

**UNIVERSIDADE FEDERAL DA PARAÍBA**

**CENTRO DE INFORMÁTICA**

**CIÊNCIA DA COMPUTAÇÃO**

**Gabriel Leite Santana – 2016000284**

**Drayton Corrêa Filho – 2016058088**

**Relatório – Trabalho Preparatório I**

**João Pessoa**

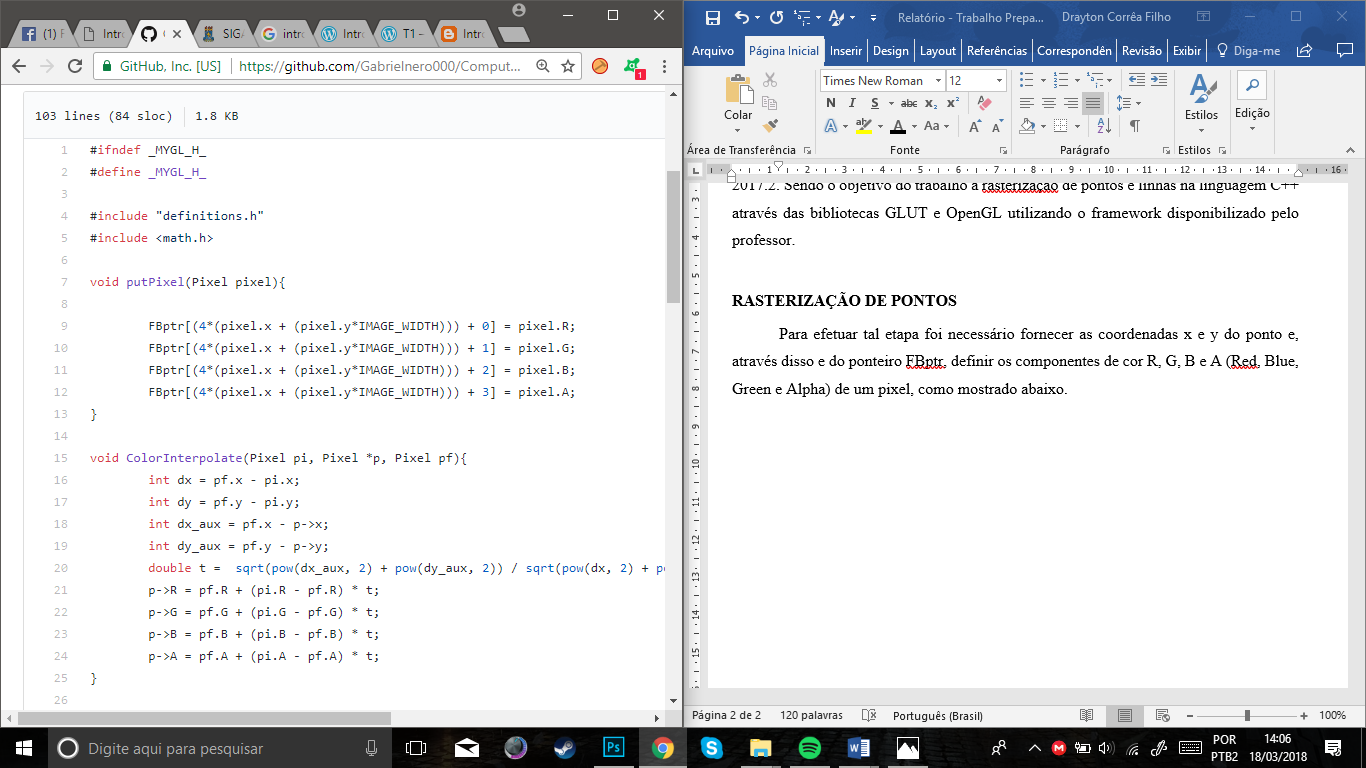
**2018**

**INTRODUÇÃO**

Esse relatório faz parte do primeiro trabalho preparatório da disciplina Introdução à Computação Gráfica, ministrada pelo professor Christian Azambuja Pagot, no período 2017.2. Sendo o objetivo do trabalho a rasterização de pontos e linhas na linguagem C++ através das bibliotecas GLUT e OpenGL utilizando o framework disponibilizado pelo próprio professor.

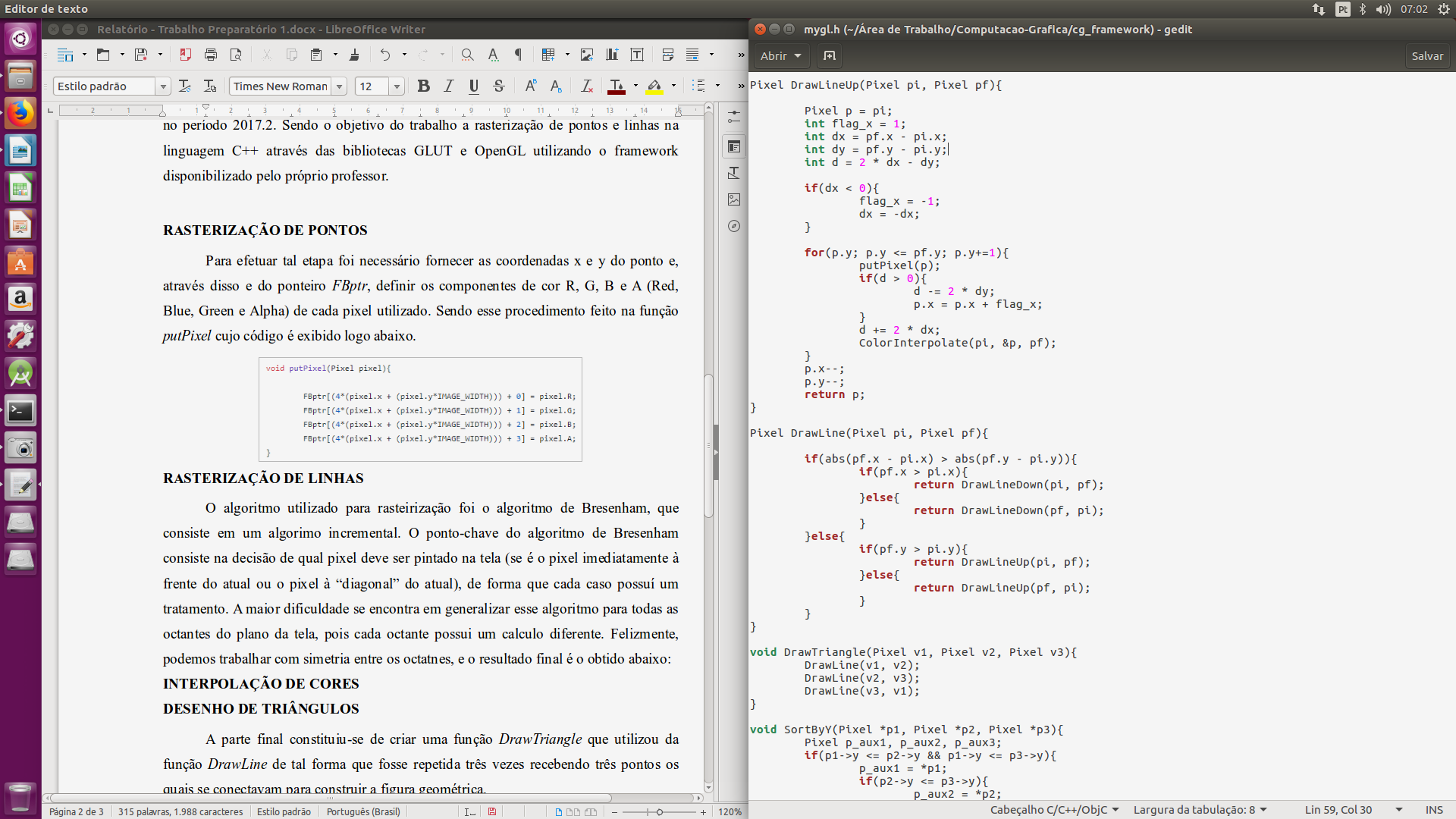
**RASTERIZAÇÃO DE PONTOS**

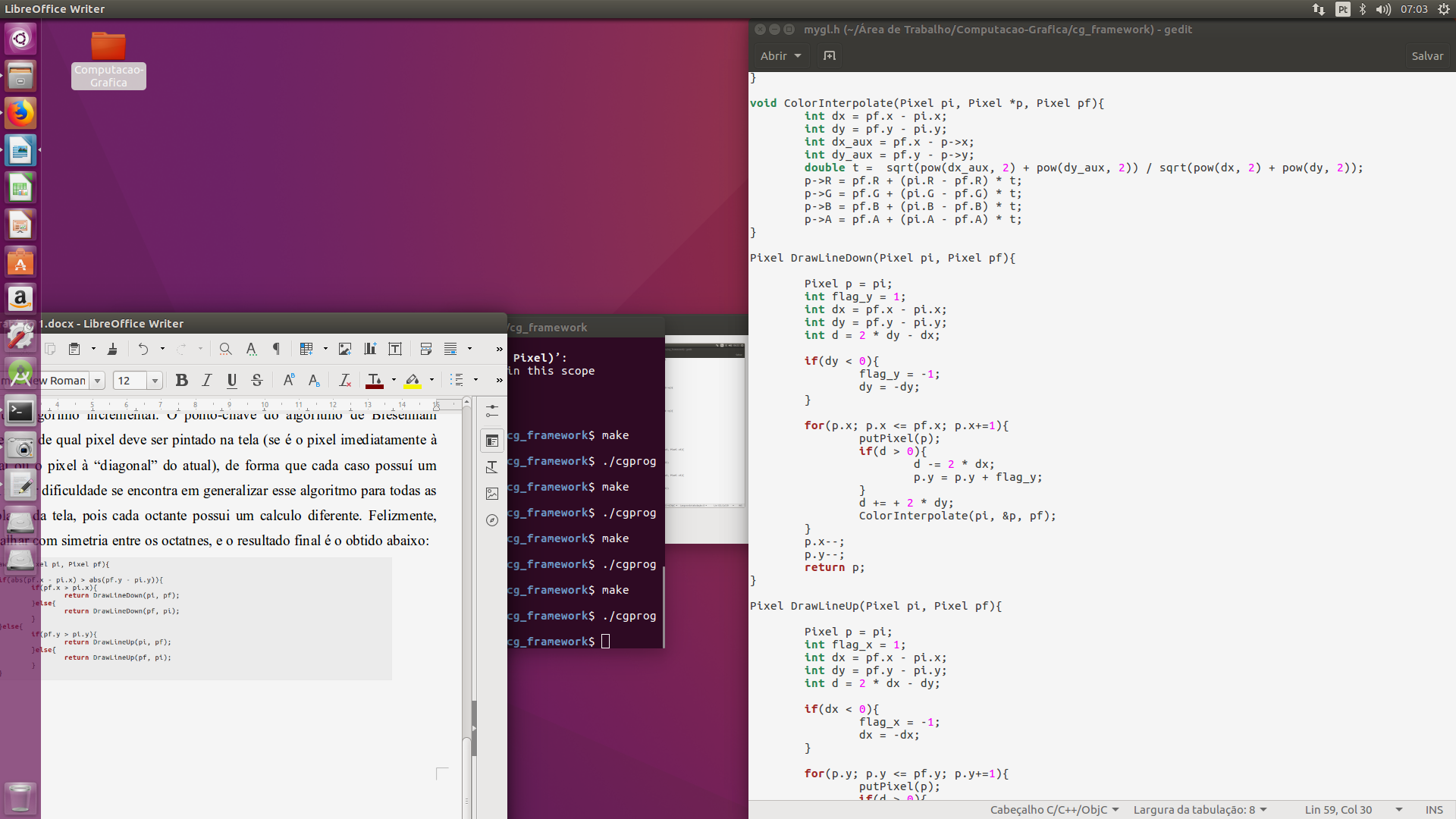
Para efetuar tal etapa foi necessário fornecer as coordenadas x e y do ponto e, através disso e do ponteiro *FBptr*, definir os componentes de cor R, G, B e A (Red, Blue, Green e Alpha) de cada pixel utilizado. Sendo esse procedimento feito na função *putPixel* cujo código é exibido logo abaixo.

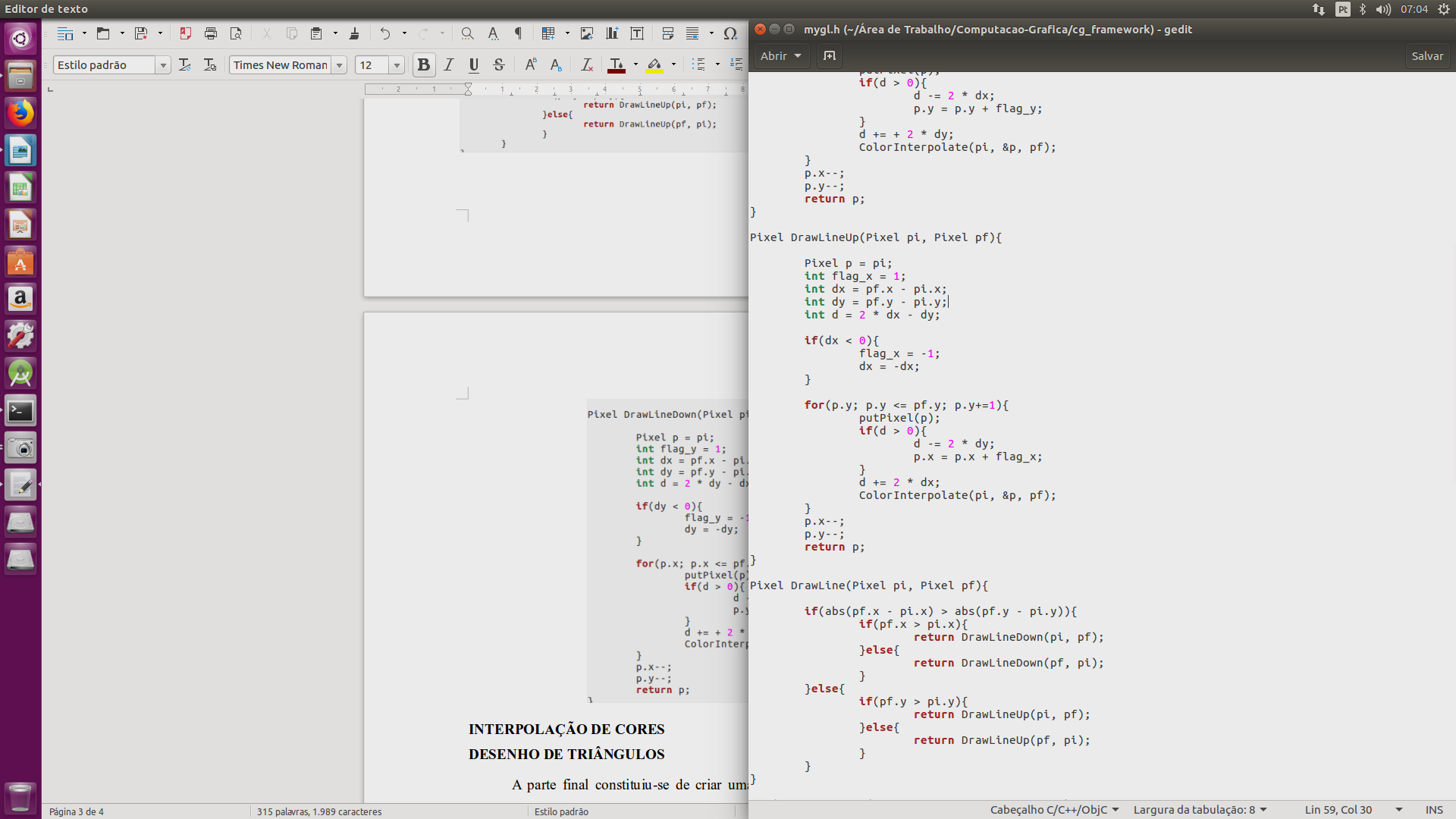


**RASTERIZAÇÃO DE LINHAS**

O algoritmo utilizado para rasteirização foi o algoritmo de Bresenham, que consiste em um algorimo incremental. O ponto-chave do algoritmo de Bresenham consiste na decisão de qual pixel deve ser pintado na tela (se é o pixel imediatamente à frente do atual ou o pixel à “diagonal” do atual), de forma que cada caso possuí um tratamento. A maior dificuldade se encontra em generalizar esse algoritmo para todas as octantes do plano da tela, pois cada octante possui um calculo diferente. Felizmente, podemos trabalhar com simetria entre os octatnes, e o resultado final é o obtido abaixo:

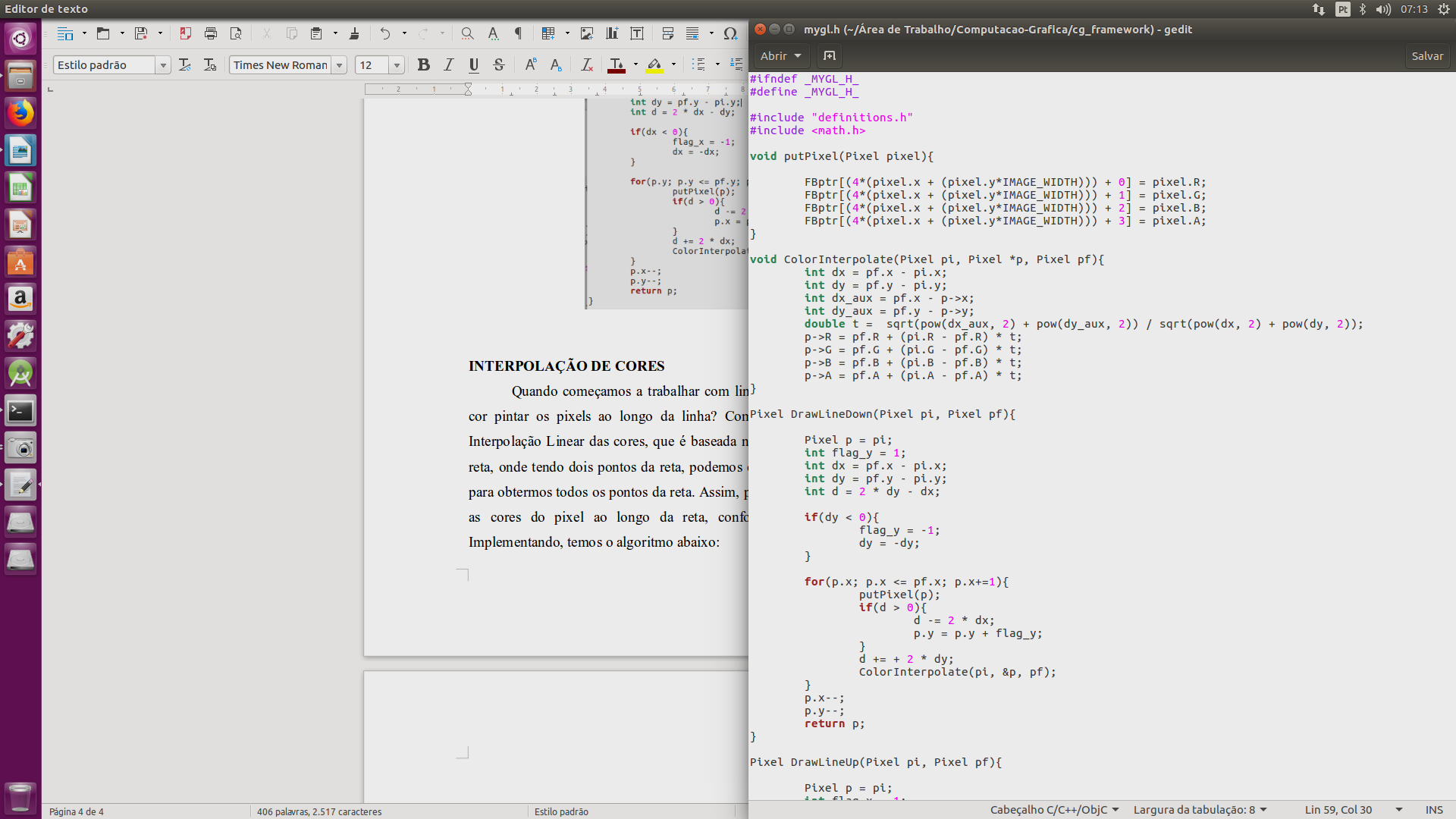






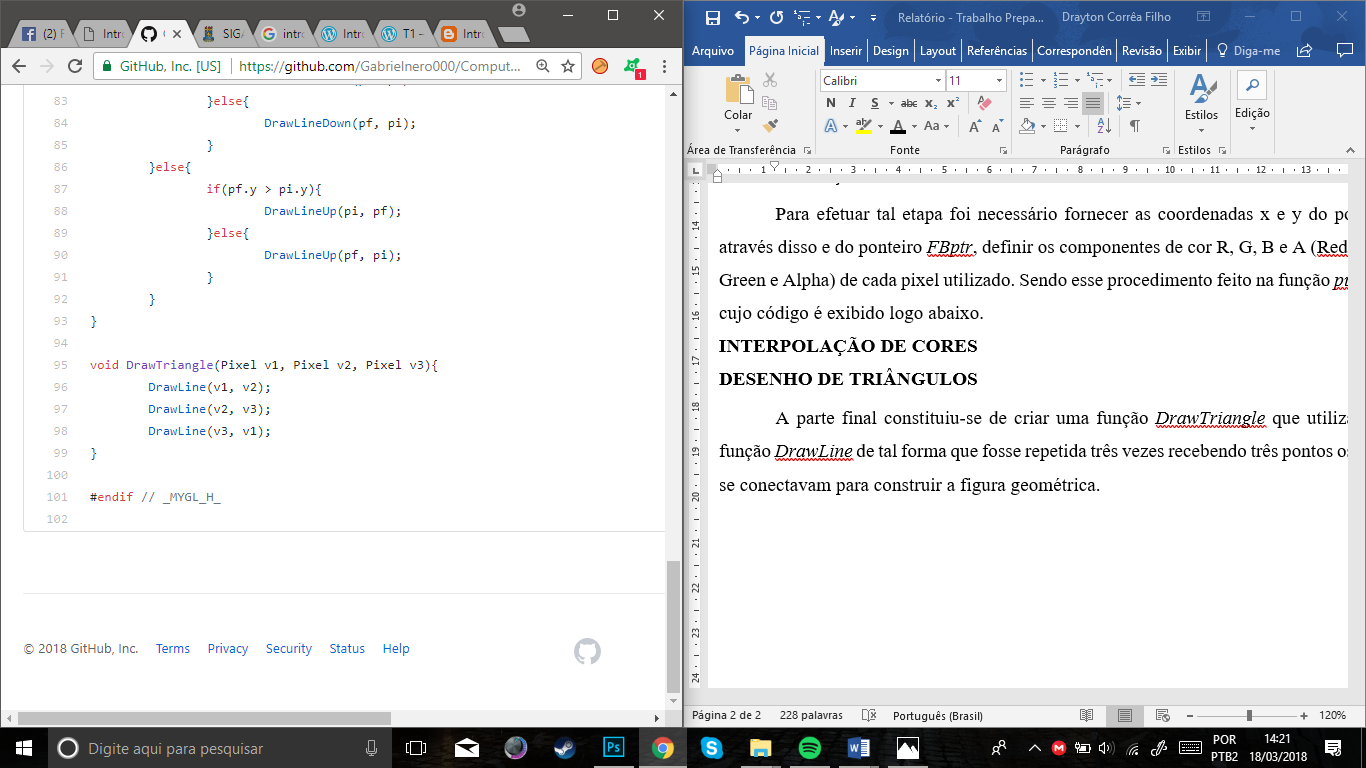
**INTERPOLAÇÃO DE CORES**

Quando começamos a trabalhar com linhas, esbarramos num problema: de que cor pintar os pixels ao longo da linha? Como solução para isso, podemos usar a Interpolação Linear das cores, que é baseada no principio da equação vetorial de uma reta, onde tendo dois pontos da reta, podemos obter um parâmetro t, onde 0 <= t <= 1 para obtermos todos os pontos da reta. Assim, podemos usar esse parâmetro para variar as cores do pixel ao longo da reta, conforme a distância dos pontos iniciais. Implementando, temos o algoritmo abaixo:



**DESENHO DE TRIÂNGULOS**

A parte final constituiu-se de criar uma função *DrawTriangle* que utilizou da função *DrawLine* de tal forma que fosse repetida três vezes recebendo três pontos os quais se conectavam para construir a figura geométrica.



Por fim, para exibir a figura foram escolhidos os pontos (50, 50), (500, 25), (300, 500) e suas respectivas cores foram a vermelha, branca e azul. Sua geração é feita na *main* através da função *MyGLDraw* cujo resultado efetivo pode ser visto abaixo.

