FICHA 5 - Culling

1 - Compare em termos computacionais os três tipos de culling apresentados na disciplina.

-Back-Face Culling: Consiste em não desenhar triângulos pertencentes a faces
viradas de costas para a câmara,corre em GPU portanto é bastante rápido

-Occlusion Culling: Consiste em não desenhar polígonos que estejam tapados por outros polígonos, no entanto isto embora possa ser testado em GPU é bastante complexo necessitando de uma visão da restante geometria.

-View Frustum Culling: Consiste na implementação de uma caixa, esta caixa vai conter 6 faces e iremos apenas desenhar os triângulos que estiverem dentro desta caixa, este método é bastante simples de implementar embora corra em CPU é o método mais utilizado atualmente.

2 - Descreva o processo matemático para obter a equação normalizada do plano que contem os pontos p1, p2 e p3.

N = V1 x
V2 /
Norma(V1x
V2)

Nx*x + Ny
* y + Nz * z
+ D = 0
Substituir x,y e z por um ponto de forma a obter D

3 - Descreva os passos necessários para implementar o algoritmo de View Frustum Culling.

_____Calcular os planos da caixa que pretendemos usar para definir os triângulos a demonstrar ao utilizador

Para cada polígono verificar se este se encontra dentro ou fora da caixa em questão, para tal podem ser usados 4 métodos para detetar colisão:

AABB - criar uma caixa ao longo do objeto : Muito comum e mais simples de implementar porém não poupa muito espaço

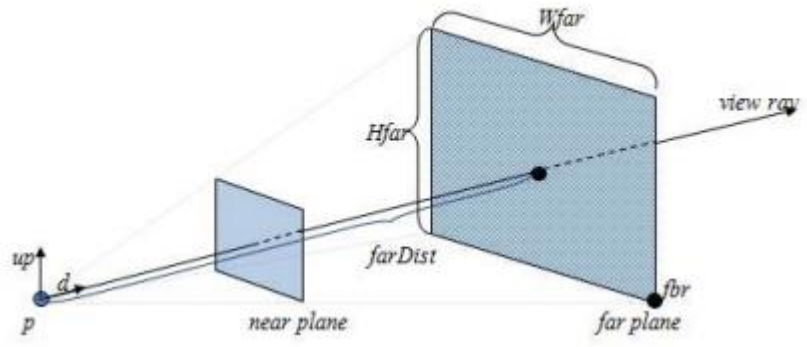
OBB - Criar uma caixa à volta do objeto alinhada com este, Complicado de implementar mas poupa mais espaço do que o anterior

Sphere - Criar uma esfera que cubra completamente o polígono, simples de implementar mas nem sempre o mais eficiente em termos de espaço utilizado

ConvexHull - Criar um objeto curvado em volta do polígono em questão. Melhor em termos de espaço poupado mas dificil de implementar e testar

4 - Considere os vectores d e up, o ponto p, e as distâncias farDist, Wfar e Hfar, apresentados na figura. Descreva o processo matemático para obter o ponto fbr.

Assumindo d e up como estando normalizados podemos obter o ponto Fbr fazendo o seguinte:



Right = d * up

F = p + d*farDist (ponto intermédio central do farPlane)

Fb = F - HFar/2*up (ponto intermédio central na junção entre o farPlane e o plano de baixo)

Fbr = Fb + WFar/2*right (ponto final)

5 - Considere agora que tem somente os dados presentes nas seguintes instruções:

gluPerspective(fov, ratio, nearDist, farDist);
gluLookAt(px,py,pz, lx,ly,lz, ux,uy,uz);

Descreva o processo matemático para obter os dados referidos na pergunta anterior: vectores d, up e right, e as distâncias Wfar e Hfar.

D =

(L -

P)/

Nor

ma(

L-

P)

Up

= U

Right = d * up

Hfar = 2 * tan(Fov/2) * farDist (decorar esta

fórmula de cáca) Wfar = Hf * ratio

6 - Apresente o algoritmo para extrair os planos do view frustum segundo a visão geométrica.

_____Calcular pontos de cada canto da caixa do frustum, utilizando o exemplo apresentado na pergunta 4 e 5, após isso iremos utilizar os devidos pontos para criar 2 vetores

perpendiculares por plano, com esses vetores utilizamos a regra da mão direita e conseguimos obter a normal ao plano, normalizando-a, sabendo a normal ao plano e 1 ponto pertencente a este, a escrita da equação do plano torna-se evidente.

7 - Descreva o algoritmo para extrair os planos do view frustum em clip space

_____Com a utilização do clip space a forma de calcular os pontos que estão envolvidos neste torna-se mais simples, passando apenas por calcular a matriz de projeção no clip space para cada frame, esta matriz pode ser obtida multiplicando a GL_MODEL_VIEW_MATRIX pela GL_PROJECTION_VIEW_MATRIX, esta matriz apenas é calculada uma vez por frame, após isso devemos multiplicar cada ponto do modelo por esta matriz, obtendo o seu equivalente no clipspace.

Depois de obter este ponto pC podemos comparar cada uma das suas componentes

com as componentes do cubo do clip space obtendo assim as seguintes inequações:

-Wc <= xpC <= Wc plano esquerdo e direito do frustum
-Wc <= ypC <= Wc plano de cima e de baixo do frustum
-Wc <= zpC <= Wc plano de trás e da frente do frustum

8 - Por forma a tornar eficiente o algoritmo de view frustum culling é necessário implementar algum mecanismo de agrupamento de triângulos. Descreva o processo de partição espacial baseado em k-D trees.

As K-D-Trees surgem como uma forma de remover o grau de liberdade da orientação dividindo a imagem em planos perpendiculares ao eixo x/y e vice-versa na próxima iteração, terminando com uma árvore binária que facilita a definição dos objetos a renderizar com o frustum.

9 - Os processos de partição espacial são em regra recursivos na construção da estrutura de dados. Indique três critérios possíveis para terminar a recursividade.

A utilizaçãop de quad(2D) trees ou octo(3D) trees tem como objetivo agrupar os vários
triângulos, consiste em (2D) a imagem total em 2 retas definindo 4 novas imagens, estas 4 serão também divididas recursivamente a menos que essa célula não possua triângulos, como forma de paragem o que acontece é a definição de 3 critérios de paragem, esses critérios são:

- Número de triângulos na célula < t
- Volume na célula < v
- Profundidade da árvore > p

10 - Num processo de partição espacial é possível que um triângulo pertença a mais que um filho. Indique quais as opções disponíveis nestes casos apresentando as vantagens e desvantagens de cada uma

A 1ª opção passa por atribuir esse triângulo à célula Pai, no entanto esta opção não é a mais correta pois o número de triângulos em que estejam no meio de uma divisão é minúsculo e fará com que a placa gráfica seja sub utilizada.

Na 2º opção podemos dividir esse triângulo de modo a que sejam criados triângulos respectivos para cada célula filha, no entanto este método é custoso pois causa a necessidade de dividir o triângulo dando origem a novos vértices que ocupam memória extra, não é necessário sendo que a maior parte das vezes os triângulos não serão partidos na célula.

A 3º opção consiste na duplicação de triângulos para cada uma das células filhas, e embora esta opção ocupe mais memória (em termos de índices) é a que faz mais sentido, pois embora nos dê a ideia de que iremos

desenhar o triângulo 2 vezes, isto não acontece pois após a renderização da primeira iremos verificar que não faz sentido desenhar a outra pois essa profundidade já se encontra utilizada.

12 - Indique os tipos de volumes envolventes que poderiam ser utilizados numa partição hierárquica, comparando a sua eficiência em termos de culling e complexidade algorítmica.

Refer to 3