Universidade Federal de Santa Maria
Departamento de Computação Aplicada
Prof. Cesar Tadeu Pozzer
Disciplina: Computação Gráfica
pozzer@inf.ufsm.br
17/02/2024

Introdução

Computação Gráfica?

Computação gráfica é uma área da computação que trabalha com assuntos relacionados à geração de imagens e modelos geométricos. Está totalmente fundamentada em princípios matemáticos e físicos, e faz uso de estruturas de dados para que os dados possam ser manipulados para gerar o resultado desejado.

Faz uso de inúmeros fundamentos e áreas da matemática, como geometria analítica, álgebra linear, cálculo, métodos numéricos e processamento de sinais. É a aplicação prática de vários conteúdos teóricos estudados durante a graduação em computação.

Para a computação gráfica, todos os processos são aproximados (não são matematicamente precisos), pois se utilizam cálculos com precisão limitada. Erros mínimos de precisão não geram nenhum efeito negativo nas imagens geradas.
Exemplos:

1. Uso de funções trigonométricas.
2. Representação de números reais com float ou double (precisão limitada).
3. Cálculo de vetores tangentes a curvas. Não são necessárias soluções analíticas de derivadas parciais. Aproximações numéricas podem ser mais que suficientes.
4. Representação da cor de pixeis usando valores inteiros no monitor.
5. Coordenadas inteiras dos pixeis na tela.

Os processos físicos são ainda mais simplificados, como por exemplo, cálculos de iluminação, simulação de fluídos e simulação de movimentação física. É impossível (e sempre será) fazer uma simulação computacional com o mesmo nível de detalhes do mundo real.

O que é preciso para se ter sucesso na disciplina?

A programação de aplicativos gráficos costuma ser muito diferente das demais cadeiras de cursos de Computação. Como é muito fundamentada em matemática, erros mínimos ou formulações erradas levam a resultados e comportamentos totalmente inesperados. Apesar de ter enfoque muito matemático, é totalmente diferente de cadeiras da matemática onde o objeto é resolver listas exercícios usando as mesmas regras (Ex: lista de exercícios de derivadas). Em computação gráfica cada problema é diferente dos demais, pois são problemas do mundo real. E cada problema exige uma solução e técnicas diferentes.
Exemplos:

1. Técnicas de animação de pessoas
2. Técnicas de animação de cabelo
3. Técnicas para representação de curvas
4. Técnicas de iluminação para água
5. Técnicas para visualização 3D

A prática e o formalismo matemático são as únicas soluções para se chegar ao resultado desejado. As soluções matemáticas devem ser genéricas para tratar todos os casos. Deve-se implementar cada aspecto do conteúdo da disciplina individualmente, por mais simples que pareçam.
Exemplos:

1. Desenho de um círculo com coordenadas polares
2. Curva de Bezier com 3 e 4 pontos de controle fixos (solução bem Ad hoc)
3. Rotação de uma linha em 2D

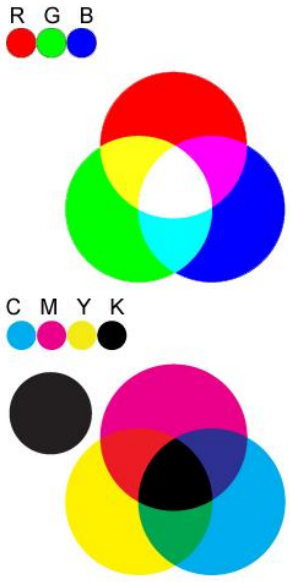
Problemas mais completos tornam-se quase impossíveis de serem solucionados sem se ter conhecimento e experiência na implementação de problemas simples. Muitos problemas tem solução com 100 linhas de código, porém **duas** dessas 100 linhas podem requerer mais tempo de programação que as outras 98. É nessas duas linhas que o aluno deve demostrar o grau de entendimento do problema.

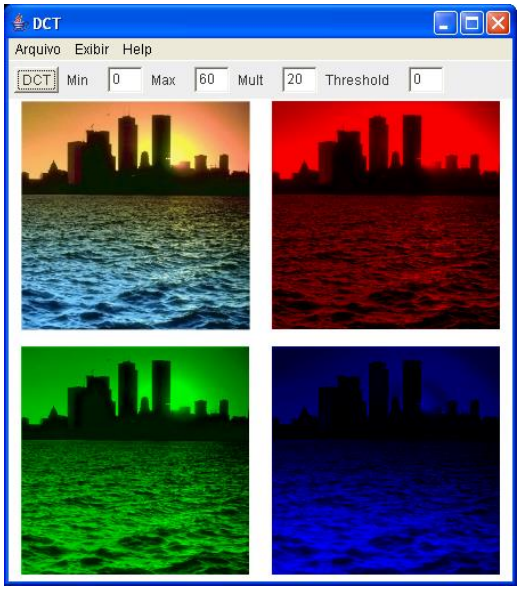
É uma disciplina divertida?

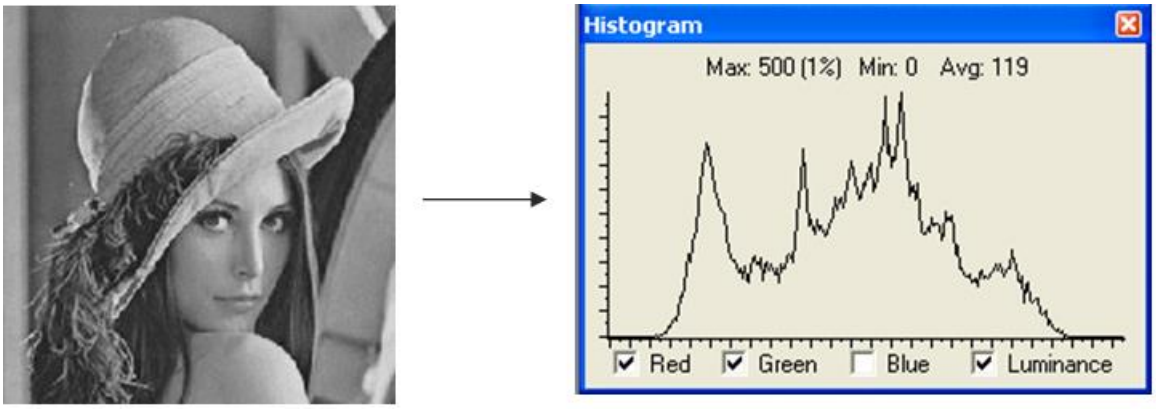
A disciplina de computação de computação gráfica é muito focada em problemas comuns no mundo real e grande parte dos problemas tem solução visual (como pode ser visto ao longo deste material). Ao contrário de muitas cadeiras que são fundamentalmente teóricas e simbólicas, computação gráfica permite a criação de programas interativos e visuais, com interfaces gráficas por onde o usuário interage com a solução gráfica. Isso torna a cadeira divertida (**quando tudo funciona como esperado**) e dá ao aluno a possibilidade de criação de jogos, efeitos especiais, simulações, dentre outros.

1. Overview do que será visto no curso (e programado)

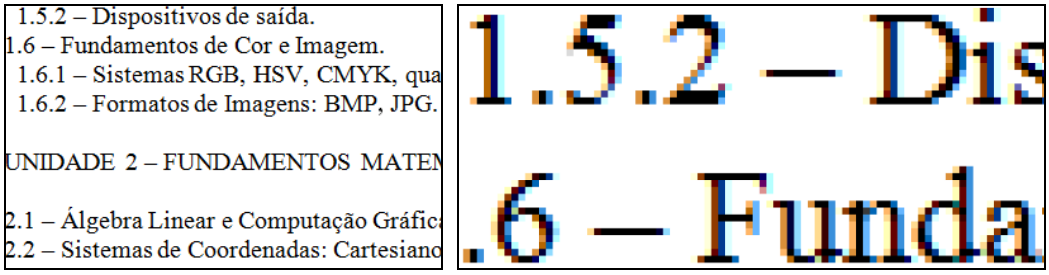
Fundamentos de Imagem e Cor



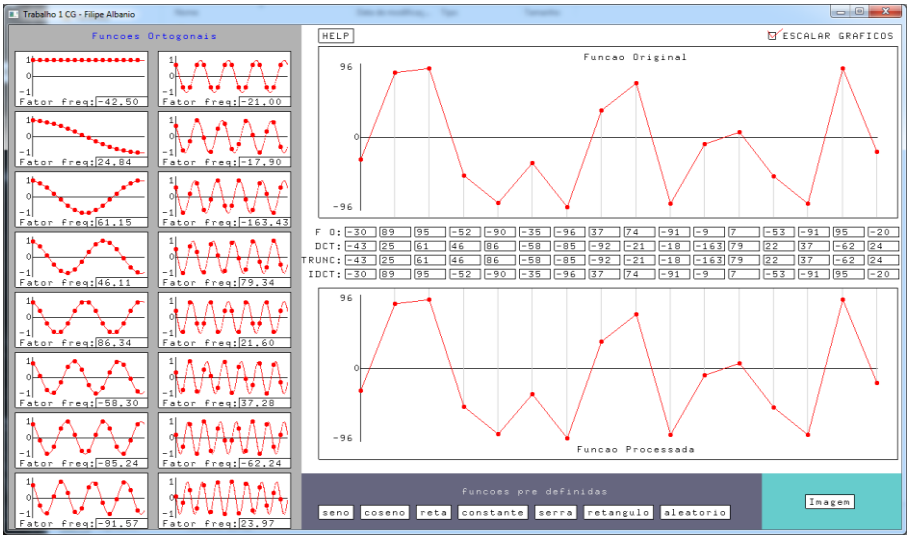




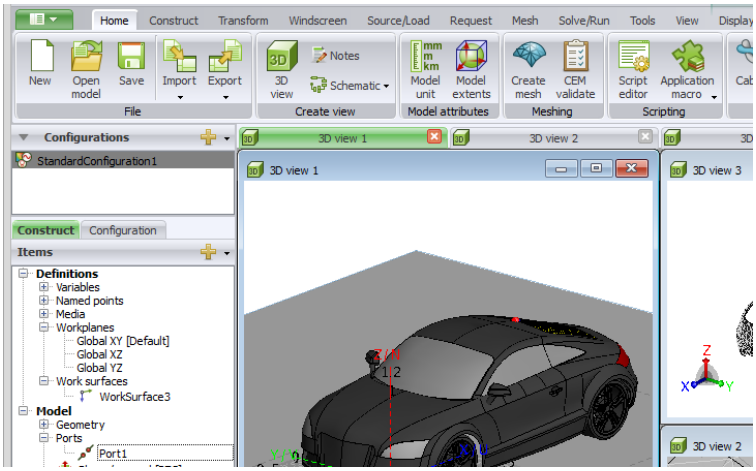
Algoritmos de Anti-aliasing



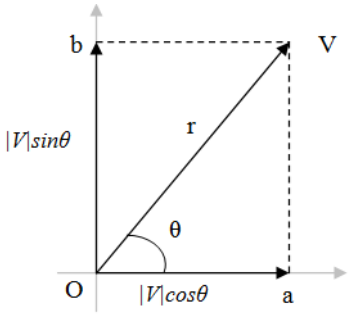
Transformada Cosseno – Processamento de sinais

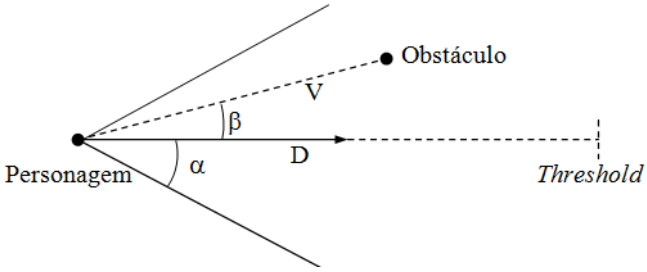


Interfaces gráficas

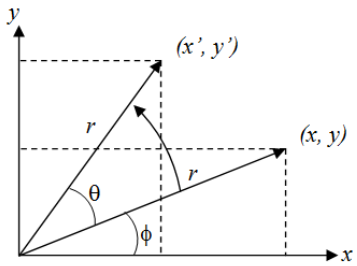


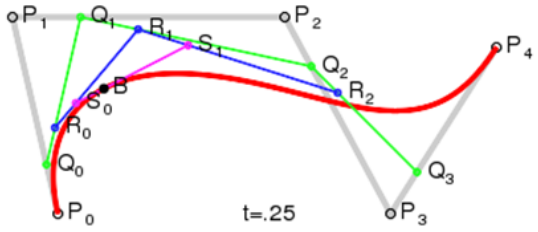
Base matemática

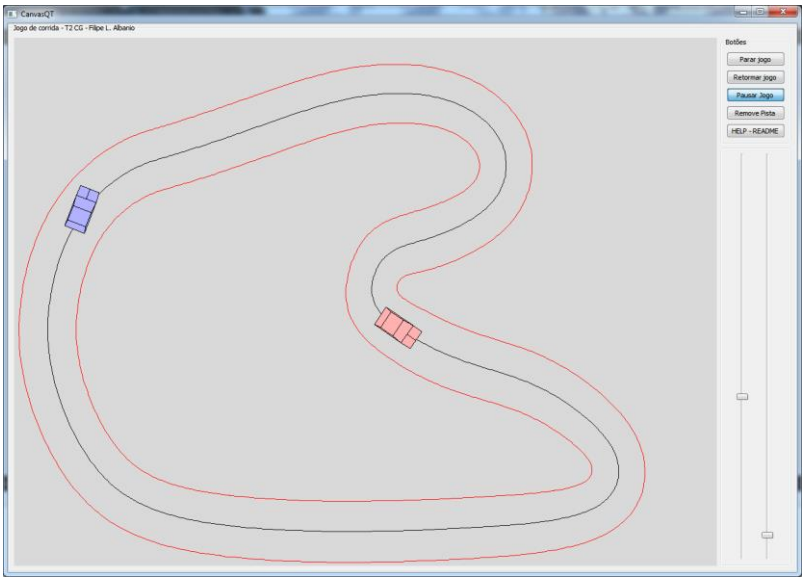




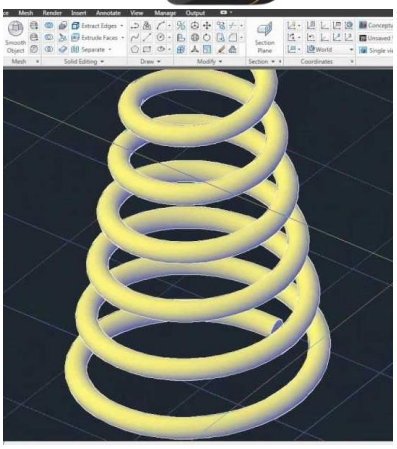
Computação Gráfica 2D

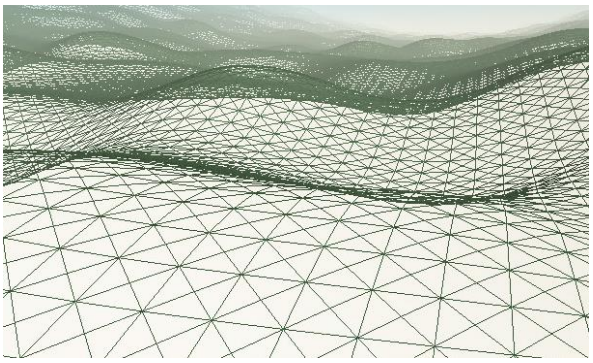


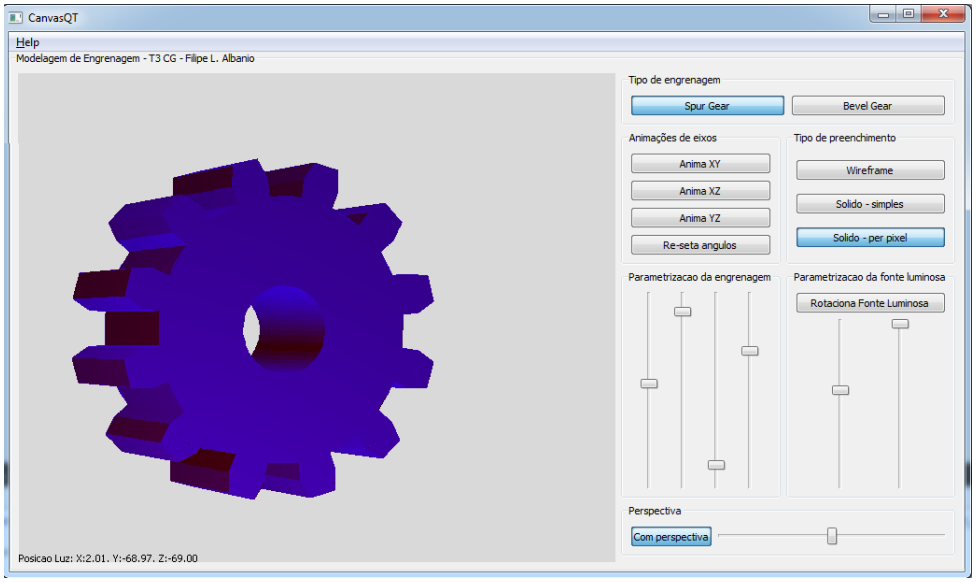




Computação Gráfica 3D







Síntese de Imagens

2. Visão geral dos Sistemas Gráficos

- CAD (*Computer-Aided Design*): projetos de engenharia para modelagem e visualização de peças, simulações de ferramentas e máquinas, plantas de casas, maquetes de edifícios;
- Apresentação de dados: gráficos, estatísticas;
- Arte computacional: Logomarcas, trabalhos artísticos, propaganda, efeitos especiais;
- Entretenimento: filmes, jogos eletrônicos;
- Educação e treinamento: cabines de simulação de vôo, etc;
- Visualização: visualização científica, funções matemáticas;
- Processamento de Imagens.

Em todas estas áreas, a CG permite visualizar objetos que:

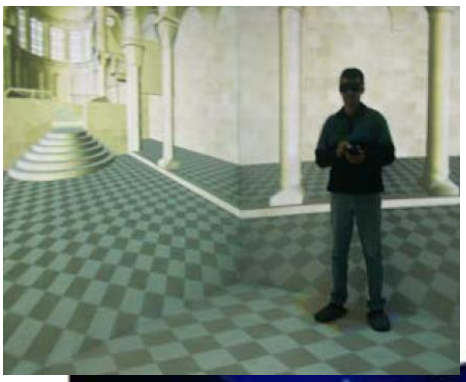
- ainda em construção
- estão fora do alcance da percepção visual
- Fogem da nossa realidade tridimensional

3. Dispositivos gráficos

- **Entrada**
  - Mouse 2D/3D
  - Teclado
  - Joystick
  - Spaceball – 6 graus de liberdade: 3 de rotação e 3 de translação (**Six degrees of freedom** (**6DOF**)) (http://www.3dconnexion.com/products/3a2.php)
  - Luva
  - Mesa e caneta digitalizadora (2/3D)
  - Scanner

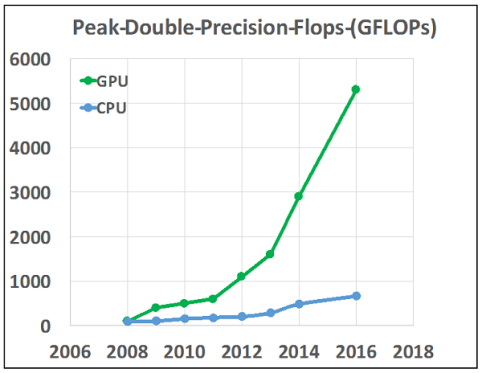
- Tela sensível ao toque

- **Saída**
  - Monitor + placa de vídeo: CRT, LCD
  - Impressora: laser, jato de tinta, BW/cor
  - Óculos (Relacionado principalmente com Realidade Virtual)
  - Canhão



4. Processamento gráfico

- Software (CPU) + hardware (GPU)



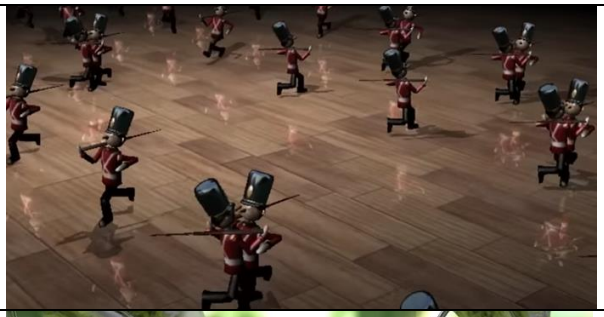
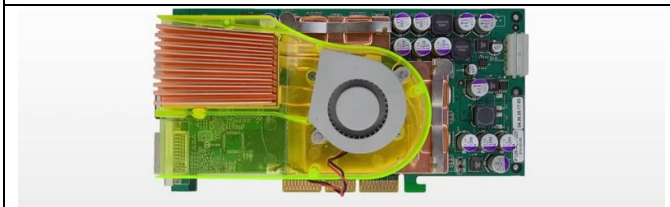
Fabricantes

- **Workstations (passado)**
  - Silicon Graphics (http://www.sgi.com/)
  - Sun (http://www.sun.com/)
- **PCs**
  - Nvidia (http://www.nvidia.com/page/home.html)
  - ATI (http://www.ati.com/)
  - Intel

5. Evolução da Área

- **Hardware**
  - Monitores
  - Placas gráficas
  - Outros dispositivos
- **Técnicas**
  - 1978 - Mapeamento de texturas (Blinn).
  - 1980 - Primeiro *Ray-tracing* (Whitted).
  - 1992 - Surge a linguagem de programação OpenGL
- **Aplicações**
  - Realidade Virtual
  - Jogos, interatividade
  - Realismo
  - Filmes

5.1 Evolução das placas de vídeo Nvidia.

 NVIDIA GEFORCE 2 GTS	
 NVIDIA GEFORCE3	
 NVIDIA GEFORCE4 Ti 4600	
 NVIDIA GEFORCE FX 5800 ULTRA	
 NVIDIA GEFORCE 6800 ULTRA	

 NVIDIA GEFORCE 7800 GTX	
 NVIDIA GEFORCE 8800 GTX	
 NVIDIA GEFORCE GTX 280	
 NVIDIA GEFORCE GTX 480	
 NVIDIA GEFORCE GTX 580	
 NVIDIA GEFORCE GTX 680	

 GTX780 NVIDIA GEFORCE GTX 780	
 GTX980 NVIDIA GEFORCE GTX 980	 UNITED STATES
 GTX 1080 NVIDIA GEFORCE GTX 1080	 29 900
 RTX 2080 NVIDIA GEFORCE RTX 2080	
 NVIDIA GEFORCE RTX 3080	
 RTX 4090	

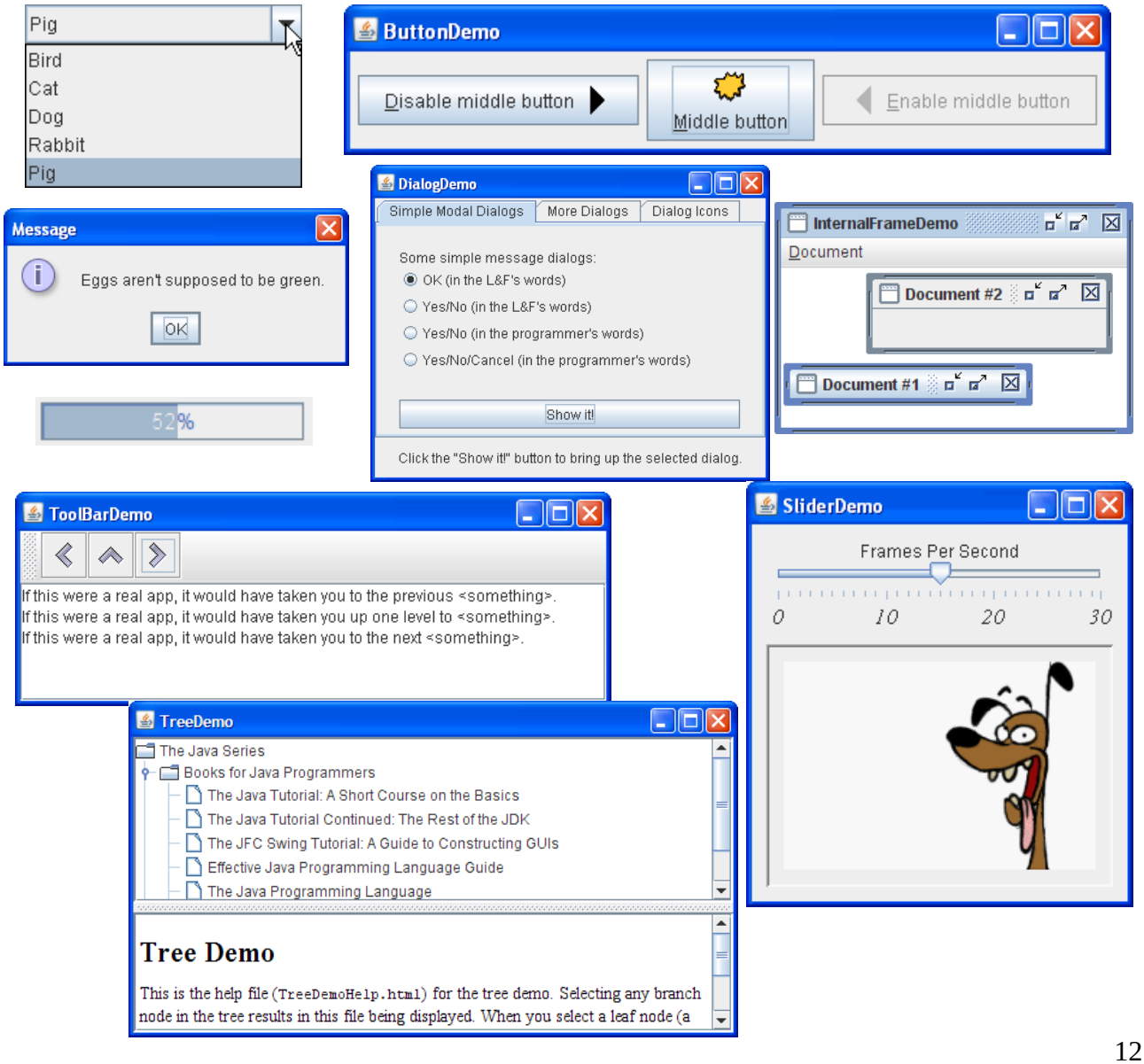
6. Interfaces Gráficas

Além de permitirem a exibição de informação gráfica como imagens, animações, etc, um dos principais objetivos de interfáces gráficas é auxiliar o usuário na **interação** com a aplicação. Surgiram para abolir a utilização de consoles textuais e passagem de argumentos por linha de comando. Temos como exemplos a interface gráfica do Windows.

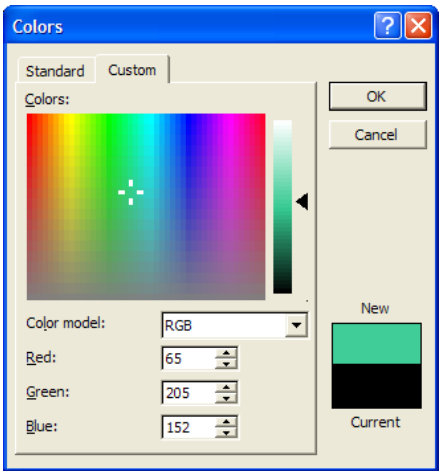
Uma interface gráfica pode disponibilizar ao programador uma simples canvas (área de desenho), como no caso da Canvas 2D (http://www-usr.inf.ufsm.br/~pozzer/), bem como também Widgets (componentes gráficos de interface) que permitem uma fácil manipulação de informação e eventos de interação com o usuário. Sob a canvas, o prograd or pode, se jugar necessário, desenvolver seus próprios widgets personalizados.

Ao se utilizar APIs gráficas, geralmente um grande número de widgets já estão disponibilizados. São exemplos de APIs para a linguagem C/C++ QT (http://www-usr.inf.ufsm.br/~avelar/tutorial_qt/), Fox, GTK, xwWidgets (http://www.vconrado.com/wx/), dentre outras. Para a linguagem Java, por exemplo, pode-se fazer uso do Toolkit Swing. O Swing foi desenvolvido para prover um conjunto mais sofisticado de componentes de interface (GUI – *Graphical User Interface*) do que o AWT (*Abstract Windows Tookit*).

Nas seguintes figuras ilustra-se alguns exemplos de widgets do Toolkit Swing do Java (http://72.5.124.55/docs/books/tutorial/uiswing/components/label.html).

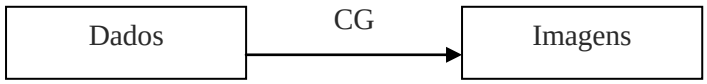


Combinando-se os diversos widgets, pode-se construir interfaces mais avançadas, como a usada para fazer a escolha de cores no modelo HSL, usada pelo programa Word da Microsoft.

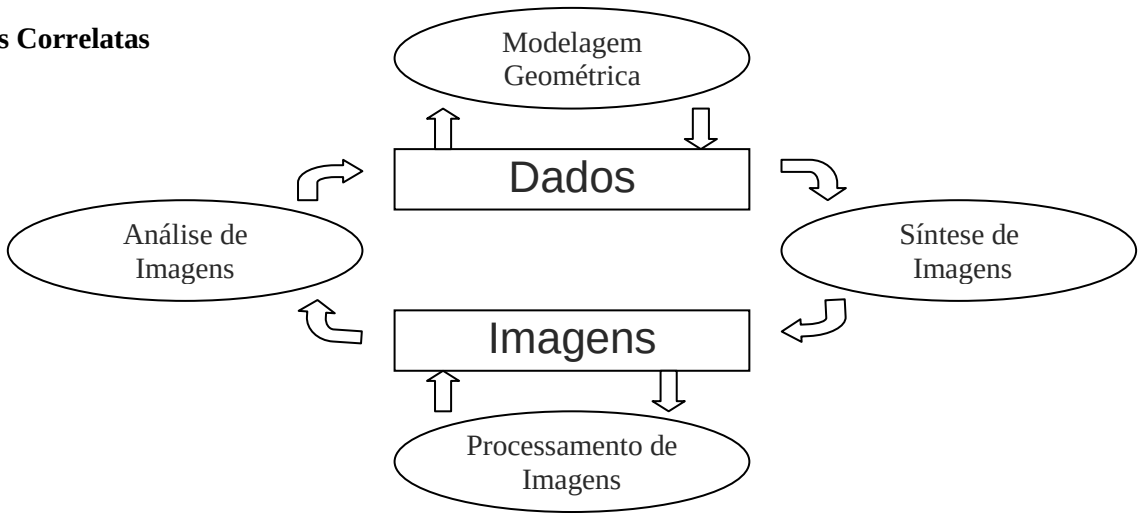


7. CG – Computação Gráfica

- Teve início na década de 50.
- Definição: "conjunto de métodos e técnicas para transformar dados em imagens através de um dispositivo gráfico".



Áreas Correlatas

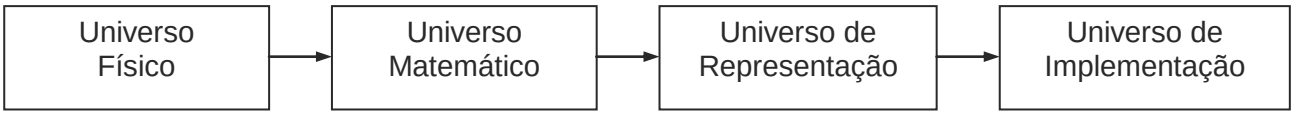


8. Paradigmas de Abstração

Na matemática aplicada que faz uso de métodos computacionais, como em CG, um paradigma de abstração (quatro universos) que se aplica em geral consiste em estabelecer quatro universos (conjuntos). Este paradigma se baseia no fato de que para estudar um determinado fenômeno ou objeto do mundo real no computador, associamos ao mesmo um modelo matemático, em seguida procuramos uma representação finita do modelo associado que seja passível de implementação [Velho].

- Universo Físico: objetos do mundo real
- Universo Matemático: descrição abstrata do mundo físico
- Universo de Representação: descrições simbólicas e finitas

- Universo de Implementação: uso de estruturas de dados para representar descrições do universo de representação.

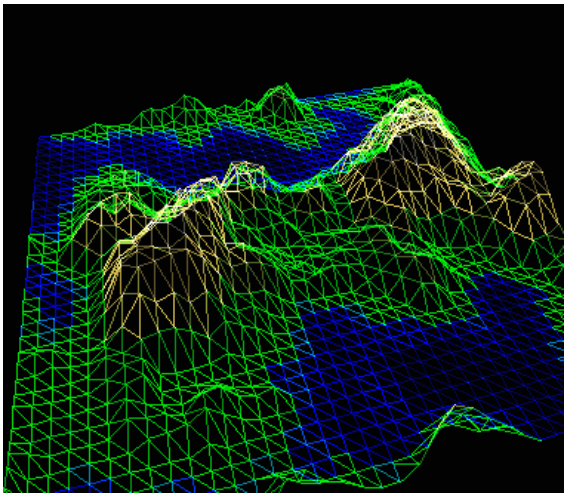


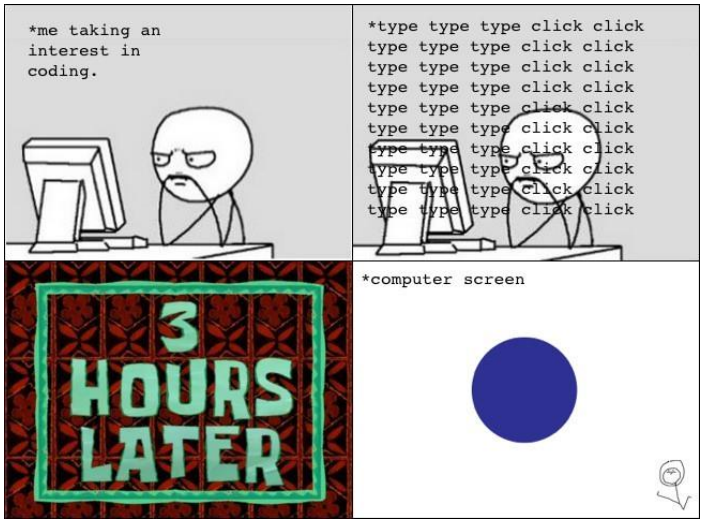
Exemplo: Representação de um terreno (Imagem)

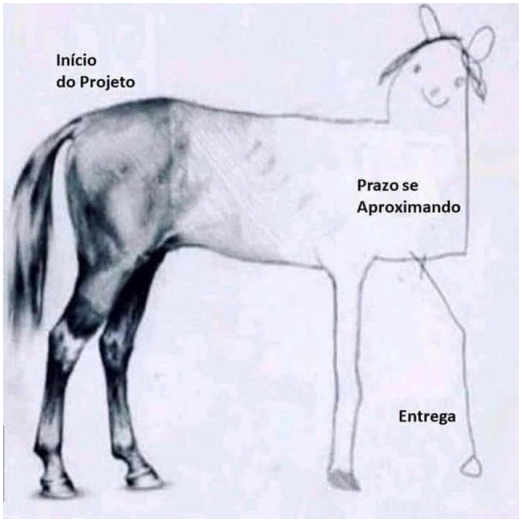
- Pode ser definido por um mapa de alturas. Cada ponto tem uma elevação.
- No universo matemático, o mapa de alturas (*height field*) define uma função
  $F : U \subset R^2 \rightarrow R,\ z = f(x, y)$
  onde (x,y) são as coordenadas do plano e z a altura corresponde.
- Geometricamente o terreno é descrito pelo gráfico da função de alturas f

$$G(f) = \{(x, y, f(x, y))\}$$

- Para **representar** o terreno, pode-se utilizar um reticulado de pontos *(xi, yj)*, obtido pelo produto cartesiano dos conjuntos *Px* e *Py*, onde
  Px = {x0 < x1 <...< xn} e
  Py={y0 < y1 < ... < yn}
  U = [a, b] x [c, d], a ≤ x ≤ b
- A representação do terreno é dada pela matriz de alturas *zij=f(xi, yj)*, onde (xi, yj) representa um vértice do reticulado.
- Esta representação é chamada representação por amostragem uniforme.
- Para **implementação**, pode-se utilizar uma matriz da linguagem C: int matriz[lin][col].







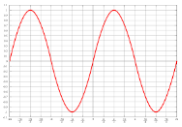
9. Lista de implementações mandatórias para a cadeira de Computação Gráfica

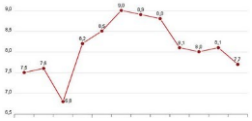
Utilizando a CanvasGlut (Disponível no site da disciplina), faça:

1. Faça um programa para carregar/exibir uma imagem colorida em formato BMP. Após, implemente funções para:
    a. Gerar o histograma de cada canal.
    b. Converter RGB para Luminância.
    c. Gerar imagem com cada canal de cor separado.
    d. Fazer rotação na imagem em 90 graus.
    e. Aumentar e diminuir o brilho da imagem.
    f. Reescalar imagem
    g. Salvar a nova imagem em disco.
2. Em relação a vetores:
    a. Implemente um relógio analógico com ponteiros para hora/minuto/segundo usando coordenadas polares.
    b. Fazer um veículo (quadrado) andar na tela com o uso de vetores controlado com o uso do teclado (setas). Usar controle de FPS.
    c. Implemente um cano de canhão que aponta continuamente na direção do cursor do mouse. Deve-se imprimir na tela o ângulo formado com o eixo x.
3. CG 2d
    a. Fazer a rotação de um quadrado na tela em relação ao centro e em relação a um canto.
    b. Desenhar um círculo com aproximação por linhas.
    c. Implementar o algoritmo de flood fill.
    d. Implementar uma curva de Bezier que passa por 4 pontos de controle estáticos. Plotar as blending functions.
    e. Implementar uma curva de B-spline que passa por 6 pontos de controle estáticos. Plotar as blending functions.
4. CG 3D
    a. Fazer a visualização de um cubo com projeção perspectiva e ortográfica em wireframe.
    b. Fazer a visualização de superfícies Bezier e B-Spline definidas por 16 pontos de controle com projeção perspectiva e ortográfica em wireframe.
5. OpenGL
    a. Explorar os tipos de primitivas do OpenGL
    b. Programa para fazer a visualização de um cubo com uso de gluLookAt, glRotate, glTranslate, glNormal em wireframe e com preenchimento.
    c. Fazer a aplicação/visualização de uma textura sobre um GL_QUAD.

10. Outras sugestões de exercícios e problema de pesquisa

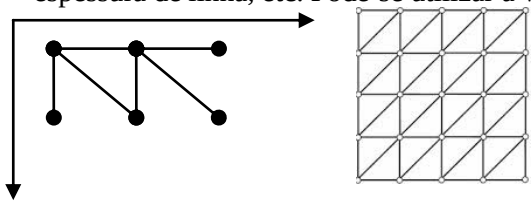
Aquecimento:

1. Programa para desenho de um quadrado verde.
2. Programa para movimentação lateral de um quadrado com o uso das setas direcionais do teclado.
3. Programa para movimentação de um círculo com o uso do mouse.
4. Programa para plotar polinômios. Ex: $3x^3 - 10x^2 + 5x + 4$
5. Programa para plotar as funções seno, cosseno e tangente. 
6. Programa para desenho de gradientes de cores. 

7. Programa para desenhar um alvo com círculos concêntricos. 
8. Programa para entrada de dados (amostras) com o uso do mouse. Ao mover o cursor, a coordenada y do mouse deve ser armazenada num vetor indexado pela coordenada x do mouse. 

Mais interessantes:

1. Faça um programa em C/C++ para gerar/ler um arquivo binário que armazena um reticulado de pontos. Cada ponto deve conter um valor que representa uma altura no intervalo [0, 255]. O arquivo deve ter um header que indica as dimensões do reticulado. Após, faça um programa para ler o arquivo gerado no exercício anterior e renderizar na tela, em 2D, a malha poligonal em wireframe, como mostrado na seguinte figura. O programa pode permitir escolha de cor, zoom, estilo de linha, espessura de linha, etc. Pode-se utilizar a valor de cada ponto para definir a cor de desenho da malha.



2. Implementar uma calculadora com um display digital. Os botões (dos números) devem ser pressionados com o uso do mouse. Os dígitos do display devem ser feitos com 7 linhas ou com 7 polígonos. 

Jogos:

1. Implementar uma versão simplificada do jogo Pac-man sem inimigos (somente a movimentação do avatar do jogador) 
2. Implementar o jogo Space Invaders sem uso de IA. 
3. Implementar o jogo da velha sem IA: apenas o tabuleiro e inserção de jogadas alternadas entre O e X. Quando for feita uma 

Interfaces gráficas:

1. Estude e desenvolva aplicativos simples com uso das seguintes GUIs: QT (https://www.qt.io/) e Dear ImGui (https://github.com/ocornut/imgui).

Bem complicados e trabalhosos

1. Implementar um emulador do videogame Atari
2. Implementar um emulador de um videogame mais moderno que o Atari.