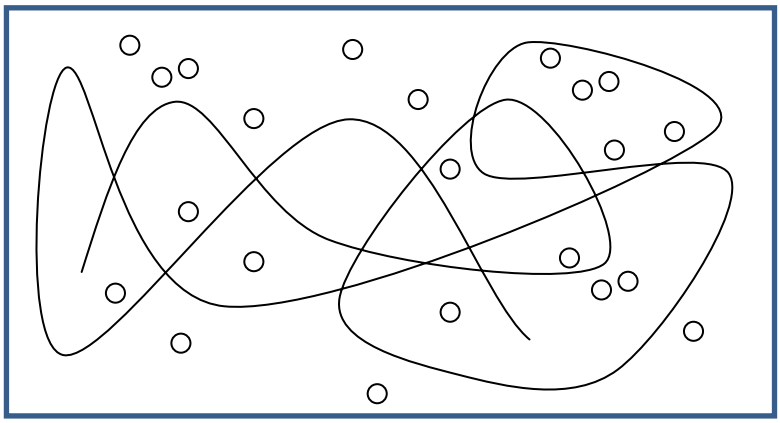
Universidade Federal de Santa Maria
Curso de Ciência da Computação
Disciplina: Computação Gráfica Avançada
Segundo Semestre de 2023
Prof. Cesar Tadeu Pozzer

Trabalho 2 – Spatial Hashing

Objetivos do trabalho:

- Uso de hash espacial
- Técnica de aceleração
- Estruturas de dados



Desenvolva um programa (Canvas2D, Unity ou Visual Studio) que implemente a técnica de hash espacial apresentada no artigo **A Hash Table Construction Algorithm for Spatial Hashing Based on Linear Memory (Pozzer et al., 2014).**

Deve-se definir uma área onde é desenhada uma curva com pelo menos 200 pontos de controle, criada de forma randômica. Após, deve-se distribuir círculos de raio r nesta área de forma que eles não fiquem sobre a linha. Uma forma de tratar interseção de círculos com linhas curvas é aproximar a curva por segmentos de reta. O tamanho da tela deve ter no máximo 1800 x 1000 pixels. Pode-se utilizar o material de interseção da disciplina.

Usar duas técnicas para determinar se os círculos estão sobre a curva:

1. com força bruta (todos com todos) e
2. com o algoritmo de hash espacial.

Comparar os tempos de execução, considerando diferentes tamanhos de células de hash e diferentes conjuntos de dados.

O programa deve permitir a visualização dos círculos e da curva. Atentem que o foco é **medir a eficiência do algoritmo de hash**, logo, não devem ser considerados tempo de rendering e nem tempo de geração/distribuição dos segmentos. Desta forma, exibindo apenas uma fração do total de segmentos já é suficiente para mostrar que as interseções foram encontradas.

O programa deve ser muito bem otimizado e comentado. Faça um programa bem interativo.

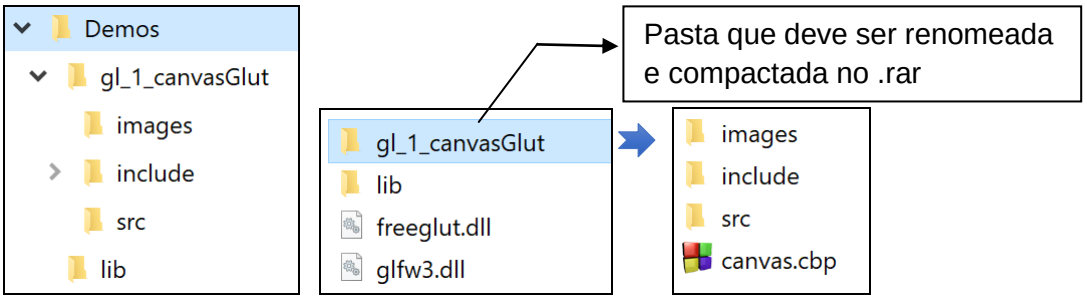
Enviar arquivo .pdf com tabelas comparativas entre das duas abordagens, mostrando o tempo de cada abordagem dadas várias configurações de números de células e número de linhas.

Extras:

1. Função de zoom sobre o plano com uso do mouse
2. Algoritmo básico de LOD para exibir um número de segmentos/círculos de acordo com o nível de zoom sobre o terreno. Senão, renderizar todos os segmentos pode travar a aplicação.
3. Fazer em GLSL com VBOs.
4.

Data e Formato de Entrega

- O trabalho deve ser apresentado durante a aula.
- O trabalho deve ser enviado pelo classroom.
- Deve-se enviar fontes e o projeto para o compilador **Code::blocks ou Unity**. Envie **apenas** os arquivos **de projeto, código fonte e modelos/texturas.**
- Se for usada a Canvas2D, deve-se utilizar como base o projeto gl_1_canvasGlut disponível nos demos da disciplina, como ilustrado na seguinte figura a esquerda.



- O programa deve ser enviado em um arquivo compactado fulano.rar (fulano = login ou nome do aluno). Dentro deste arquivo deve haver um diretório com o mesmo nome do arquivo e dentro deste diretório os arquivos do trabalho.
- Ex: o arquivo pozzer.rar deve conter um diretório chamado pozzer, e dentro do diretório devem estar os arquivos do trabalho.
- Não devem ser enviadas lib, exe, obj, DLL.
- Enviar arquivo .pdf com tabelas comparativas entre das duas abordagens.

Critério de Avaliação

- Documentação: descrever no cabeçalho de cada arquivo a ideia geral do código e detalhes específicos de partes que mereçam uma explicação – não comente por exemplo o que faz b++.

- README.txt: incluir um arquivo “README.txt” contendo informações sobre quais funcionalidades foram implementadas (requisitos e extras).
- Pontualidade: Trabalhos não entregues na data não serão avaliados e receberão nota zero.
- Legibilidade: nome de variáveis, estruturação do código. O código digital a ser entregue deve ter 4 espaços de identação e não deve possuir tabulações.
- Clareza: facilidade de compreensão – evite códigos complexos e desnecessários. Adote a solução mais simples possível.
- Funcionalidade: o programa deve satisfazer todos os requisitos. Programas que não compilarem ou que não atenderem nenhum requisito receberão nota 0 (zero).

Você pode discutir estratégias e ajudar o colega na implementação, porém evite passar código fonte. Programas semelhantes terão nota 0 (zero).