Projeto 1

Computação gráfica e interfaces

Tiago Fernandes 57677
António Ferreira 58340

Introdução
O programa encontra-se funcional e foram aplicadas todas as transformações necessárias para completar os desafios opcionais do trabalho. Este relatório apresenta e explica os atributos e uniforms presentes nos shaders do projeto, assim como algumas opções tomadas pelo grupo.

shader1.vert
Vertex shader responsável pela criação dos pontos da grelha e onde se realizam os cálculos do campo elétrico responsáveis pela alteração da posição dos pontos móveis.

Atributos:
- vPosition: vec4 que recebe as coordenadas do ponto da grelha, independentemente deste ser móvel ou não. Os seus valores são atribuídos ao gl_Position do mesmo vertex shader.
- isMoving: float que funciona na prática como um booleano. Recebe 1.0 caso o ponto seja móvel e 0.0 caso seja fixo. Faz-se a verificação do mesmo na main deste vertex shader.

Uniforms:
- table_width e table_heigth: floats que recebem a largura e altura, respetivamente, da mesa de trabalho. Este valor é usado para fazer a conversão das coordenadas dos pontos (em World Coordinates) para WebGL (Clipping Coordinates).
- protonPos[MAX_CHARGES] e eletronPos[MAX_CHARGES]: vetores de vec2 com comprimento MAX_CHARGES (valor arbitrário), que recebem as posições de todas as cargas positivas e negativas, respetivamente, existentes no momento. Usados para calcular o valor do campo elétrico incidente no ponto móvel.
- protonPosSize e eletronPosSize: inteiros que recebem o tamanho dos dos vetores correspondentes às cargas positivas e negativas, respetivamente.

Charge.vert
Vertex shader responsável pela criação das cargas existentes.

Atributos:
- vPosition: vec4 que recebe as coordenadas da carga. Os seus valores são atribuídos ao gl_Position do mesmo vertex shader.

Uniforms:
- table_width e table_heigth: floats que recebem a largura e altura, respetivamente, da mesa de trabalho. Este valor é usado para fazer a conversão das coordenadas dos pontos (em World Coordinates) para WebGL (Clipping Coordinates).

Charge.frag
Fragment shader responsável pelo processamento de cada fragmento gerado. Neste shader fizémos alterações para que o ponto das cargas fossem circulares e aprentassem "+" ou "-", fazendo com que apenas alguns fragmentos não fossem pintados.

Uniforms:
- color: vec4 que recebe a cor usada para pintar o fragmento.

Decisões do grupo

O grupo decidiu, devido à crença de que facilitaria o nosso trabalho, em criar dois vetores diferentes para as cargas, sendo que um guardava cargas positivas e outro guardava as negativas. Para isto foram necessárias alterações no buffer para se reservar espaço para ambos os vetores, assim como criação de 2 atributos ao invés de 1 no shader1.vert, para guardar as posições das cargas de diferentes sinais.

Decidimos também usar apenas um buffer. Percebemos que pode facilitar o trabalho a criação de dois, mas como não nos deparámos com problemas depois de usarmos apenas um buffer, optámos por não criar um segundo apenas para as cargas.