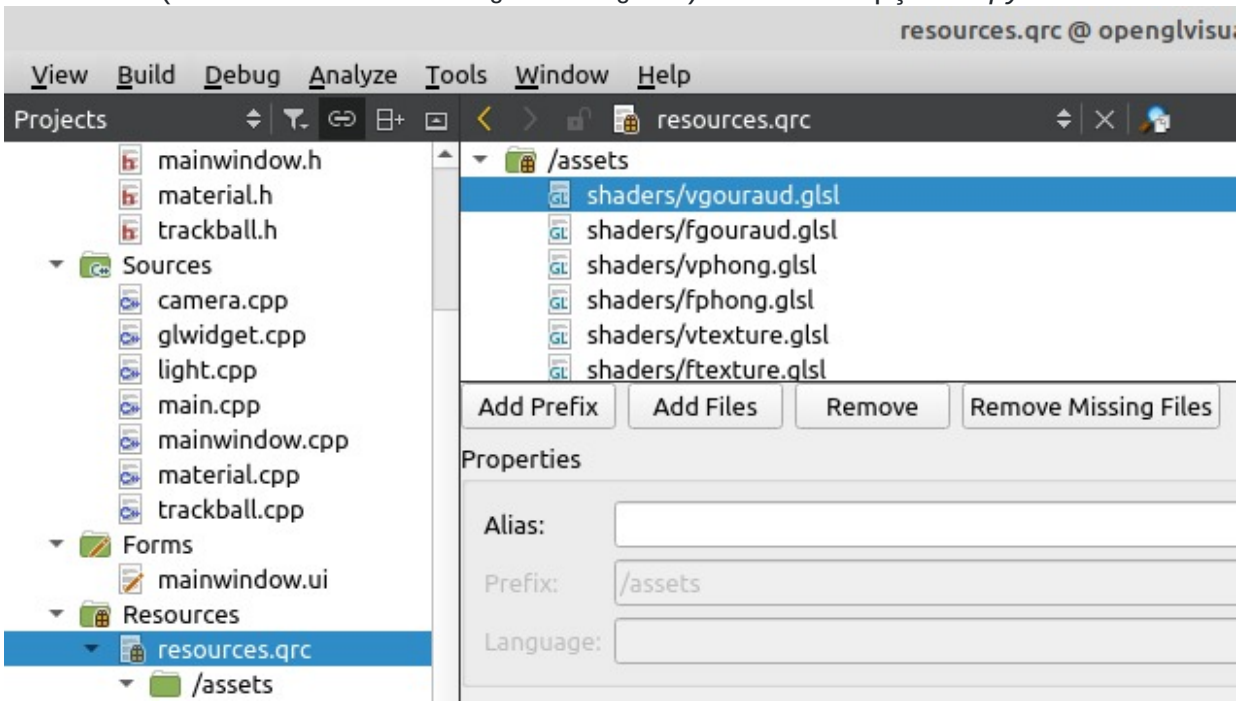
# Problemas encontrados

Usando Qt Creator 4.13.3, Qt 5.15.2 e seguindo o artigo *Interactive Graphics Applications with OpenGL Shading Language and Qt* encontramos problemas em alguns pontos:

- Antes de usar a versão 5.15.2 do Qt, tentamos usar a versão 5.9 que era a versão padrão ao instalar pelo Linux com `sudo apt-get install -y qt-default` , porém tivemos problemas com essa versão em relação ao OpenGL. Então, para outros que venham a fazer esse trabalho, recomendo que usem o instalador online disponibilizado no site oficial do Qt.
- Não é possível selecionar trechos inteiros de código no artigo pois ao selecionar um trecho você acaba pegando outros que não estão relacionados ao código.
- Foi necessário incluir os módulos opengl e widgets no projeto. No final ficou dessa forma: `QT += core gui opengl widgets`
- A função qMax na versão 5 do Qt aparentemente não aceita pegar o maior de dois valores do tipo double. Dessa forma, foi necessário trocar algumas variáveis para float no código.

- Tivemos problemas também com o *resource path* de alguns *shaders* e texturas por não entender bem como fica o caminho de um recurso após adicioná-lo ao projeto. Por isso, para não ter dúvidas sobre o caminho de um recurso, recomendo clicar com o direito no recurso (nesse caso `shaders/vgouraud.glsl`) e clicar na opção *Copy Resource Path to Clipboard*.




- O código feito no artigo pressupõe que o objeto do arquivo .off tem faces triangulares. Dessa forma, foi necessário adaptar alguns arquivos .off quebrando faces quadradas em triângulos. Para isso, criamos um *script* Python que faz essa conversão:

```
"""
This script reads an .off file with faces that have more than 3 vertices
and breaks them into multiple triangules creating new triangular faces.
As an example, a square would be transformed into 4 triangules.
ABC, ABD, ACD, BCD
A          B
------------
| ..    .. |
|  ..  ..  |
|   ....   |
|   ....   |
|  ..  ..  |
```

```
| ..    .. |
|..      ..|
------------
C          D
USAGE: python3 break-faces-into-triangules.py < cone.off > cone-tri.off
"""

import sys

from itertools import combinations

numVertices = 0
numFaces = 0
numEdges = 0

lines = sys.stdin.readlines()
new_lines = [line for line in lines if len(line.strip()) != 0]
lines = new_lines
i = 0

line = lines[i].strip()
i += 1

while line[0] == '#':
    line = lines[i].strip()
    i += 1

if '#' in line:
    hastag_index = line.index('#')
    line = line[:hastag_index]

# OFF header
print(line)
line = lines[i].strip()
i += 1

while line[0] == '#':
    line = lines[i].strip()
```

```
    i += 1

if '#' in line:
    hastag_index = line.index('#')
    line = line[:hastag_index]

# Num. of vertices, faces and edges
lineSplit = line.split()
numVertices = int(lineSplit[0])
numFaces = int(lineSplit[1])
numEdges = int(lineSplit[2])

new_lines = []
newNumFaces = 0
newNumEdges = 0

line = lines[i].strip()
i += 1

# All vertices
for j in range(numVertices):
    while line[0] == '#':
        line = lines[i].strip()
        i += 1

    if '#' in line:
        hastag_index = line.index('#')
        line = line[:hastag_index]

    lineSplit = line.split()
    vertexXYZ = [lineSplit[0], lineSplit[1], lineSplit[2]]
    new_lines.append(' '.join(vertexXYZ))

    line = lines[i].strip()
    i += 1

# All faces
for j in range(numFaces):
```

```
        while line[0] == '#':
            line = lines[i].strip()
            i += 1

        if '#' in line:
            hastag_index = line.index('#')
            line = line[:hastag_index]

        lineSplit = line.split()
        numVerticesForTheFace = int(lineSplit[0])
        faceVertices = []

        for k in range(numVerticesForTheFace):
            faceVertices.append(lineSplit[k + 1])

        # Divide the polygon face into triangules
        triangulesVertices = combinations(faceVertices, 3)

        for trianguleVertices in triangulesVertices:
            newNumFaces += 1
            newNumEdges += 3
            new_lines.append(f'3 {" ".join(trianguleVertices)}')

        if j < numFaces - 1:
            line = lines[i].strip()
            i += 1

    print(f'{numVertices} {newNumFaces} {newNumEdges}')
    for line in new_lines:
        print(line)
```

- Dado o problema de não poder selecionar trechos inteiros de código, acabou que enfrentamos problemas com a textura simplesmente por ter digitado errado alguns trechos que estavam no artigo.

Para fazer a textura e algumas outras coisas funcionar no Windows, foi necessário alterar alguns trechos de código:

1º

```
  glViewport(0, 0, width , height);
```

para

```
  QOpenGLFunctions glFuncs(QOpenGLContext::currentContext());
  glFuncs.glViewport(0, 0, width , height);
```

2º

```
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glActiveTexture ( GL_TEXTURE0 );
  glBindTexture ( GL_TEXTURE_2D , texID [0]) ;
  glActiveTexture ( GL_TEXTURE1 );
  glBindTexture ( GL_TEXTURE_2D , texID [1]) ;
  glDrawElements(GL_TRIANGLES , numFaces * 3, GL_UNSIGNED_INT , 0);
```

para

```
  QOpenGLFunctions glFuncs(QOpenGLContext::currentContext());
  glFuncs.glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glFuncs.glActiveTexture ( GL_TEXTURE0 );
  glFuncs.glBindTexture ( GL_TEXTURE_2D , texID [0]) ;
  glFuncs.glActiveTexture ( GL_TEXTURE1 );
  glFuncs.glBindTexture ( GL_TEXTURE_2D , texID [1]) ;
  glFuncs.glDrawElements(GL_TRIANGLES , numFaces * 3, GL_UNSIGNED_INT , 0);
```

3º

```
  glClearColor(1, 1, 1, 1);
  glClearColor(0, 0, 0, 1);
```

para

```
QOpenGLFunctions glFuncs(QOpenGLContext::currentContext());
glFuncs.glClearColor(1, 1, 1, 1);
glFuncs.glClearColor(0, 0, 0, 1);
```

4º

```
glEnable ( GL_DEPTH_TEST );
glActiveTexture ( GL_TEXTURE0 );
glActiveTexture ( GL_TEXTURE1 );
```

para

```
QOpenGLFunctions glFuncs(QOpenGLContext::currentContext());
glFuncs.glEnable ( GL_DEPTH_TEST );
glFuncs.glActiveTexture ( GL_TEXTURE0 );
glFuncs.glActiveTexture ( GL_TEXTURE1 );
```