

Computação Gráfica – Seleção e Picking

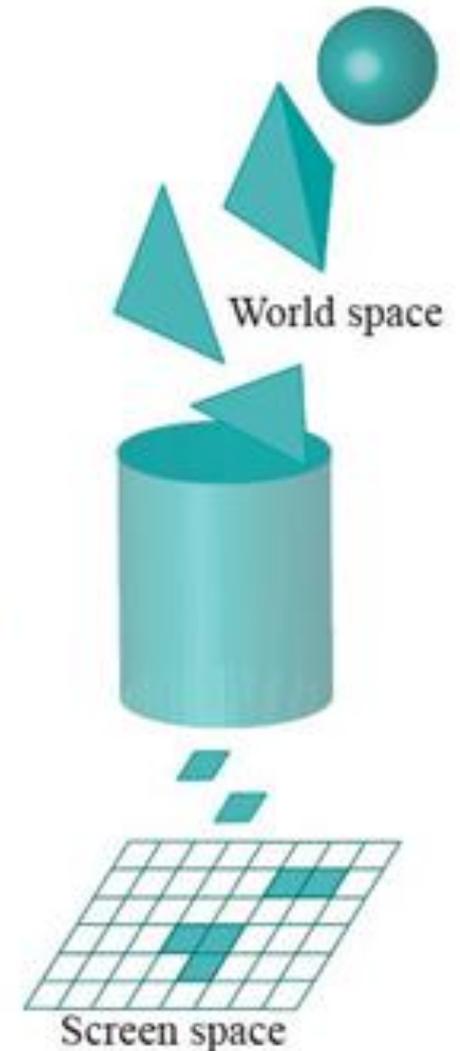
Profa. Mercedes Gonzales
Márquez

Tópicos

- Seleção
- Picking

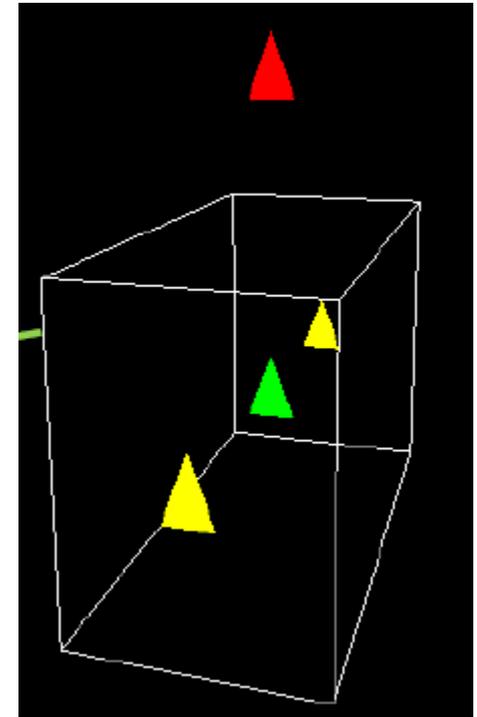
Seleção - Problema

- Muitas vezes podemos requerer realizar aplicações interativas como identificar objetos na tela e manipulá-los. A questão é:
- Como saber em que Z foi feito clique para determinar qual objeto foi clicado? Lembremos que o clique do mouse é 2D (no espaço da imagem) e os objetos estão descritos em coordenadas do mundo e na maioria dos casos em 3D.



Seleção - Problema

- A ideia é permitir que o usuário especifique um volume e então encontre os objetos que intersetem (*hit*) este volume. Para isso usamos o modo seleção do OpenGL o qual retornará uma lista de primitivas que intersetem este volume de seleção (*hit records*). IDs (names) são associados com objetos ou conjunto de primitivas.



Seleção - Passos

1. Especificar um buffer para ser usado para retornar os hit records com `glSelectBuffer()`.
2. Entrar no modo de seleção especificando `GL_SELECT` com `glRenderMode()`.
3. Definir o volume que você deseja usar na seleção. Usualmente é um volume diferente do que o utilizado na renderização, portanto é interessante voltar ao estado “original” com: `glPushMatrix()` e `glPopMatrix()`.
4. Inicializar a pilha de nomes usando `glInitNames()` e `glPushName()`.

Seleção - Passos

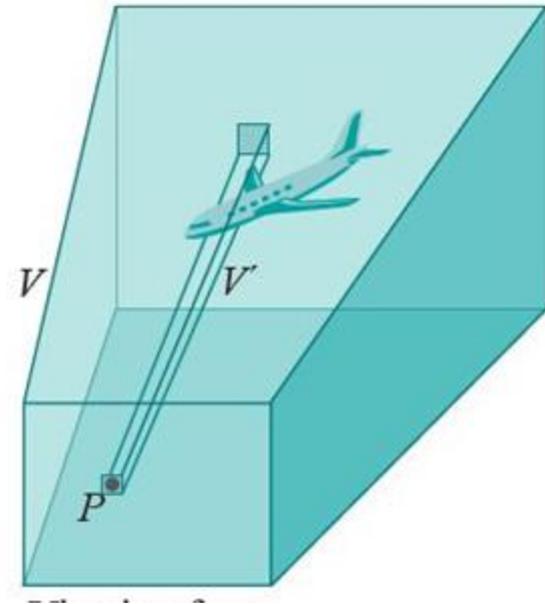
5. Especificar comandos de desenho e comandos para manipular a pilha de nomes tal que cada primitiva de interesse tenha um nome. Exemplo:

```
glLoadName(1);  
drawRectangle(0.0, 0.0, 3.0, 1.0, 0.0, 0.0);
```
6. Sair do modo seleção e processar os registros do buffer de retorno os quais incluem id e informação de profundidade.

Veja o programa `selection.cpp` e identifique os 6 passos da seleção.

Picking - Problema

- Muitas vezes queremos verificar qual o objeto que intersesta o quadrado em volta do cursor.
- OpenGL permite definir uma matriz para obter objetos que foram desenhados próximos a uma região de pixels.
- `void gluPickMatrix(GLdouble x, GLdouble y, GLdouble width, GLdouble height, GLint viewport[4]);` Onde `x` e `y` são a posição do centro (cursor) e `width` e `height` são o tamanho da região e `viewport` são as dimensões da viewport.



Picking - Passos

1. Inicializar *buffer* de retorno
2. Entrar modo de seleção
3. Inicializar *stack* de nomes simbólicos
4. **Definir matriz de sensibilidade baseada na posição do dispositivo de entrada (usar evento do mouse)**
5. Definir volume de seleção
6. “Desenhar” a cena incluindo o nome simbólico dos objetos
7. Sair do modo seleção e processar os registos do *buffer* de retorno

Picking - Exemplo

Veja o programa `BallAndTorusPicking.cpp` e identifique os 7 passos do picking.

Picking – Mais Exemplos

Rode os seguintes exemplos que usam picking:

1. PickLine.c : Pressiona o botão esquerdo do mouse para entrar em modo picking. Você obtém dois hits se você apertar o mouse enquanto o cursor está onde as linhas intersectam.
- PickDepth: Três retângulos sobrepostos são desenhados. Quando o botão esquerdo do mouse é pressionado entramos no modo picking. Os retângulos que são desenhados “abaixo” da posição do cursor são “pegos”.

Picking – Mais Exemplos

Os retângulos são desenhados em profundidades diferentes. Já que um único nome é usado para identificar todos os três retângulos, somente um hit pode ser registrado. Porém, se mais do que um retângulo é “pego” aquele único hit tem valores diferentes de profundidade mínima e máxima.

Exercício:

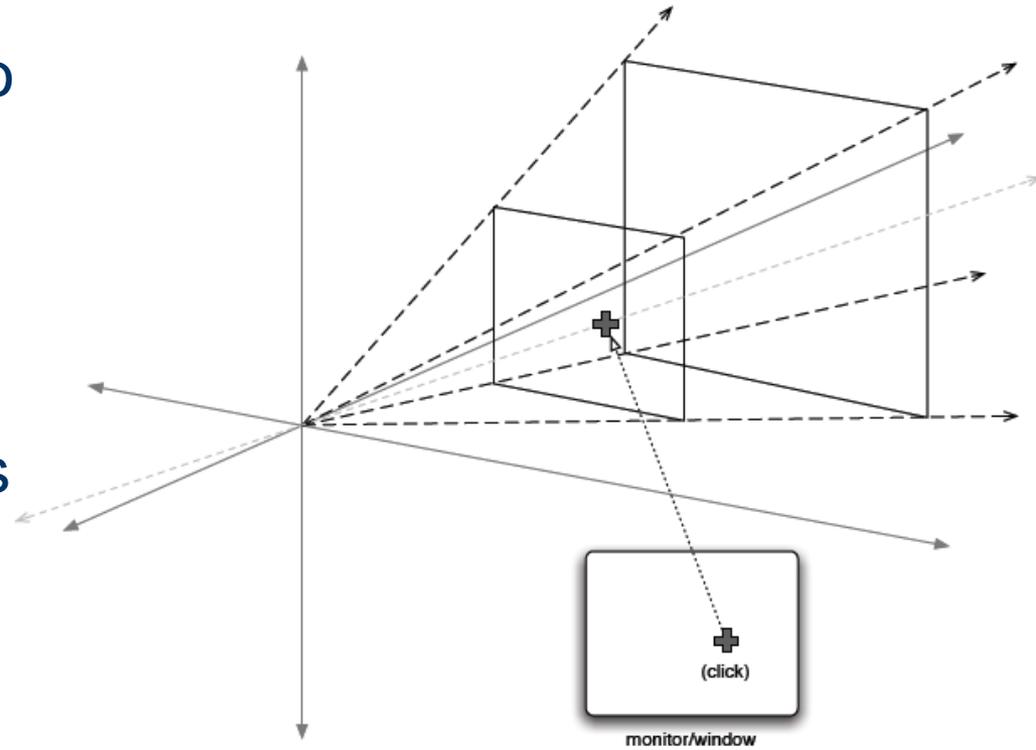
Melhore o programa `canvas.cpp` tal que as figuras na área de desenho possam ser clicadas e arrastadas.

Picking

- Exemplo:
 1. Rode o programa drag.c e identifique o seguinte:
 - (a) Como é feita a seleção do retângulo
 - (b) Como é feita a seleção quando temos mais de um objeto na mesma posição.
 - (c) Como é feito o arraste do retângulo antes de clicar na posição definitiva.
 - (d) Como é feita a armazenagem definitiva da nova posição após o arraste.

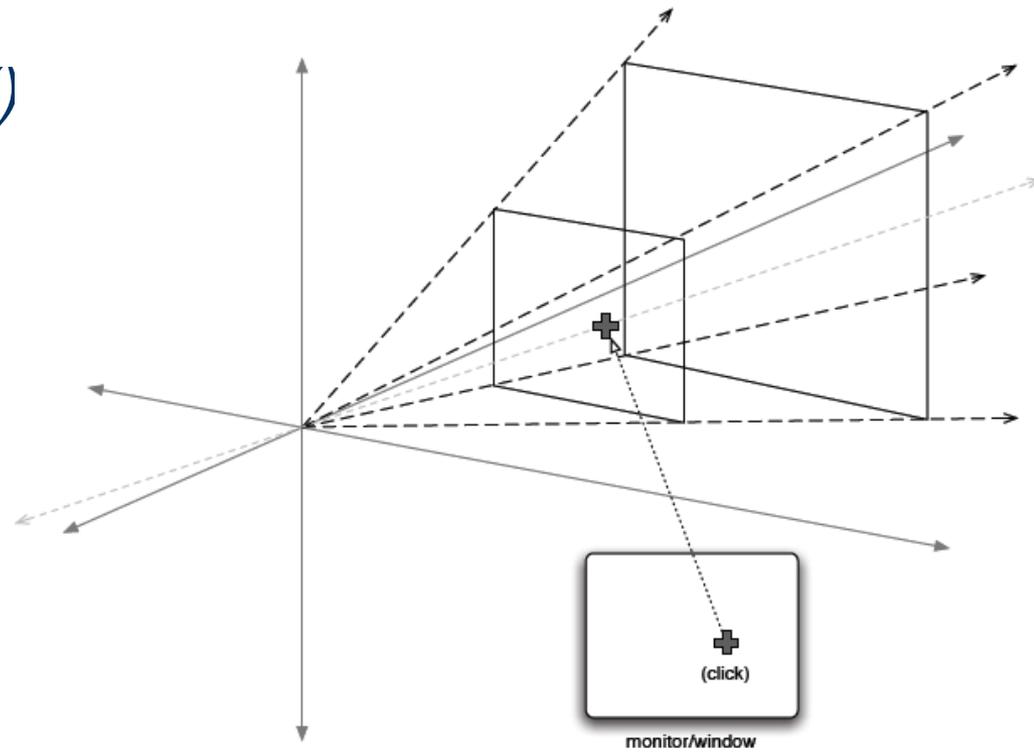
GluUnProject

- Dado um evento de mouse (clique, movimento ou arraste), como traduzir as coordenadas do evento de mouse em coordenadas do espaço tridimensional?
- Isto é: Dado um ponto na tela 2D, quais são as suas correspondentes coordenadas no espaço tridimensional?



GluUnProject

- Perceba que um clique 2D não provém de um único ponto, mas de um raio.
- O comando *gluUnProject()* fornecerá os pontos extremos desse raio.



GluUnProject

- *gluUnProject()* inverte o cálculo da projeção, ao invés de ir de um ponto no espaço tridimensional a um ponto na tela, que é o que fizemos até agora.

Ponto no “mundo” -> transformação modelagem -> transformação de projeção -> transformação em coordenadas da tela -> Ponto na tela.

- ...nós tomamos um ponto na tela e vamos na direção contrária:

Ponto na tela -> transformação em coordenadas da tela inversa -> projeção inversa -> modelagem inversa -> ponto no “mundo”

GluUnProject

- Assim nós temos o comando *gluUnProject*

```
GLint gluUnProject (GLdouble winX, GLdouble winY,  
GLdouble winZ, const GLdouble *model, const GLdouble  
*proj, const GLint *view, GLdouble* objX, GLdouble* objY,  
GLdouble* objZ);
```

Mapeia as coordenadas da tela (winx, winy, winz) em coordenadas do objeto usando as transformações definidas por matriz modelview (modelMatrix), matriz de projeção (projMatrix) e viewport. As coordenadas do objeto resultante são retornadas em objx, objy, and objz.

GluUnProject

- A função retorna *GL_TRUE*, indicando sucesso, ou *GL_FALSE*, caso contrário (caso de uma matriz não inversível).
- Por quê winZ ?
- $winZ == 0.0$ (a tela) corresponde a $objZ == -N$ (o plano near), e $winZ == 1.0$ corresponde to $objZ == -F$ (o plano far)
- Para obter então os extremos do raio no mundo que corresponde ao ponto clicado na tela você precisa chamar *gluUnProject()* duas vezes, uma vez com $winZ == 0.0$ e com $winZ == 1.0$.

GluUnProject

Exercícios

1. Unproject.c demonstra **gluUnProject()** lendo a posição do mouse e determinando os pontos tridimensionais nos planos de corte near e far a partir dos quais foi transformado. As coordenadas calculadas são impressas.
2. Unproject2.c apresenta **gluUnProject()** lendo a posição do mouse e traduzindo-a em um cursor 3D representado por um cubo. Uma grade representa o plano xz.