Mini EP 4: Brincando com a GPU.

Alfredo Goldman, Elisa Silva e Luciana Marques

MAC 0219-5742 – Programação Concorrente e Paralela 2021

Entrega até 17 de maio de 2021

1. Introdução

Neste mini EP vamos implementar um programa que gera a imagem do conjunto mandelbrot de duas formas diferentes. A primeira será usando JavaScript e a API Canvas para gerar a imagem de forma sequencial e a segunda será usando GLSL e WebGL para gerar a mesma imagem porém explorando o paralelismo da GPU.

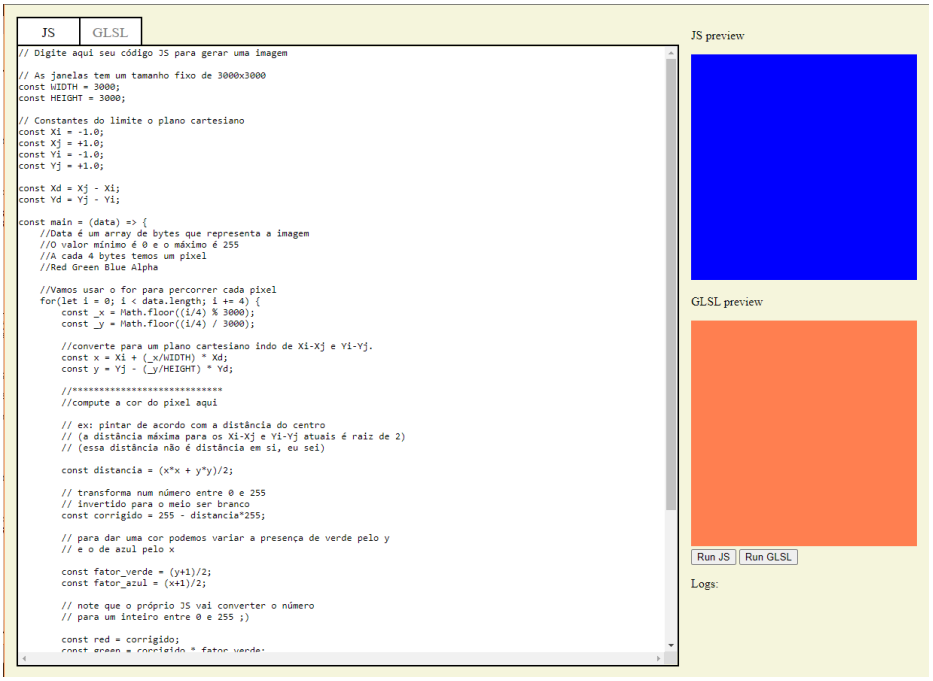
Como ambas as versões usam tecnologias disponíveis em navegadores modernos, desenvolvemos um site para facilitar o desenvolvimento e execução de ambos os programas.

GLSL é uma linguagem de programação baseada em C usada para escrever *shaders*. Este são programas que a GPU executa dentro da pipeline de OpenGL para as mais diversas funções (definir sombra, deformar objetos em 3D, aplicar texturas e afins) e vamos explorar o Fragment Shader que é executado para computar a cor de cada pixel.

2. Tarefas

Acesse http://blog.silva.moe/toys/shadermini.html.

Você deverá observar a página desse jeito:



Utilize as abas no canto superior esquerdo para alternar entre os editores de JS e GLSL e os botões “Run JS” e “Run GLSL” para executar as versões desejadas.

Ao executar qualquer uma das versões, será inserido no canto inferior direito um valor estimado do tempo de execução. Devido a certas restrições de como as APIs do WebGL funcionam, não é possível estimar com precisão o tempo de execução, bem como a variância pode ser muito grande devido a diversas razões técnicas de como o WebGL funciona.

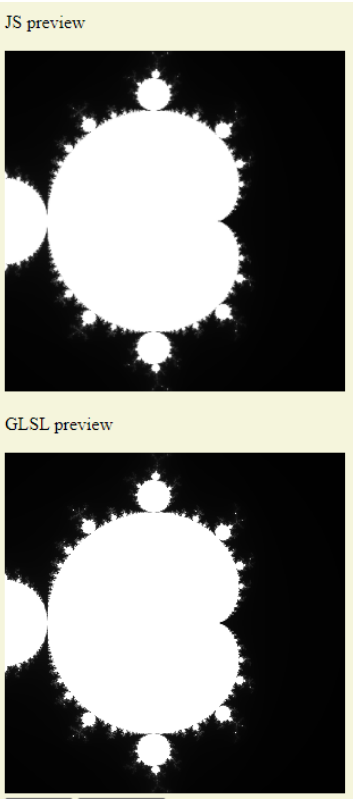
Ao executar os códigos já preenchidos você deve observar nas prévias algo similar ao apresentado na imagem à direita.

Você deve implementar o seguinte algoritmo para gerar o conjunto Mandelbrot em JS e GLSL:

```
Para cada ponto(x, y) {
  zx = 0; zy = 0; steps = 0;
  enquanto steps < 255 {
    zx_novo = zx*zx - zy*zy + x;
    zy_novo = 2*zx*zy + y;
    zx = zx_novo;
    zy = zy_novo;
    se (zx*zx + zy*zx) > 4 {
      break;
    }
    steps = steps + 1;
  }
  define cor com base no steps
}
```

Procure a região demarcada por "/********...". Nela já estão definidas as variáveis x e y bem como mostra como definir a cor do pixel em cada linguagem.

Ao final você deve observar nas prévias, uma imagem similar a imagem abaixo:



2.1 Detalhes para a implementação em JS

JS não exige a declaração do tipo da variável. Para declarar a variável você pode usar `let foo = valor;`. Ele também não faz muitas distinções entre float a int. Note que o código incluso na página já usa a variável `i` para percorrer os pixels, então cuidado para não alterar seu valor sem perceber.

Os valores possíveis para cada canal de cor varia de 0 a 255, sendo 0 a intensidade mínima daquela cor e 255 a máxima. O critério mais fácil para definir a cor é definir o valor de cada canal (red, green e blue) como o valor da variável steps. Isso vai gerar uma imagem com os pontos que fazem parte do conjunto na cor branca e preta para os pontos que não fazem parte.

2.2 Detalhes para a implementação em GLSL

GLSL já é uma linguagem similar a C, deste modo exige a declaração do tipo antes do nome da variável, como `float a = 1.0; int b = 0;`. GLSL é bem pedante quanto a conversões e constantes numéricas, por isso é bem comum que na hora de executar

ele reclame de que certa operação não foi definida entre ints e floats. Um exemplo é a linha `zy_novo = 2*zx*zy + y;`, em GLSL é preciso escrever `2.0` para o programa rodar corretamente. Conversões são feitas como em C++: `int a = 5; float b = float(a);`.

Diferente da versão em JS, o valor de cada canal espera um float variando de 0.0 a 1.0. Neste caso é só usar o valor da variável steps dividida por 255. Outra diferença é que não existe um bloco for para iterar pelos pixels uma vez que a GPU e o seu driver vão cuidar disso pra gente.

2.3 Alterando a região renderizada e dicas para definir as cores

A região renderizada é definida pelo valor das constantes Xi, Xj, Yi e Yj definidas no início dos programas. Note que para a versão em GLSL elas precisam do ponto decimal para serem tratadas como um float.

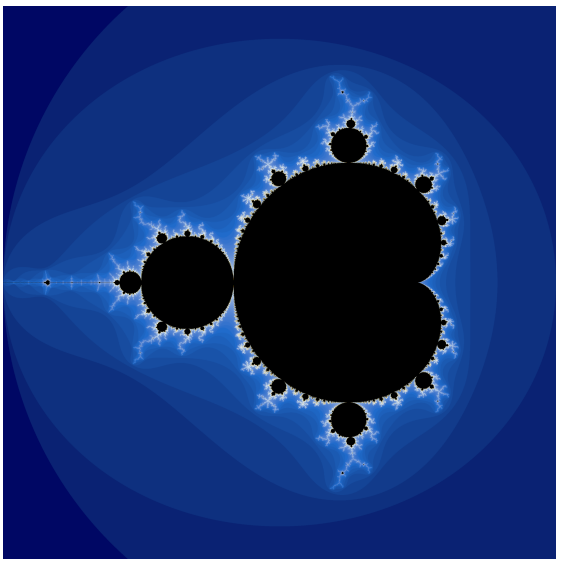
A cor pode ser definida com uma tabela de cores com base nos steps usando interpolação. Considere esta tabela de cores para a versão em JS:

Steps	**Red**	**Green**	**Blue**
0	0	7	100
10	32	107	203
80	237	255	255
150	255	170	0
200	100	120	0
255	0	0	0

A interpolação pode usar uma relação quadrática ao invés de uma linear, exemplo:

```
Interpola(A, B, porcentagem) {
    return sqrt(A*A*(1-porcentagem) + B*B*(porcentagem));
}
```

Se definir os valores de Xi, Xj, Yi e Yj para -2.0, +1.0, -1.5 e +1.5 respectivamente você obterá a imagem clássica do conjunto de Mandelbrot:



3. Entrega

Quando aparecer o termo NUSP no código, troque ele pelo seu número USP.

Salve o seu código JS em um arquivo com o nome "NUSP.js" e o seu código GLSL em um arquivo com o nome "NUSP.glsl". Comprima os dois arquivos em um arquivo .ZIP com o nome "NUSP.zip" e envie-o pelo eDisciplinas da matéria.

Você também deve preencher um formulário com o seu feedback sobre a atividade: https://forms.gle/JuUZVegf9fvzWMXy5. Não iremos exigir que se identifique.

Entrega até 17 de maio de 2021.

4. Critério de Avaliação

Os Mini EPs usam um critério de avaliação binária (ou 1 ou 0). Para tirar 1 envie o arquivo ZIP no eDisciplinas conforme especificado.

Vale reforçar parágrafo II do artigo 23 do **Código de Ética da USP**:

> Artigo 23 - É vedado aos membros do corpo discente e demais alunos da Universidade:
> [...]
> II. lançar mão de meios e artifícios que possam fraudar a avaliação do desempenho, seu ou de outrem, em atividades acadêmicas, culturais, artísticas, desportivas e sociais, no âmbito da Universidade, e acobertar a eventual utilização desses meios.

Mini EPs plagiados receberão nota 0.

Se tiver dúvidas, envie uma mensagem no fórum do curso ou envie e-mails para elisa@silva.moe, lucianadacostamarques@gmail.com ou gold@ime.usp.br com *[miniEP4]* no assunto do e-mail. Divirta-se!

5. Extras

Para saber mais sobre o conjunto de Mandelbrot, recomendamos este vídeo: https://www.youtube.com/watch?v=FFftmWSzgmk.

O editor é bem simples e ao recarregar a página, seu código será perdido. Recomendamos usar outro editor de texto para escrever o código em si e apenas colar na interface para evitar que perca seu trabalho.

Também é possível em alguns navegadores, clicando com o botão direito sobre as prévias, aparecer a opção de salvar a imagem (aparece no Chrome e no Edge). Ela será salva na resolução completa (3000x3000).