Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso Superior de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD1 2° semestre de 2017.

1) Faça uma pesquisa sobre a área de Realidade Virtual. Descreva que subáreas da Computação Gráfica colaboram para resolver os problemas dentro do escopo da Realidade Virtual(1.0 ponto).

Segundo Popolin et al. [Popolin2015], Realidade Virtual é a uma forma de experiência onde o usuário encontra-se imerso em um mundo virtual (sintetizado), o qual reage as suas ações e no qual ele possui controle dinâmico do ponto de vista. Alguns autores afirmam que Realidade Virtual é uma forma avançada de interface com o usuário. Suas principais características são: imersão, interação e envolvimento. Para que tais características se manifestem, é necessário o estimulo aos sentidos de visão, tato e audição, através de equipamentos como HMD (*Head Mounted Displays*) e sistemas multiprojeção, as chamadas CAVEs. Em geral, as imagens são estereoscópicas para gerar a sensação de profundidade. Os dispositivos de interação incluem 3DUIs (interfaces 3D com usuários), dispositivos de captura de gestos, como o Kinect, luvas e outros. Bibliotecas de realidade virtual devem oferecer suporte à interface para dispositivos de interação, operação e gerenciamento de monitores e projetores, multiprocessamento, cálculo de perspectiva do ponto de vista do usuário e configuração para diferentes sistemas de RV.

Como pode ser visto, existem diferentes problemas e tarefas que devem ser realizadas em um processo envolvendo RV, incluído geração de imagens estereoscópicas a partir do ponto de vista do usuário (Síntese de imagens), interação com o usuário (IHC), reconhecimento de gestos (Análise de Imagens).

Referência: [Popolin2015] M. Popolin Neto, I. A. Agostinho, D. R. C. Dias, I. A. Rodello e José R. F. Brega. A Realidade Virtual e o Motor de Jogo Unity. Tendências e Técnicas em Realidade Virtual Aumentada, v. 5, p. 9-23, maio/2015.

2) Discuta sobre a dicotomia eficiência versus realismo na Computação Gráfica. (1.0 ponto).

Na Computação Gráfica a dicotomia eficiência versus realismo sempre foi considerada desde o seu surgimento. Para conseguir efetuar tarefas de forma rápida, em muitos casos, é necessário simplificar os cálculos ou realizar pré-computações através de estruturas de dados especiais. Quanto maior for o detalhamento dos objetos a serem representados e quanto maior a complexidade dos processos envolvidos, como iluminação, animação e simulação, maior será o impacto sobre a eficiência da geração das imagens. Portanto, toda técnica de Computação Gráfica deve considerar o compromisso entre complexidade visual/realismo e eficiência. Um exemplo prático de como esta questão é considerada na Computação Gráfica é o uso de técnicas de nível de detalhamento variável (LOD – *Level of Detail*), onde formas são representadas em diferentes níveis de detalhe e uma delas é escolhida para o passo de renderização, em função da distância do objeto à câmera. Objetos distantes são exibidos em sua forma mais simplificada, enquanto que os mais próximos são exibidos com geometria e textura mais detalhadas. Outro exemplo é o uso de algoritmos de tonalização. Algoritmos de iluminação local são preferidos, quando deve-se

considerar aplicações em tempo real. Por outro lado, os métodos de iluminação global, mais sofisticados, são usados para geração de imagens *off-line*, que não necessitem ser geradas em tempo real nem de interação, com na produção de filmes na indústria cinematográfica.

3) Descreva alguns dos principais dispositivos de entrada e de saída utilizados em Computação Gráfica (1.0 ponto).

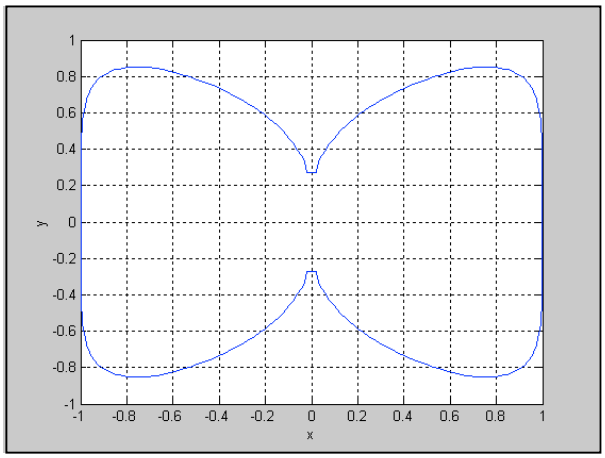
Os dispositivos de entrada e saída são responsáveis pela interação entre o usuário, no sentido amplo, e a máquina. Os dispositivos de entrada são os de captação de informações gráficas e os dispositivos de saída são os responsáveis pela visualização dos dados.

Tratando-se de dispositivos de entrada e saída gráficos, estes estão relacionados com o formato dos dados com que trabalham. Dois são estes formatos: *raster* ou matricial e vetorial.

Os dispositivos do tipo raster ou matriciais, representam os dados através de uma matriz MxNxC. Esta matriz é vista de forma tri-dimensional onde M representa o número de colunas, N o número de linhas e C representa a cor. Cada posição da matriz é um pixel da imagem e seu conteúdo é representado por C. Exemplos de dispositivos matriciais de entrada são: scanners e a máquinas fotográficas digitais. Como exemplos de dispositivos matriciais de saída temos o monitor CRT ou LCD e as impressoras.

O segundo formato representa a imagem de forma vetorial; as informações são armazenadas na forma de coordenadas de um espaço vetorial. Exemplos de dispositivos de entrada vetoriais são: *light pen*, *tablet*, *touch pannel*, 3D-digitizer e os mais comuns e conhecidos de todos, o mouse e o joystick, os quais possuem um sistema de coordenadas absolutas. Os dispositivos de saída vetorial são os que produzem imagens traçando segmentos de retas e curvas descritas por coordenadas de seus pontos iniciais e finais. Exemplos desses dispositivos são: display caligráfico, display de armazenamento e os traçadores.

4) Considere a curva dada pela equação $f(x,y) = x^6+y^6-x^2$. Classifique cada ponto da lista L = {(0.4,0.6),(0.0,1.0),(-0.2,0.0),(0.8,0.3) } como *interior* ou *exterior*. Descreva o método utilizado para classificar os pontos (1.0 ponto).



A curva é descrita em forma implícita, logo, basta aplicar a equação sobre as coordenadas de cada um dos pontos e verificar o sinal do resultado. Se o sinal for negativo então o ponto

pertence à região(é um ponto interior), caso contrário, ele não pertence (é um ponto exterior).

Pontos	***f(x,y) = $x^6+y^6-x^2$***	***Resultado***
(0.4, 0.6)	*f(x,y) = $0.4^6+0.6^6-0.4^2$ = -0.1092*	*(interior)*
(0.0, 1.0)	*f(x,y) = $0.0^6+1.0^6-0.0^2$=1.0*	*(exterior)*
(-0.2, 0.0)	*f(x,y) = $-0.2^6+0.0^6-(-0.2)^2$=-0.0401*	*(interior)*
(0.8, 0.3)	*f(x,y) = $0.8^6+0.3^6-0.8^2$ = -0.3771*	*(interior)*

5) Descreva um método para computar a interseção entre duas retas *r* e *s* no plano, onde ambas as retas são representadas de forma paramétrica (1.0 ponto).

Seja *r* a reta que passa por um par de pontos p_0 e p_1 e *s* a reta que passa por dois pontos p_2 e p_3. Pode-se então escrever as equações paramétricas de *r* e *s* respectivamente como:

$$p_a(t_a) = p_0 + t_a(p_1 - p_0)$$
$$p_b(t_b) = p_2 + t_b(p_3 - p_2)$$

Fazendo $p_a(t_a) = p_b(t_b)$, tem-se duas equações em duas incógnitas t_a e t_b.

$$x_1 + t_a(x_1 - x_0) = x_2 + t_b(x_3 - x_2)$$
$$y_1 + t_a(y_1 - y_0) = y_2 + t_b(y_3 - y_2)$$

Resolvendo para t_a e t_b obtem-se:

$$t_a = \frac{(x_3 - x_2)(y_0 - y_2) - (y_3 - y_2)(x_0 - x_2)}{(y_3 - y_2)(x_1 - x_0) - (x_3 - x_2)(y_1 - y_0)}$$
$$t_b = \frac{(x_1 - x_0)(y_0 - y_2) - (y_1 - y_0)(x_0 - x_2)}{(y_3 - y_2)(x_1 - x_0) - (x_3 - x_2)(y_1 - y_0)}$$

Se os denominadores de t_a e t_b forem zero, então as retas suporte são paralelas, e se tanto os denominadores quanto os numeradores forem zero, as retas suporte são coincidentes. Para encontrar o ponto de interseção basta avaliar a equação (1) ou (2) para os parâmetros t_a e t_b, respectivamente.

6) Que tipo de objeto gráfico você utilizaria para representar um modelo de uma rocha para fins de computar sua permeabilidade (https://pt.wikipedia.org/wiki/Permeabilidade_(geologia)) (1.0 ponto). Explique sua escolha (1.0 ponto).

Como é necessário compreender as estruturas internas da rocha, um modelo volumétrico é o mais apropriado. Uma outra razão é que o tipo de instrumento utilizado para adquirir dados digitais sobre rochas usa algum mecanismo de tomografia computadorizada, que gera dados naturalmente volumétricos. Entretanto, é ainda possível utilizar um Modelo B-Rep que descreva a interface que separe o interior da rocha de seus poros, o que pode ser uma tarefa complexa, haja visto o nível de complexidade de tais estruturas e a presença de ruído.

7) O formato .obj é uma formato aberto para representação de formas 3D introduzido pela Wavefront. Considere o arquivo .obj abaixo que utiliza um subconjunto do formato .obj para

descrever um objeto e proponha uma estrutura de dados na linguagem de sua preferência para que o represente em memória principal. (1.0 ponto).

```
  # Lista de coordenadas dos vértices
  0.123 0.234 0.345 1.0
  ...

  # Lista das normais dos vértices na forma x,y,z; as normais não precisam ser
vetores unitários.

  0.707 0.000 0.707
  ...

  ...
  # Lista de Faces poligonais na forma (v0 v1 ... vn) onde vi é o índice do i-
ésimo vértice
  f 1 2 3
  f 3 4 5
  f 6 3 7
  f 7 8 9
  f ...
```

Utilizaremos a linguagem C para definir a estrutura de dados para descrever um objeto descrito em um formato .obj simplificado. A estrutura basicamente contém uma lista de vértices e uma lista de faces, onde cada face contém um array de índices da lista de vértices.

```
typedef int indice_vert_t;

typedef struct vertice{
        float coord[3];
        float normal[3]
}vertice_t;

typedef struct lista_vertices{int n_vertices;
                              índice_vert_t *v;}lista_vertices_t;

typedef struct face{
        v_indices_t * vertices; // array contendo índices da lista de vértices
        int    n_vertices;
}face_t;

typedef struct lista_vertices{int n_faces;
                              face_t *v;}lista_faces_t;
```

8) Dada uma curva paramétrica definida entre um intervalo [a,b], determine um método para computar seu comprimento de forma aproximada (1.0 pontos).

Seja $c(t):[a,b]\rightarrow R^2$ a curva paramétrica definida no intervalo [a,b]. O método consite em computar um conjunto finito, formado por *n* amostras da curva, através da aplicação da função paramétrica

sobre n pontos, no intervalo de parâmetros, determinados através de uma partição uniforme. O conjunto de amostras gera uma aproximação poligonal da curva. Para cada par consecutivo de amostras obtidas, avaliar sua distância e somar o total. Estas ideias são expressas no algoritmo abaixo:

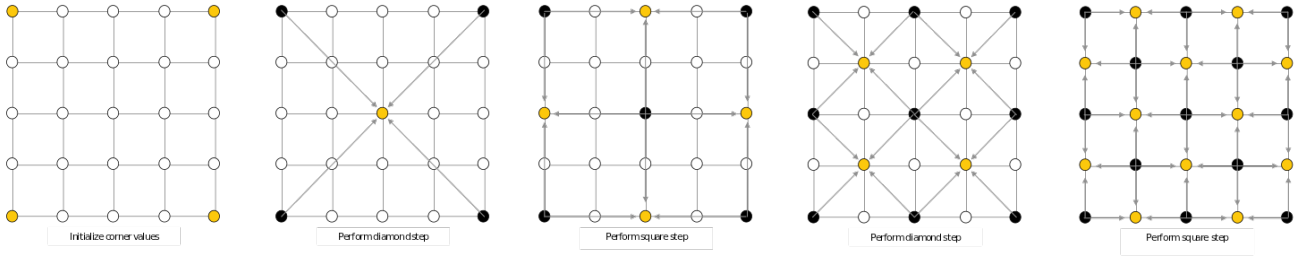
Algoritmo
comprimento = 0
Δ=(a-b) / n
px = c(a).x
py = c(a).y
Para i=0 ate n-1 faça
*q.x = c(a+i*Δ).x*
*q.y = c(a+i*Δ).y*
comp_arest = $\sqrt{(q.x - p.x)^2 + (q.y - p.y)^2}$
comprimento = comprimento + comp_aresta
Fim-para
Fim-Algoritmo

9) Faça uma pesquisa sobre como gerar um modelo de terreno computacionalmente (1.0 ponto)

Diferentes algoritmos podem ser usados para gera um modelo de terreno. Aqui apresentamos o algoritmo Diamond Square que surgiu pela primeira vez no trabalho de Fournier et al [Fournier82].

O algoritmo é relativamente simples. Ele parte de um reticulado de dimensão 2^nx2^n definido em um subconjunto do plano, conforme a figura abaixo obtida de:

https://commons.wikimedia.org/wiki/File:Diamond_Square.svg.



Em um primeiro passo, são definidos valores aleatórios para os cantos extremos do quadrado de lado igual a 2^n. Em seguida, é aplicado o passo chamado Diamante (*Diamond Step*), onde o valor do vértice no centro do quadrado, determinado pelos extremos é computado como a média dos valores nos cantos. Logo, é aplicado o passo Quadrado (*Square Step*), no qual o valor nos vértices no centro das arestas do quadrado (ver terceira figura), são calculados como a média dos valores dos vértices que definem a aresta que o contém. Os passos Diamante e Quadrado são repetidos de forma recursiva nos quadrados de dimensão 2^i até que todos os vértices do reticulado tenham seus valores calculados.

[Fournier82] Alain Fournier, Don Fussell, and Loren Carpenter. 1982. Computer rendering of stochastic models. *Commun. ACM* 25, 6 (June 1982), 371-384. DOI=http://dx.doi.org/10.1145/358523.358553

10) Como você adaptaria a técnica utilizada na questão anterior para modelar um planeta, isto é, gerar um terreno sobre uma superfície esférica (1.0 ponto).

Para modelar um planeta, precisamos definir o mapa de alturas calculado no algoritmo descrito na questão 9 sobre uma superfície esférica. Para isto, pode-se considerar uma representação paramétrica de uma superfície esférica usando a equação:

$$x(\theta,\varphi) = r\cos\varphi\cos\theta$$
$$y(\theta,\varphi) = r\cos\varphi\sin\theta$$
$$z(\theta,\varphi) = r\sin\varphi$$
$$0 \le \theta \le 2\pi$$
$$-\pi/2 \le \varphi \le \pi/2$$

Cria-se então um reticulado no espaço de parâmetros, com um número de amostras igual a uma potência de 2, nos eixos correspondentes aos ângulos θ e φ, e aplica-se o algoritmo *Diamond-Square*, computando para cada amostra (θ_i,φ_i) um valor $h(\theta_i,\varphi_i)$.

Ao final do processo, aplica-se a função $(x(\theta,\varphi),y(\theta,(\theta,\varphi)),z(\theta,\varphi))$ para transferir o mapa de alturas para a superfície, resultando assim em um deslocamento $h(v_i)$ para cada vértice v_i. As coordenadas $(x(v_i),y(v_i),z(v_i))$ de cada vértice v_i, correspondente a uma amostra na superfície da esfera, devem ser deslocadas na direção normal $(n_x(v_i), n_y(v_i), n_z(v_i))$ pela altura $h(v_i)$ computada pelo algoritmo *Diamond-Square*, resultando em novas coordenadas $(x'(v_i) = x(v_i)+h(v_i)n_x(v_i), y'(v_i) = y(v_i) + h(v_i)n_y(v_i), z'(v_i) = z(v_i)+ h(v_i)n_y(v_i))$.

Observe que o conjunto de vértices com coordenadas paramétricas $\varphi=-\pi/2$ e $\varphi=\pi/2$ são todos mapeados nos dois polos da esfera discretizada. Uma forma de tratar este problema é fazer com que os vértices mapeados no polo norte (polo sul) recebam a média das alturas computadas para os vértices tais que $\varphi=\pi/2$ ($\varphi=-\pi/2$). Finalmente, o reticulado pode ser triangulado no espaço de parâmetros para a criação da malha que descreve a superfície através da parametrização da esfera.