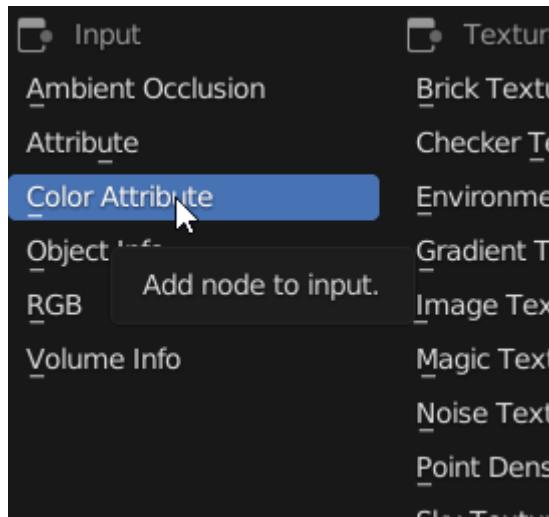
Troquei o "Principled BSDF" por "Emission", que é o que faz sentido para a "radiosidade", pois estamos calculando direto a radiância das superfícies.



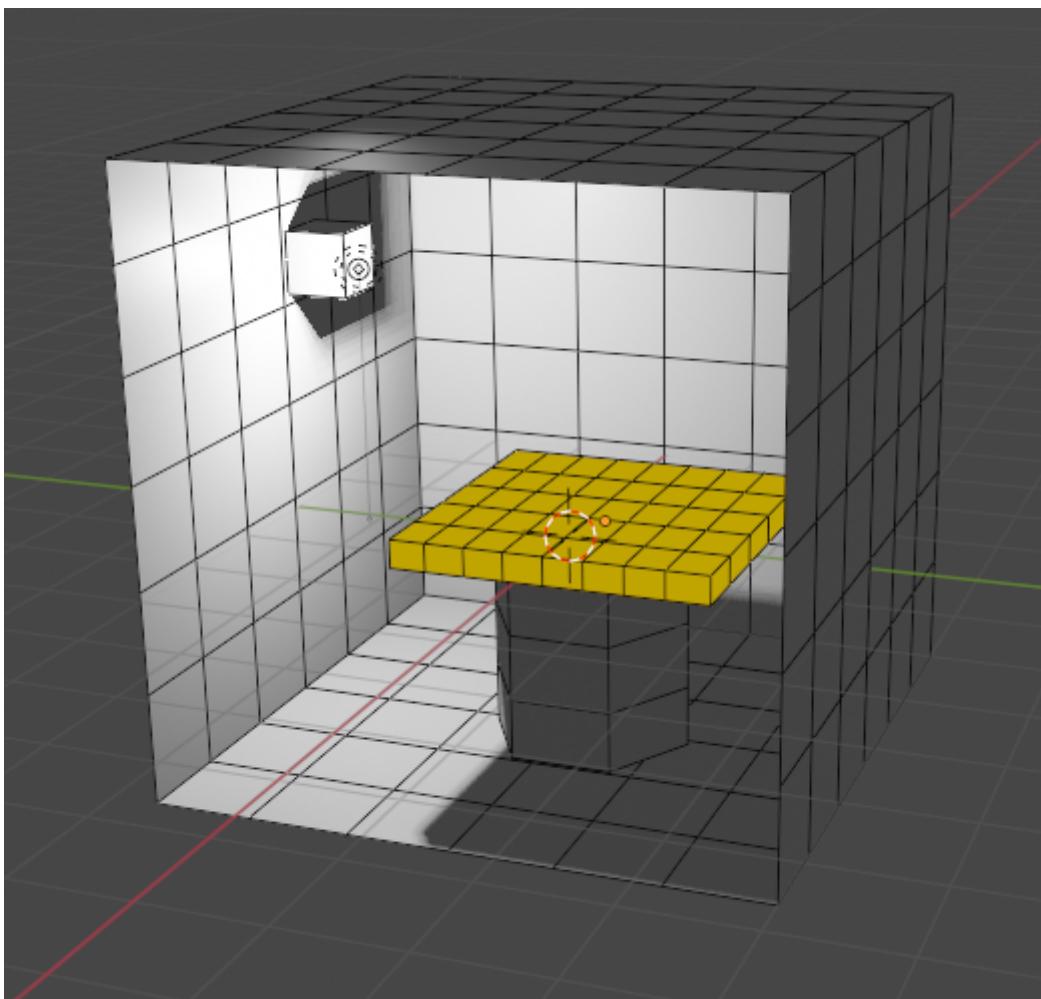
Clique na bolinha amarela do campo "Color"



Escolha "Color attribute"



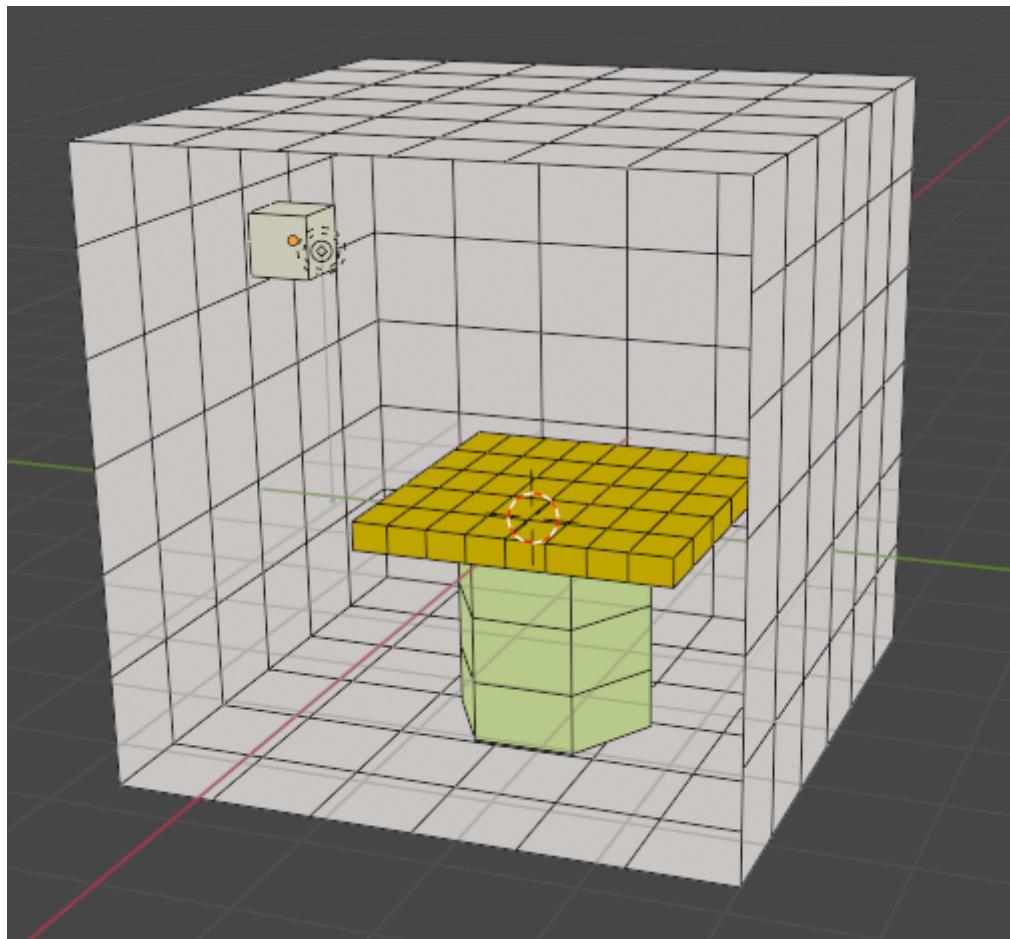
E pronto, temos um tampo de mesa emissivo de cor marrom.



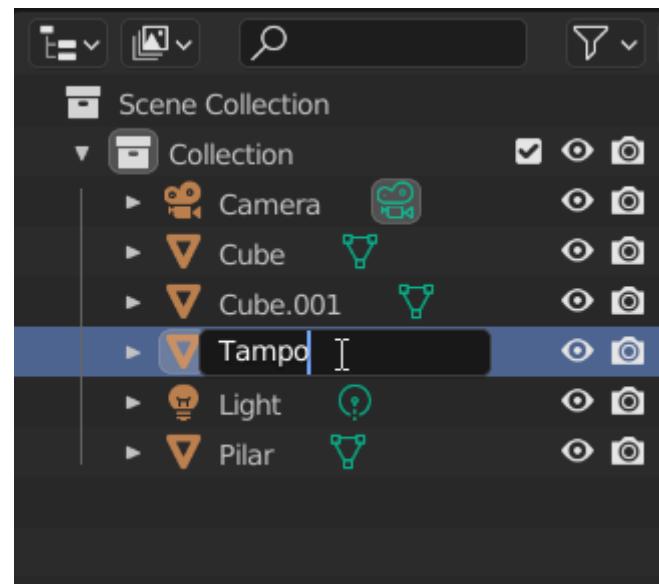
Completando para os outros objetos: (como já explicado)

Selecione o objeto. Aba "material", escolha "vtxcolor".

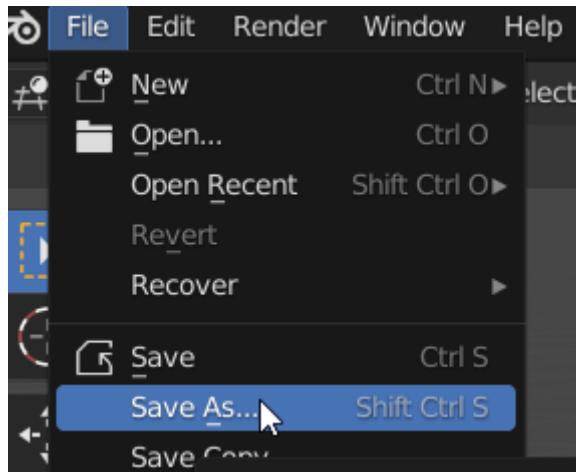
Aba "geometria". Crie o layer de informação de cor dos vértices.



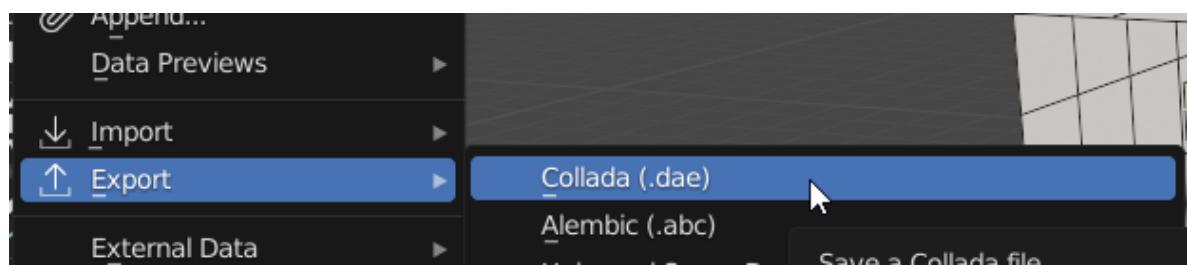
Vamos dar nomes decentes aos objetos. Nomes que nos façam sentido para selecionar rapidamente os objetos.



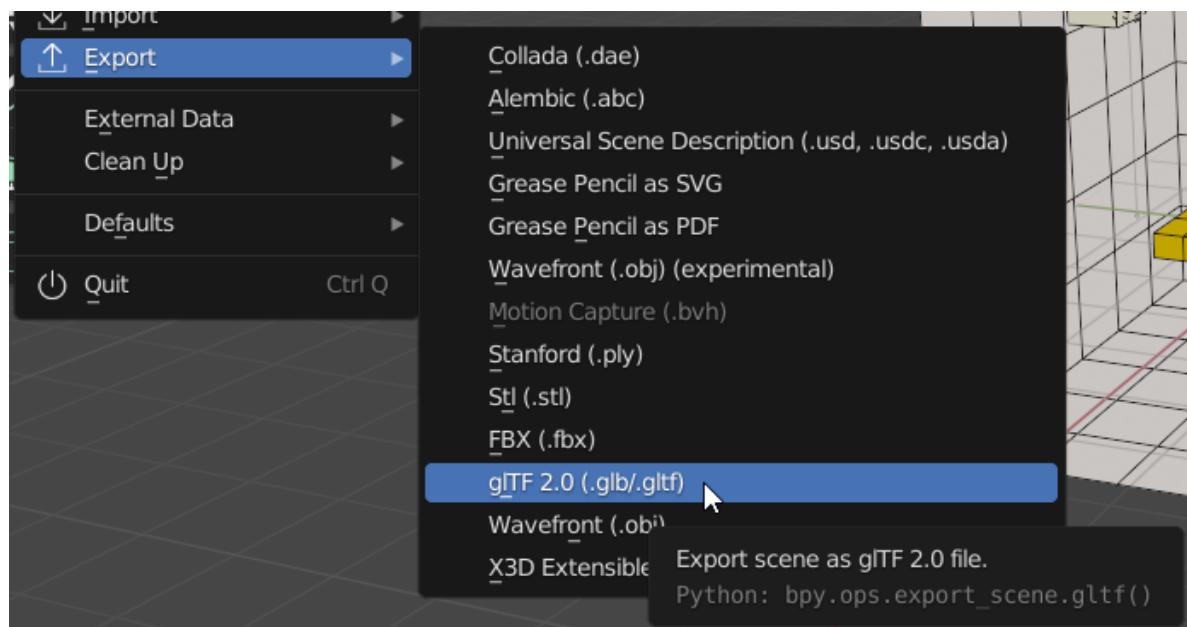
Salvar o ".blend"



Exportar para COLLADA

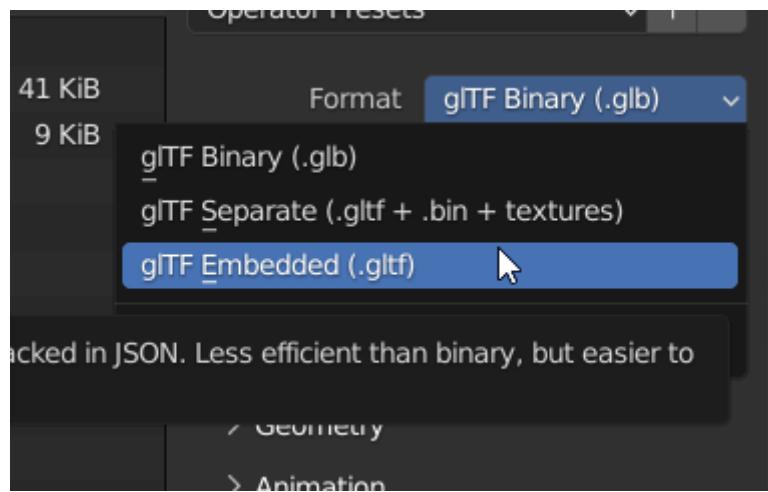


Exportar para GLTF

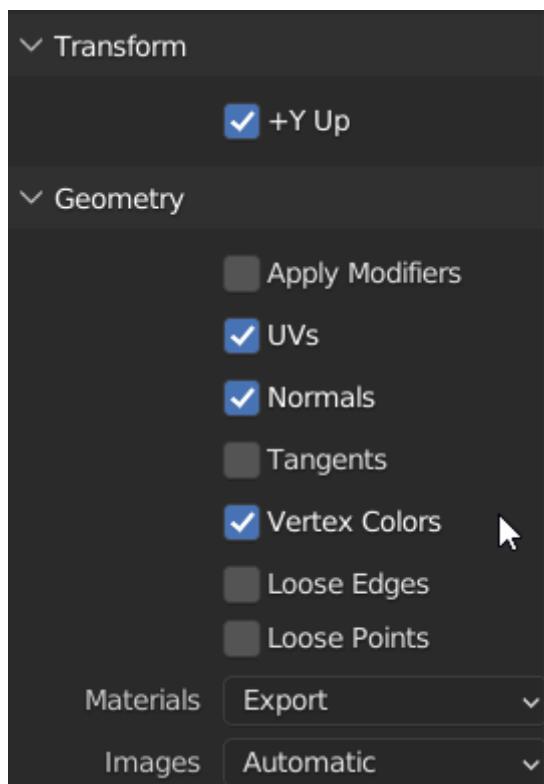


Aqui tomar cuidado com as opções:

Escolher o formato embedded (mais fácil para o JS, porque já vem embutido no JSON em baseURL) ou "separate" (mais fácil para o Python ou outras).



Faça questão de que venham as "vertex colors":

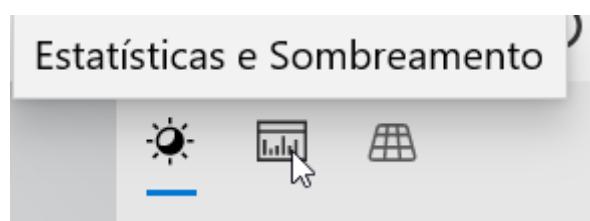


Pronto. Exportamos.

Para checar, faça "file->new", delete o cubo infeliz, "Import->Collada" e traga o arquivo exportado com extensão ".DAE". Algumas coisas mudaram, por exemplo, as faces são todas triângulos agora.

Para checar o "GLTF" no Windows, basta clicar duas vezes e abrir o visualizador 3D.

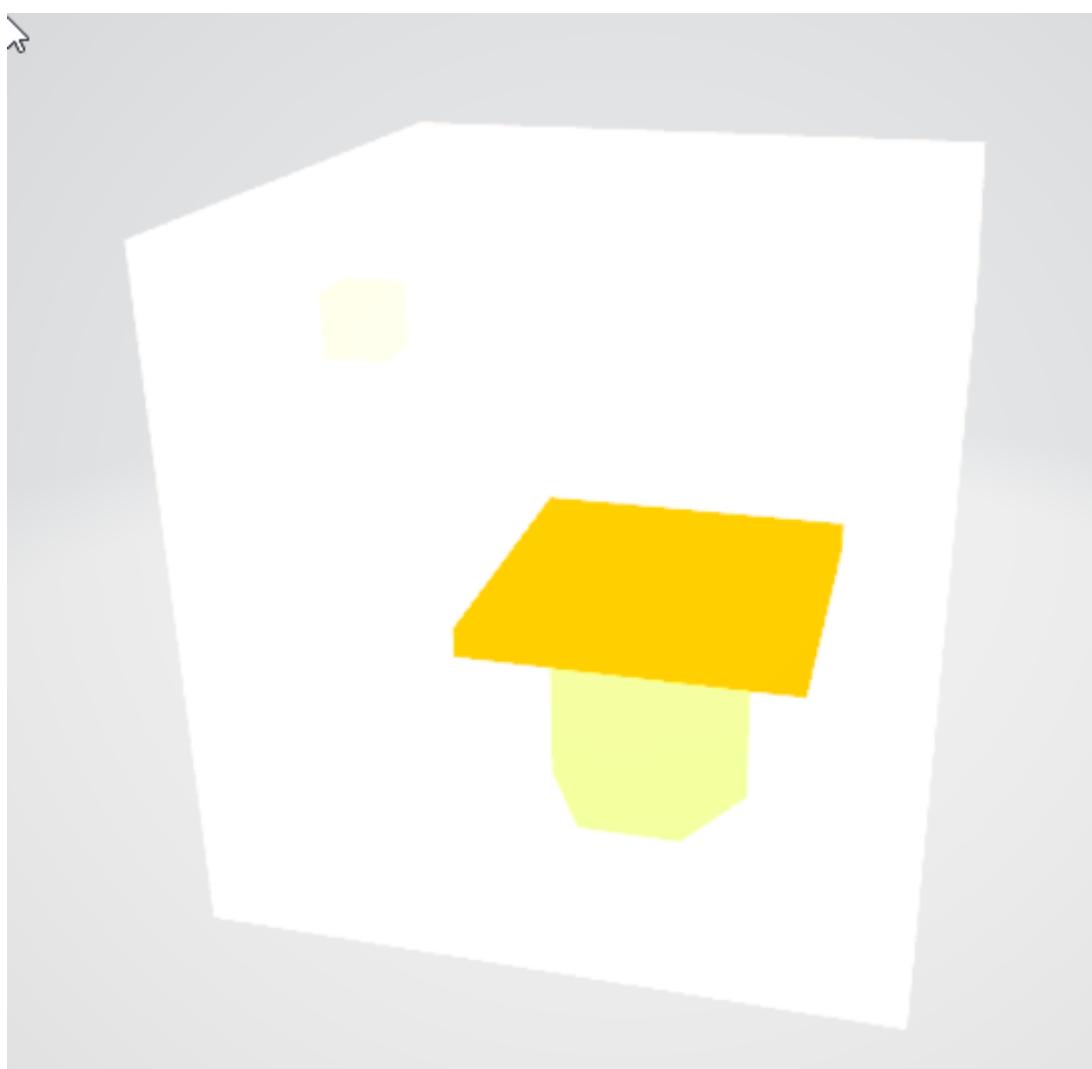
Escolha "Estatística e Sombreamento"



Escolha "Cores de vértices", tem que estar "SIM"

Dados de Malhas	
Triângulos	576
Vértices	509
Conjunto UV 0	Sim
Conjunto UV 1	Não
<input checked="" type="checkbox"/> Cores de Vértices	Sim
IDs de Material	1

E ai estamos:



Parte 2 - Extrair as informações dos arquivos

Vamos entender como funcionam os arquivos. Começando pelo COLLADA - arquivo .DAE.

O arquivo é um texto XML. Você deve encontrar os objetos nos campos "" com o nome que foi dado ao objeto.

```
<geometry id="Cube_002-mesh" name="Cube.002">
```

Em seguida aparecem os buffers de dados.

- As coordenadas de vértices:

```
<float_array id="Cube_002-mesh-positions-array" count="324">-1 -1 -1 -1 1 -1 1 -1 -1 1 1 1  
-1 -1 1 -1 1 1 1 -1 1 1 1 -1 0.7721158 -1 -1 0.5221158 -1 -1 0.2721157 -1 -1 0.0221157 -1 -1  
-0.2278842 -1 -1 -0.4778842 -1 -1 -0.7278842 -1 -1 -0.7278842 1 -1 -0.4778842 1 -1 -0.2278842  
1 -1 0.0221157 1 -1 0.2721157 1 -1 0.5221158 1 -1 0.7721158 1 1 -0.7278842 -1 1 -0.4778842 -1  
1 ..... -0.7278842 -1</float_array>
```

- as normais de vértice
- as coordenadas de textura
- as cores:

```
<source id="Cube_002-mesh-colors-Color" name="Color">  
    <float_array id="Cube_002-mesh-colors-Color-array" count="2544">0.8588235 0.6509804  
0 1 0.8588235 0.6509804 0 1 0.8588235 0.6509804 0 1 0.8588235  
0.6509804 0 1 0.8588235 0.6509804 0 1 ....
```

No próprio arquivo está explicado como funcionam as cores:

```
<accessor source="#Cube_002-mesh-colors-Color-array" count="636" stride="4">  
    <param name="R" type="float"/>  
    <param name="G" type="float"/>  
    <param name="B" type="float"/>  
    <param name="A" type="float"/>  
</accessor>
```

No meu caso aqui são 2544 valores correspondendo a 636 vetores de 4 números representando a cor.

Cada face é um triângulo e cada vértice de cada face tem um valor de cor associado. Um vértice pode estar em mais de uma face, mas pode ter um valor diferente para cada face. Assim as cores são atribuídas para cada canto de cada face. Então o número de cores é o número de faces vezes 3. No caso 212 faces.

Por fim, vem a definição dos triângulo (faces):

```
<triangles material="Vtxcolor-material" count="212">  
    <input semantic="VERTEX" source="#Cube_002-mesh-vertices" offset="0"/>  
    <input semantic="NORMAL" source="#Cube_002-mesh-normals" offset="1"/>  
    <input semantic="TEXCOORD" source="#Cube_002-mesh-map-0" offset="2" set="0"/>  
    <input semantic="COLOR" source="#Cube_002-mesh-colors-Color" offset="3" set="0"/>  
<p>21 0 0 2 0 1 1 8 0 2 2 43 1 3 3 6 1 4 4 36 1 5 5 35 2 6 6 4 2 7 7 22
```

Entre o `<p>` e o `</p>` está uma lista de números inteiros, que correspondem aos índices de cada objeto.

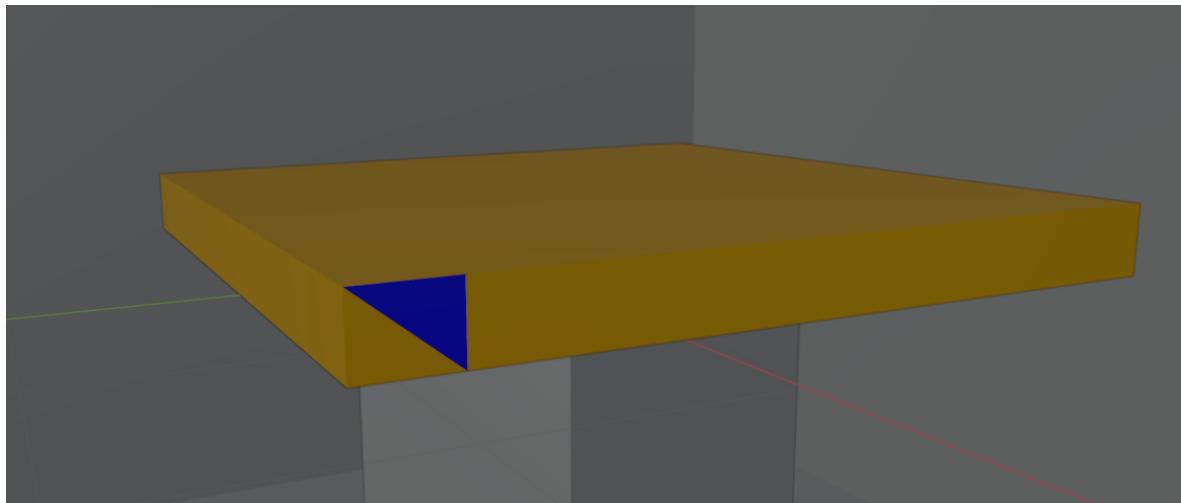
No próprio XML está explicado como ler. Cada um dos 212 triângulos possui 3 cantos. Cada canto possui um índice de vértice, um índice de normal, um índice de coordenada de textura e um índice de cor.

Assim, o primeiro triângulo contém os vértices 21, 2 e 8. Suas posições estão no buffer de positions descrito acima. Veja que as direções normais são 0, 0 e 0, ou seja, a mesma. Os índices de cor são "0", "1" e "2", ou seja, as três cores iniciais discutidas (vetores de 4 números).

Com isso, podemos pegar os valores das cores armazenados no modelo e colocar como coeficiente " ρ " de reflexão das faces.

Quando calcularmos as cores finais de cada face, podemos substituir nesse arquivo XML.

Por exemplo, após alterar no arquivo texto as três primeiras cores, eis o resultado no Blender:



Modificação feita no texto:

```
<float_array id="Cube_002-mesh-colors-Color-array" count="2544">0.08588235 0.06509804 0.9 1  
0.08588235 0.06509804 0.9 1 0.08588235 0.06509804 0.9 1 0.8588235 0.6509804 0 1 0.8588235  
0.6509804 0 1 0.8588235 0.6509804 0 1 0.8588235 0.6509804 0 1 0.8588235 0.6509804 0 1  
0.8588235 0.6509804 0 1 0.8588235 0.6509804 0 1
```

Troquei (0.85;0.65;0;1) por (0.08;0.06;0.9;1)

Para verificar a posição, a primeira face possui índices de vértice (21, 2, 8). Vou modificar o vértice 2, que é o terceiro trio de valores:

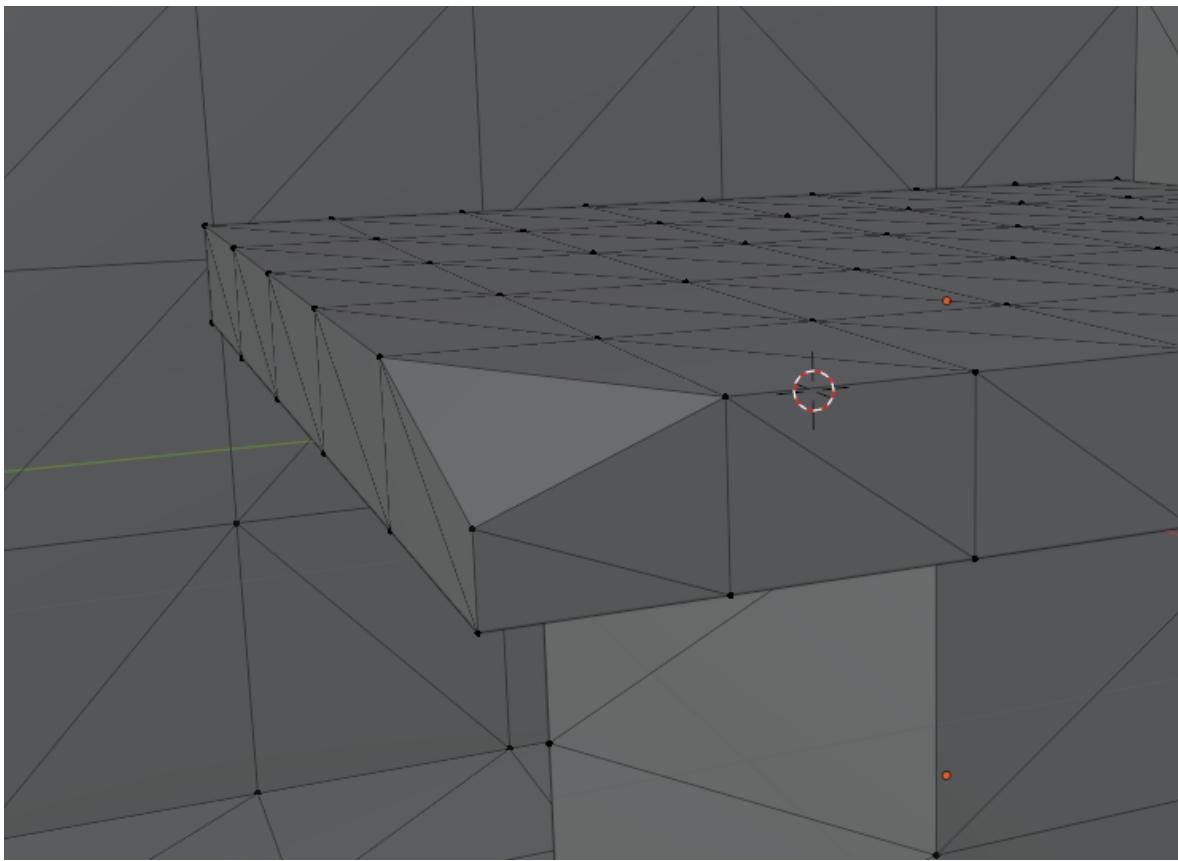
original:

```
<float_array id="Cube_002-mesh-positions-array" count="324">-1 -1 -1 -1 -1 1 -1 1 -1 -1 1 1 1  
-1 -1 1 -1 1 1 1 ....
```

alterado o valor de "Z" do terceiro vértice (vértice índice 2):

```
-1 -1 -1 -1 -1 1 -1 1 -1 0
```

Resultado:



Reposicionamos o vértice no eixo Z.

Parte 3 - Computar a Radiosidade e substituir as cores nos arquivos

Em posse dos arquivos de exportação, precisamos montar o sistema para resolver a Radiosidade.

Precisamos extrair dos arquivos:

- coordenadas dos três vértices de cada face
- cor dos três vértices de cada face (para compor o ρ da face, que pode ser por exemplo a média da cor dos vértices)
- calcular a normal da face a partir das coordenadas dos três vértices (as normais que estão nos arquivos são normais de vértice, ou seja, a média das faces do vértice, mas queremos normais da face)
- calcular a área de cada face (pelas coordenadas dos vértices)
- calcular o centróide de cada face

Precisamos também selecionar quais faces vão atuar como fontes de luz. No caso, colocamos $E_i = 1$ por exemplo.

Para calcular os fatores de forma F_{ij} (simplificadamente/ aproximadamente)

- checar a visibilidade do par de faces i e j : isto consiste em traçar um segmento de reta entre os centróides das faces e verificar se este segmento intercepta uma terceira face. Se positivo, o fator de forma é zero.
- checar também as direções das normais para ver se as faces se enxergam. O ângulo entre a normal e o raio incidente deve ser maior que 90°.

- Caso contrário, utilizar uma fórmula para cálculo aproximado do fator de forma a partir dos ângulos, áreas e distância.

Completar as matrizes do sistema de equações da radiosidade:

- completar com as informações de ρ_i , E_i , F_{ij}

Finalmente, resolver o sistema da radiosidade.

Se necessário, normalizar as cores obtidas.

Substituir as cores obtidas, substituindo os valores do arquivo DAE (grave em outro arquivo o resultado para não perder o original).