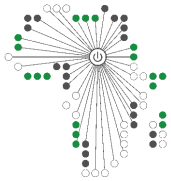


Universidade Virtual Africana

INFORMÁTICA APLICADA: CSI 4303

COMPUTAÇÃO GRÁFICA

Fernando Sanha

Prefácio

A Universidade Virtual Africana (AVU) orgulha-se de participar do aumento do acesso à educação nos países africanos através da produção de materiais de aprendizagem de qualidade. Também estamos orgulhosos de contribuir com o conhecimento global, pois nossos Recursos Educacionais Abertos são acessados principalmente de fora do continente africano.

Este módulo foi desenvolvido como parte de um diploma e programa de graduação em Ciências da Computação Aplicada, em colaboração com 18 instituições parceiras africanas de 16 países. Um total de 156 módulos foram desenvolvidos ou traduzidos para garantir disponibilidade em inglês, francês e português. Esses módulos também foram disponibilizados como recursos de educação aberta (OER) em oer.avu.org.

Em nome da Universidade Virtual Africana e nosso patrono, nossas instituições parceiras, o Banco Africano de Desenvolvimento, convido você a usar este módulo em sua instituição, para sua própria educação, compartilhá-lo o mais amplamente possível e participar ativamente da AVU Comunidades de prática de seu interesse. Estamos empenhados em estar na linha de frente do desenvolvimento e compartilhamento de recursos educacionais abertos.

A Universidade Virtual Africana (UVA) é uma Organização Pan-Africana Intergovernamental criada por carta com o mandato de aumentar significativamente o acesso a educação e treinamento superior de qualidade através do uso inovador de tecnologias de comunicação de informação. Uma Carta, que estabelece a UVA como Organização Intergovernamental, foi assinada até agora por dezenove (19) Governos Africanos - Quênia, Senegal, Mauritânia, Mali, Costa do Marfim, Tanzânia, Moçambique, República Democrática do Congo, Benin, Gana, República da Guiné, Burkina Faso, Níger, Sudão do Sul, Sudão, Gâmbia, Guiné-Bissau, Etiópia e Cabo Verde.

As seguintes instituições participaram do Programa de Informática Aplicada: (1) Université d'Abomey Calavi em Benin; (2) Université de Ougagadougou em Burkina Faso; (3) Université Lumière de Bujumbura no Burundi; (4) Universidade de Douala nos Camarões; (5) Universidade de Nouakchott na Mauritânia; (6) Université Gaston Berger no Senegal; (7) Universidade das Ciências, Técnicas e Tecnologias de Bamako no Mali (8) Instituto de Administração e Administração Pública do Gana; (9) Universidade de Ciência e Tecnologia Kwame Nkrumah em Gana; (10) Universidade Kenyatta no Quênia; (11) Universidade Egerton no Quênia; (12) Universidade de Addis Abeba na Etiópia (13) Universidade do Ruanda; (14) Universidade de Dar es Salaam na Tanzânia; (15) Universite Abdou Moumouni de Niamey no Níger; (16) Université Cheikh Anta Diop no Senegal; (17) Universidade Pedagógica em Moçambique; E (18) A Universidade da Gâmbia na Gâmbia.

Bakary Diallo

O Reitor

Universidade Virtual Africana

Créditos de Produção

Autor

Fernando Sanha

Par revisor(a)

Arlete Ferrão

UVA - Coordenação Académica

Dr. Marilena Cabral

Coordenador Geral Programa de Informática Aplicada

Prof Tim Mwololo Waema

Coordenador do módulo

Florence Tushabe

Designers Instrucionais

Elizabeth Mbasu

Benta Ochola

Diana Tuel

Equipa Multimédia

Sidney McGregor

Barry Savala

Edwin Kiprono

Kelvin Muriithi

Victor Oluoch Otieno

Michal Abigael Koyier

Mercy Tabi Ojwang

Josiah Mutsogu

Kefa Murimi

Gerisson Mulongo

Direitos de Autor

Apoiado por

Projeto Multinacional II da UVA financiado pelo Banco Africano de Desenvolvimento.

Índice

Visão Geral do Curso

Bem-vindo a Computação Gráfica

Este curso tem como objectivo aprender criar e manipular imagens tridimensionais usando métodos e técnicas gráficas.

Saber tratar da descrição, especificação e representação do suporte geométrico de objetos gráficos que é chamada de modelagem.

Pré-requisitos

1. Programação orientada a objetos:
2. Básica e Avançada
3. Propriedades da Matemática e processamento de imagem.

Materiais

1. Os materiais necessários para concluir este curso são:
2. Um software de interface de programa de aplicação para a definição 2 e 3 imagens gráficas tridimensionais, como o OpenGL programação.
3. Um programa que traduz o código fonte (escrito em uma idioma, como C ++) em código executável computador. Exemplo é o C ++ Compilar.
4. Um circuito eletrônico especializado para a manipulação e alteração de memória para acelerar a criação de imagens. Isso é chamado de Unidade de Processamento Gráfico (GPU) ou Unidade de Processamento Visual (VPU)de:..

Objetivos do Curso

1. Após a conclusão deste curso, o aluno deve ser capaz de:
2. Definir os conceitos básicos e os elementos de computação gráfica.
3. Identificar e definir várias transformação de imagem técnica.
4. Empregar técnicas de transformação em gráficos 2D e 3D0:..
5. Criar e manipular gráficos simples usando OpenGL

Unidades

Unidade 0: Matemática avançada

Pré-avaliação

Esta unidade introduz conceitos básicos de pré-requisitos que são necessários, a fim de realizar o Módulo de Computação Gráfica. Nesta unidade, os temas em matemática avançada e programação orientada a objetos, que são essenciais para computação gráfica, serão introduzidos.

Unidade 1:

Noções básicas e Elementos de Computação Gráfica

Esta unidade introduz os conceitos e aplicações da computação gráfica básica. Elementos de computação gráfica, como a cor, representados em computadores serão abordados nesta unidade

Unidade 2:

Transformações nesta imagem da unidade, serão introduzidas técnicas de manipulação de imagem, tais como tradução, transformação e rotação. Estes são necessários para que uma imagem ou uma imagem não é redesenhada mas sim manipulada para produzir uma versão, tamanho ou aparência de que é diferente 3.

Unidade 3:

Criação de gráficos com OpenGL

Unidade 4:

Gráficos 3D

Esta unidade apresenta aos alunos os gráficos 3D. Os alunos irão aprender como construir e manipular gráficos 3D.

Avaliação formativa,

Avaliações de uso para verificar o progresso do aluno, são incluídas em cada unidade.

Avaliações somativas, tais como testes finais e atribuições, são fornecidas no final de cada módulo e cobertura de conhecimentos e habilidades a partir de todo o módulo.

Somativas avaliações são administradas ao critério da instituição responsável pelo curso. O plano de avaliação sugerida é a seguinte:

1	Testes de	20%
2%	Atribuições	20
3	Exame Final	60%

Programação

Unidade de	Atividades	Tempo estimado de
Unidade 0: Pré-avaliação	0.1 Matemática avançada Trigonometria coordenada polar Coordenar sistema de 3D Pontuação Vectores Matrizes 0.2 programação orientada a: O “objeto” conceito orientado a objetos paradigma Abstração Polimorfismo Herança Encapsulation	

Unidade 1 Objetos: Noções básicas e elementos de computação gráfica	1.1 Propriedades da imagens básicas e processamento de imagem básica básico cor de computador básico de visão 1.2 O que é computação gráfica 2:? 1.3 Aplicativos gráficos de computador 1.4 Polylines 1.5 Texto 1.6 Regiões preenchidas 1.7 Representação a cores de dispositivos de entrada e saída (CRT)	
Unidade Imagem Transformações	2.1 O que é a transformação? 2.2 Representação e transformação básica 2.3 Tradução de pontos Matriz 2.4 Rotação 2.5 Escala 2.6 Visualizando gasoduto 2D	

Unidade 3 Criando gráficos usando o OpenGL	3.1 O que é OpenGL 3.2 Tipos de dados OpenGL 3.3 Gráficos de inicialização 3.4 Criação de gráficos de janela 3,5 Gráficos Básicos rotinas primitivos 3.6 Interação com mouse e teclado 3.7 Veja as portas 3.8 Mapeamento de exibição da janela 3,9 Clipping 3.10 Zooming	
Unidade 4: Gráficos 3D	4.1 classificação lógica de dispositivos de entrada 4.2 técnicas interativo de entrada 4,3 Transformações 3D 4.4 visualização gasoduto 3D 4.5 Remoção de linha Invisível e oculta	

Leituras e outros recursos

As leituras e outros recursos neste curso são:

Unidade 0

Leituras exigidas e outros recursos essenciais:

- Matemática para Computação Gráfica rápida, Vince John de 2001, Springer
- 4 grandes princípios de programação orientada a objetos, Raymond Wallen (accecible de http://codebetter.com/raymondlewallen/2005/07/19/4-major-principles-of-object-oriented-programming/)

Leituras opcionais e outros recursos:

- Titulo, autor, e outras informações de referência

Unidade 1 Noções básicas e elementos de computação gráfica

Leituras exigidas e outros recursos:

- Fundamentos de Processamento de Imagens, Ian T. Young, Jan J. Gerbrands, Lucas J. van Vliet, 1998, Delft University of Technology (para download gratuito a partir de http://www.ebook3000.com/Fundamentals-of_Princípios_-Processing_60576.html -image
- De Computação Gráfica: Teoria e Prática Usando OpenGL e Maya®. New York:. Springer Science, Shalini Govil-Pai (2004)
- Matemática Essencial para a Computação Gráfica rápidos, Vince John de 2001, Springer

leituras opcionais e outros recursos:

- Título, autor, e outras informações de referência

Unidade 2

Leituras exigidas e outros recursos:

- Shalini Govil-Pai (2004). Princípios de Computação Gráfica: teoria e prática usando OpenGL e Maya®. New York: Springer Science
- Matemática essenciais para computação gráfica rápida e Vince John, 2001

Leituras opcionais e outras referências

- Recursos:título, autor, e outras informações de

Unidade 3

Leituras exigidas e outros recursos:

- Francis S Hill Stephen M Kelley (2006) Computação Gráfica Usando OpenGL (3rd Edition),Prentice Hall

Leituras opcionais e outros recursos:

- Título, autor, e outras informações de referência

Unidade 4

Leituras exigidas e outros recursos:

- Princípios de Computação Gráfica: teoria e prática usando OpenGL e Maya®. New York:. Springer Science, Shalini Govil-Pai (2004)
- Matemática Essencial para a Computação Gráfica rápidos, Vince John de 2001, Springer

Leituras opcionais e outros recursos:

- Título, autor, e outras informações de referência

Unidade 0: O objetivo

Pré-avaliação dessa unidade é determinar alcance de um aluno de conhecimentos relacionados a este curso. Ele apresenta e testa um estudante sobre noções básicas de pré-requisitos que são necessários para realizar o Módulo de Computação Gráfica. Serão introduzidos alguns temas em matemática avançada e programação orientada a objetos de.:

Objetivos da unidade

1. Após a conclusão desta unidade, você deve ser capaz de:
2. Ilustrar funções trigonométricas básicas de um ângulo (seno, co-seno e tangente)
3. Definir e ilustrar um ponto, um vector e uma matriz
4. Ilustrar o cartesiano (x, y) e polar (r, θ) coordenar sistemas
5. Ilustrar a cartesiano 3D sistema de coordenadas
6. Definir o OOP termos
7. Listar e descrever / definir as propriedades de um objeto (abstração, polimorfismo, herança,

Termos-chave

Programação orientada a objetos (OOP); Paradigma de programação representa o conceito de objetos que têm campos de dados e procedimentos associados, conhecidos como métodos

Avaliação da unidade

Questões de Revisão

- Como são representados os objetos?
- O que é um circulo trigonométrico?
- Transforme a equação trigonométrica em uma equação trigonométrica com apenas uma única função trigonométrica como variável.

Esta é uma unidade de pré-avaliação; notas atribuídas nesta unidade não contam para a avaliação do módulo global avaliação:.

Itens

Matemática avançada

1. Usando um diagrama, ilustram as funções trigonométricas básicas de um ângulo e suas expressões matemáticas A..
2. Diferenciar entre um ponto, um vector e uma matriz
3. Posição de um ponto é definido especificando a distância (raio) a partir de um ponto fixo chamado como .....
4. Diferenciar entre cartesiana e polar sistemas de coordenadas.
5. Usando um diagrama, ilustram coordena o cartesiano 3D.
6. Um ponto no sistema de coordenadas cartesianas pode ser definida especificando dois números, chamados como .... e o ..... desse ponto.

Programação orientada a objetos

7. Defina os seguintes termos, tal como aplicado no paradigma orientado a objetos

   - Abstração
   - Polimorfismo
   - Herança
   - Encapsulation

8. Dê um exemplo de qualquer aplicação do mundo real em que abstração pode ser usada a classe
9. Faça duas das vantagens de herança, tanto quanto OOP está em causa pública...
10. Diferenciar entre e abstração encapsulamento
11. Leia o seguinte trecho de código e, em seguida, responda às seguintes perguntas:

classe Vehicle public abstract

```
{Wheels

int abstratas
públicas;}
bicicleta   : Veículo
{public
override int Wheels  {return
 ()
 2;}}
```

```
Car classe pública: Veículo
{public
override int Wheels  {return
 ()
 4;}}


Truck classe pública: Veículo
{public
override int Wheels  {return
()
18;}
}
```

- Nome da classe base e sub-classes.
- O item na classe base é genérico para todas as outras sub-classes? Explique por que, com base no seu entendimento sobre polimorfismo.

Leituras de unidade e Outros recursos

As leituras desta unidade podem ser encontrados no “Leituras e Outros seção do nível do Recursos “curso”.

Unidade 1: Fundamentos e Elementos da Computação Gráfica

Introdução

Esta unidade introduz os conceitos e aplicações da computação gráfica básica. As noções básicas de imagens digitais serão abordadas, bem como a cor é representada em computadores. A unidade também irá abordar elementos de computação gráfica, incluindo exemplos de computação gráfica, onde pode ser aplicado de.:

Objetivos da unidade

Após a conclusão desta unidade, você deve ser capaz:

1. Definir os seguintes termos: computação gráfica,definir imagem digital
2. Listar as propriedades de uma imagem digital
3. Descrever como a cor é representada em uma lista de computador
4. Em exemplos de aplicações de computação gráfica
5. Ilustrar os elementos de uma imagem, como polilinhas, texto e cheia regiões
6. Listar lista de entrada dos dispositivos de saída

Termos-chave

Pixel: a menor unidade de resolução de imagem

Digital:. o número de pixels em uma imagem

Computação gráfica

Cor, Imagem, Matriz

Programação orientada ao objecto

Atividades aprendizagem

Atividade 1 - Imagem Básica Propriedades

Introdução

Nesta atividade, a definição de imagem aplicada ao objeto em Computação Gráfica é apresentado

Detalhes da atividade

Objetos do mundo real, tais como carros, casas, árvores, etc, são geralmente em coordenadas mundo tridimensional. Estes objetos podem ser representadas de um computador utilizando imagens.

Uma imagem, ou simplesmente uma imagem, pode ser representada matematicamente utilizando uma matriz bidimensional de intensidade ou dados com um valor de cor em cada elemento da matriz. Na computação gráfica, esses valores são uma coleção de discretos elementos de imagem (digital) chamados pixels. Um pixel é o menor elemento de uma imagem que pode ser representada na tela de um dispositivo como um computador. Figura 1.1 (a) mostra uma relação entre a imagem bidimensional (X e Y) e os pixels dentro da imagem(a):.

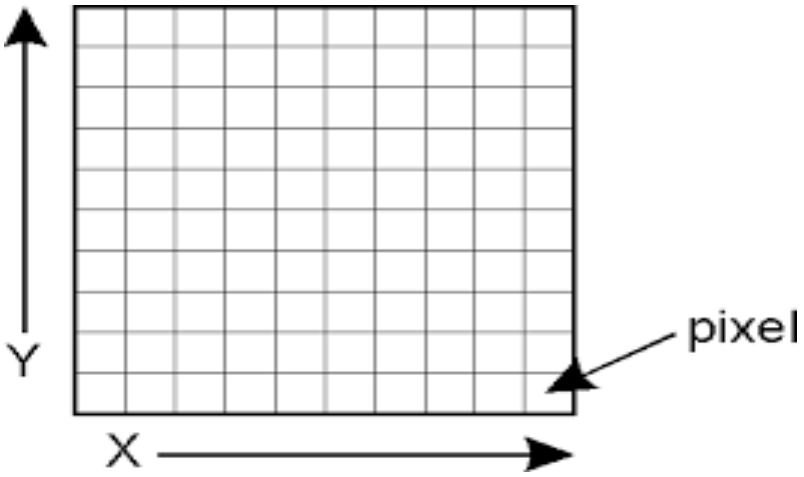


Figura 1.1 Pixels de uma imagem

[Fonte: http://nmita.iowa.uiowa.edu/instr_co O número de .htm] Pixels em uma imagem determina a sua resolução.Resoluções típicas variam de 320 por 200 pixels para 2000 por 1500 pixels. A intensidade de cada pixel é descrita por um número. Por exemplo, para uma imagem a preto e branco, os pixels podem ser expressos entre 0.0 (preto) e 1,0 (branco). Desde computadores armazenam dados em formato binário, estes valores são armazenados como números inteiros entre 0 (preto) e 255 (branco). Figura 1.1 (b) mostra uma imagem em cor, bem como a preto e branco(b):.

Figura 1.1b Preto e branco vs Imagem da cor

[Fonte:http://www.learnquebec.ca/en/content/professional_development/media_literacy/literacy-today-classroom/photo-language-photograph.html]

In imagens a cores, por outro lado, os pixels são descritos por três números que representam a intensidade das três cores (primários) básicos: vermelho, verde e azul. Por exemplo, o vermelho puro é (255,0,0); verde é puro (0,255,0), etc. Uma combinação destes valores de cores resultar em outras cores (secundários), como representado na Figura 1.1 (c),(c):.

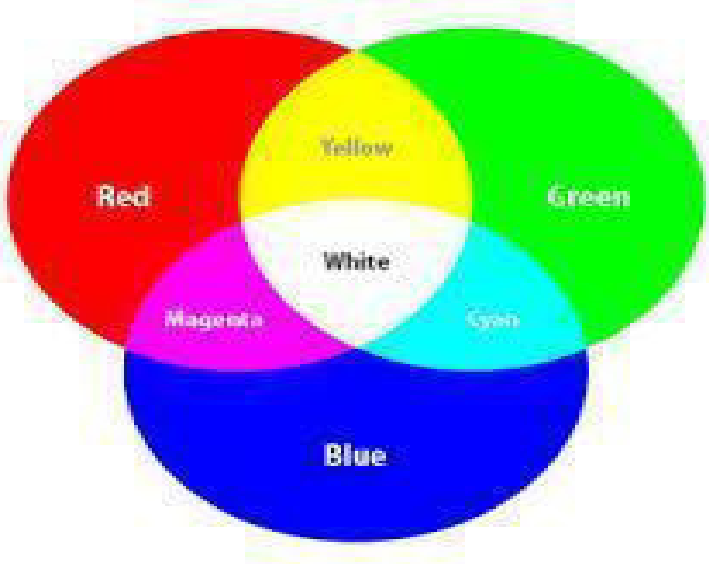


Figura 1.1c Colour Space

[Fonte: http://07268grum.wordpress.com/2013/11/08/colour-space-greyscale-rgb-yuv-lumiance-chrominance-hsv-hue-saturation-value/]

Conclusão

Objectos são representados em um computador usando imagens bidimensionais. Imagens consistem em valores pequenos, discretos chamados pixels. Pixels são representados em um computador usando valores inteiros entre 0 e 255. pixels de cor são representados através de três números que representam a intensidade do vermelho, verde e azul.

Avaliação

1. Defina uma imagem.
2. Definir um objeto.
3. Definir um pixel.
4. O que é resolução de imagem?
5. Como é uma imagem representada Matematicamente-?

Atividade 2 Computação Gráfica e suas Aplicações

Introdução

Computação Gráfica envolve as maneiras pelas quais as imagens podem ser exibidas, manipulados e armazenados usando um computador. Computação gráfica fornece o software e hardware técnicas ou métodos para gerar imagens. Nesta atividade, a definição de Computação Gráfica será apresentado, juntamente com exemplos de onde computação gráfica é aplicada.

Atividade do computador

Definição

detalhes gráfica é uma arte de imagens de desenho, linhas, gráficos, etc, usando computadores com a ajuda de programação. Os gráficos de computador são constituídos por número de pixels. Um pixel é a menor imagem gráfica ou unidade representada na tela do computador interativa:..

gráficos em um computador pode ser interativo ou não interativo

computação gráfica É a computação gráfica em que um usuário pode interagir com a imagem na tela do computador . Existe uma comunicação de duas vias entre o usuário e a imagem. A imagem está totalmente sob o controlo de um utilizador. Exemplo: Jogando jogos de computador em um computador no qual um usuário controla a imagem completamente interativa:.

Não computação gráfica É a computação gráfica em que um usuário não tem qualquer tipo de controlo sobre a imagem. A imagem é apenas o produto de programa armazenado estática e funcionará de acordo com as instruções dadas na linearmente o programa. A imagem está totalmente sob o controlo de instruções do programa, e não sob o utilizador. Exemplo:. Proteções de tela gráfica do computador podem ser classificados em duas categorias: Raster ou de bitmap gráficos e vetor ou Orientada a Objetos gráficos

1..Gráficos Raster Estas são

- (Bitmap) de pixels com base gráfica e os pixels podem ser modificados individualmente.
- As imagens são fáceis de editar na memória e visualização em monitores de TV, devido ao arranjo dos pixels em uma matriz retangular.
- O tamanho da imagem é determinado com com base na resolução da imagem facilmente.;
- Estas imagens não podem ser escalados redimensionamento não funcionam muito bem e pode distorcer significativamente a imagem(a):...
- Os gráficos de bitmap são usadas para imagens de uso geral e em fotografias particulares

Figura 1.2 (a) ilustra o conceito de imagens bitmap

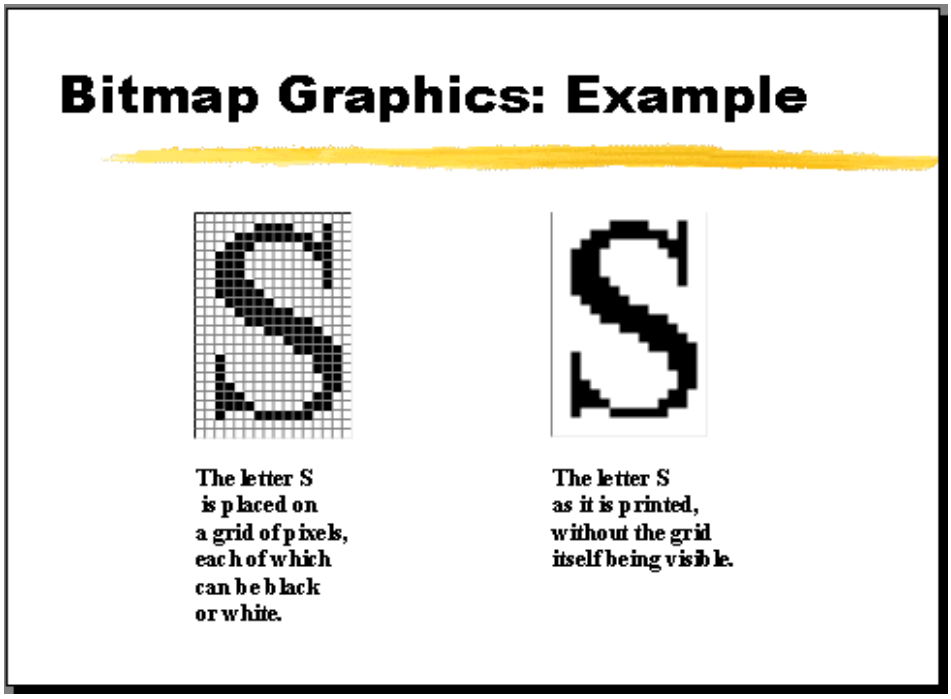


Figura 1.2a Imagem Bitmap

[Fonte: http : //ecee.colorado.edu/~mathys/ecen1200/hwcl07/]

2.Vector Graphics (-orientado a objeto)

- Imagens são estes gráficos baseados matematicamente.
- Vector baseado imagens têm bordas lisas e, portanto, usado para armazenar imagens compostas de linhas, círculos e polígonos.
- Estas imagens facilmente pode ser redimensionados e rodados.
- Eles não podem facilmente acomodar imagens complexas, tais como fotografias, onde informações de cor varia de pixel para pixel.
- Os gráficos vetoriais são adequados para gráficos, por exemplo, em folhas de cálculo e de fontes escaláveis, por exemplo, fontes PostScript

Figura 1.2 (b) ilustra o conceito de imagens vetoriais.

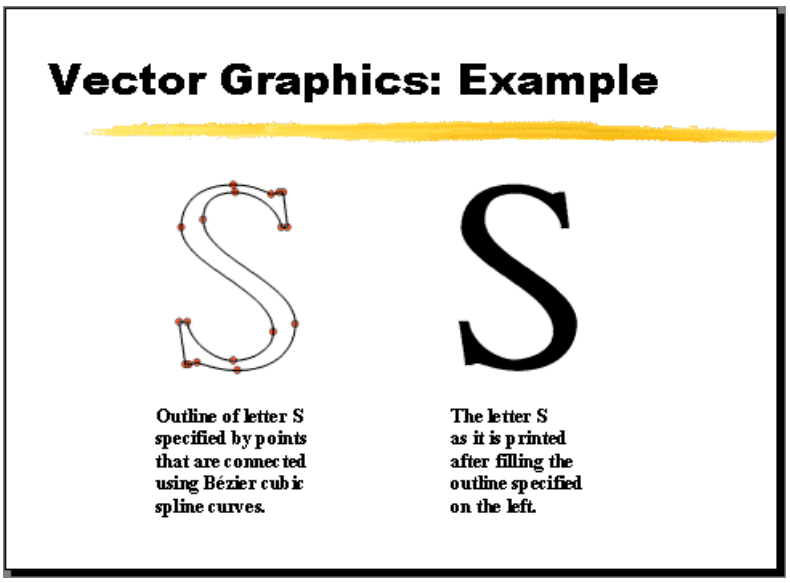


Figura 1.2 (b) : Imagem do vetor

[Fonte: http://ecee.colorado.edu/~mathys/ecen1200/hwcl07/]

Figura 1.2 (c) ilustra a diferença entre uma imagem de bitmap em vetor e(c):.

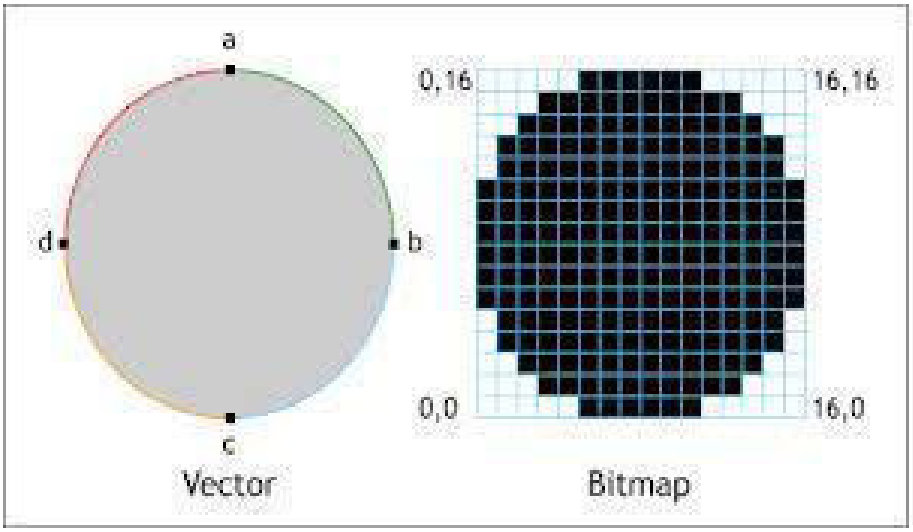


Figura 1.2(c) Vector vs Imagem de bitmap

[Fonte: Aplicações http://www.xaraxone.com/webxealot/xealot30/html/features.htm]
Computação Gráfica Gráfica

Computação pode ser aplicado em várias áreas. Exemplos são como segue:

Desenho assistido por computador:

Em sistemas de engenharia e arquitetura, os produtos são modelados usando computação gráfica comumente referido como CAD (Computer Aided Design). Em muitas aplicações de design como automóveis, aeronaves, veículos espaciais, etc., os objetos são modelados em um esboço wireframe que ajuda o designer de observar a forma geral e as características internas dos objetos Art.:

Computador

Uma variedade de métodos de computador estão disponíveis para os artistas para projetar e especificar movimentos de um objeto. O objeto pode ser pintado eletronicamente em um tabela gráfico usando caneta com diferentes cursos da escova, escova larguras e cores. Os artistas também pode usar a combinação de pacotes de modelagem 3d, mapeamento de textura, programas de desenho e software CAD para pintar e visualizar qualquer objeto

Entertenimento.:

Métodos gráficos de computador são amplamente utilizados em fazer filmes, vídeos de música e programas de televisão. Objetos gráficos podem ser combinadas com ações ao vivo ou pode ser usado com técnicas de processamento de imagem para transformar um objeto para outra formação.:

Educação

Os gráficos de computador pode fazer melhor a compreensão do funcionamento de um sistema. Em sistemas físicos, sistemas biológicos, as tendências demográficas, etc., modelos torná-lo mais fácil de entender. Em alguns sistemas de formação, modelos gráficos com simulações ajudar um estagiário para treinar em ambiente de realidade virtual. exemplo, por a sessão de prática ou de formação de capitães de navio, pilotos de aeronaves, pessoal de controle de tráfego aéreo imagem.:

Processamento

Processamento de imagem fornece técnicas para modificar ou interpretar as imagens existentes. Pode-se melhorar a qualidade da imagem por meio de técnicas de processamento de imagem. Por exemplo, em aplicações médicas, técnicas de processamento de imagem pode ser aplicado para melhorias de imagem e está sendo amplamente utilizado para CT (Computer raios-X Tomography) e PET (Posição Emission Tomography) imagensInterface.:

Uso Gráfico

GUI é comumente usado para fazer um pacote de software mais interativo. Existem vários sistemas de janelas, ícones, menus, o que permite uma configuração de computador a ser utilizado de forma mais eficiente.

Conclusão

Computação Gráfica envolve maneiras em que as imagens podem ser exibidas, manipulados e armazenados usando computadores. Imagens de computação gráfica podem ser categorizadas em gráficos raster, que, como gráficos baseados em pixel, e gráficos de vetor, que são matematicamente representados. Computação gráfica é aplicável em diversas áreas, como design assistido por computador, arte de computador, entertenimento, bem como na educação e formação.

Avaliação

1. Listar duas categorias de imagens em computação gráfica.
2. Qual sistema gráfico é melhor para imagens realistas?
3. Em qual sistema gráfico são imagens facilmente escalável?
4. Listar quatro aplicações de computação gráfica.
5. Explicar como gráficos de computador são úteis na indústria de entretenimento e na educação-.

Atividade 3 Elementos de computador Gráficas

Introdução

(Primitives)

Esta atividade apresenta como primitivas de imagens gráficas de saída aparecer. Estas primitivas são polilinhas, texto e regiões preenchidas. Todos os outros elementos gráficos são construídos a partir destas primitivas.

Detalhes da atividade

Polylines

Uma polilinha é uma seqüência conectada de linhas retas. Para o olho, uma polilinha pode aparecer como uma curva suave. Atributos polilinha simples são cor e espessura. A polilinha mais simples é um único segmento de reta. Um segmento de linha é especificado pelas suas duas extremidades, tais como (x1, y1) e (x2, y2). Quando existem várias linhas de um polígono, cada um é chamado uma aresta,e duas linhas adjacentes se encontram num vértice.

As extremidades de uma poligonal pode atravessar um ao outro, mas um polígono não tem de ser fechada. Um polígono tem seus primeiros e últimos pontos conectados por uma aresta. Se há duas bordas se cruzam, o polígono é chamado de um polígono simples. Um exemplo de um polígono é mostrado na figura 1.3 (a), e um polígono é mostrado na figura 1.3 (b),[Fonte:.

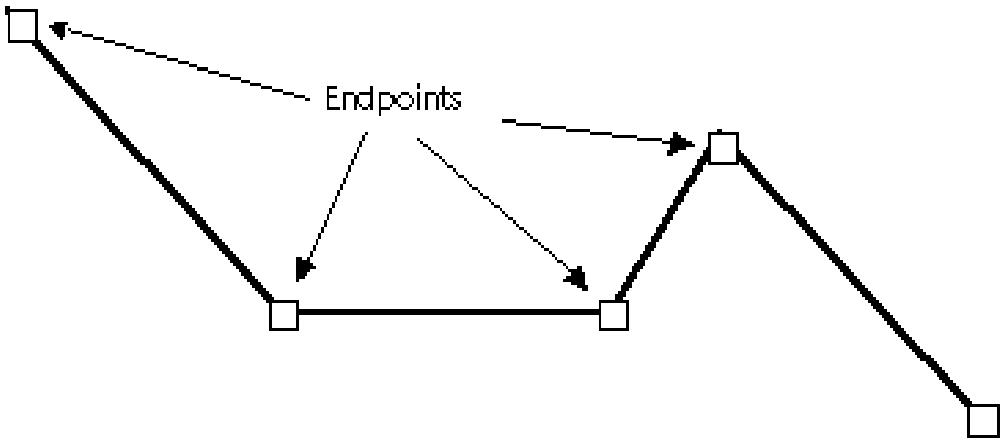


Figura 1.3 (a) Polilinha

Http://www.webopedia.com/TERM/P/polyline.html]

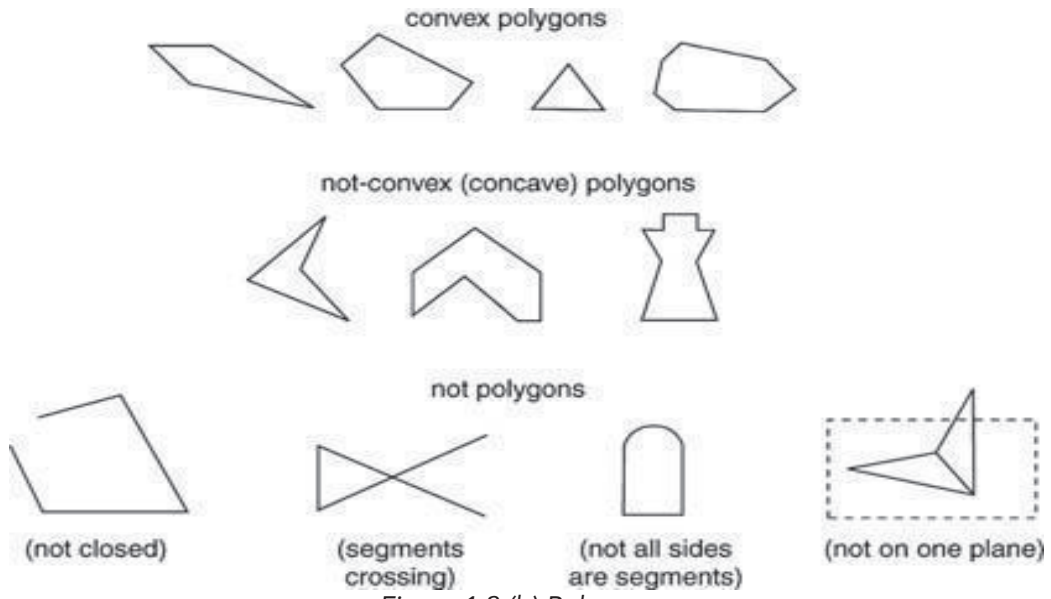


Figura 1.3 (b) Polygon

[Fonte:http://www.cliffsnotes.com/math/geometry/polygons/classifying-polygons]

Texto

Alguns dispositivos de vídeo têm dois modos de exibição distintos: o modo de texto e gráficos. No modo de texto, o texto é gerada usando um gerador de caracteres embutida. Texto em modo gráfico é desenhado. Atributos de texto são como a cor, tamanho, fonte, espaçamento e orientação.

Regiões preenchidas

A região preenchida é uma forma preenchida com alguma cor ou padrão. Um exemplo é um polígono preenchido conforme mostrado na figura 1.3 (c)preenchidas.

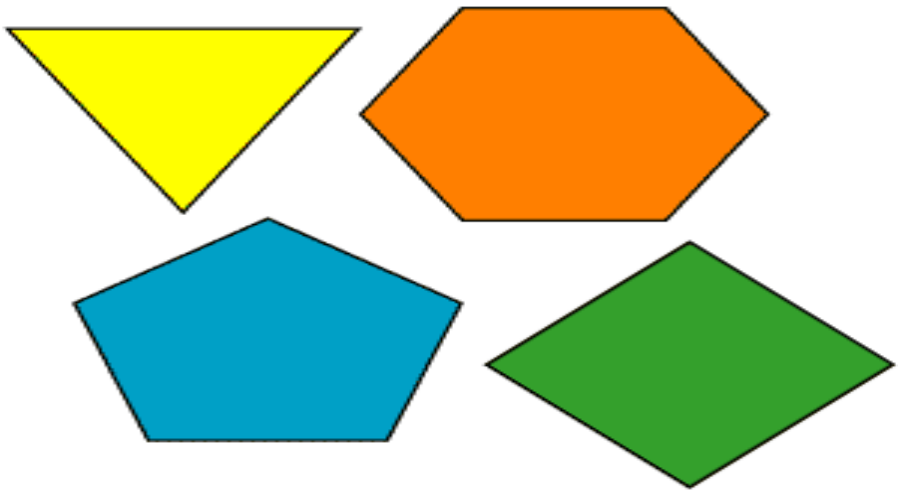


Figura 1.3 (c) regiões

[Fonte: http://www.barryscientific.com/lessons/polygon.html]

Conclusão

Nesta atividade, os primitivos gráficos básicos : polylines, texto e regiões preenchidas foram apresentados. Eles são caraterizados por atributos tais como a cor, tamanho, fonte e espessura. Estas primitivas são usados para criar outros gráficos.

Avaliação

1. Listar os primitivos gráficos básicos.
2. Que atributos caracteriza cada uma das primitivas?

Resumo da Unidade

Esta unidade apresentou os conceitos básicos de computação gráfica, incluindo da imagem Propriedades básicas e como imagens e cores são representadas em computadores. Os elementos de computação gráfica, ou seja, polylines, texto e regiões preenchidas, foram abordados. Por último, foram apresentados pedidos de computação gráfica em educação, entretenimento, arte de computador, etc.

Leituras de unidade e Outros recursos:

As leituras desta unidade se encontram no campo de leituras de nível outros recursos.e

Unidade 2: Imagem Transformações

Introdução da unidade

As transformações são uma das operações fundamentais realizadas em computação gráfica. Na computação gráfica, as imagens não são só criadas, mas também transformados para fazê-las parecer diferente do que elas se parecem. Esta unidade apresenta definições e técnicas de transformação básica da imagem para atributos de objeto, como alterar a sua posição, tamanho e orientação como visto em dispositivos de exibição. A unidade apresenta também a visualização 2D gasoduto onde coordenar mundo 2D imagens do sistema são mapeadas para coordenadas do dispositivo de.:

Objetivos da unidade

1. Após a conclusão desta unidade, você deve ser capaz:
2. Definir transformação do objeto da coisas real
3. Identificar os tipos de operações de transformação
4. Ilustrar representação de um ponto usando matriz
5. Executar a tradução, escala e rotação em imagens de

Termos-chave

Coordenadas tridimensionais que definem a localização de objetos no mundo real

Ponto de Vista (Câmara) coordenadas:. As coordenadas são baseadas no ponto de vista do observador, e mudanças como eles mudam sua visão

Coordenadas de tela: coordenadas físicas dos pixels na tela do computador, com base na resolução da tela atual.

Atividades de aprendizado

Atividade 1 - Definições e transformação Básica

Introdução

Quando forem necessárias objetos definidos em um sistema de coordenadas a ser observado em outro sistema de coordenadas, técnicas de transformação são normalmente utilizados. A transformação é realizada tal como para transferir uma imagem a partir de um lugar para outro lugar num ecrã, ou para aumentar ou diminuir o seu tamanho. Esta atividade irá apresentar a definição de transformação, representação matricial de pontos, e as noções básicas de transformação

http://www.pling.org.uk/cs/cgv.html).

Detalhes da atividade

(Notas / Materiais adotadas a partir de:

O que é transformação?

A transformação é um processo que é usado para obter um tipo diferente de imagem a partir de um já existente através da manipulação dos atributos da imagem original. Estes atributos são a posição, o tamanho e orientação da imagem.

Representação matricial pontos A

Das imagens pode ser pensado como sendo uma combinação de pontos. A idéia é identificar os pontos que formam uma imagem que está sendo desenhado ou exibido. A tela é também pensado para ser um composto de um número de pixels, cada um pixel que corresponde a um ponto. Esses pixels, que formam uma parte da imagem que está sendo desenhada são feitas para iluminar para que a imagem fique visível na tela.

Baseado no sistema de coordenadas cartesianas, um ponto é representada por dois valores comoy).(x, representa Xa distância do ponto de origem na direção horizontal y e na direção vertical. No contexto de gráficos, um ponto é representado como uma entidade 3 valorizado [x, y, 1]. X e Y são as coordenadas, e 1 é um valor que é utilizado para a representação. . Figura 2.1 (a) mostra um ponto num sistema de coordenadas Cartesiano na posição (4, 2)

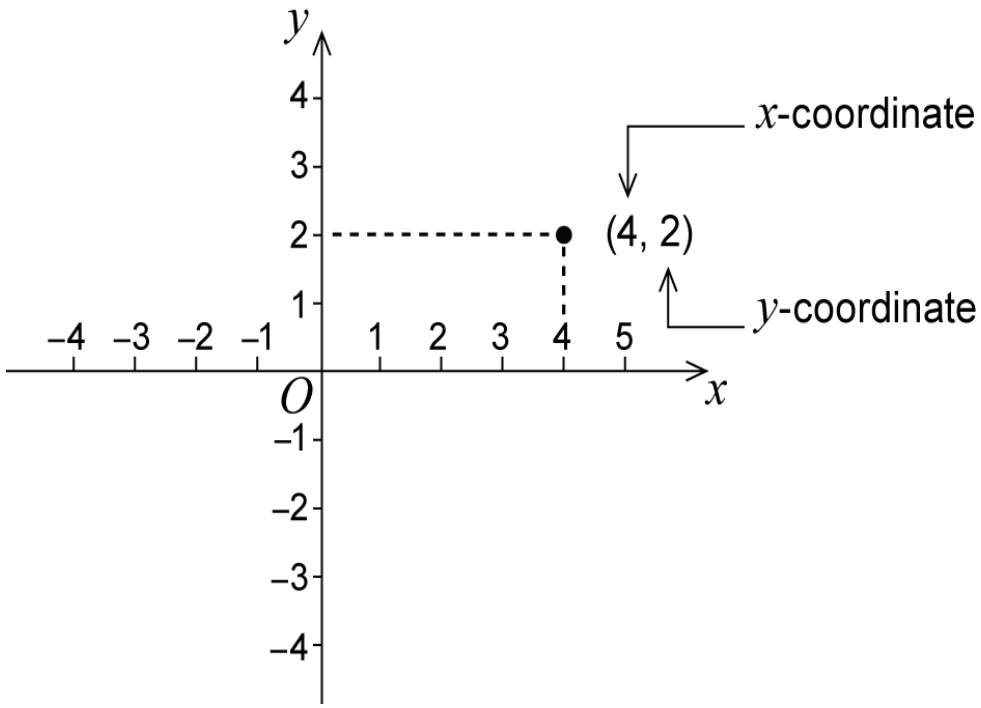


Figura 2.1 (a) Um ponto em coordenadas cartesianas

[Fonte: http://syllabus.bos.nsw.edu.au/glossary/ mat /-sistema de coordenadas cartesianas /?] ajax

Noções básicas de transformação

Manipulação de imagens em uma tela envolve dois tipos de processos. O primeiro lida com a alteração da aparência da imagem, alterando sua cor e textura da superfície, bem como alterar sua aparência, alterando o modelo de relâmpago que o computador usa para torná-lo visível.

O segundo processo é geométrica, em que o parente ou localização absoluta da imagem no sistema de coordenadas é alterada. A imagem neste caso pode ser movido ao redor da tela, ou parecem tornar-se maior ou menor. Por exemplo, uma imagem pode ser deslocada do centro para o canto superior esquerdo da tela. A imagem pode também ser aumentado ou diminuído para o dobro ou metade do seu tamanho, respectivamente. A imagem também pode ser ligado / girada 45 ou 90 graus são:.

As três transformações básicas

1. Tradução
2. Rotação
3. Escala

Estes serão discutidos em detalhes na próxima atividade redimensionadas.

Conclusão

Imagens em telas de computador pode ser traduzido para diferentes posições na tela, ou , ou girado para aparecer de forma diferente da imagem original. Essa atividade apresentou a definição de transformação, e introduziu os tipos de transformações de imagem

Avaliação

1. Definir transformação
2. Por que é transformação realizada?
3. Listar três maneiras em que uma imagem pode ser transformada

Atividade 2 Tipos de transformações

Introdução

Como introduzido na atividade anterior, há três transformações básicas: translação, rotação e escala. Estas transformações matemáticas são essencialmente as operações que são executadas em pontos e, por conseguinte, imagens, uma vez que uma imagem pode ser vista como um conjunto de pontos.

Detalhes da atividade

Tradução

Refere-se ao deslocamento de um ponto para outro local, cuja distância em relação ao presente momento é conhecida. Este processo altera a posição de um ponto de um conjunto de coordenadas para o outro.

Considere-se um ponto P (x1,y1) a ser traduzido para outro ponto Q (X2,Y2).Se os valores dos pontos de (X2,Y2)são conhecidos, o ponto pode ser directamente deslocado para Q, exibindo o pixel em (X2,Y2).Se a distância através da qual o ponto está a ser deslocado é conhecido, por exemplo o tX e Ty ao longo do x e y eixos respectivamente, então as coordenadas podem ser derivados por x2 = x1 + Te Y X2 = Y1+ Ty.

Exemplo: se um triângulo com as coordenadas de um (20,10), B (30,100) e C (40,70) está a ser deslocado por 20 unidades ao longo do eixo X e 10 unidades ao longo do eixo y, em seguida, o novo triângulo será proporção de 1na(20 + 20, 10 + 10) B1(30 + 20, 10 + 10) C1(40 + 20, 70 + 10).

Na forma de matriz, [x2 y2 1] = x[1 Y1 1] *

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Tx & Ty & 1 \end{matrix}$$

Rotação

Rotação é o ponto de rotação de uma volta de um eixo. O eixo pode ser qualquer uma das coordenadas ou qualquer outra linha especificado. Suponhamos que um ponto (x1 Y1)é para ser rodado no sentido horário por meio de um ângulo θ em relação à origem do sistema de coordenadas. Matematicamente as novas coordenadas do ponto (x2 y2)pode ser expressa como:

x2 = x1cos + y1sinθ

y2 = x1sinθ - y1cos

Estas equações aplicam-se apenas se a rotação é sobre a origem . A

matriz para [x2 y2 1] = [x1 Y1 1] *

$$\begin{matrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{matrix}$$

Escalonamento

É a aumentar ou diminuir o tamanho de uma imagem em uma ou ambas as direções. O tamanho da imagem for alterada, aumentando ou diminuindo a distância entre os pontos de extremidade da imagem e mudando os pontos intermédios de acordo com requisitos.

Suponhamos que um ponto (x1 Y1)é para ser escalada por um SX fator ao longo da direção x e por um fator sy ao longo da direção y, as novas coordenadas tornar-se:

x2 = 1x * sx

y = 2y *1sy

Dimensionamento de um ponto significa fisicamente mover o ponto de distância, mas não ampliar o ponto. No entanto, quando uma imagem é dimensionado, cada um dos pontos são dimensionados de modo diferente e, portanto, as dimensões das mudanças de imagem.

A tradução, rotação e dimensionamento são resumidos na Figura 2.2 (a).

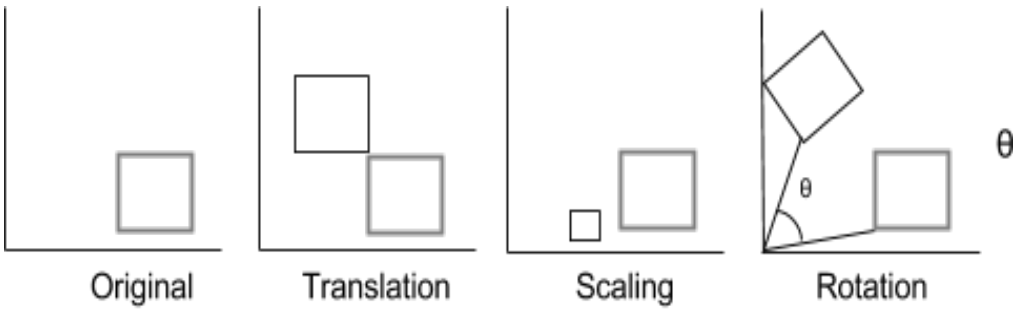


Figura 2.2 (a) tipos de transformações

[redesenhado : http://www.pling.org.uk/cs/cgv.html]

Conclusão

Essa atividade tem apresentado os tipos de transformações, ou seja, tradução, escala e rotação. Na tradução, a posição de uma imagem é alterada, alterando suas coordenadas. Em escala, o tamanho de uma imagem é alterada, ou seja aumentado ou diminuído, enquanto em rotação, uma imagem é girada sobre um eixo através de um ângulo, no sentido horário ou anti-horário. Estas operações são necessárias em computação gráfica de modo que imagens podem ser manipuladas e transformadas em termos de tamanho, posição e orientação, para dar vários efeitos das imagens.

Avaliação

1. Se hum Ponto (x, y) e movido a hum ponto que está em uma distância de Tx ao longo do eixo x, o que é sua nova posição?
2. Se um ponto (x, y) é movido para um ponto que fica a uma distância de Ty ao longo do eixo Y, o que é a sua nova posição?
3. Se um ponto (x, y) é rodado no sentido contrário através de um ângulo sobre a origem, quais são as suas novas coordenadas?

Atividade 3 - Visualização 2D Pipeline

(Notes adopted from: https://www.cg.tuwien.ac.at/courses/CG1/textblaetter/englisch/03%202D%20Viewing%20%28engl%29.pdf)

Introdução

Em muitos casos, o tamanho de dispositivos de exibição é geralmente limitados em termos da dimensão física bem como a resolução. Quando a exibição de imagens dos dispositivos, as suas dimensões podem ser limitado pela dimensão do dispositivo de visualização. Em tais casos, as imagens têm de ser "preparados" no dispositivo de visualização, e é submetido a uma série de transformações, as quais são designadas por um entrando tubagem, de modo que pode ser exibida num dispositivo físico.

Detalhes da atividade

O termo Visualizando Pipeline descreve uma série de transformações, que são passados por dados de geometria para acabar como os dados de imagem a ser exibido em um dispositivo. O oleoduto visualização 2D descreve este processo para dados 2D, como mostrado na Figura 2.3 (a).

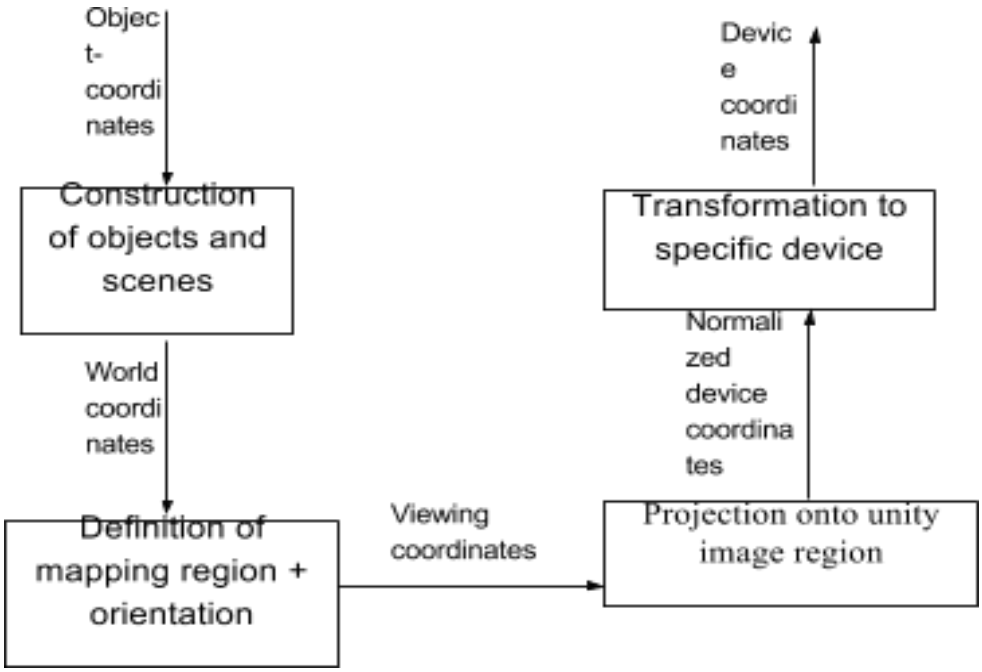


Figura 2.3 (a) Visão Pipeline

[Fonte: https://www.cg.tuwien.ac.at/courses/CG1/textblaetter/englisch/03%202D%20Viewing%20%28engl%29.pdf]

- As coordenadas no qual os objetos individuais (modelos) são criados são chamados de modelo (ou objeto) coordena.
- Quando vários objetos são montados em uma cena, eles são descritos por coordenadas mundo.
- Após a transformação no sistema de coordenadas da câmara (visualizador) tornam-se a visualização coordenadas.
- Sua projeção sobre um plano comum (janela) produz coordenadas normalizadas independentes de dispositivo.
- Finalmente, depois do mapeamento essas coordenadas normalizadas para um dispositivo específico, obtemos coordenadas do dispositivo.

Mais recursos para a tubagem de visão pode ser encontrada em:

https://www.cg.tuwien.ac.at/courses/CG1/textblaetter/englisch/03%202D%20Viewing%20%28engl%29.pdf

Conclusão

O pipeline de visualização permite a transformação da imagem de coordenadas para um formato que pode ser exibido em um dispositivo de exibição física.

Avaliação

1. Defina o termo “Vendo Pipeline”.
2. Delinear a importância da visualização de gasoduto.

Sumário

Esta unidade apresentou as operações fundamentais realizadas em imagens; tradução ou seja, rotação e escala. A unidade também apresentou o pipeline de visualização 2D onde coordenar mundo 2D imagens do sistema são mapeadas para coordenadas do dispositivo.

Leituras de unidade e Outros recursos

As leituras desta unidade encontram-se em curso leituras de nível e outros recursos.

Unidade 3: Criar Gráfico Com o Programa OpenGL

Unidade Introdução

Computação gráfica é maioritariamente dominado por praticar; tal como por escrito e teste de programas que produzem uma variedade de imagens. Um ambiente que permite gravar e executar programas é necessária. O ambiente deve geralmente incluir hardware para a exibição de imagens, e ferramentas de software que programas escritos pode usar para realizar o desenho real de fotos. Esta unidade apresenta a Interface de Programação de Aplicativos (API) para uso na produção de gráficos.

Objetivos da unidade

Após a conclusão desta unidade, você deve ser capaz de:

1. Definir programa OpenGL.
2. Descrever os elementos fundamentais do OpenGL.
3. Escrever programas para fazer gráficos simples em OpenGL
4. Ilustre visualização da janela do objeto e executar recorte em gráficos.

Termos-chave

Interface para hardware gráfico Imagem, e objectos

Ferramenta de software

Aplicativos (API) pixels

Atividades de Aprendizado

Atividade 1 - OpenGL: Definição e Fundamentos

Introdução

Nesta atividade, a definição e fundamentos do OpenGL como uma API será apresentado.

Detalhes da atividade

Definição

OpenGL é uma interface de programação de aplicativo (API) para a criação de imagens gráficas em 2D e 3D. OpenGL especifica um conjunto de comandos em que cada comando dirige uma ação de desenho ou provoca efeitos especiais. OpenGL é independente de janelas características de cada sistema operacional.

OpenGL Fundamentos

[Referência: www.glprogramming.com/blue/ch01.html]

OpenGL permite desenhar gráficos primitivos (linhas, polígonos, pontos, etc.). Primitives são definidos por um grupo de um ou mais vértices. Um vértice define um ponto, de uma extremidade de uma linha, ou um canto de um polígono, onde os dois bordos se encontram. Os dados de coordenadas que consiste de vértices, cores coordenadas de textura, etc está associada com um vértice, e cada vértice e seus dados associados são processados independentemente, na ordem e na mesma maneira.

Os comandos são sempre processada em ordem em que são recebidos, embora possa haver um atraso intermediário antes de um comando tem efeito. Isto significa que cada primitiva é desenhada completamente antes de qualquer comando subsequente produz efeitos.

Operação básico OpenGL

A operação básica do OpenGL é como mostrado na Figura 3.1 (a).

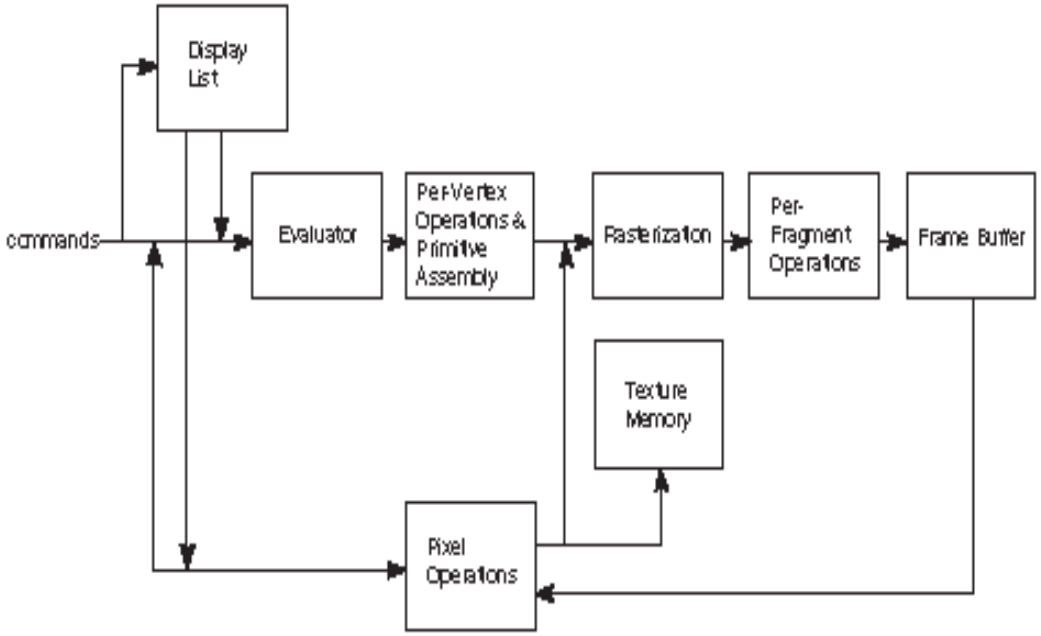


Figura 3.1 (a) OpenGL Operação básica

[Fonte: http://www.glprogramming.com/blue/ch01.html]

Como mostrado pelo primeiro bloco no diagrama, em vez de ter todos os comandos de proceder de imediato através do oleoduto, alguns dos comandos podem ser acumulados em uma lista de exibição para o processamento num momento posterior.

O estágio avaliador de processamento fornece um meio eficiente de aproximação de curva e geometria da superfície avaliando comandos polinomiais de valores de entrada. Durante a fase seguinte, por vértice operações de montagem e primitiva, OpenGL processa-primitivas geométricas pontos, segmentos de linha, e polígonos, todos os quais são descritos por vértices. Vértices são transformados e iluminados, e primitivos são cortados para a janela de exibição, em preparação para a próxima fase.

Rasterization produz uma série de endereços de frame buffer e valores associados usando uma descrição bidimensional de um ponto, segmento de linha, ou polígono. Cada fragmento produzido é alimentado na última etapa, por fragmento de operações, que executa as operações finais sobre os dados antes de serem armazenados como pixels no frame buffer.

Os dados de entrada pode estar na forma de pixels, em vez de vértices. Tais dados, que pode descrever uma imagem para utilização em mapeamento de textura, salta para a primeira fase de processamento descrita acima e em vez disso é processada como pixels, na fase de pixel. O resultado desta etapa é armazenada quer como memória de textura, para utilização na fase rasterization, ou reticulado e os fragmentos resultantes fundido no frame buffer apenas como se eles foram gerados a partir de dados geométricos.

Todos os elementos de estado OpenGL, incluindo o conteúdo da memória de textura e do mesmo tampão de trama, pode ser obtida por uma aplicação OpenGL.

Conclusão

Esta atividade tem apresentado a definição e os fundamentos de OpenGL como uma API para criar gráficos.

Avaliação

1. Definir OpenGL.
2. Descrever a forma como desenhar gráficos simples usando OpenGL pode ser alcançado.

Atividade 2 - Desenhos Simples usando OpenGL

[Referência: Computação Gráficas Usando OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, Chapter 2]

Introdução

Cada programa gráfico começa com algumas inicializações para estabelecer o modo de exibição desejado e criar um sistema de coordenadas para pontos especificando, linhas, etc. desenhos gráficos simples usar o sistema de coordenadas cartesianas com coordenadas x e y representam pixels que se estendem para a direita e para baixo / para cima instruções respectivamente.

Detalhes da atividade

Figura 3.2 (a) mostra os diferentes modos em que gráficos simples podem ser extraídas. Na parte (a) a tela inteira é usada para o desenho: o display é inicializado, alternando-lo em "modo gráfico", e o sistema de coordenadas é estabelecida conforme mostrado. Coordenadas X e Y são medidos em pixels, com cada vez maior x para a direita e Y aumentando para baixo.

Na parte (b) um sistema mais moderno ", baseado windows" é mostrado. Pode suportar um número de diferentes janelas retangulares na tela do monitor ao mesmo tempo. Inicialização envolve a criação ea "abertura", uma nova janela (que pode ser chamado de janela na tela) para gráficos. Comandos gráficos utilizam um sistema de coordenadas que está ligado à janela: geralmente x aumenta para a direita e para baixo y aumenta. Parte (c) mostra uma variante em que o sistema de coordenadas inicial é "lado direito para cima", com y aumentando para cima.

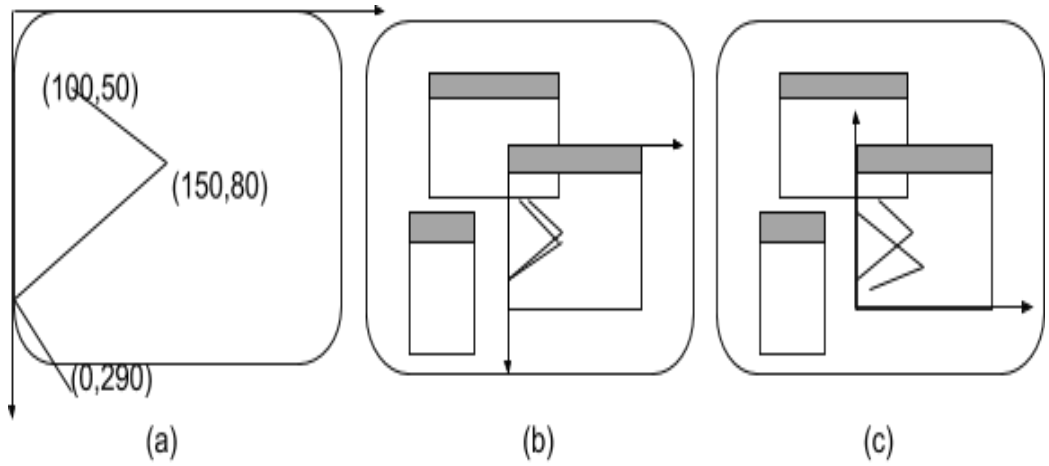


Figure 3.2(a): Display Layouts

[Source: Computer Graphics Using OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, Chapter 2]

Desenho básico

Cada sistema normalmente tem algumas ferramentas de desenho elementares que ajudam a começar. A mais básica tem um nome como setPixel (x, y, cor): que define o pixel individualmente no local (X, Y) para a cor especificada pela cor. Às vezes vai por nomes diferentes, tais como PutPixel (), SetPixel (), ou drawPoint (). Junto com setPixel () há quase sempre uma ferramenta para desenhar uma linha reta, linha de (x1, y1, x2, y2), que traça uma linha entre (x1, y1) e (x2, y2). Em outros sistemas que poderia ser chamado drawLine () ou Linha (). os comandos

- Linha(100, 50, 150, 80);
- Linha(150, 80, 0, 290);

Gostaria de chamar as imagens mostradas em cada sistema na Figura 3.2 (a). Outros sistemas não têm nenhuma linha de comando (), mas sim usar mover para (x, y) e lineto (x, y). Eles derivam da analogia de uma caneta plotter, onde a caneta tem alguma posição atual. A noção de que é mover para (X, Y) se move a caneta de forma invisível para o local (X, Y), definindo assim a posição actual para (x, y); lineto (x, y) desenha uma linha a partir da posição atual para (x, y), em seguida, atualiza a posição atual para este (x, y). Cada comando move a caneta de sua posição atual para uma nova posição. A nova posição torna-se então a posição actual. As imagens na Figura 3.2 (a) seria desenhado usando os comandos:

- Mover para(100, 50);
- Lineto(150, 80);
- Lineto(0, 290);

Em geral, um conjunto de ferramentas de funções pode ser desenvolvido a partir de ferramentas elementares simples de construir uma biblioteca de rotinas gráficas. No entanto, uma vez que cada exibição de gráficos utiliza diferentes comandos básicos para "conduzí-lo", e cada ambiente tem uma coleção diferente de ferramentas para produzir os primitivos gráficos, torna-se difícil para a porta de um programa a partir de um ambiente para outro, portanto, isso pode exigir grandes alterações na estrutura global de uma biblioteca ou aplicação, e esforço significativo programador.

Programação orientada a eventos

Baseados no Windows A maioria dos programas são evento-dirigido; que é o programa responde a vários eventos, como um clique do mouse, o pressionar de uma tecla do teclado, ou o redimensionamento de uma janela de tela. O sistema gerencia automaticamente uma fila de eventos, que recebe mensagens de que certos eventos ocorreram, e lida com eles em um primeiro a chegar, primeiro a ser servido. Um programador organiza um programa como um conjunto de funções de retorno de chamada que são executados quando ocorrem eventos. A função de retorno é criada para cada tipo de evento que possa ocorrer. Quando o sistema remove um evento da fila simples-mente executa a função de retorno relacionado com o tipo de evento que.

OpenGL vem com um Toolkit Utility que fornece ferramentas para ajudar com gerenciamento de eventos. Por exemplo excesso rato Func (meu Mouse) registra o meu Mouse função () como a função a ser executada quando ocorre um evento de mouse. O prefixo "excesso" indica que é parte do GLUT. O programador coloca código no meu Mouse () para lidar com todas as possíveis ações do mouse de interesse. Um exemplo de um esqueleto principal () para uma função de origem num evento é mostrado na figura 3.2(b).

```
void main()
{
initialize things, create a screen window
glutDisplayFunc(myDisplay); // register the redraw function
glutReshapeFunc(myReshape); // register the reshape function
glutMouseFunc(myMouse); // register the mouse action function
glutKeyboardFunc(myKeyboard); //register the keyboard action
function
glutMainLoop(); // enter the unending main loop
}
```

Figure 3.2(b): Code skeleton for main function for an event-driven program using OpenGL

[Source: Computer Graphics Using OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, Chapter 2]

Excesso de Display Func (minha Display); Sempre que o sistema determina que uma janela de tela deve ser redesenhada ele emite um "redesenhar" do evento. Isto acontece quando a janela é aberta pela primeira vez, e quando a janela é exposta movendo outra janela fora do mesmo. Aqui a função de minha Display () é registrada como a função de retorno de chamada para um evento de redesenho.

Excesso Reshape Func (minha Reshape); Janelas de tela pode ser reformulado pelo usuário, geralmente arrastando um canto da janela para uma nova posição com o mouse. (Basta mover a janela não produz um evento de remodelagem.) Aqui o myReshape function () é registrado com o evento "remodelar". Como veremos, a minha Reshape () passa automaticamente argumentos que relatam a nova largura e altura da janela reformulado.

Excesso rato Func (meu Mouse); Quando um dos botões do mouse é pressionado ou liberado um evento de mouse é emitido. Aqui meu Mouse () é registrada como a função a ser chamada quando ocorre um evento de mouse. meu Mouse () está argumentos que descrevem a localização do mouse, a natureza da ação do botão passou automaticamente.

Excesso Keyboard Func (meu teclado); Isso registra a função de meu teclado () com o evento de pressionar ou liberar alguns tecla do teclado. meu teclado () é argumentos que indicam qual tecla foi pressionada passou automaticamente. Convenientemente, também é passado como dados para a localização do rato no momento em que a tecla foi pressionada.

Excesso de Loop Main (). Quando isso é executado o programa desenha a imagem inicial e entra em loop infinito, no qual ele simplesmente espera para que os eventos ocorrem. (Um programa normalmente é encerrado clicando na caixa "ir embora", que está ligado a cada janela).

Criando janela de gráficos

A primeira tarefa é a de abrir uma janela de tela para desenhar. Porque funções OpenGL são dispositivo independente, eles não fornecem suporte para controle de janela em sistemas específicos. Figura 3.2 (c) mostra um esqueleto para a função inteira main () para um programa que vai desenhar gráficos em uma janela de tela. As cinco primeiras chamadas de função usar o kit de ferramentas para abrir uma janela para desenhar com OpenGL. As cinco primeiras funções inicializar e exibir a janela de tela em que o programa vai produzir gráficos. Uma breve descrição do que cada um faz é a seguinte:

```
// appropriate #includes
void main(int argc, char** argv)
{
      glutInit(&argc, argv); // initialize the toolkit
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set the display mode
      glutInitWindowSize(640,480); // set window size
      glutInitWindowPosition(100, 150); // set the window position on
      screen
      glutCreateWindow("my first attempt"); // open the screen window

      // register the callback functions
      glutDisplayFunc(myDisplay);
      glutReshapeFunc(myReshape);
      glutMouseFunc(myMouse);
      glutKeyboardFunc(myKeyboard);

      myInit(); // additional initializations as necessary
      glutMainLoop(); // go into a perpetual loop
}
```

Figura 3.2 (c): a função principal para abrir a janela inicial para o desenho

[Fonte: Computodor Gráficos Usando OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, capítulo 2]

- Excesso Init (& argc, argv); Essa função inicializa o kit de ferramentas. Os seus argumentos são os padrões para passar informações de linha de comando.
- Excesso modo Init Display (GLUT_SINGLE | GLUT_RGB); Esta função especifica como o monitor deve ser inicializado. A built-in constantes GLUT_SINGLE e GLUT_RGB, que são OR'd em conjunto, indicam que um buffer de exibição único deve ser atribuído e que as cores são especificados usando quantidades desejadas de vermelho, verde e azul. (Mais tarde vamos alterar esses argumentos: por exemplo, usaremos o buffer duplo para a animação suave.)
- Tamanho de Janela Init excesso (640.480); Esta função especifica que a janela de tela deve, inicialmente, ser 640 pixels de largura por 480 pixels de altura. Quando o programa está sendo executado, o usuário pode redimensionar esta janela se o desejar.
- Excesso Init Posição da Janela (100, 150); Esta função especifica que canto superior esquerdo da janela deve ser posicionado na tela 100 pixels da beirada esquerda e 150 pixels de cima para baixo. Quando o programa está sendo executado, o usuário pode mover esta janela sempre que desejar.
- Criar excesso de janela ("minha primeira tentativa"); Esta função, na verdade, é aberto e exibe a janela de tela, colocando o título de "a minha primeira tentativa" na barra de título.
- As funções restantes no main () registrar as funções de retorno de chamada como descrito anteriormente, executar quaisquer inicializações específicas para o programa em questão, e iniciar o processamento principal ciclo de eventos. O programador implementa cada uma das funções de retorno de chamada, bem como minha Init ().

Um exemplo de um programa completo OpenGL

Figura 3.2 (d) mostra um programa OpenGL completa. Isto pode ser estendido para desenhar objectos mais complexos e interessantes.

A inicialização no meu Init () define-se o sistema de coordenadas, o tamanho de ponto, a cor de fundo e a cor desenho. O desenho é encapsulado no retorno de chamada funcionar meu Display (). Como este programa é não-interativo, há outras funções de retorno de chamada são usados. Lavar GL () é chamado após os pontos são desenhados para assegurar que todos os dados são completamente processado e enviado para o mostrador. Isso é importante em alguns sistemas que operam em uma rede: dados é tamponado na máquina host e apenas enviados para o display remoto quando o amortecedor ficar cheio ou um resplendor gl () é executado. A saída do programa é como mostrado na Figura 3.2 (e).

```
#include <windows.h> // use as needed for your system
#include <gl/Gl.h>
#include <gl/glut.h>
//<<<<<<<<<<<<<<<<<<<< myInit >>>>>>>>>>>>>>>>>>>
void myInit(void)
{
glClearColor(1.0,1.0,1.0,0.0); // set white background color
glColor3f(0.0f, 0.0f, 0.0f); // set the drawing color
glPointSize(4.0); // a 'dot' is 4 by 4 pixels
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0, 640.0, 0.0, 480.0);
}
//<<<<<<<<<<<<<<<<<<<<<<< myDisplay >>>>>>>>>>>>>>>>>
void myDisplay(void)
{
glClear(GL_COLOR_BUFFER_BIT); // clear the screen
glBegin(GL_POINTS);
glVertex2i(100, 50); // draw three points
glVertex2i(100, 130);
glVertex2i(150, 130);
glEnd();
glFlush(); // send all output to display
}
//<<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>
void main(int argc, char** argv)
{
glutInit(&argc, argv); // initialize the toolkit
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // set display mode
glutInitWindowSize(640,480); // set window size
glutInitWindowPosition(100, 150); // set window position on screen
glutCreateWindow("my first attempt"); // open the screen window
glutDisplayFunc(myDisplay); // register redraw function
myInit();
glutMainLoop(); // go into a perpetual loop
}
```

Figura 3.2 (d): programa completo para OpenGL desenho 3 pontos

[Fonte: Computer Graphics Usando OpenGL, FS Colina Jr., Stephen M. Kelley, Prentice Hall, 2006, capítulo 2]

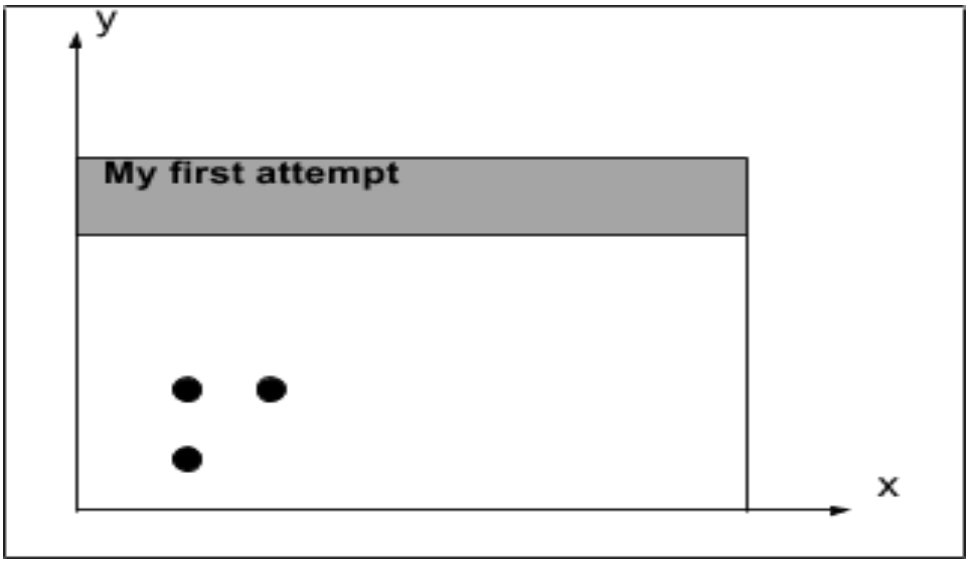


Figura 3.2 (e) Desenho três pontos, usando OpenGL

[Redesenhados a partir de: Computação Gráfica Usando OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, capítulo 2]

Interação com Mouse e Teclado

Dados sobre o mouse é enviada para o aplicativo, projetando a função de retorno meu Mouse () para tirar quatro parâmetros, para que ele tenha o protótipo:

invalidar o meu mouse (botão int, estado int, int x, int y);

Quando ocorre um evento de mouse do sistema chama a função registrada, fornecendo-o com os valores para esses parâmetros. O valor do botão será um dos seguintes:

GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON,

com a interpretação óbvia, e o valor de Estado será um dos seguintes: GLUT_UP ou GLUT_ DOWN. Os valores x e y relatório à posição do rato no momento do evento.

Nota: O valor de x é o número de pixels a partir do lado esquerdo da janela como esperado, mas o valor de y é o número de pixels para baixo a partir do topo da janela.

Exemplo: Colocar pontos com o mouse.

Cada vez que um usuário pressiona o botão esquerdo do mouse um ponto é desenhado na janela de tela na posição do mouse. Se o usuário pressiona o botão direito do mouse no programa termina. A versão do meu Mouse () mostrado a seguir faz o presente. Porque o valor de y da posição do rato é o número de pixels a partir do topo da janela do ecrã, um ponto é desenhado, não em (x, y), mas em (x, screenHeight - Y), onde screenHeight é assumido a altura da janela de pixels.

```
invalidar o meu mouse (botão int, estado int, int x, int y)
{
if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
drawDot(x, screenHeight -y);
else if(button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
exit(-1);
}
```

Mais exemplos são mostrados no livro de referência: Computação Gráfica Usando OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, capítulo 2.

Interação com teclado

Pressionando uma tecla no teclado filas um evento de teclado. A função de retorno de meu teclado () está registrado com esse tipo de evento através de excesso Keyboard Func (meu teclado). Deve ter protótipo:

invalidar o meu teclado (unsigned int chave, int x, int y);

O valor da chave é o valor ASCII da tecla pressionada. Os valores X e Y relatório da posição do mouse no momento em que o evento ocorreu. (Como antes y mede o número de pixels para baixo a partir do topo da janela).

O programador pode aproveitar muitas teclas do teclado para oferecer ao usuário um grande número de opções para chamar a qualquer momento em um programa. A maioria das implementações de meu teclado () consistem em um grande interruptor.

Declaração, com um caso para cada chave de interesse. Um exemplo é mostrado abaixo, onde pressionando 'p' desenha um ponto na posição do mouse; pressionando a tecla de seta para a esquerda acrescenta um ponto para alguma lista (global), mas não faz nenhum desenho; prementes saídas "E" do programa. Observe que, se o usuário mantém pressionada a tecla 'p' e move o mouse em torno de uma rápida seqüência de pontos é gerado para fazer um desenho "carta branca".

invalidar o meu teclado (char não assinado a chave, int mouseX, int mouseY)

```
{
GLint x = mouseX;
GLint y = screenHeight - mouseY; // flip the y value as always
switch(theKey)
{
case 'p':
drawDot(x, y); // draw a dot at the mouse position
```

```
break;
case GLUT_KEY_LEFT: List[++last].x = x; // add a point
List[ last].y = y;
break;
case 'E':
exit(-1); //terminate the program
default:
break; // do nothing
}
}
```

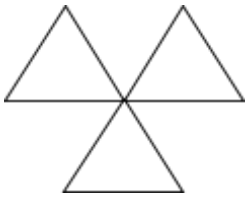
Conclusão

OpenGL é uma API que fornece rotinas útil para desenhar gráficos simples. Esta atividade tem apresentado as noções básicas de desenho gráfico simples usando OpenGL. A maioria dos aplicativos gráficos são escritos para um ambiente baseado em janelas. O programa abre uma janela na tela que pode ser movida e redimensionada pelo usuário, e ele responde a cliques de mouse e teclado traços.

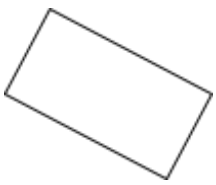
Avaliação

Escreva um programa em OpenGL para desenhar o seguinte:

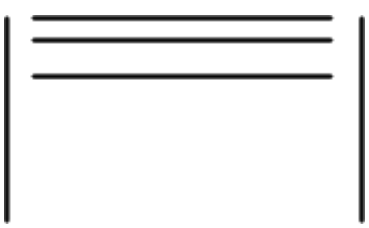
a. Três triângulos, como mostrado no diagrama



b. Um rectângulo, utilizando um rato, como se mostra no diagrama a seguir



c. As linhas a tracejado, como mostrado no diagrama a seguir, utilizando um teclado



Atividade 3 - Viewports, Mapeamento e Clipping

Referência: Computação Gráfica Usando OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, capítulo 3.

Introdução

Em um determinado problema, pode ser muito mais natural pensar em termos de x variando de, por exemplo, -1 a 1, e y varia entre -100,0 a 20,0 e não somente os valores positivos de x e y. Esta atividade apresenta métodos que permitem que um programador descrever objetos em qualquer sistema de coordenadas melhor se adapta ao problema na mão, e ter a imagem dimensionada e mudou para que ele caiba na janela da tela automaticamente.

Detalhes da atividade

Veja as portas

Na actividade anterior, desenhos usado o sistema básico de coordenadas da janela do ecrã: coordenadas que são, essencialmente, em pixels, que se estende de 0 a alguns tela valor de largura - 1 em x, e de 0 a alguns tela valor -1 Altura em y. Isto significa que somente valores positivos de x e y podem ser utilizados, e os valores deve estender-se ao longo de um grande intervalo (várias centenas de pixels) para se obter um desenho de alguns de tamanho razoável. Em um determinado problema, pode ser muito mais natural pensar em termos de x variando de, por exemplo, -1 a 1, e y varia entre -100,0 a 20,0. A imagem pode ser escalado e se mexeu para que ele caiba na janela da tela automaticamente.

O espaço em que os objetos são descritos é chamado de coordenadas mundo. É o sistema de coordenadas cartesianas xy-habitualmente utilizado em matemática, com base em qualquer unidades são convenientes. Uma janela retangular mundo é definido nessas coordenadas mundo. A janela mundo especifica qual parte do "mundo" deve ser desenhado. O entendimento é que tudo o que está dentro da janela deve ser elaborado; tudo o que está fora deve ser cortada fora e não sacados. Além disso, nós definimos uma janela retangular na janela de tela na tela. Um mapeamento (que consiste em escaladas e deslocamentos) entre a janela do mundo e janela de exibição é estabelecido de modo a que, quando todos os objetos no mundo são desenhadas, as peças que se encontram dentro da janela do mundo são automaticamente mapeados para o interior da janela de exibição. Assim, um programador pensa em termos de "olhar através de uma janela" para os objetos sendo desenhado, e colocando um "instantâneo" de tudo o que é visto na janela para que a janela de exibição na tela. Esta abordagem janela / porta de visualização faz com que seja muito mais fácil de fazer as coisas naturais, como "zoom in" em um detalhe na cena, ou "garimpando em torno de" uma cena.

Mundo do Windows e exibição de janela

Um exemplo é usado para ilustrar o uso de janelas e janelas de exibição. Sinc (x), uma função de processamento de sinal em famoso, é expressa como se segue:

$$\sin c(x) = \frac{\sin(\pi x)}{\pi x}$$

Como x varia de -∞ a ∞the valor de sinc (X) varia ao longo de uma gama de -1 a 1. Por exemplo, um lote de sinc (x) que é centrada em (0,0) e mostrando sinc (x) para espaçadas entre os valores de x, diz -4.0 e 4.0 pode ser desenhada utilizando OpenGL para gerar uma trama, como mostrado na Figura 4.3 (a)

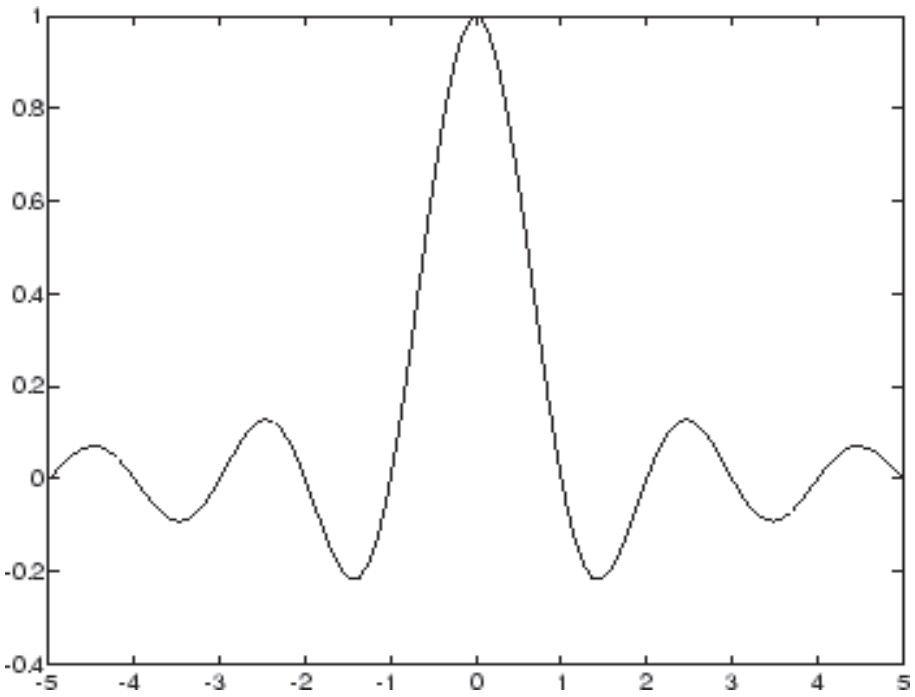


Figura 3.3 (a). Sinc (x) Plot

[Fonte: http://www.mathworks.com/help/signal/gs/waveform-generation-time-vectors-and-sinusoids.html]

O código de função de exibição OpenGL é a seguinte:

anular a minha Display (void)

```
{
glBegin(GL_LINE_STRIP);
for(GLfloat x = -4.0; x < 4.0; x += 0.1)
{
GLfloat y = sin(3.14159 * x)/(3.14159 * x);
glVertex2f(x, y);
}
glEnd();
glFlush();
}
```

O código nestes exemplos opera num sistema de coordenadas natural para o problema: X é feita variar em pequenos incrementos de -4,0 a 4,0. A questão-chave aqui é a forma como os vários valores (x, y) se tornar escalado e se mexeu para que a imagem seja exibida corretamente na janela do ecrã.

Janela para Mapeamento Viewport

[Fonte: http://www.asksatyam.com/2011/01/window-to-viewporttransformation.html]

Para visualizar a janela o mapeamento de porta é usado para mapear coordenadas mundiais para a tela ou coordenadas do dispositivo. A janela é uma área que define o que é para ser visualizado, enquanto um visor, é a área que define onde ele é para ser exibido. Figura 4.3 (b) mostra uma janela em coordenadas globais e janela de exibição em coordenadas de tela ou visor. Seja qual for a área é selecionada na janela é enviada para a janela de exibição. O tamanho da janela de visualização pode ser menor ou maior do que a janela.

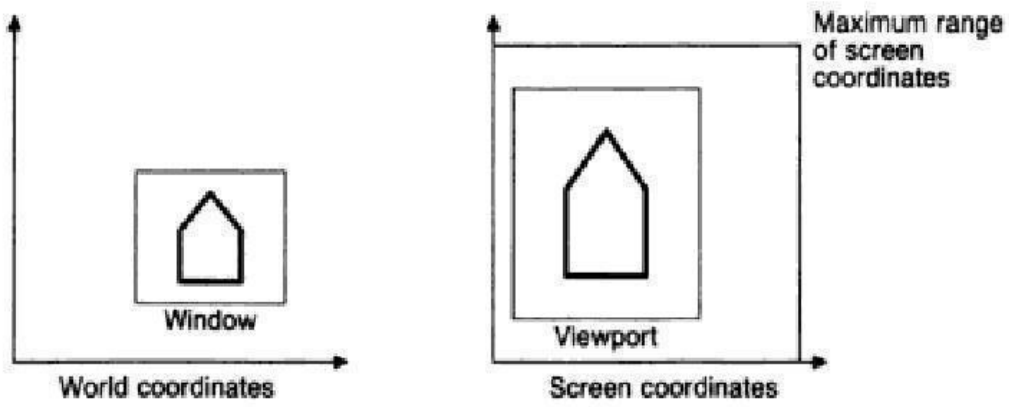


Figura 3.3 (b): A janela em coordenadas globais e janela de exibição em coordenadas de tela

[Fonte: http://www.asksatyam.com/2011/01/window-to-viewporttransformation.html]

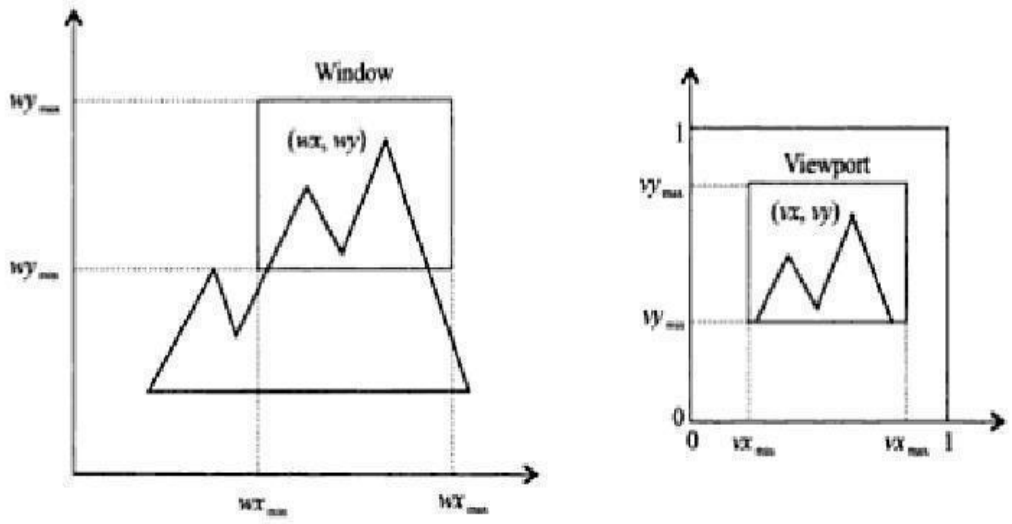


Figura 3.3 (c): Janela para viewport exemplo transformação

[Fonte: http://www.asksatyam.com/2011/01/window-to-viewporttransformation.html]

A transformação porta-janela de visão mantém a posição relativa de um ponto na janela, bem como em vista a porta. Um ponto na posição (xw, yw) na janela é mapeado para a posição (xv, yv) na porta de exibição associada. Um exemplo é mostrado na figura 4.3 (c).

Mais informações e detalhes podem ser encontrados a partir de: http://www.di.ubi.pt/~agomes/cg/teoricas/04e-windows.pdf

Clipping

Qualquer parte de uma imagem que está além dos limites da tela não pode ser

exibidas, portanto, métodos para lidar com imagens que são maiores do que o tamanho da tela de exibição disponíveis são usados. Suponhamos que o tamanho da imagem a ser mostrado é maior do que o tamanho do ecrã, em seguida, apenas uma parte da imagem pode ser exibida. O contexto é semelhante à da visualização de uma cena de fora da janela. Enquanto a cena externa é bastante grande, uma janela irá permitir ver apenas aquela parte da cena como pode ser visível a partir da janela - o último é limitado pelo tamanho da janela. Se nós presumimos que a tela, o que nos permite ver as imagens como uma janela, então qualquer imagem cujas partes se encontram fora dos limites da janela não pode ser mostrado e para fins de algoritmos, eles têm de ser "cortado"

Mais informações podem ser obtidas a partir de:

- [Computador Gráficos usando OpenGL, FS Hill Jr., Stephen M. Kelley, Prentice Hall, 2006, Capítulo 3]

Conclusão

Esta atividade tem apresentado como objetos em um sistema de coordenadas podem ser mapeados para a outra, ou seja, mundo para rastrear / coordena dispositivo. Isto permite que a imagem a ser escalada e deslocado de modo que possa encaixar na janela do dispositivo.

Avaliação

1. Definir janela e ponto de vista.
2. Definir recorte..
3. Explique porque é importante recorte em desenho gráfico.
4. Descreva como janela para mapeamento janela de exibição é executada.

Resumo da Unidade

Esta unidade tem coberto os aspectos básicos em imagem de desenho usando OpenGL como uma API. Também são apresentados os programas OpenGL simples para desenhos de imagem simples, como linhas e retângulos. Transformação de imagem a partir da janela de visor também é apresentada, em que o mapeamento de um quadro de um sistema para um sistema de coordenadas é necessário coordenar realizada. A unidade apresentada também como imagens pode ser cortada a fim de encaixar em dispositivos de exibição.Leituras de

unidade e Outros recursos

As leituras desta unidade estão a ser encontradas em curso leituras de nível e outros recursos.

Unidade 4: 2D e Gráficos 3D

Introdução unidade

Este tópico irá apresentá-lo aos princípios básicos sobre os objetos de visualização em duas dimensões e três dimensões. O tópico também irá cobrir os vários dispositivos físicos que estão disponíveis para a entrada de dados em estações de trabalho gráficas com a sua classificação lógica.

Objetivos da unidade

Após a conclusão desta unidade, você deve ser capaz de:

1. Visualização de objetos em duas dimensões
2. Descrever os dispositivos de entrada de base em gráficos
3. Ver e transformar objetos em três dimensões

Atividades de Aprendizado

Atividade 1

Introdução

Esta secção irá apresentar-lhe o mecanismo formal para a exibição de pontos de vista de uma imagem em um dispositivo de saída. Qualquer sistema de coordenadas cartesianas, conveniente, referido como o sistema de referência mundo coordenada, pode ser utilizado para definir a imagem. Para obter uma imagem bidimensional, uma vista é seleccionado especificando uma subzona da área total da imagem. Transformações do mundo em coordenadas do dispositivo envolve tradução, rotação e escala de operações, bem como os procedimentos para apagar as partes da imagem que estão fora dos limites de uma área de exibição selecionada.

Detalhes da atividade

A área mundial de coordenadas selecionado para exibição é chamado de janela. Uma área em um dispositivo de exibição de uma janela que é mapeado é chamado uma janela de exibição. A janela define o que deve ser visto; a janela de exibição define onde está a ser exibido. Muitas vezes, janelas e janelas de exibição são retângulos em posição normal, com as bordas do retângulo paralelo aos eixos coordenados. Outros janela ou porta de visualização geometrias, tais como círculos e formas poligonais geral, são utilizados em algumas aplicações, mas estas formas demorar mais tempo a processar. Em geral, o mapeamento de uma parte de uma cena ao mundo coordenada coordenadas do dispositivo é referido como uma transformação de visualização. Por vezes, a transformação de visualização bidimensional é simplesmente referida como a transformação janela-visor para a transformação ou de janelas.

Mas, em geral, visualização envolve mais do que apenas a transformação da janela para a janela de exibição. Figura 4.1.1 ilustra o mapeamento de uma secção de imagem que cai dentro de uma janela rectangular para uma janela de exibição designada & angular. Na terminologia de computação gráfica, a janela do prazo previsto originalmente para uma área de uma imagem que é selecionado para visualização, conforme definido no início desta seção. Infelizmente, o mesmo termo agora é usado em sistemas de window-manager para se referir a qualquer área da tela retangular que pode ser movido sobre, redimensionada, e fez ativo ou inativo. Neste capítulo, nós só vai usar a janela de termos para se referir a uma área de uma cena mundial coordenada que foi selecionado para exibição

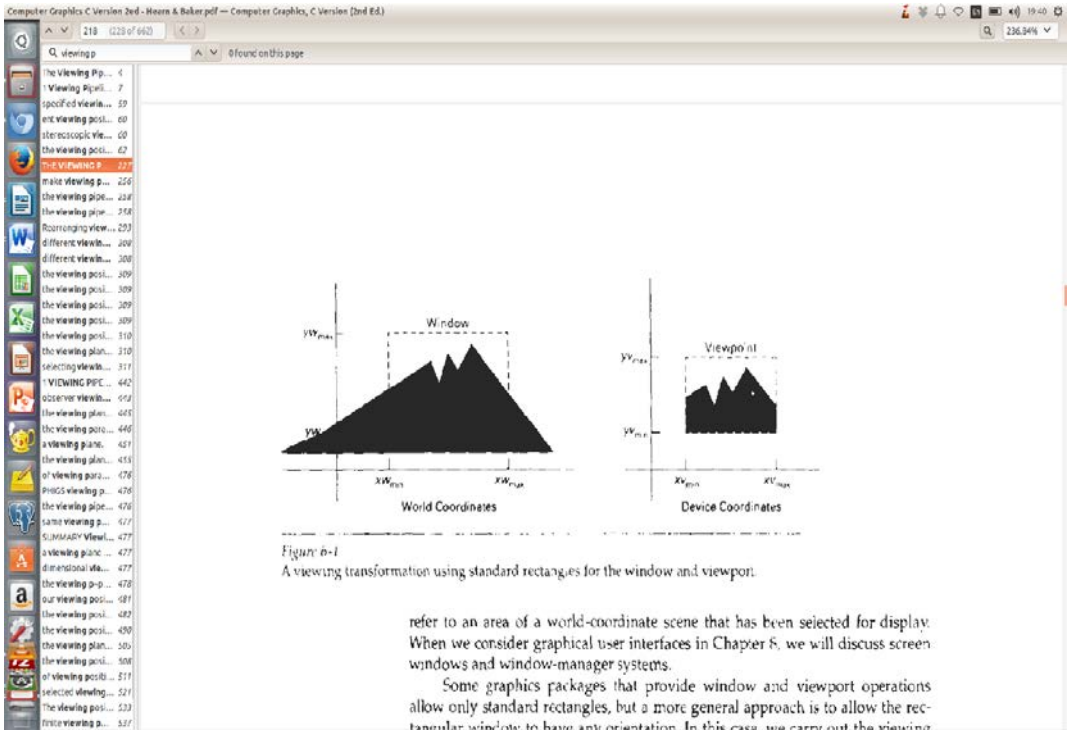
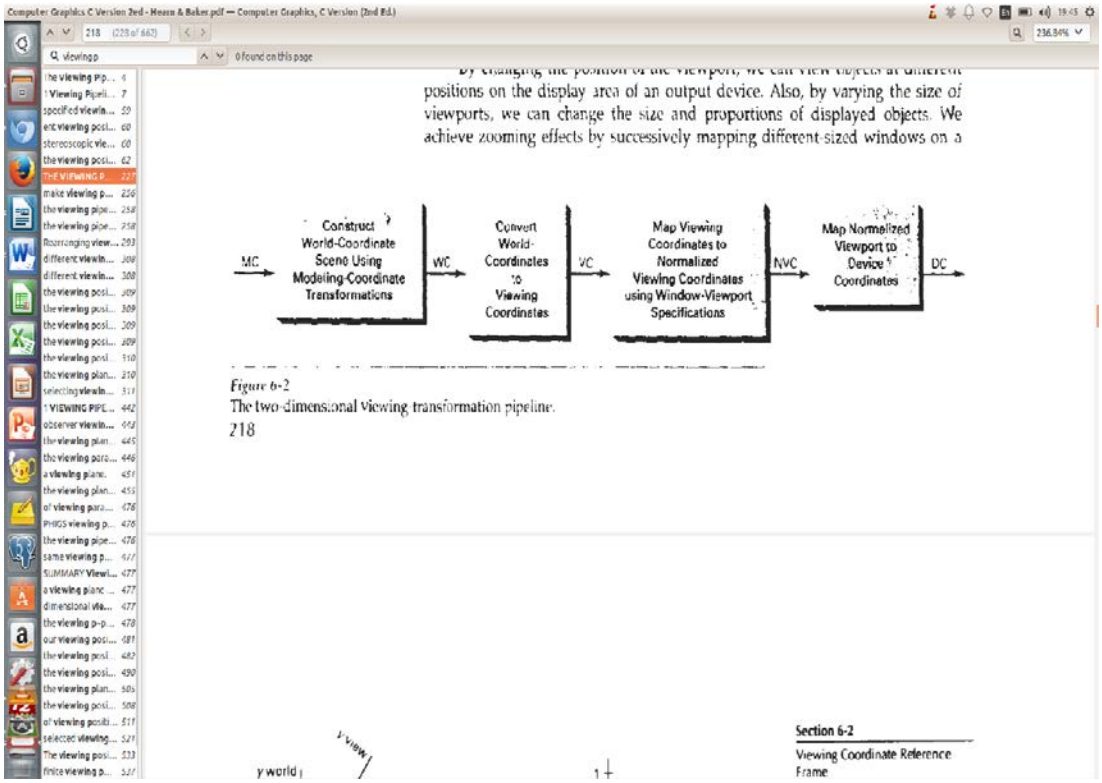


Figura 4.1.1: A transformação de visualização usando retângulos padrão para a janela e porta de visualização.

(Hearn & Baker, "Computer Graphics C Versão 2ed")

Alguns pacotes gráficos que fornecem operações de janela e porta de visualização permitem retângulos único padrão, mas uma abordagem mais geral é a de permitir que a janela retangular para haw qualquer orientação. Neste caso, realizar a transformação visualização em várias etapas, como indicado na Figura 4.1.2 abaixo

d ")



4.1.2: O gasoduto bidimensional visualização de transformação d ").

(Hearn & Baker ", Versão " de computador Gráficos C 2ed")

Conclusão

Nesta seção, vimos como mapear, coordenar um mundo bidimensional e um dispositivo de exibição de cena. A visualização do gasoduto de transformação inclui a construção da cena, usando transformações em modelagem de transferências para visualização mundiais das coordenadas de mapeamento de objetos de descrição coordenadas e normalizadas para mapear coordenadas do dispositivo. Finalmente, as coordenadas normalizadas são especificadas na gama de 0 à 1, usadas para fazer pacotes de visualização independente de particular dispositivo de saída.

Atividade 2

Introdução

Esta seção irá apresentá-lo para os vários dispositivos físicos que estão disponíveis para a entrada de dados em estações de trabalho gráficas. Estes dispositivos serão agrupados de acordo com sua classificação lógica.

Detalhes da atividade

Keyboard é um teclado alfanumérico em um sistema de gráficos é usado principalmente como um dispositivo para introduzir sequências de texto. O teclado é um dispositivo eficiente para a introdução de tais dados não gráficos como etiquetas de imagem associados a uma exibição de gráficos. Teclados também podem ser fornecidas com recursos para facilitar a entrada de coordenadas de tela, opções de menu, ou as funções gráficas..

Um rato é a caixa de mão pequeno usado para posicionar o cursor na tela. Rodas ou rolos na parte inferior do mouse pode ser usado para registrar a quantidade e direção do movimento. Outro método para detectar o movimento do rato é com um sensor óptico. Para estes sistemas, o rato é movido sobre uma almofada especial do rato que tem uma grade de linhas horizontais e verticais. O sensor óptico detecta o movimento entre as linhas da grelha.

Trackball é uma bola que pode ser girado com os dedos ou palma da mão, para produzir o movimento screen-cursor. Potenciômetros, ligado à bola, medir a quantidade e sentido de rotação. Trackballs são freqüentemente montados em teclados ou outros dispositivos, como o mouse Z.

Enquanto um trackball é um dispositivo de posicionamento de duas dimensões, uma bola de espaço proporciona seis graus de liberdade. Ao contrário do trackball, uma bola de espaço na verdade não mover. Os extensómetros medir a quantidade de pressão aplicada para a bola para proporcionar espaço de entrada para o posicionamento e orientação espacial como a bola é empurrado ou puxado em várias direcções. Bolas de espaço são usados para tridimensional operação de posicionamento e seleção

Um joystick consiste de uma pequena alavanca, vertical (chamada vara) montada sobre uma base que é usada para dirigir o cursor em torno. A maioria dos joysticks selecionar posições de tela com movimento real vara; outros respondem à pressão sobre o stick. Alguns joysticks são montados sobre um teclado; outros funcionam como unidades autônomas.

Digitadores são um dispositivo comum para desenho, pintura, ou interativamente selecionando coordenar posições sobre um objeto é um digitalizador. Estes dispositivos podem ser utilizados para introduzir valores de coordenadas em qualquer uma de duas dimensões ou um espaço tridimensional. Normalmente, um digitalizador é usado para digitalizar em um desenho ou objeto e para introduzir um conjunto de posições discretas coordenar, que podem ser unidas com segmentos de reta, que harmoniza as curva ou superfície formas.

Desenhos, gráficos, fotos coloridas e em preto-e-branco, ou o texto podem ser armazenados para o processamento de computador com um scanner de imagem por meio de um mecanismo de leitura óptica sobre a informação a ser armazenada. As gradações de tons de cinza ou cor são então gravados e armazenados em uma matriz. Uma vez que temos a representação interna de uma imagem, podemos aplicar transformações para girar, escala, ou cortar a imagem para uma área da tela particular.

Painéis de toque de permitir que objetos exibidos ou posições de tela para ser selecionado com o toque de um dedo. Uma aplicação típica de painéis de toque é para a seleção de opções de processamento que são representados com ícones gráficos.

Canetas de luz são dispositivos em forma de lápis são usados para selecionar as posições de tela através da detecção da luz que vem de pontos na tela do CRT. Eles são sensíveis à curta explosão de luz emitida a partir do revestimento de fósforo no instante em que o feixe de electrões atinge um determinado ponto.

Sistemas de voz são reconhecimento de voz são usados em algumas estações de trabalho gráficas como dispositivos de entrada para aceitar comandos de voz. A entrada do sistema de voz pode ser usado para iniciar operações de gráficos ou para inserir dados. Estes sistemas operam, combinando uma entrada contra um dicionário pré-definido de palavras e frases.

Classificação lógica dos dispositivos de entrada:

Dispositivos Locator são usados para fornecer uma posição coordenada na localização desejada. Dispositivos comuns são thumbwheel, trackball, joystick, rato e painéis de toque.

Stoke dispositivos utilizados para fornecer uma série de posições coordenadas. É essencialmente várias chamadas para o dispositivo de um localizador. Dispositivos comuns são tablet e caneta de luz.

Dispositivos de corda são usados para fornecer texto. O dispositivo seqüência comum é o teclado. Outros dispositivos são dispositivos tempos com reconhecedores de caracteres.

Dispositivos avaliadores são usados para fornecer valores escalares de ângulos de rotação, fatores de escala, parâmetros de aplicação. Valuators dispositivos comuns são o teclado (entrada numérica)

Dispositivo de escolha são usados para selecionar as opções de menu. Dispositivos escolha comum são os painéis de toque, canetas de luz

Escolher dispositivos são usados para selecionar os componentes de imagem. Um dispositivo de picareta comum é a caneta de luz.

Conclusão

Neste capítulo, vimos para a entrada gráfica, temos uma gama de dispositivos para escolher e como esses dispositivos podem ser categorias logicamente.

Avaliação

Q1. Listar todos os dispositivos com base em sua classificação lógica

Q2. Identificar os dispositivos localizadores, que também podem ser utilizados como os dispositivos de Stoke.

Atividade 3

Introdução

Métodos de transformações geométricas e modelagem de objetos em três dimensões são estendidos a partir de métodos bidimensionais, incluindo considerações para a coordenada z. Nós agora traduzir um objecto tridimensional, especificando um vetor de translação três, que determina o quanto o objecto está a ser movido em cada uma das três direções de coordenadas.

Detalhes da atividade

TRADUÇÃO

Em uma representação homogénea de coordenadas tridimensional, um ponto é traduzida a partir da posição P = (x, y, z) na posição P '= (x', y ', z'). Parâmetros tx, ty, e tz, especificando distâncias de tradução para as direcções x, y, z e coordenar, são atribuídos quaisquer valores reais. A representação da matriz é equivalente para as três equações abaixo

x'=:r+ tx,

y'=y+ ty,

z ' = z + tz;

ROTAÇÃO

Para gerar uma transformação de rotação de um objeto, é preciso designar um eixo de rotação (sobre a qual o objeto deve ser girado) e a quantidade de rotação angular. Ao contrário das aplicações bidimensionais, em que todas as transformações são levadas a cabo no plano xy, uma rotação tridimensional pode ser especificada em torno de qualquer linha no espaço. Os eixos de rotação mais fácil de manusear são aqueles que são paralelos aos eixos coordenados. Além disso, podemos usar combinações de rotações do eixo de coordenadas (juntamente com traduções apropriadas) para especificar qualquer rotação geral.

ESCALA

A expressão da matriz de referência, a transformação de dimensionamento de uma posição P = (x, y, z) em relação à origem das coordenadas pode ser escrito como mostrado abaixo, onde os parâmetros de escalonamento sx, sy, e sz, são atribuídos quaisquer valores positivos.

P' = SP

As expressões explícitas para as transformações de coordenadas para escala em relação à origem são

x' = x . sx,

y' = y . sy,

z' = Z . Sz,

O dimensionamento de um objecto com a transformação altera o tamanho do objecto e reposiciona o objeto em relação à origem das coordenadas. Além disso, se os parâmetros de transformação não são todas iguais, as dimensões relativas do objecto são alteradas.

Conclusão

Representações de tradução e de escala são extensões diretas de representações de transformação bidimensionais. Para rotações, no entanto, precisamos de representações mais gerais, uma vez que objetos pode ser girado sobre qualquer eixo especificado no espaço. Qualquer rotação tridimensional pode ser representada como uma combinação de rotações de base em torno dos eixos x, y, e z.

Avaliação

Q1. Mencione outros métodos que podem ser utilizados para transforma objectos em três dimensões.

Atividade 4

Introdução

Em aplicações de gráficos bidimensionais, as operações de transferência de um entrando em posições a partir do plano de coordenadas mundo para posições de pixel no plano do dispositivo de saída. Para aplicações graylucs tridimensionais, a situação é um pouco mais complexo, uma vez que agora temos mais opções quanto à forma como são vistas a ser gerado. Primeiro de tudo, nós podemos visualizar um objeto a partir de qualquer posição espacial: de frente, de cima, ou na parte de trás.

Detalhes da atividade

As etapas para a geração de computador de uma visão de uma cena tridimensional são um tanto análogo aos processos envolvidos na tomada de uma fotografia. Para tirar um instantâneo, primeiro precisamos para posicionar a câmera em um determinado ponto no espaço. Então precisamos decidir sobre a orientação da câmara (veja a figura abaixo):

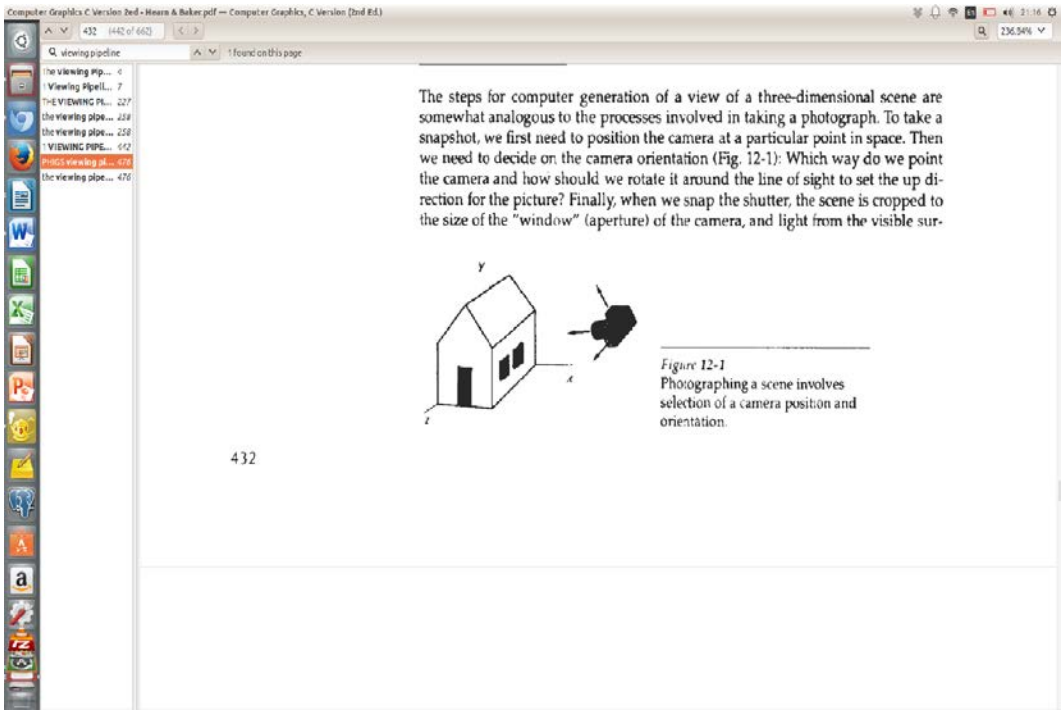

The steps for computer generation of a view of a three-dimensional scene are somewhat analogous to the processes involved in taking a photograph. To take a snapshot, we first need to position the camera at a particular point in space. Then we need to decide on the camera orientation (Fig. 12-1): Which way do we point the camera and how should we rotate it around the line of sight to set the up direction for the picture? Finally, when we snap the shutter, the scene is cropped to the size of the "window" (aperture) of the camera, and light from the visible sur-

Figure 12-1
Photographing a scene involves selection of a camera position and orientation.

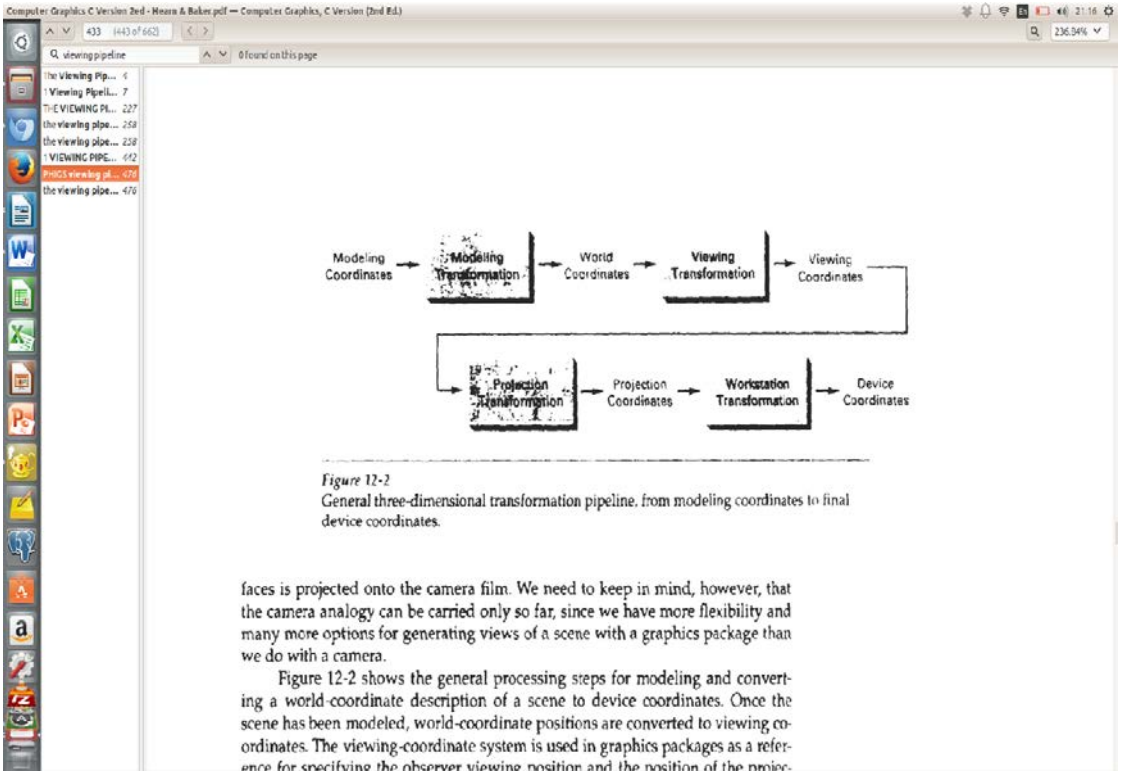
432

4.4.1: Fotografar uma cena envolve a seleção de uma posição da câmera e orientação

(Hearn & Baker, "Computação Gráfica C Versão 2ed")

Que maneira nós apontar a câmera e como devemos girá-lo ao redor da linha de visão para definir a direção para a foto? Finalmente, quando tirar o obturador, a cena é cortada para o tamanho da "janela" (abertura) da câmera, luz e das superfícies visíveis é projetada sobre o filme câmera. Precisamos ter em mente, no entanto, que a analogia câmara só pode ser realizado até agora, uma vez que temos mais flexibilidade e muitas mais opções para gerar pontos de vista de uma cena com um pacote de gráficos do que nós com uma câmera.

A figura abaixo mostra as etapas de processamento gerais para modelagem e conversão de uma descrição mundo coordenada de uma cena para as coordenadas do dispositivo. Uma vez que a cena foi modelado, mundialmente coordenar posições são convertidos em coordenadas de visualização. O sistema de visualização de coordenadas será utilizada em pacotes gráficos como referência para especificar a posição de observador a ver ea posição do plano de projeção, o que podemos pensar em analogia com o plano do filme câmera. Em seguida, as operações de projecção são realizadas para converter a descrição de coordenadas de visualização da cena para coordenar as posições no plano de projecção, o que irá então ser mapeado para o dispositivo de saída. Os objetos fora dos limites especificados de visualização são cortados hm uma análise mais aprofundada, e os objetos restantes são processados através de procedimentos de identificação e superfície-renderização visível de superfície para produzir a exposição dentro da janela de exibição do dispositivo.



4.4.2: Geral pipeline de transformação em três dimensões, a partir de modelagem coordenadas para coordenadas finais dispositivos (Hearn & Baker, "Computação Gráfica C Versão. 2ed")

Conclusão

Visualizando procedimentos para cenas tridimensionais seguir a abordagem geral utilizado na visualização bidimensional. Isto é, primeiro criamos uma cena mundial coordenada a partir das definições dos objetos em coordenadas de modelagem. Então montamos uma visão coordenada quadro de referência e descrições de objetos de transferência de coordenadas mundiais para as coordenadas de visualização. Finalmente, a visualização de coordenadas descrições são transformadas em coordenadas do dispositivo.

Ao contrário de visualização bidimensional, no entanto, a visualização tridimensional requer rotinas de projecção para transformar descrições de objectos para um plano de visualização antes da transformação de coordenadas do dispositivo. Também as operações de visualização tridimensional envolvem parâmetros mais espaciais. Nós podemos usar a analogia câmera para descrever parâmetros de visualização tridimensional, que incluem a posição da câmera e orientação

Avaliação

Q1. Explique brevemente como um objeto pode ser transformado a partir do Mundo coordenada para o Dispositivo Coordenado em três dimensões de visualização.

Resumo da Unidade

Esta unidade apresentou os princípios básicos de objetos de visualização em duas e três dimensões. Vários dispositivos físicos para entrada de dados em estações de trabalho gráficas foram apresentados, em conjunto com a sua classificação lógica. Métodos de transformações geométricas e modelagem de objetos em três dimensões também foram apresentadas pelo qual as transformações são prorrogado a partir de métodos bidimensionais, incluindo considerações para a coordenada z.

Sede da Universidade Virtual africana

The African Virtual University
Headquarters

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oer@avu.org

Escritório Regional da Universidade Virtual Africana em Dakar

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org