Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso Superior de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD1 1° semestre de 2018.

1) Faça uma pesquisa sobre algum dispositivo de Realidade Virtual. Descreva o seu funcionamento básico e cite aplicações. (1.25 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.

O funcionamento básico de sistemas de Realidade Virtual dependem da aplicação específica e dos tipos de dispositivos utilizados. Entretanto, pode-se afirmar que existem algumas características que estão presentes em todos os sistemas. Praticamente todo sistema de Realidade Virtual envolve um sistema computacional, o *motor de realidade virtual*, que recebe um conjunto de sinais de entrada gerados pelo usuário e por dispositivos de captura. Dentre estas entradas pode-se citar o posicionamento e orientação capturados pelo dispositivo de *tracking* do HMD e interações com *joysticks* ou *trackballs*. As entradas capturadas são processadas por um conjunto de componentes de software, que acessam a base de dados do sistema e que contém o modelo do mundo virtual e os dados do domínio da aplicação. Como resultado do processamento são realizadas mudanças no mundo virtual e enviados ao usuário um conjunto de sinais, que se manifestam como estímulos visuais, auditivos e às vezes táteis, nos dispositivos de saída.

Um esquema simplificado de arquitetura de RV pode ser visto na Figura 1:

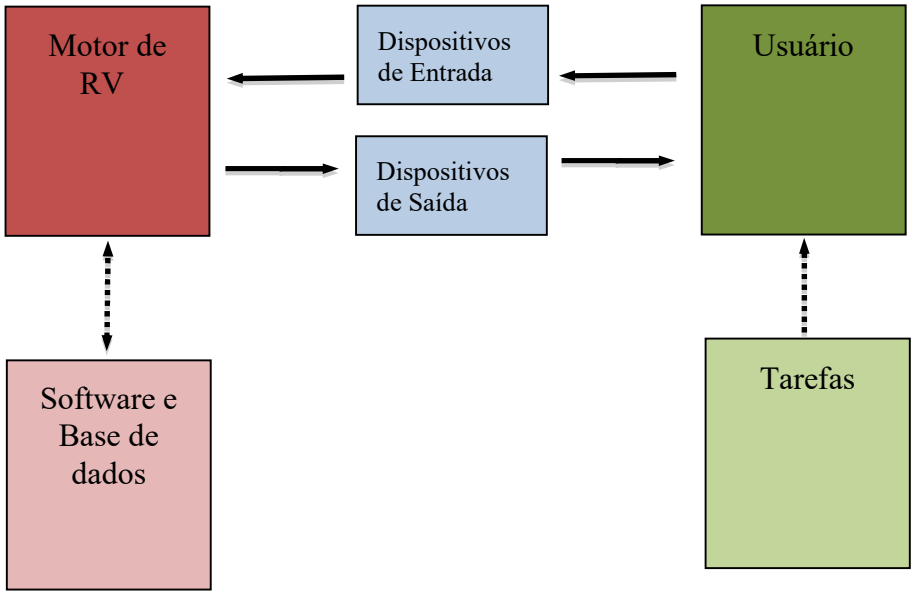


Figura 1 – Arquitetura de um sistema de RV (baseada nos slides obtidos de [Stankovic2012])

As técnicas de Realidade Virtual podem ser aplicadas em inúmeras áreas incluindo entretenimento, visualização científica, esportes, na indústria cinematográfica, na medicina, por exemplo, em cirurgias a distância e muitas outras.

Referências:

[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.

[Stankovic2012] Stanislav Stankovic. Curso de Realidade Virtual da Tampere University of Technology. Introduction to Virtual Reality. SGN http://www.cs.tut.fi/kurssit/SGN-5406/lectures/VR1-introduction.pdf. (acessado em 26-02-2018 - 11:32)

2) Explique o que é Computação Visual (*Visual Computing*). Cite um problema investigado pela Computação Visual (1.25 ponto).

A Computação Visual é uma área da Ciência da Computação de natureza interdisciplinar, que busca resolver problemas que envolvem a forma e a aparência dos objetos e o modo como estes interagem com a luz e os sistemas visuais. A Computação Visual utiliza conceitos, métodos e ferramentas provenientes das áreas de Computação Gráfica, Processamento de Imagens, Visão Computacional, Geometria Computacional e Interação Homem Computador, além de outras áreas da ciência. Problemas de natureza visual são formulados e resolvidos com base na integração dos conceitos e métodos provenientes das áreas afins. Neste sentido, a Computação Visual é mais do que um simples agregado de subáreas, mas uma nova forma de tratar problemas relacionados a objetos de natureza visual e gráfica.

Um problema investigado em Computação Visual é a aquisição de geometria e aparência de obras arquitetônicas. Este problema específico envolve diferentes subáreas como Visão Computacional, que colabora para inferir a geometria e atributos de cor e material da cena, a partir de imagens e mapas de profundidade, Modelagem Geométrica e Geometria Computacional, na modelagem dos objetos capturados e correção dos dados reconstruídos (por exemplo, na geração de malhas de melhor qualidade) e Computação Gráfica na síntese final da cena virtual reconstruída no display para fins de visualização do processo.

3) Considere o problema de selecionar um município em um mapa. Como você representaria o conjunto de municípios no mapa? Qual teste deveria ser feito para verificar se o cursor do mouse está sobre um dado município? (1.25 ponto).

Uma forma simplificada para representação dos objetos em questão é aquela baseada em uma lista de curvas poligonais, onde cada curva poligonal descreve as fronteiras que delimitam um município. Conhecida a posição (x,y) do clique do mouse, deve ser feita uma transformação de coordenadas para o sistema de coordenadas usado na descrição dos entes geográficos. Posteriormente, pode-se efetuar um teste ponto-polígono, usando o algoritmo que se baseia na contagem de interseções de uma semi-reta com origem em T(x,y), as coordenadas transformadas. Se o número de interseções for par determina-se que o ponto é interior, caso contrário, sendo ímpar, determina-se que é exterior a região em questão.

4) Considere o objeto gráfico dado por:

$$x(t) = a\cos(t)(1-\cos(t))$$
$$y(t) = a\cos(t)(1-\sin(t))$$

Classifique tal objeto (1.25 ponto).

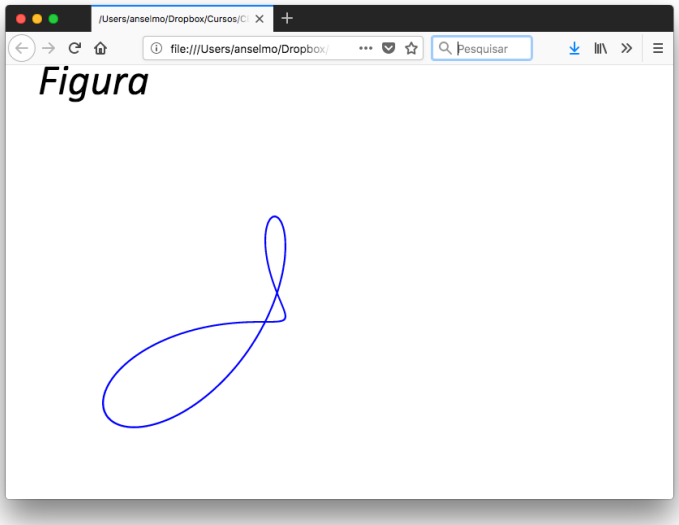
O objeto gráfico descrito acima é uma curva paramétrica. Logo, um objeto gráfico unidimensional e planar, uma vez que associa duas coordenadas a um valor do parâmetro t.

5) Faça uma pesquisa sobre HTLM5 canvas. Plote a figura do objeto gráfico do exercício 4 usando as ferramentas do HTLM5 canvas (1.25 ponto).

O HTML5 Canvas é um elemento da linguagem de descrição HTML5 que cria uma superfície de desenho. O Canvas possui um tamanho especificado pelos atributos *width* e *height*. Ao ser criado, o canvas expõe um contexto (*context*) que é usado para criar e manipular formas, tanto bidimensionais quanto tridimensionais (usando WebGL, por exemplo). Através da linguagem Javascript, é possível obter um contexto a partir do elemento canvas, que é visto como um nó em uma estrutura DOM. Uma vez de posse do contexto, pode-se efetuar diferentes desenhos de formas e modificações de seus atributos. Por exemplo:

```
...
// Definição do canvas com 640x640 pixels
<canvas id="myCanvas" width="640" height="640"></canvas>
   <script>
     // Obtém o elemento canvas a partir do nó DOM com id myCanvas
     var canvas = document.getElementById('myCanvas');
     // Obtém o contexto
     var context = canvas.getContext('2d');
...
```

A plotagem da figura do exercício 4 pode ser feita de forma simples utilizando o seguinte código HTML5/Javascript a seguir.



Referência:

https://en.wikipedia.org/wiki/Canvas_element (acessado em 23 de fev. 18 14:33h)
https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial (acessado em 23 de fev. 18 14:33h)

```
<!DOCTYPE HTML>
<html>
 <head>
  <style>
   body {
    margin: 0px;
    padding: 0px;
   }
  </style>
 </head>
 <body>
  <canvas id="myCanvas" width="640" height="640"></canvas>
  <script>
   // Pega o elemento canvas do documento
   var canvas = document.getElementById('myCanvas');
   // Obtem o contexto
   var context = canvas.getContext('2d');

   // Configura a fonte e escreve um texto em azul
   context.font = 'italic 40pt Calibri';
   context.fillText("Figura",40,40);


   // Determina o numero de amostras, o intervalo de amostragem(delta)
   // Determina o centro do canvas (cx, cy)
   var delta = 0.1;
   var scale = 100;
   var numSamples = 300;
   x0 = -numSamples/2;
   y0 = (x0*x0)/scale;
   cx = canvas.width/2;
   cy = canvas.width/2;
   // Move o sistema de coordenadas para o primeiro ponto da curva
   context.moveTo(x+cx,cy-y);

   //Gera uma amostra para um conjunto discretizado de valores do ângulo teta
   var x=0,y=0;
   var i;
   var teta0 = 0;

   // Cria um path (uma linha poligonal)
   context.beginPath();
   for (i=0;i<numSamples;i++){
    teta = teta0 + i*delta;
    x = scale*Math.cos(teta)*(1-Math.cos(teta));
    y = scale*Math.cos(teta)*(1-Math.sin(teta));
    // Desenha um segmento de reta o ponto corrente até o especificado no comando
    context.lineTo(x+cx,cy-y);
   }

   // Determina a lagura da linha e o estilo do tracejado
   context.lineWidth = 2;
   context.strokeStyle = 'blue';
   context.stroke();


  </script>
 </body>
```

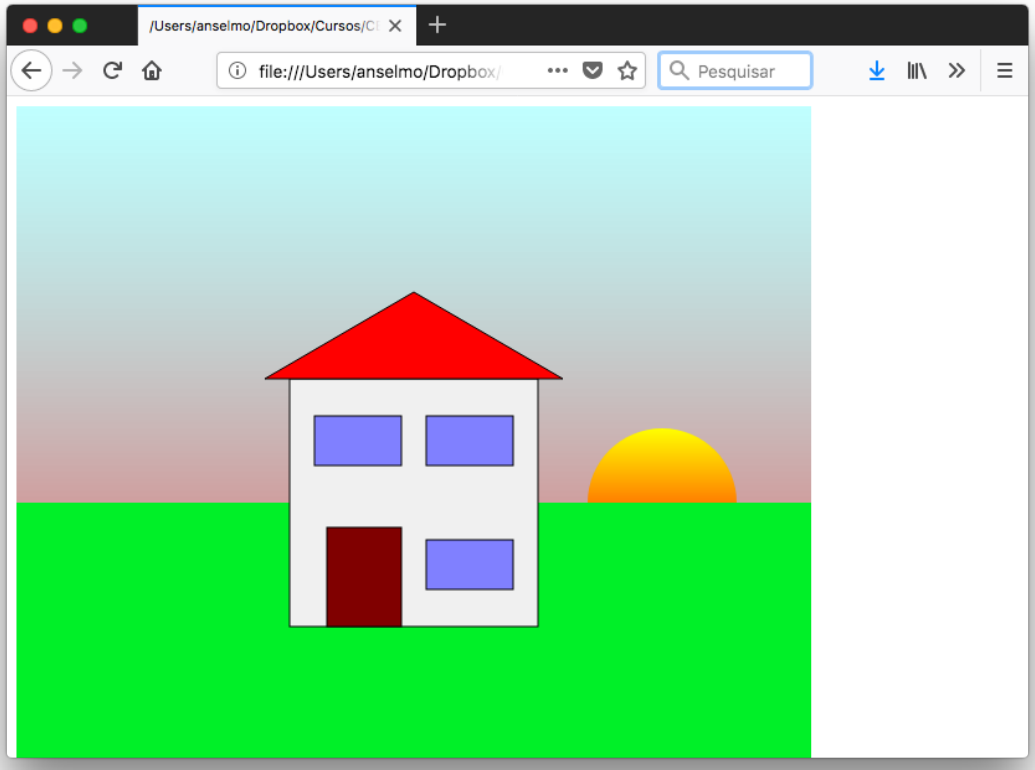
6) Faça uma pesquisa sobre a linguagem de descrição de gráficos SVG. Explique de forma sucinta este formato e ilustre com um exemplo (1.25 ponto).

O SVG é um padrão para especificação de formas vetoriais em navegadores (*browsers*) baseado em XML. Diferentemente do elemento canvas do HTML5, um arquivo de especificação em SVG descreve objetos gráficos que são armazenadas em um grafo de cena para posteriormente serem renderizados em um mapa de bits. Um *grafo de cena* é uma estrutura hierárquica que descreve transformações e objetos de uma cena. Quando um atributo é modificado, o *browser* automaticamente refaz o desenho. No HTML5 canvas, pode-se dizer que o modelo é especificado pela linguagem Javascript e não persiste, uma vez que o desenho é feito. Por outro lado, em um formato SVG, o modelo é mantido em memória na estrutura do grafo de cena.

Uma outra diferença importante é que o SVG permite a associação de tratamento de eventos (*event handlers*) aos objetos gráficos de forma mais natural. Em HTML5 canvas, é necessário fazer o vínculo explicito entre um evento e um objeto, usando programação em Javascript e o conhecimento do tipo de evento e seus parâmetros. Por exemplo, em SVG é possível vincular uma figura ao evento onClick. Em HTML5, deve-se obter as coordenadas (x,y) e checar se elas estão no interior da figura em questão, via programação.

Referências:
https://en.wikipedia.org/wiki/Canvas_element (acessado em 23 de fev. 18 14:33h)
https://developer.mozilla.org/en-US/docs/Web/SVG (acessado em 23 de fev. 18 14:33h)

A seguir segue um exemplo de descrição em SVG



```
<!DOCTYPE html>
<html>
<body>

<svg height="640" width="640">

 <defs>
  <linearGradient id="grad1" x1="0%" y1="0%" x2="0%" y2="100%">
    <stop offset="0%" style="stop-color:rgb(128,255,255);stop-opacity:0.5" />
    <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
  </linearGradient>
 </defs>

 <defs>
  <linearGradient id="grad2" x1="100%" y1="0%" x2="100%" y2="100%">
    <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
    <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
  </linearGradient>
 </defs>

 <rect width="640" height="640" x="0" y="0" fill="url(#grad1)";stroke-width:1;stroke:rgb(0,0,0)" />


 <circle cx="520" cy="320" r="60" stroke="black" stroke-width="0" fill="url(#grad2)" />
 <rect width="640" height="320" x="0" y="320" style="fill:rgb(0,240,40);stroke-width:0;stroke:rgb(0,0,0)" />

 <rect width="200" height="200" x="220" y="220" style="fill:rgb(240,240,240);stroke-width:1;stroke:rgb(0,0,0)" />
 <polygon points="200,220 440,220 320,150" style="fill:red;stroke:black;stroke-width:1" />
 <rect width="60" height="80" x="250" y="340" style="fill:rgb(128,0,0);stroke-width:1;stroke:rgb(0,0,0)" />
 <rect width="70" height="40" x="330" y="350" style="fill:rgb(128,128,255);stroke-width:1;stroke:rgb(0,0,0)" />
 <rect width="70" height="40" x="240" y="250" style="fill:rgb(128,128,255);stroke-width:1;stroke:rgb(0,0,0)" />
 <rect width="70" height="40" x="330" y="250" style="fill:rgb(128,128,255);stroke-width:1;stroke:rgb(0,0,0)" />

 Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

7) Considere um objeto volumétrico onde cada célula armazena um valor de densidade $d_{ijk,}$ $0 \leq d_{ijk} \leq 1$. Descreva um algoritmo que compute o número de regiões conectadas tal que dijk>0.8. Dica: use um algoritmo recursivo semelhante ao *Flood Fill* (https://en.wikipedia.org/wiki/Flood_fill) (1.25 ponto).

```
Algoritmo FloodFill(d, v, i, j, k, cor)
Entrada – d – array de densidades; dx,dy,dz – dimensões de d
          v – array de voxels já coloridos(visitados)
          cor – cor de preenchimento
          i, j, k – coordenadas da célula inicial (semente)
    Se d_ijk>0.8 ∧ v_ijk = 0  ∧ Interior(i,j,k,dx,dy,dz) então
       v_ijk ← cor
       {Considera-se apenas os 6 vizinhos de cada célula para definição da conectividade.}
       {Entretanto é possível considera todos os 26 elementos como vizinhos}
       FloodFill(d, v, i+1, j, k, cor)
       FloodFill(d, v, i-1, j, k, cor)
       FloodFill(d, v, i, j-1, k, cor)
       FloodFill(d, v, i, j+1,k, cor)
       FloodFill(d, v, i, j, k+1, cor)
       FloodFill(d, v, i, j, k-1, cor)
    Fim_se
Fim_Algoritmo
```

```
Algoritmo RegiõesConectadas(d, v, i, j, k, cor)
Entrada – d – array de densidades; dx,dy,dz – dimensões de d
          v – array de voxels já coloridos(visitados)
          cor – cor de preenchimento
          i, j, k – coordenadas da célula inicial (semente)
     Para i=0...dx faça
        Para j=0..dy faça
           Para k=0...dz faça
              Se d_ijk>0.8 ∧ v_ijk =0  então
                   n = n+1
                   FloodFill(d, v, i, j, k, n)
              Fim_se
           Fim_para
        Fim_para
     Fim_para
Fim_Algoritmo
```

No algoritmo FloodFill, o procedimento Interior(...) apenas retorna verdadeiro ou falso, dependendo se os índices i, j e k estiverem contidos no domínio do array.

8) Faça uma pesquisa sobre superfícies de subdivisão. Utilizando o software *Blender*, crie um modelo baseado em subdivisão de superfícies, por exemplo, usando o algoritmo de *Catmull-Clark* (1.25 ponto).

Superfícies de subdivisão é o nome dado a uma técnica de modelagem capaz de gerar superfícies como o limite de um processo de refinamento sucessivo. Elas são vistas como generalização das superfícies do tipo Spline.

O esquemas de subdivisão basicamente funcionam através de um passo de refinamento (introdução de novos vértices na superfície) seguido da aplicação de uma máscara de suavização, que efetua uma média ponderada das coordenadas de vértices originais e dos introduzidos no passo de refinamento. A combinação de ambos os passos, com uma escolha adequada dos pesos da máscara, é que permite que a forma convirja para uma superfície limite, normalmente com um certo grau de suavidade.

As superfícies de subdivisão podem aproximar ou interpolar a superfície inicial original. Nos esquemas que aproximam, a superfície resultante não necessariamente passa pelos vértices originais, em oposição aos esquema interpolantes. O esquema de Catmull-Clark é um esquema de aproximação.

Seja S uma malha de polígonos que descreve uma superfície. O esquema funciona da seguinte forma:

1) Seja f uma face de S, com vértices $v_0, v_1, ..., v_k$, onde k é o número de vértices de f. Cria-se um novo **vértice de face** *vf* cujas coordenadas são dadas pela média das coordenadas dos vértices de f (Figura 2).

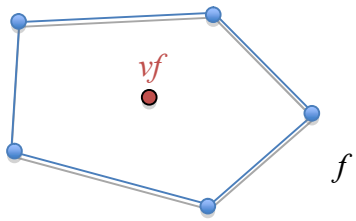


Figura 2 – Inserção de um vértice em uma face

2) Seja e uma aresta de S. Cria-se um novo **vértice de aresta** *ve* cujas coordenadas são dadas pela média das coordenadas dos pontos extremos da aresta e dos vértices vf_1 e vf_2 (introduzidos no passo 1, nas faces incidentes em e)(Figura 3).

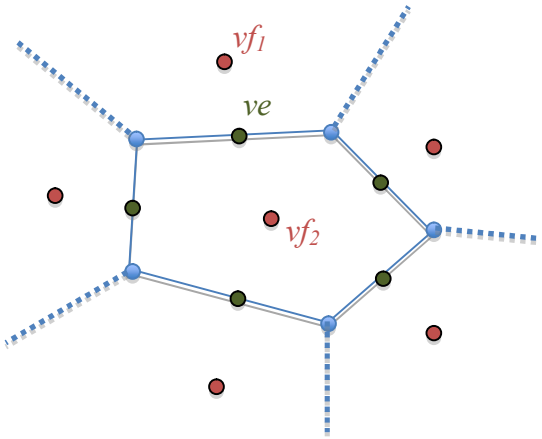


Figura 3 – Inserção de novos vértices em arestas

3) Para cada novo vértices de face *vf* criar arestas conectando *vf* aos novos **vértices de aresta** ve_i criados na face f no qual *vf* foi inserido (Figura 4).

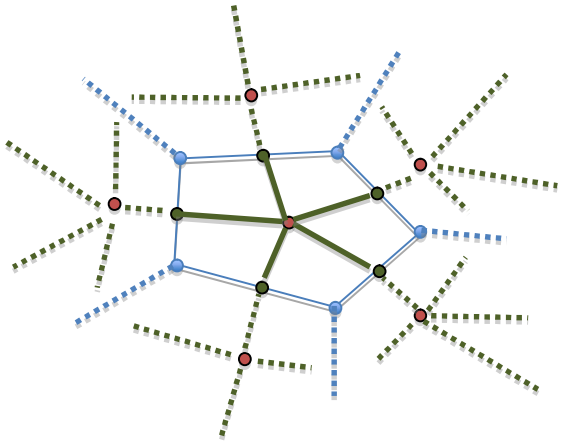


Figura 4 – conexão do vértice de face com os vértices de aresta

4) Para cada vértice original v, definir a média F de todos os novos **vértices de face** (amarelos) criados para as faces incidentes a v (roxo, no exemplo) (Figura 5).

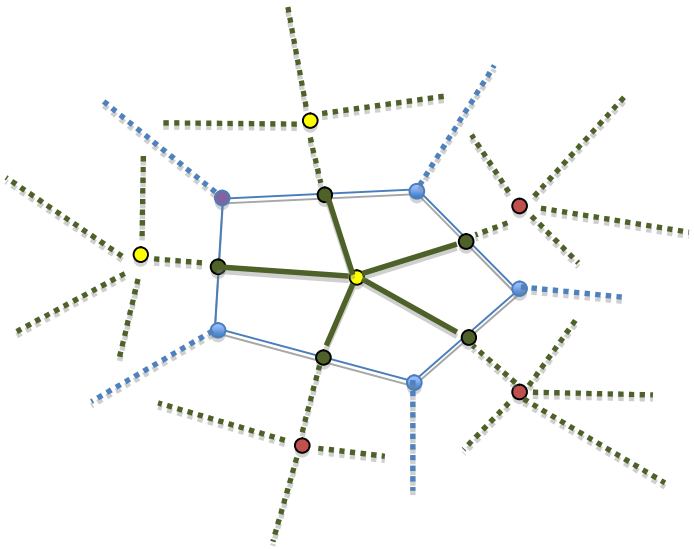


Figura 5 – Média dos novos vértices nas faces incidentes a v.

5) Para cada vértice original v, definir a média R de todos os novos **vértices de arestas** criados (laranjas) para as arestas incidentes a v (vértice roxo) (Figura 6).

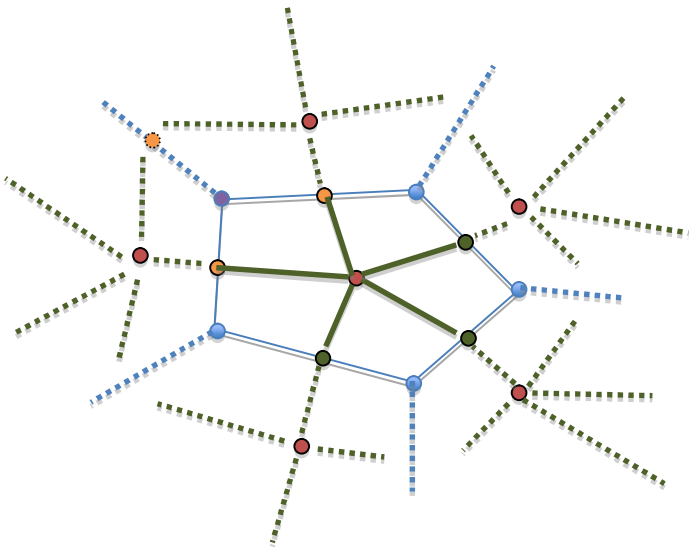
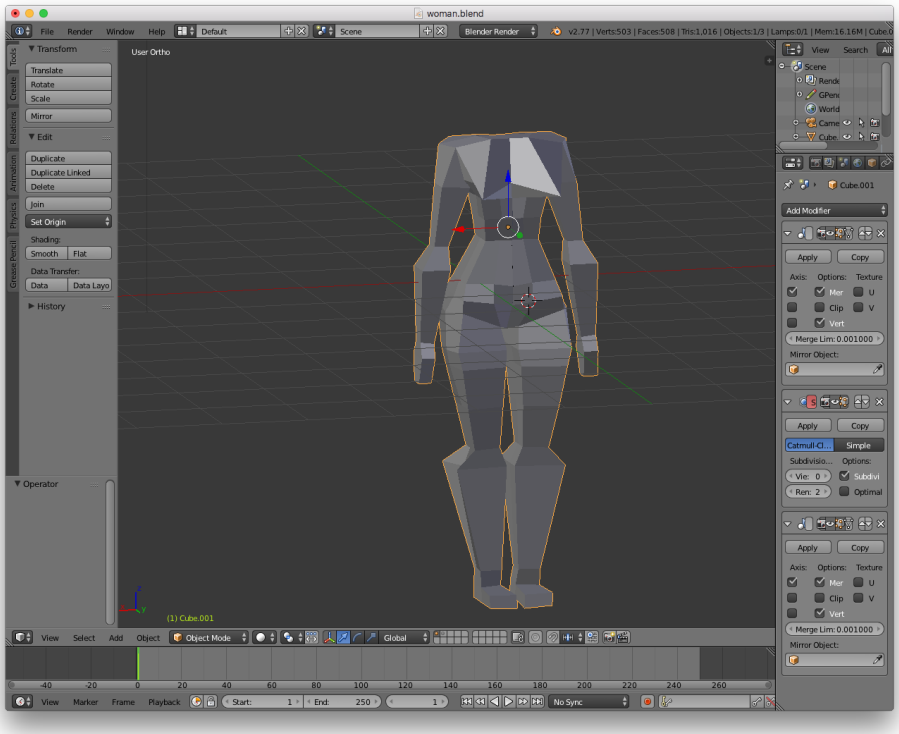


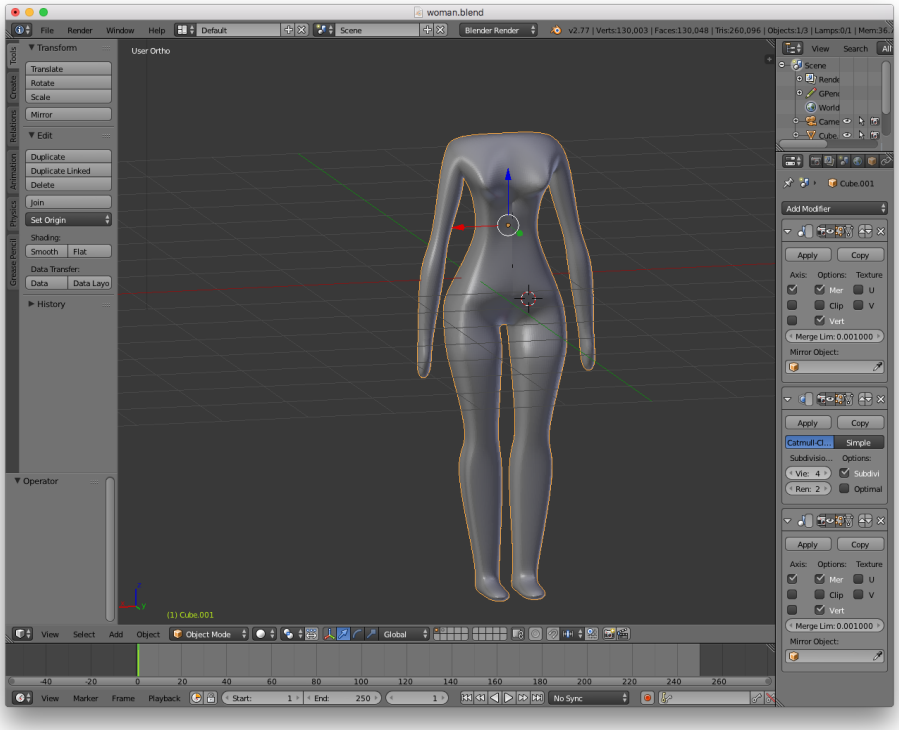
Figura 6 – Média dos vértices inseridos nas arestas incidentes a v.

6) Mover cada vértice original v para a posição dada por $\frac{F+2R-(n-3)P}{n}$, onde P são as coordenadas originas de v.

7) Construir, na estrutura de dados topológica, as faces corretas para as novas arestas criadas.

A seguir, nas Figuras 7 e 8, pode-se observar um modelo original e sua posterior subdivisão utilizando quatro passos do esquema Catmull-Clark usando o software Blender. Observe que a forma final converge para uma superfície suave que aproxima a forma original, mas não a interpola.


woman.blend
File Render Window Help Default Scene Blender Render
User Ortho
Transform
Translate
Rotate
Scale
Mirror
Edit
Duplicate
Duplicate Linked
Delete
Join
Set Origin
Shading:
Smooth Flat
Data Transfer:
Data Data Layo
History
Operator
(1) Cube.001
View Select Add Object Object Mode Global
View Marker Frame Playback Start: 1 End: 250 1 No Sync
Cube.001
Add Modifier
Apply Copy
Merge Lim: 0.001000
Mirror Object:
Catmull-Cl... Simple
Subdivisio... Options:
Optimal


woman.blend
File Render Window Help Default Scene Blender Render
User Ortho
Transform
Translate
Rotate
Scale
Mirror
Edit
Duplicate
Duplicate Linked
Delete
Join
Set Origin
Shading:
Smooth Flat
Data Transfer:
Data Data Layo
History
Operator
(1) Cube.001
View Select Add Object Object Mode Global
View Marker Frame Playback Start: 1 End: 250 1 No Sync
Cube.001
Add Modifier
Apply Copy
Merge Lim: 0.001000
Mirror Object:
Catmull-Cl... Simple
Subdivisio... Options:
Optimal