PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Documentação Curvas – Computação Gráfica

Aluno: Adhonay Júnior Profa.: Rosilane Mota

Ambiente:

- O programa foi desenvolvido utilizando a plataforma NetBeans IDE 8.1 for Java -Developers.

-O sistema operacional utilizado foi o Windows 8.1

Implementação Algoritmos:

Bézier:

A curva de Bézier é uma curva polinomial expressa como a interpolação linear entre alguns pontos representativos, chamados de pontos de controle.

```
public boolean BezierCurvaPri(Ponto pt[], int num, Ponto result[], int n[])
{
   int oldN = n[0];

   if (enough == ENOUGH && num > 6) enough += 3;
     if (enough > ENOUGH) enough -= 5;
   nRecur = 0;
   // em caso de excesso de pilha de recursão, continue tentando
   while (!BezierCurvaSec(pt, num, bpt, n))
   {
      n[0] = oldN;
      enough += 5;
      nRecur = 0;
   }
   return true;
}
```

A função BezierCurvaPri Basicamente faz a chamada principal que seria da função BezierCurvaSec, também é verificado se os pontos são suficientemente possível para chamada da função onde a condição imposta seja maior que seis, assim caso sendo satisfeita, ela chama a função BezierCurvaSec dentro de uma repetição até que a curva seja feita retornando a mesma.

```
protected boolean BezierCurvaSec(Ponto pt[], int num, Ponto result[], int n[])
{
  Ponto   qt[],rt[];        // Dividir
  int d[],i,max;

  nRecur++;
  if (nRecur > RECURSION) return false;

  d = new int[MAXCNTL];
  for (i=1;i<num-1;i++) d[i]=Distancia(pt[i],pt[0],pt[num-1]);
  max=d[1];
  for (i=2;i<num-1;i++) if (d[i]>max) max=d[i];
  if (max <= enough || nRecur > RECURSION) {
       if (n[0]==0) {
          if (bnum > 0)
             result[0].copy(pt[0]);
          else
             result[0] = new Ponto(pt[0]);
          n[0]=1;
       }
       //reuse
       if (bnum > n[0])
             result[n[0]].copy(pt[num-1]);
       else
             result[n[0]] = new Ponto(pt[num-1]);
       n[0]++;
       if (n[0] == MAXPOINT-1) return false;
  }
  else {

       qt = get_buf(num);
       rt = get_buf(num);
       DivCurva(pt,qt,rt,num);
       if (!BezierCurvaSec(qt,num,result,n)) return false;
       put_buf(qt);
       if (!BezierCurvaSec(rt,num,result,n)) return false;
       put_buf(rt);
  }
  return true;
}
```

A função BezierCurveSec recebe uma estrutura ponto (onde e tratado todos pontos pelo clique do mouse é os pontos de controle para modificação da curva) e um valor para o parâmetro result, e um valor para o tamanho da lista (em número de pontos). A lista é então implementada como uma curva de Bézier, para desenhar superfícies e chamada recursivamente passando o localização dos pontos desejados através do click do mouse , o tamanho da lista de pontos e a lista result com o resultado de todos eles.

Interpolada:

A curva interpolada foi tradada com o polinômio de lagrange(consiste em aproximar uma função contínua e definida no intervalo [0,+1], f ( ) x , por um polinômio de grau (m-1)) .

```
switch (mode){
case(INTERPOLADA):
    g.setColor(Color.RED);

  if (np>1){
    float a[] = new float[np];
    float t1;
    float t2;
    float h[] = new float[np];
    for (int i=1; i<=np-1; i++){
      h[i] = x[i] - x[i-1];
    }
    if (np>2){
      float sub[] = new float[np-1];
      float diag[] = new float[np-1];
      float sup[] = new float[np-1];

      for (int i=1; i<=np-2; i++){
        diag[i] = (h[i] + h[i+1])/3;
        sup[i] = h[i+1]/6;
        sub[i] = h[i]/6;
        a[i] = (d[i+1]-d[i])/h[i+1]-(d[i]-d[i-1])/h[i];
      }
      SistemaLi(sub,diag,sup,a,np-2);
    }

    oldt=x[0];
    oldy=d[0];
    g.drawLine((int)oldt,(int)oldy,(int)oldt,(int)oldy);
    for (int i=1; i<=np-1; i++) {   // loop entre intervalos entre nós
      for (int j=1; j<=precision; j++){
        t1 = (h[i]*j)/precision;
        t2 = h[i] - t1;
        y = ((-a[i-1]/6*(t2+h[i])*t1+d[i-1])*t2 +
             (-a[i]/6*(t1+h[i])*t2+d[i])*t1)/h[i];
        t=x[i-1]+t1;
        g.drawLine((int)oldt,(int)oldy,(int)t,(int)y);
        oldt=t;
        oldy=y;
      }
    }
```

Foi feita a chamada do método dado como SistemaLi que e tratado a resolução da matriz, para a utilização do mesmo, passando os respectivos parâmetros para solução da mesma , após o cálculo dela e realizado o

desenho da linha onde e passado uma “precision” desejada x para o desenho da curva através dos pontos destacados pelo usuário final com suas respectivas coordenas no plano.

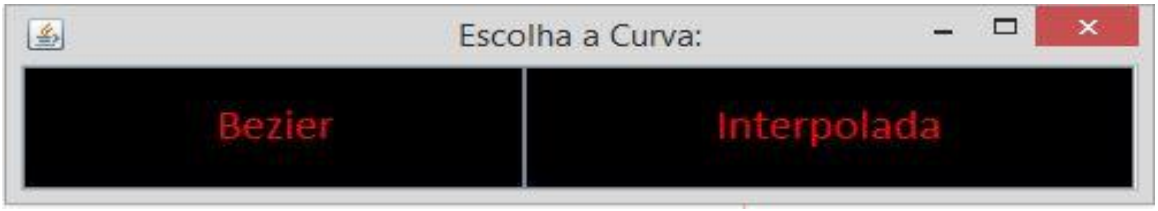
```
private void SistemaLi(float sub[], float diag[], float sup[], float b[], int n){
                sistema linear linear com tridiagonal n por matriz n
                 usando a eliminação gaussiana * sem * rotação
                 onde a (i, i-1) = sub [i] para 2 <= i <= n
                         a (i, i) = diag [i] para 1 <= i <= n
                         a (i, i + 1) = sup [i] para 1 <= i <= n-1
                 (os valores sub [1], sup [n] são ignorados)
                 o vetor do lado direito b [1: n] é substituído pela solução
                 NOTA: 1 ... n é usado em todos os arrays, 0 é inutilizado
                  OU SEJA METODO DE ESCALONAMENTO COMECANDO PELA PRIMEIRA COLUNA */
  int i;
               factorização e substituição*/
  for(i=2; i<=n; i++){
    sub[i] = sub[i]/diag[i-1];           // encontrar multiplicador
    diag[i] = diag[i] - sub[i]*sup[i-1]; // atualizar diagonal superior. matriz
    b[i] = b[i] - sub[i]*b[i-1];         //atualizar o vetor rhs
  }
  b[n] = b[n]/diag[n];                   // resolver o diagonal superior. sistema por trás subst.
  for(i=n-1;i>=1;i--){
    b[i] = (b[i] - sup[i]*b[i+1])/diag[i];
  }
}
```

A ideia foi gerar o polinômio interpolador, obtido através de uma matriz cuja resolução do sistema linear é simples e direta, para isso foi usado o método SistemaLi que e chamado acima para montagem da curva , para o método em questão foi utilizada a lógica de eliminação de Gauss (ou método de escalonamento) é um algoritmo para se resolver sistemas de equações lineares. Este método consiste em aplicar sucessivas operações elementares num sistema linear, para o transformar num sistema de mais fácil resolução, que apresenta exatamente as mesmas soluções que o original.

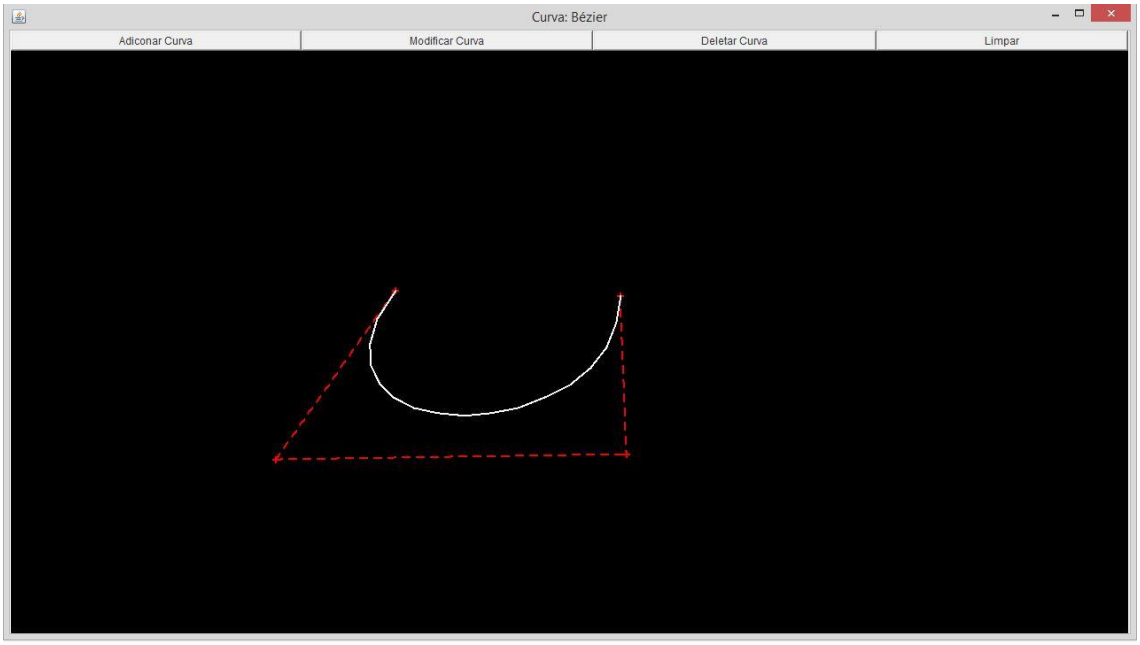
Modo de uso:

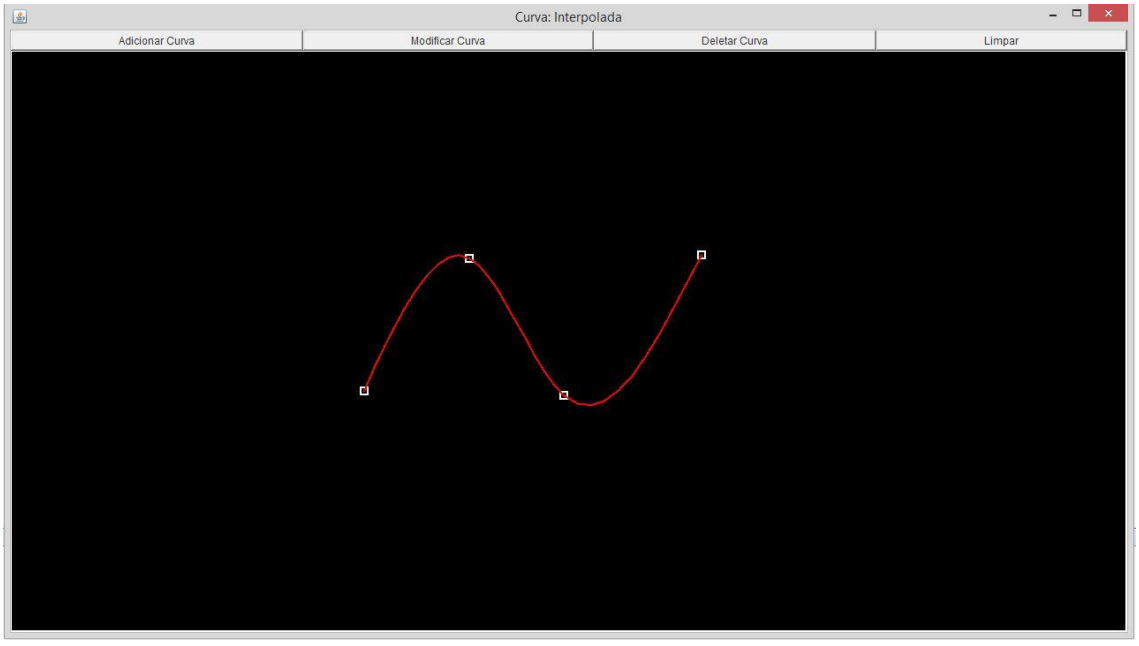
As execuções dos algoritmos estão sendo demostradas no Slide “Apresentação Slide” para melhor entendimento, abaixo segue como usar o programa:

Após a compilar e executar o algoritmo será exibida a seguinte tela para escolher a curva desejada:



Após selecionar a curva em questão será exibido um outro painel respectivo a curva desejada, abaixo segue as duas possíveis opções e com um exemplo das curvas já desenhadas. (Clique na imagem para mostrar sua execução)





Independente da curva selecionada o painel terá as mesmas opções, mas com propósitos diferentes para o botão “Deletar curva” que no caso da bézier e excluído toda curva e no da interpolada e excluído o ponto que o usuário clicou.

O botão “Adicionar Curva” basicamente após clicar nele, basta selecionar os pontos no painel onde deseja que as curvas sejam criadas, no caso da interpolada o programa já vai mostrando a geração da curva a cada ponto clicado, já para a bézier e mostrado após o último clique, quando você for selecionar no painel o último ponto que deseja basta dar um duplo clique no botão esquerdo no local onde deseja finalizar para mostrar a curva desejada, que será gerada automaticamente após essa ação.

O botão “Limpar” basicamente serve para apagar todas curvas desenhas no painel para ambos os casos.

O botão “Modificar Curva” serve para arrastar os pontos já plotados no painel, ou seja, com a curva já desenhada, caso queria modificar a localização de algum monto ou da curva toda, basta clicar nos “quadradinhos” da curva no caso da bézier os pontos de controle, para arrastar os mesmos.

O programa permite que você ultrapasse 4 pontos como realizado na apresentação e nos exemplos (caso deseje apenas 4 pontos basta clicar com o botão direito para finalizar no mesmo).