Determinação de Superfícies Visíveis para jogos na Plataforma Playstation Usando Árvores BSP

Acadêmico: Marcelo Odebrecht

Orientador: Paulo César Rodacki Gomes

Roteiro

- Motivação
- Objetivos
- Jogos interativos
- Árvores *BSP*
- Plataforma *Playstation*
- Protótipo
- Conclusões e Extensões

Motivação

- Jogos por computador
  - qualidade gráfica;
  - bom desempenho.
- Desenvolvimento para a plataforma *Playstation*

Objetivos

- Desenvolver na plataforma *IBM PC* um protótipo para a plataforma *Playstation*
- Árvore *BSP* para determinação de superfícies visíveis
- Cenários 2D
- Renderização 2D
- Execução do protótipo no *Playstation*

Jogos interativos

- Interface gráfica
  - Animação
    - *Sprites*
    - Quadro-a-quadro
    - Renderização
- Jogos por computador
  - Enredo
  - Interface interativa
  - Motor

Árvores *BSP*

Binary Space Partition Tree (*BSP tree*)

Árvore binária de divisão espacial (Árvore *BSP*)

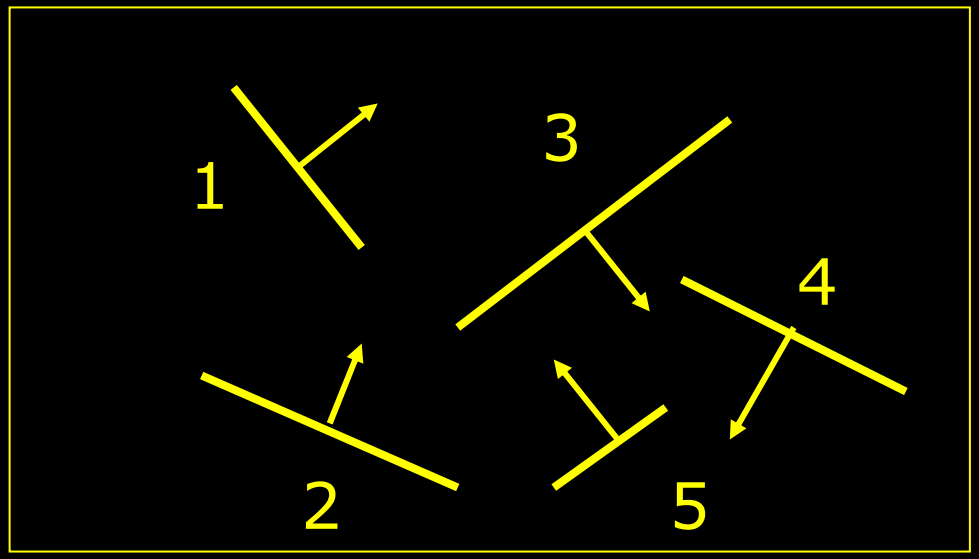
Utilização:

- **determinação de superfícies visíveis**;
- detecção de colisões;
- planejamento de movimento robótico;
- modelagem sólida;
- *ray tracing*.

Árvores *BSP*

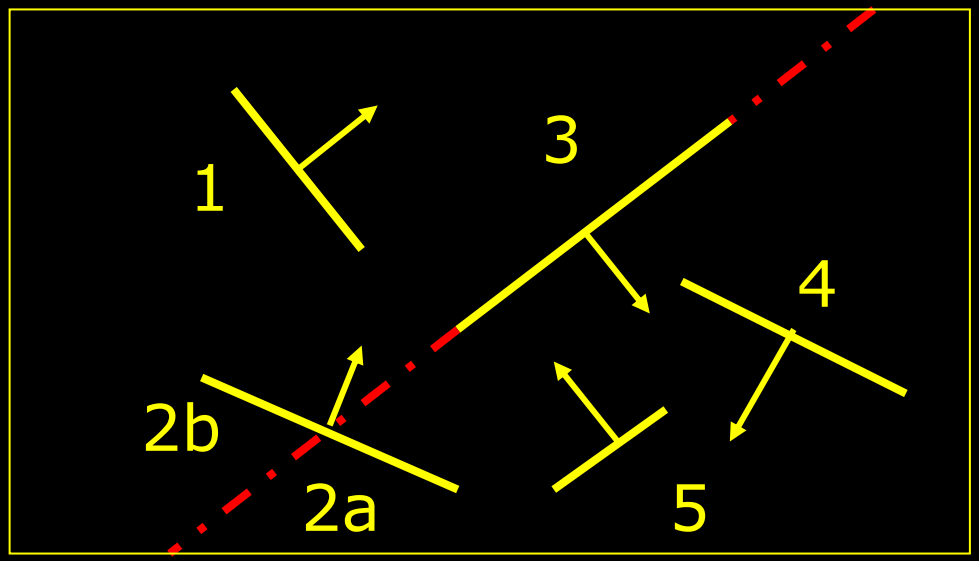
Construção

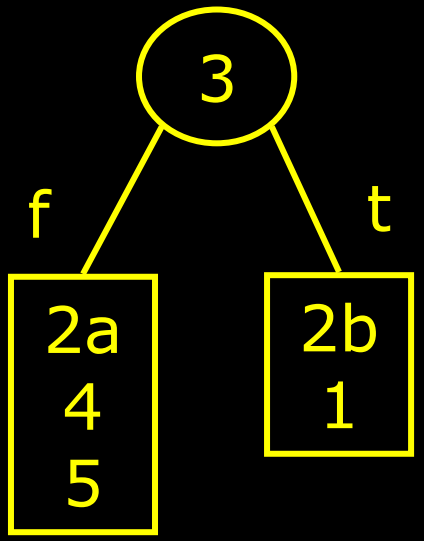
- Selecionar face
- Testar posição das faces restantes



Árvores *BSP*

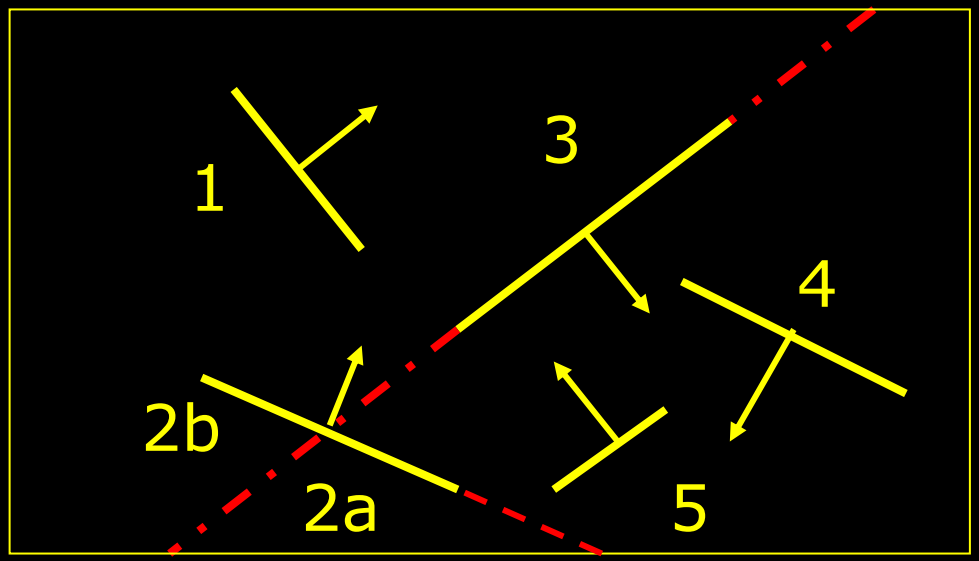
Construção

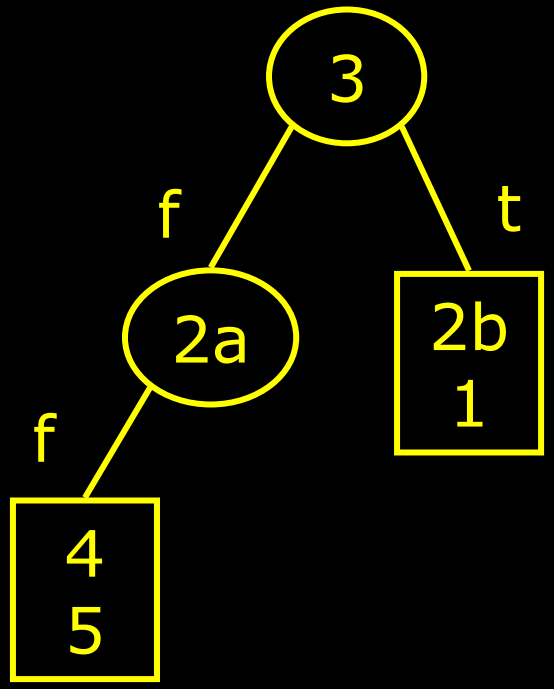




Árvores *BSP*

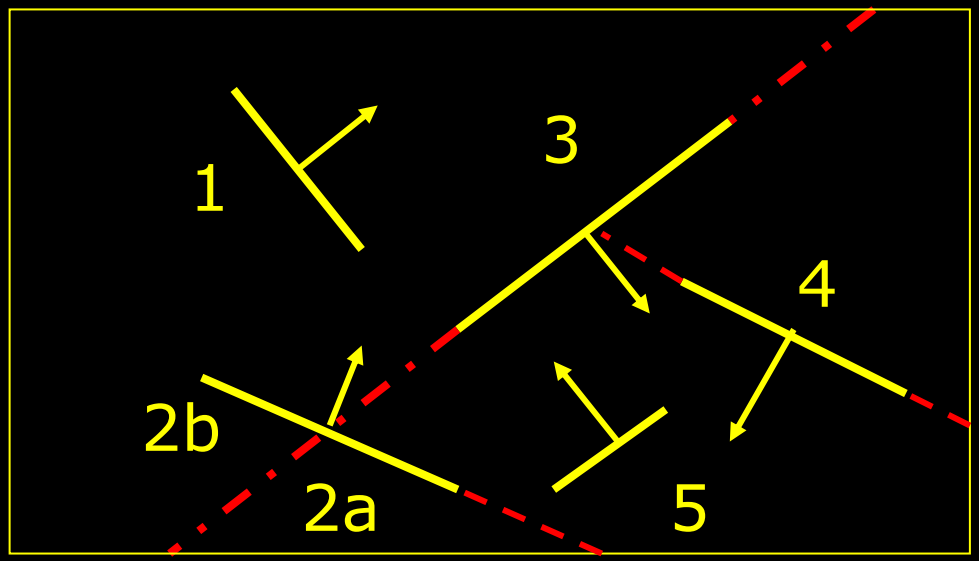
Construção

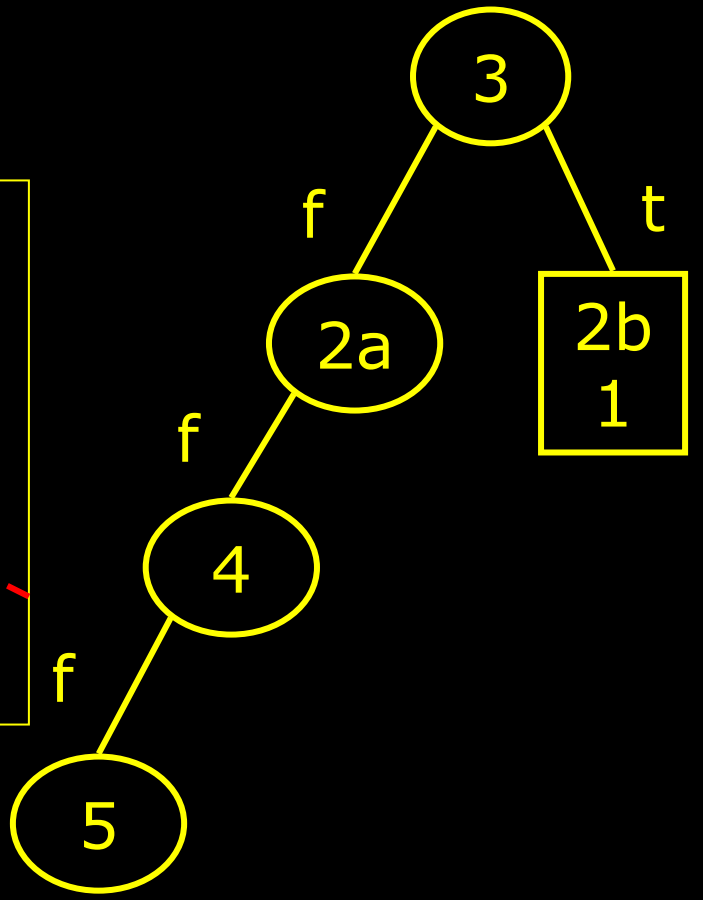




Árvores *BSP*

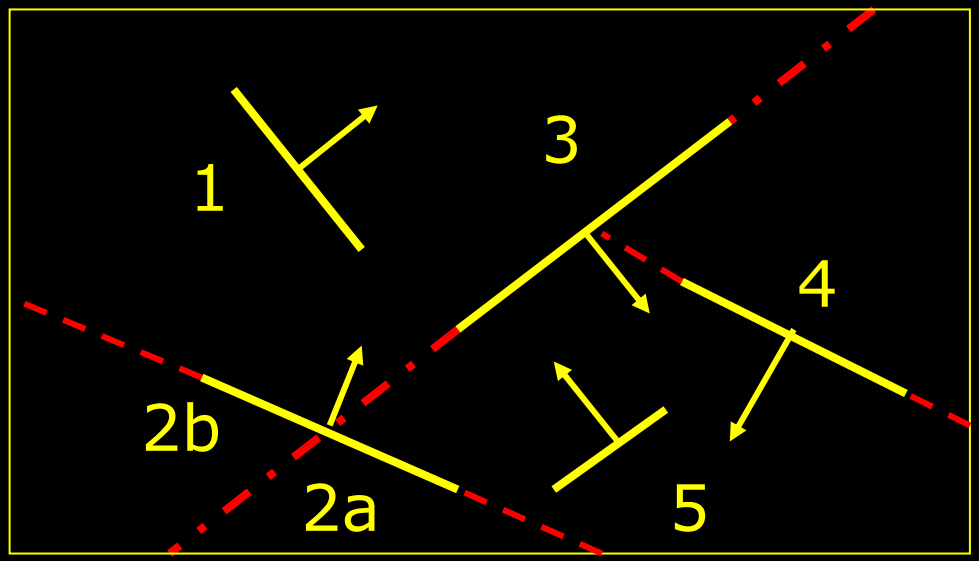
Construção

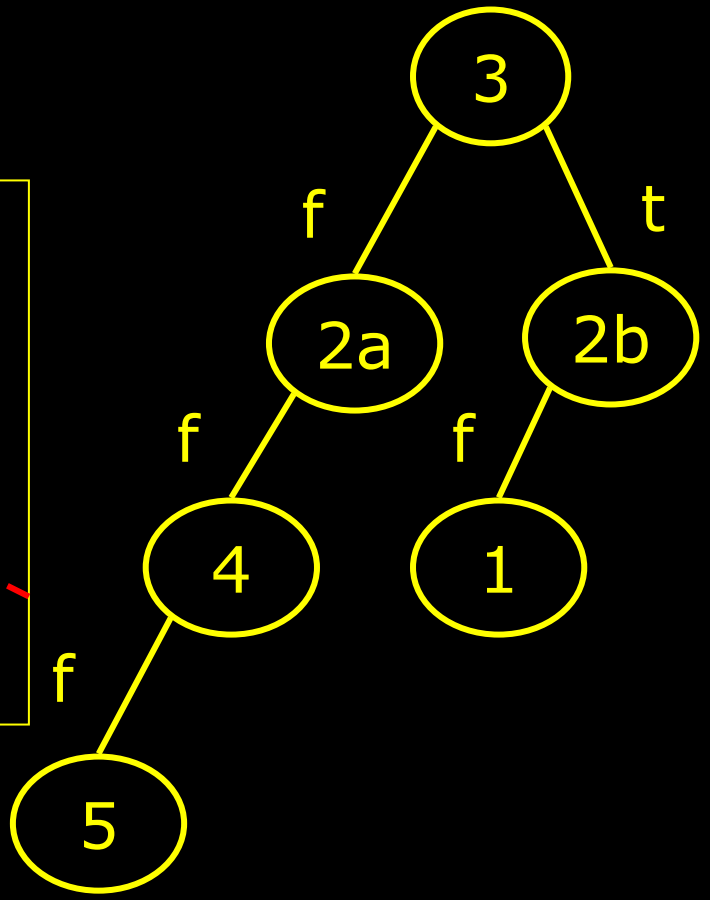




Árvores *BSP*

Construção





Árvores *BSP*

Pseudo código

```
ArvoreBSP* ConstroiArvoreBSP(ListaFaces ListaFaces)
{
   NovoNodo = AlocarNodoDaArvore();
   FaceEscolhida = SelecionarFace(ListaFaces);
   enquanto (ListaFaces não for nula) faça
   {
 FaceAtual = SelecionarFace(ListaFaces);
 Resultado = Classificar(FaceEscolhida, FaceAtual);
 caso (Resultado)
 {
    FRENTE: Insere(FaceAtual, ListaFrente);
    ATRAS: Insere(FaceAtual, ListaAtras);
    INTERCEPTA:
        {
           DividirFace(FaceAtual, PontoDeInterseção);
           InserirPonto(PontoDeInterseção);
           InserirFace(PontoInicial, PontoDeInterseção,ListaFaces);
           InserirFace(PontoDeInterseção, PontoFinal,ListaFaces);
        }
        COLINEAR: Insere(FaceAtual, NovoNodo->ListaColineares);
     }
   }
   se (ListaFrente não estiver vazia) então
      NovoNodo->Frente = ConstroiArvoreBSP(ListaFrente);
   se (ListaAtras não estiver vazia) então
      NovoNodo->Atras = ConstroiArvoreBsp(ListaAtras);
   retorna NovoNodo;
}
```

Árvores *BSP*

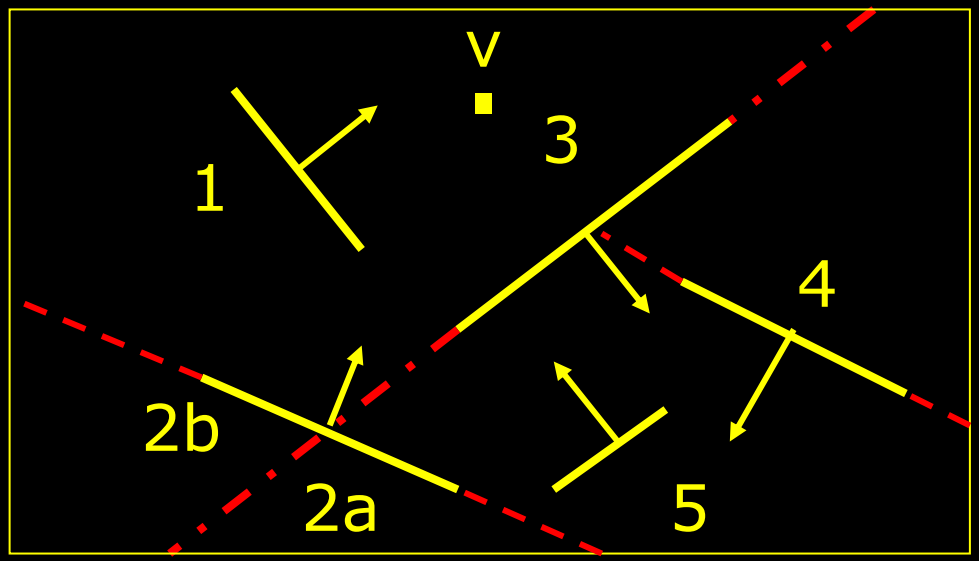
Quando parar ?

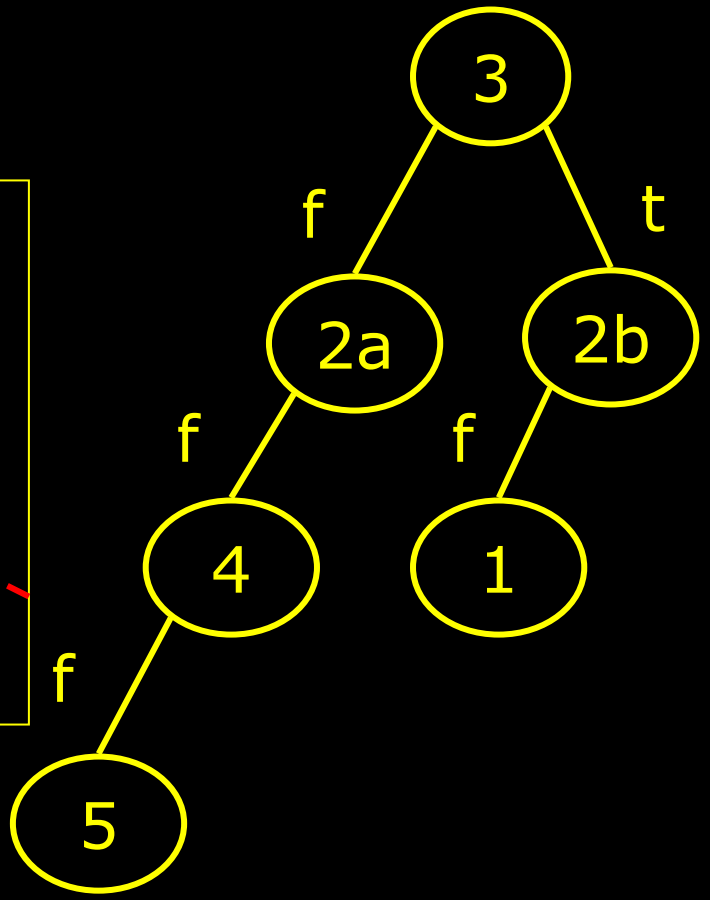
- número máximo de nodos;
- profundidade atingida;
- todos os objetos no espaço.

Árvores *BSP*

Percorrimento

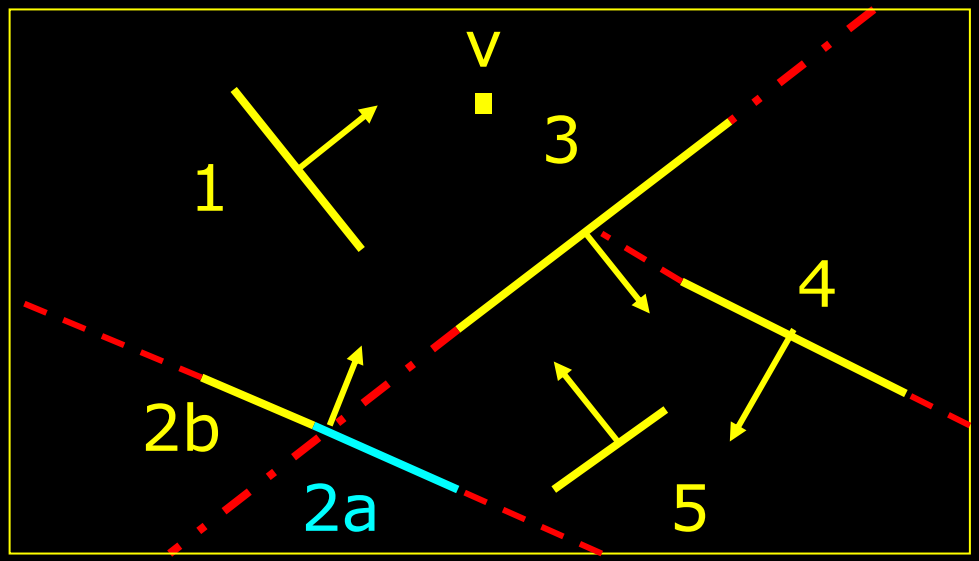
Ponto de vista “v”

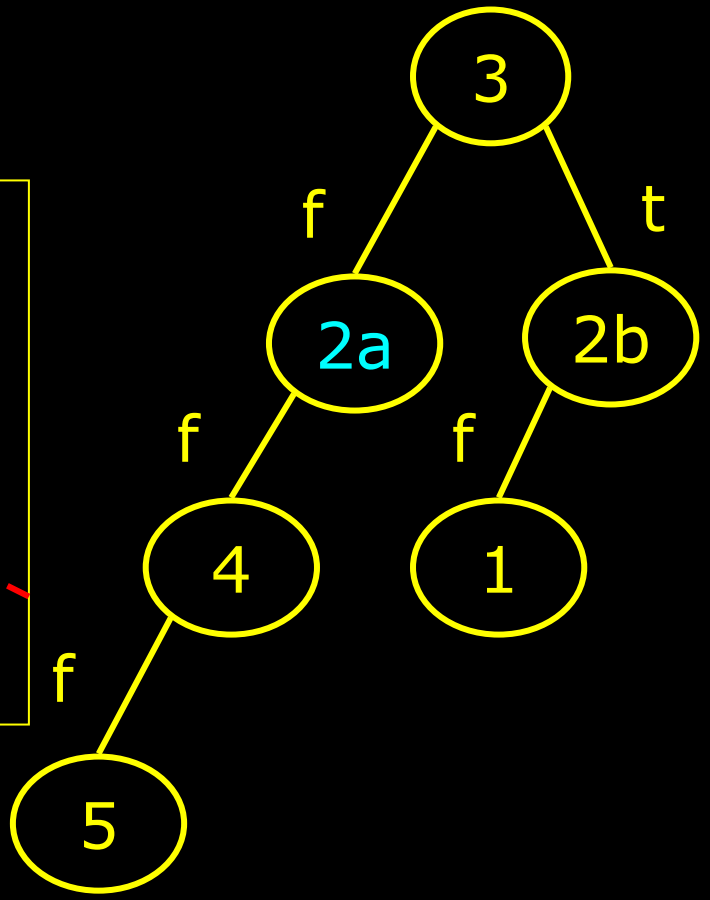




Árvores *BSP*

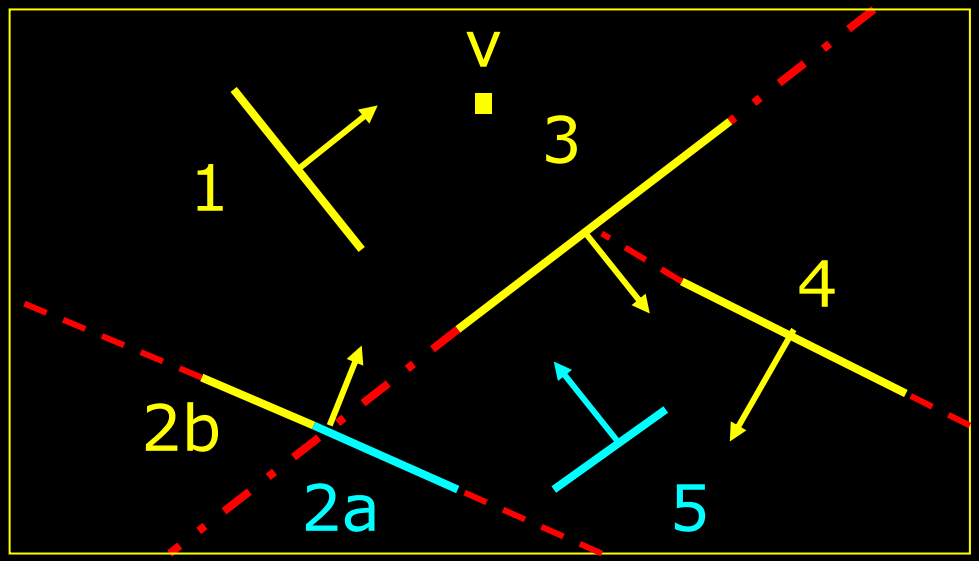
Percorrimento

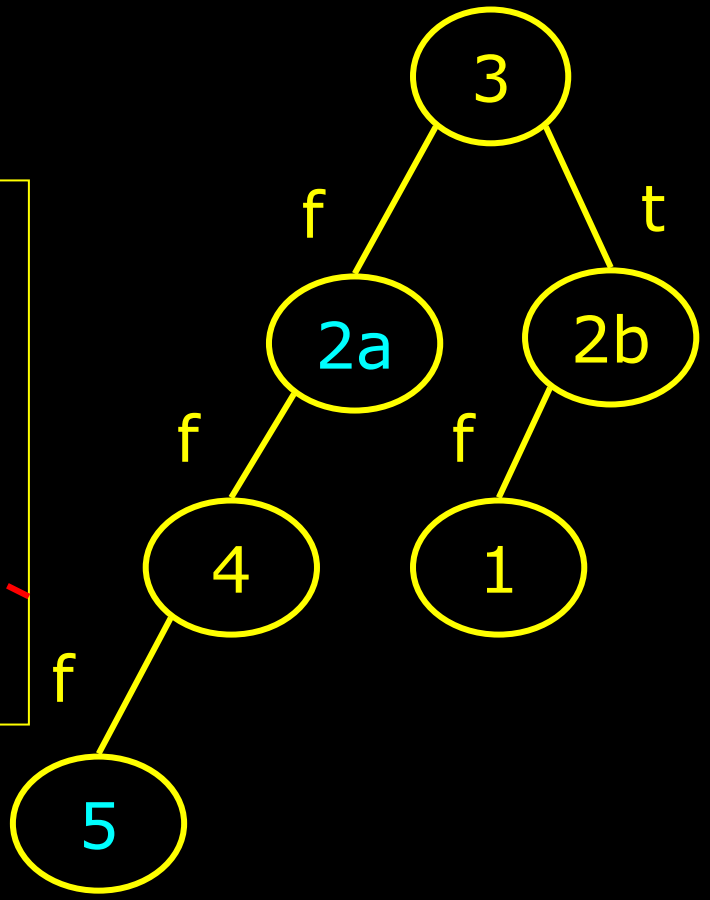




Árvores *BSP*

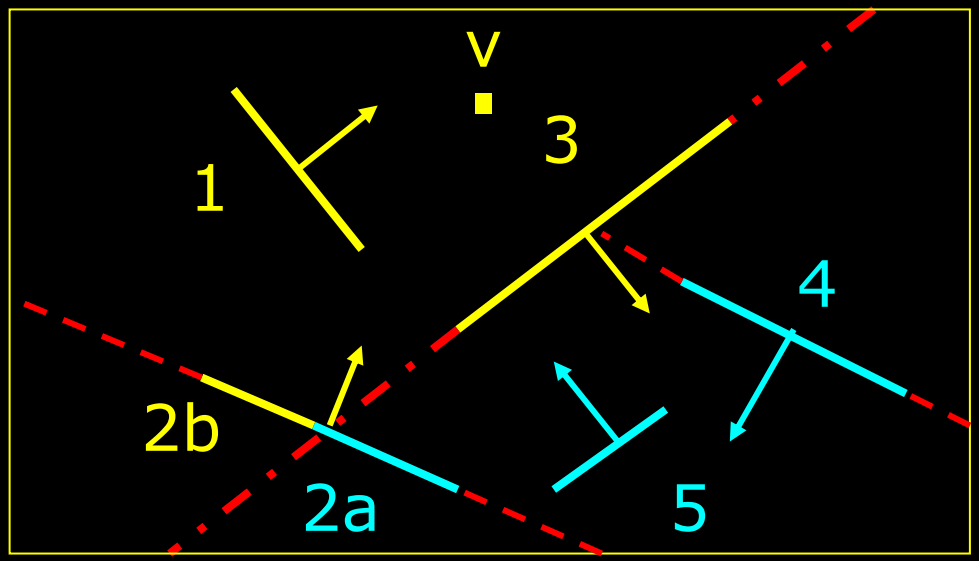
Percorrimento

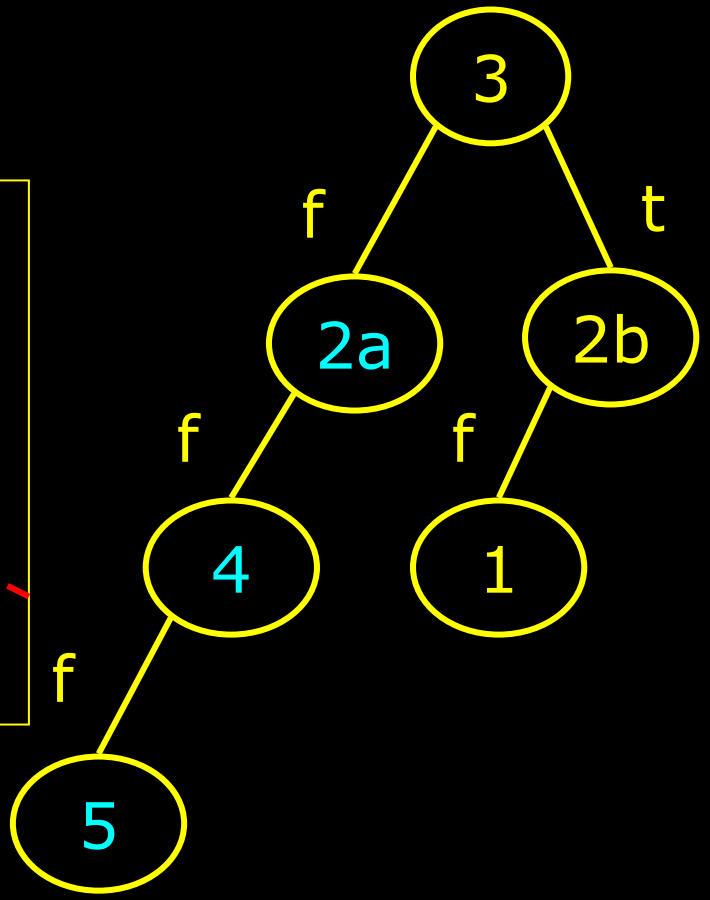




Árvores *BSP*

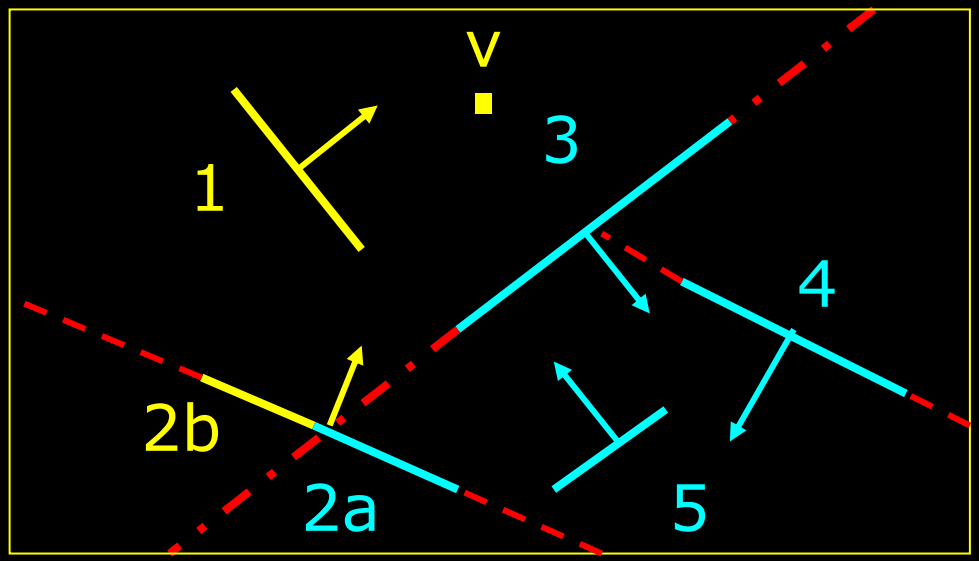
Percorrimento

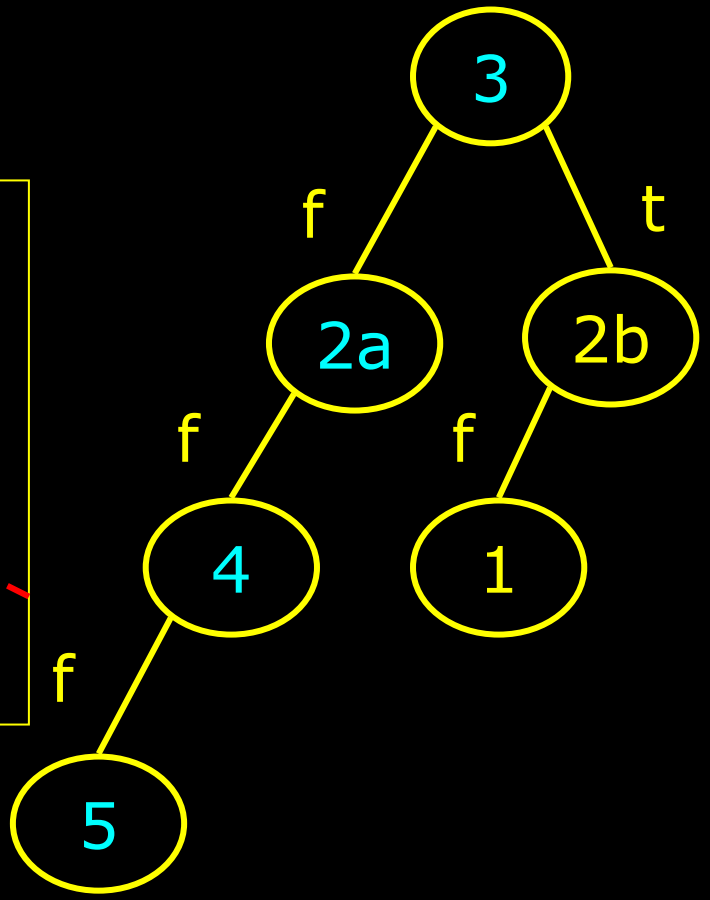




Árvores *BSP*

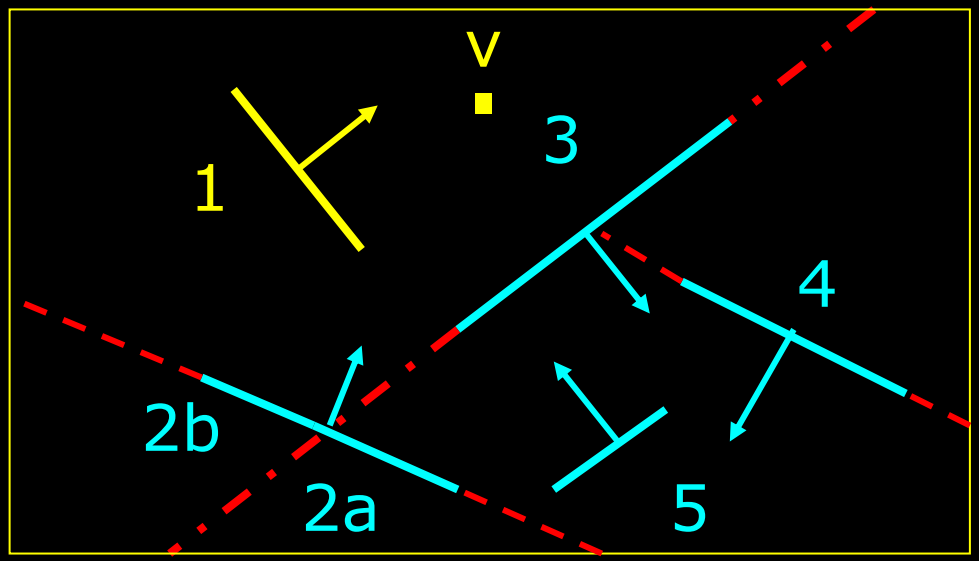
Percorrimento

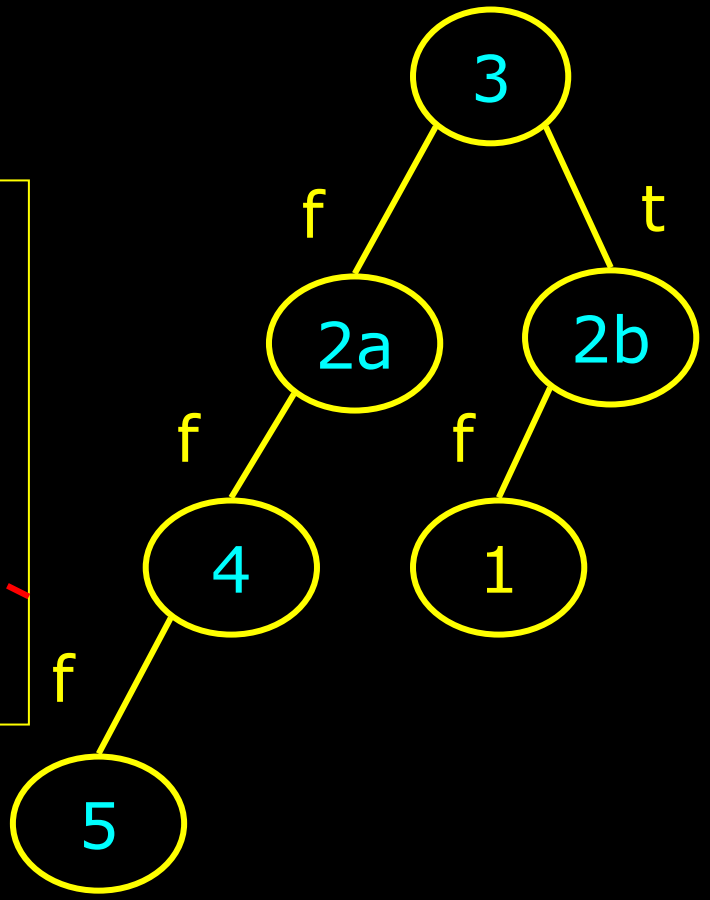




Árvores *BSP*

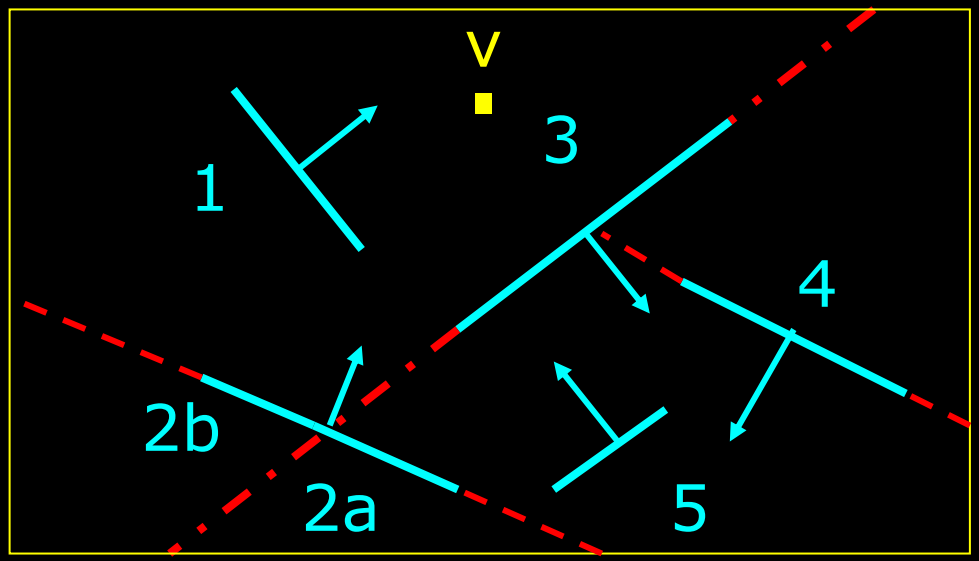
Percorrimento

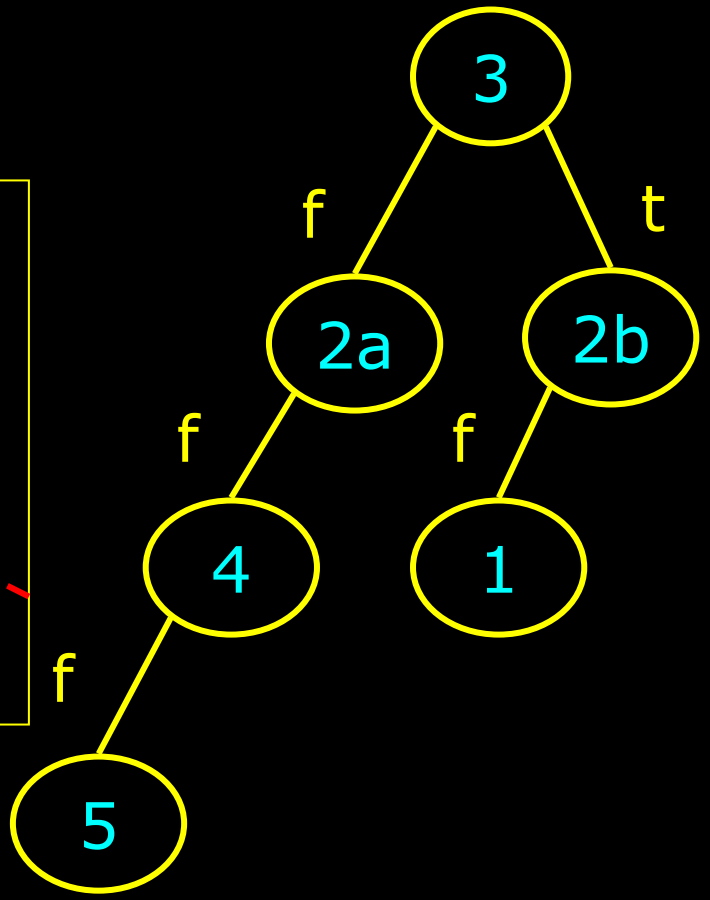




Árvores *BSP*

Percorrimento





Árvores *BSP*

Pseudo código

```
void PercorreArvoreBSP(ArvoreBSP* Arvore, real x, real y)
{
Resultado = Classificar(x, y,     SelecionaFace(Arvore->ListaFaces));
se (Resultado igual a FRENTE) então
{
   //o ponto de vista está a frente da face
   PercorreArvoreBSP(Arvore->tras, x, y);
   DesenhaArvoreBSP(Arvore->ListaFaces);
   PercorreArvoreBSP(Arvore-frente, x, y);
    }
senão
{
   //o ponto de vista está atrás ou coincide com a face
   PercorreArvoreBSP(Arvore-frente, x, y);
   DesenhaArvoreBSP(Arvore->ListaFaces);
   PercorreArvoreBSP(Arvore->tras, x, y);
    }
}
```

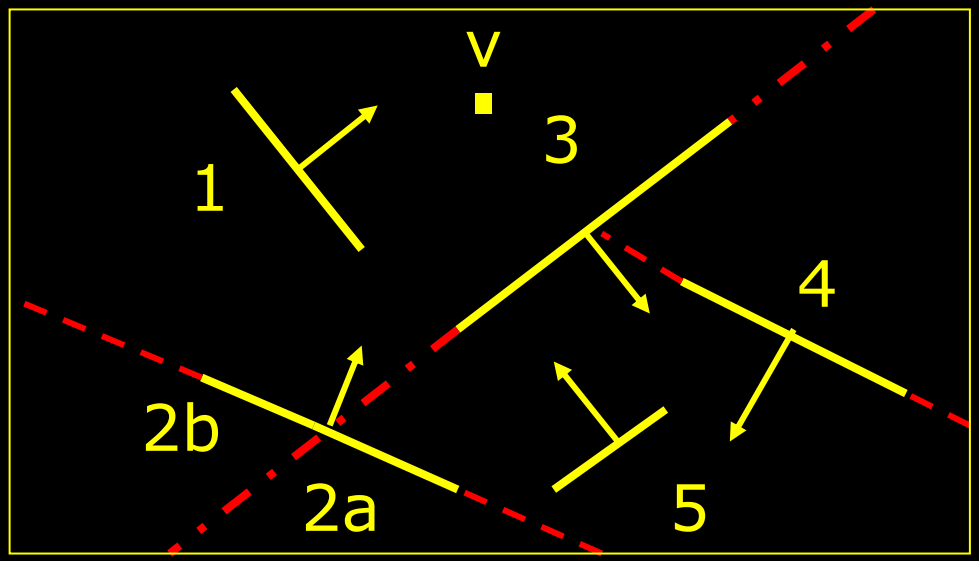
Árvores *BSP*

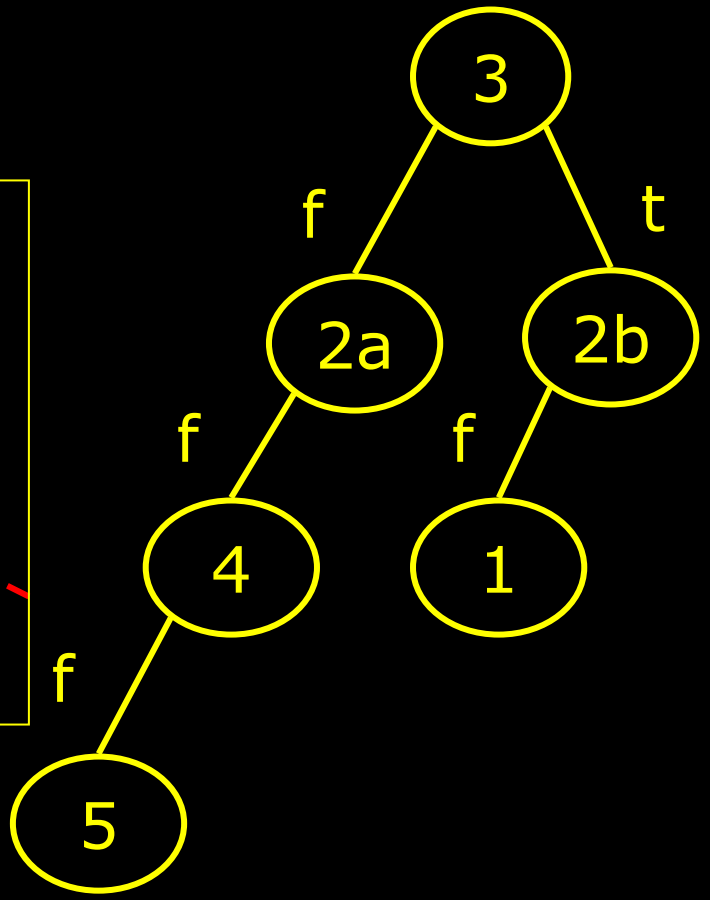
- Otimizações
  - *back face culling*;
  - *view-frustum culling*.

Árvores *BSP*

Otimizações

Back face culling

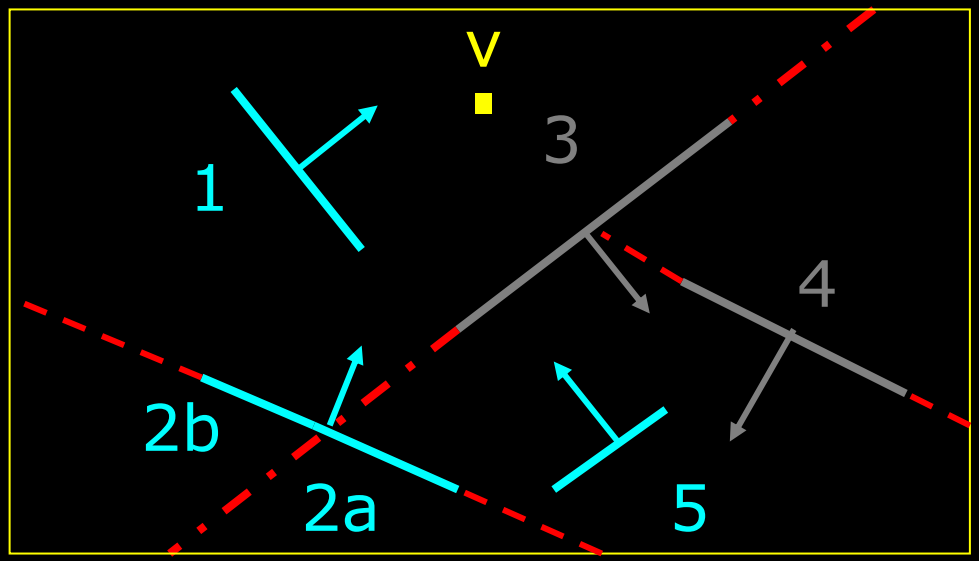


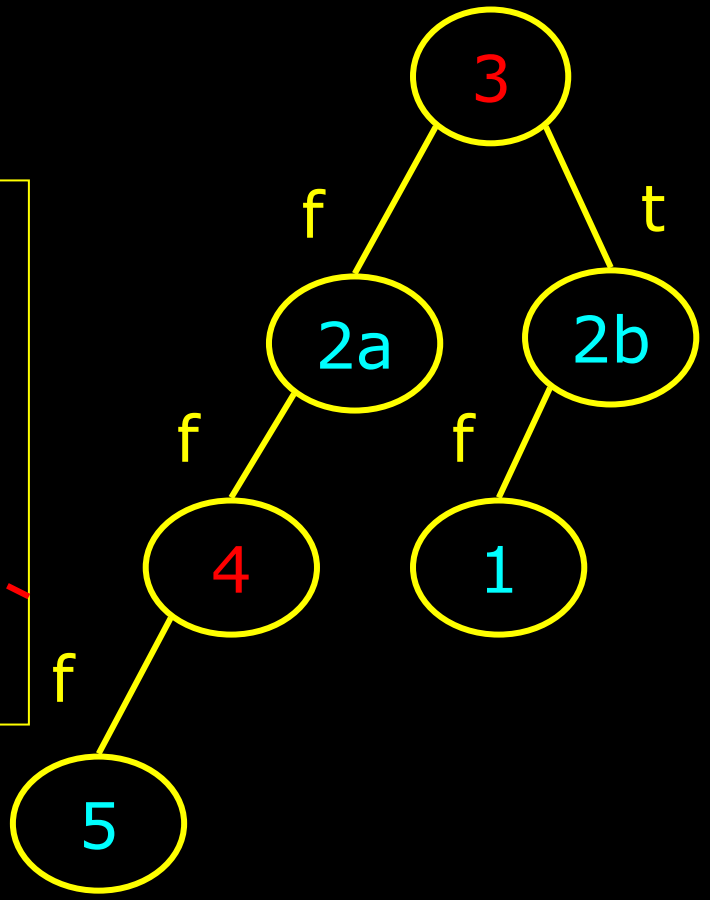


Árvores *BSP*

Otimizações

Back face culling

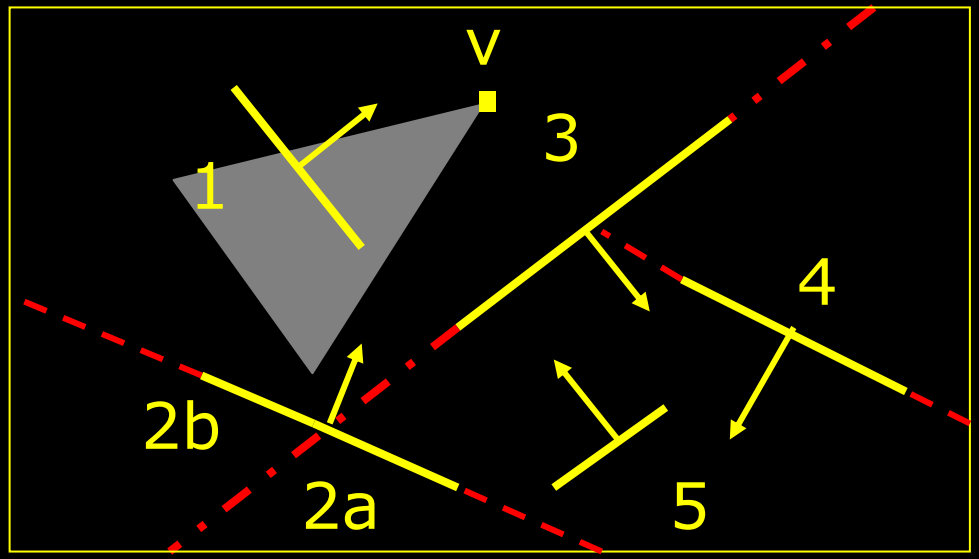


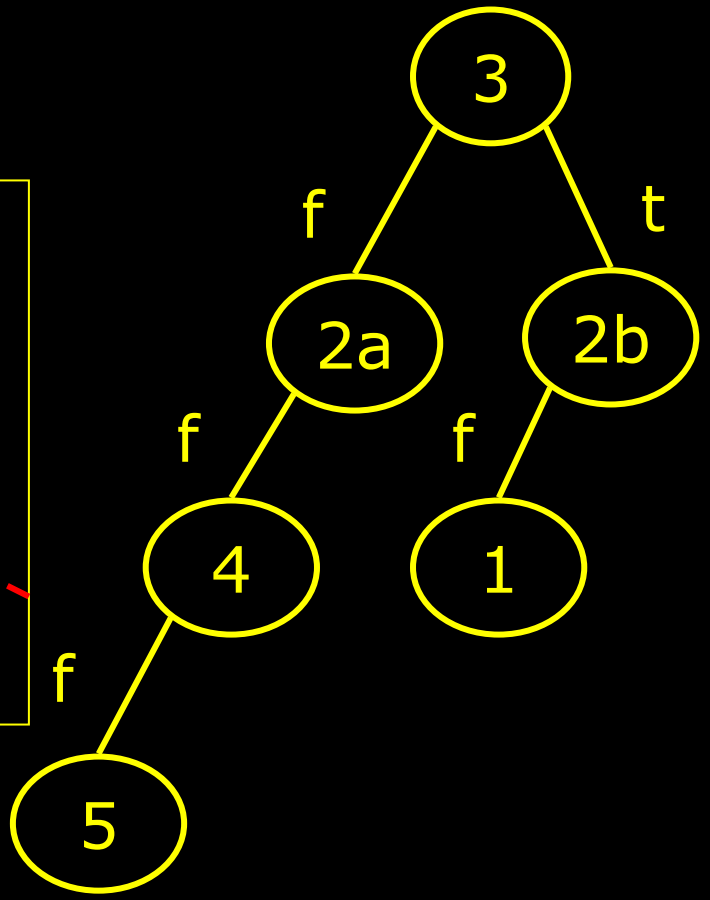


Árvores *BSP*

Otimizações

View-frustum culling

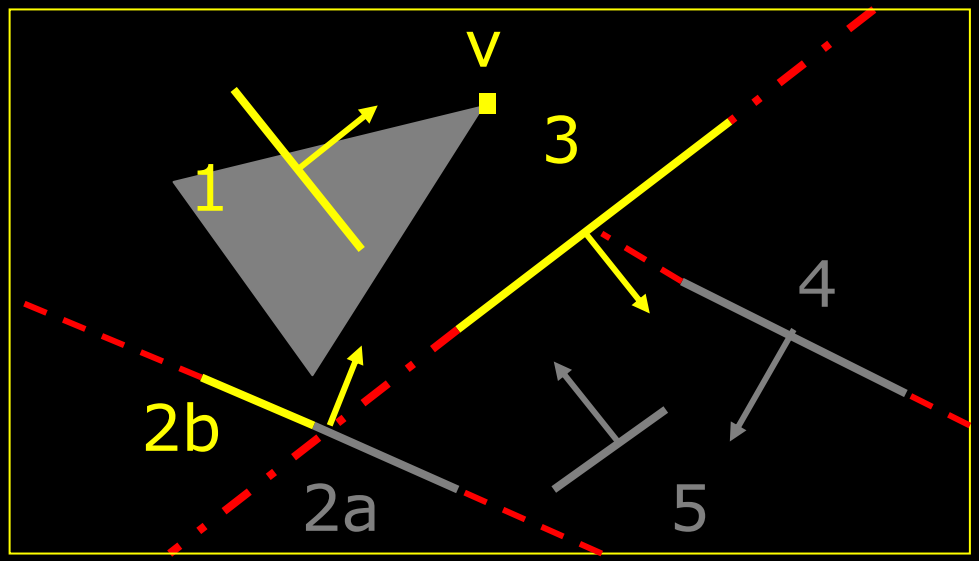


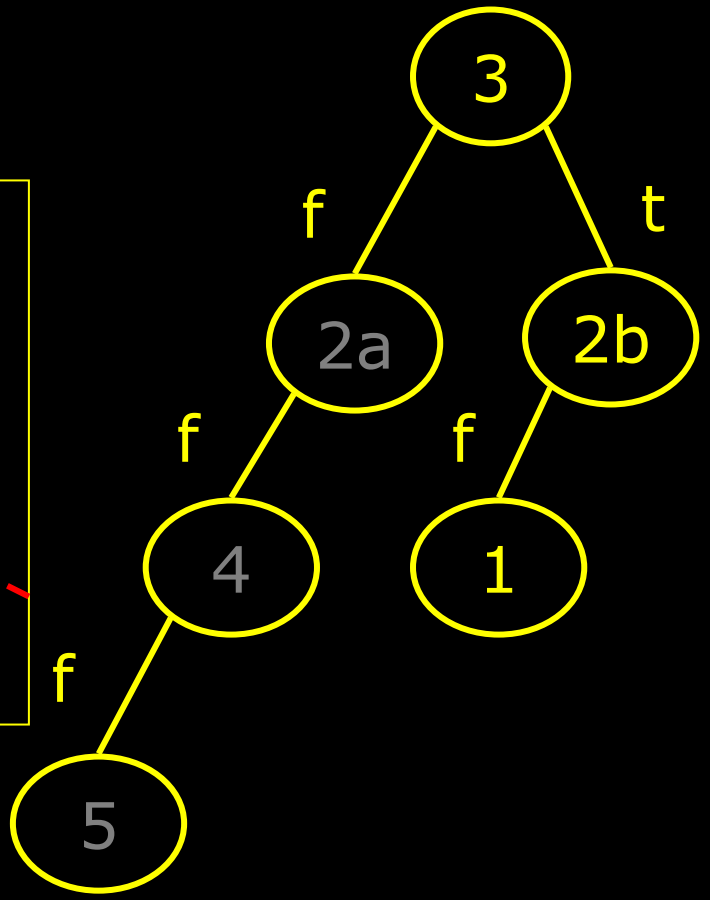


Árvores *BSP*

Otimizações

View-frustum culling

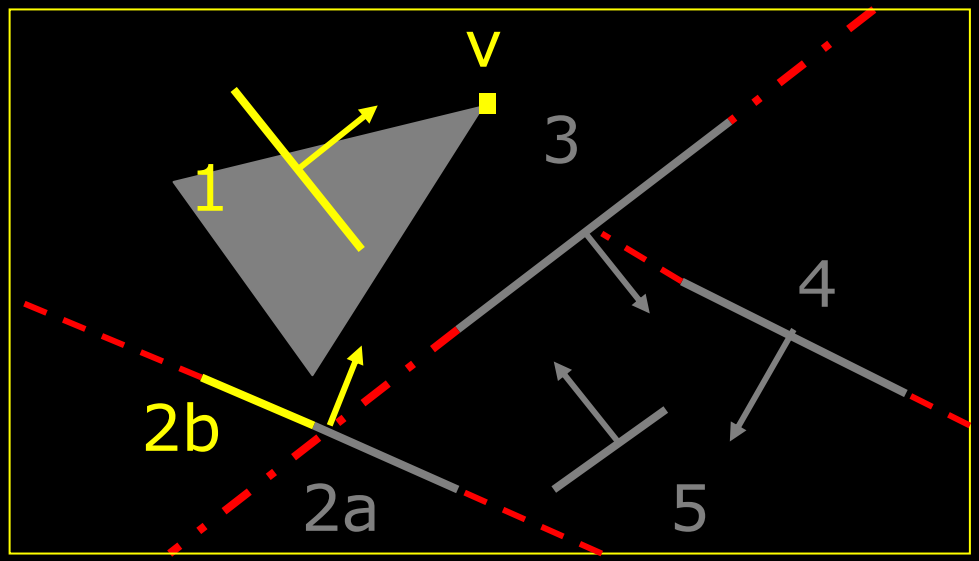


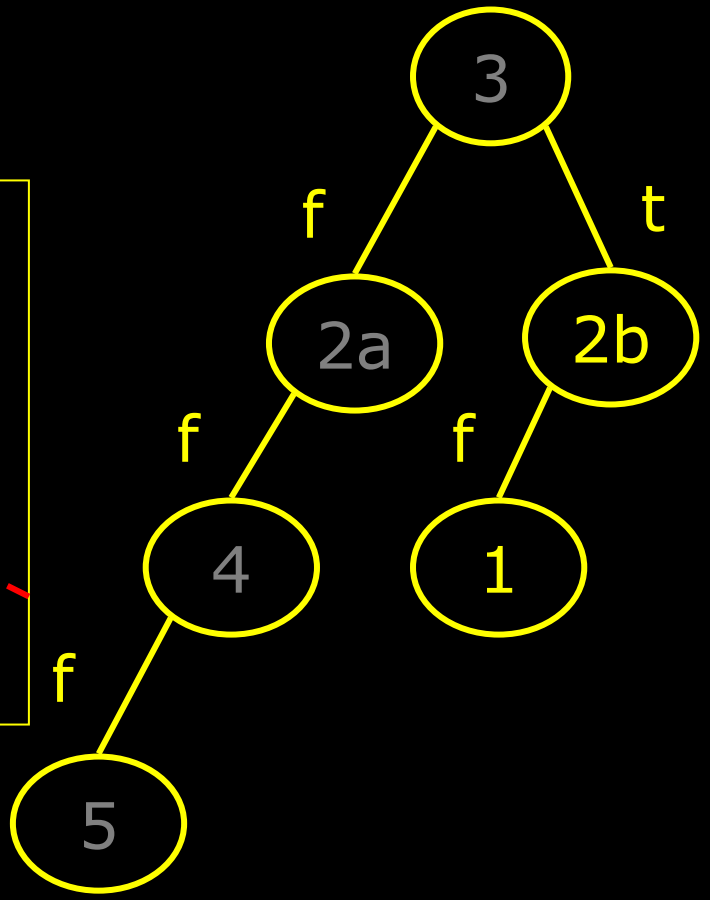


Árvores *BSP*

Otimizações

View-frustum culling

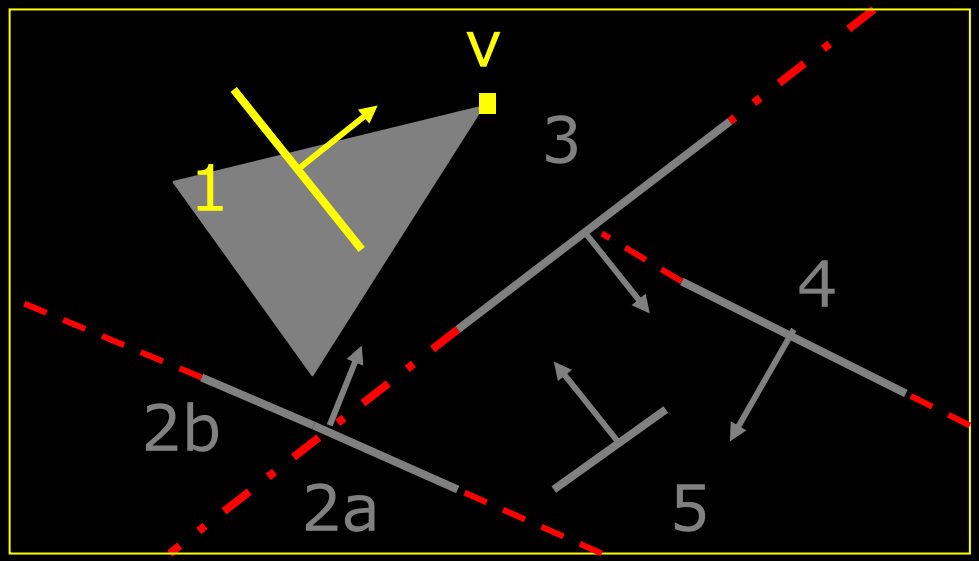


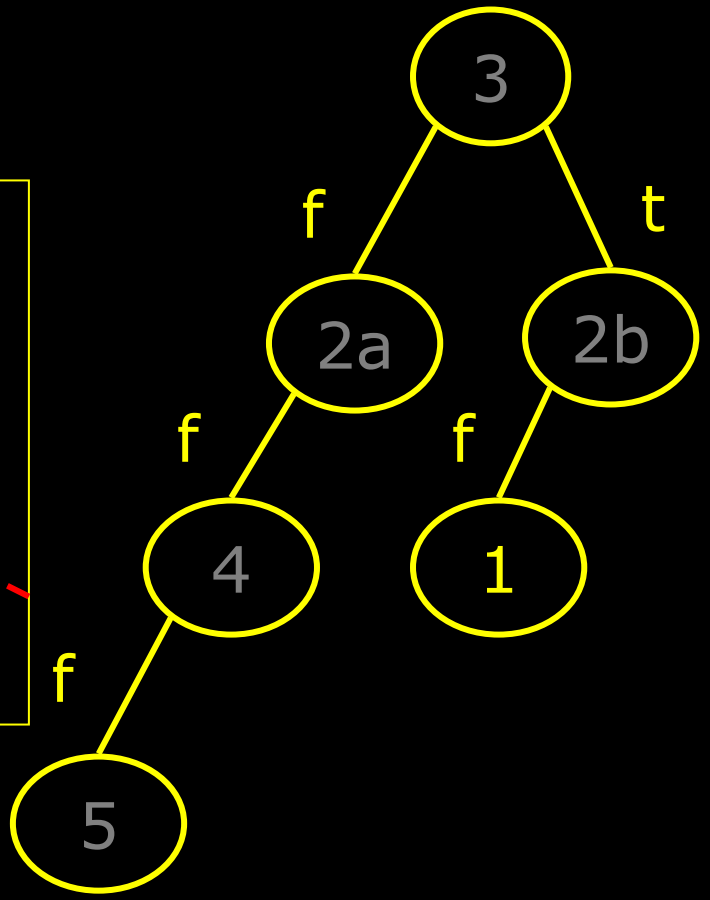


Árvores *BSP*

Otimizações

View-frustum culling

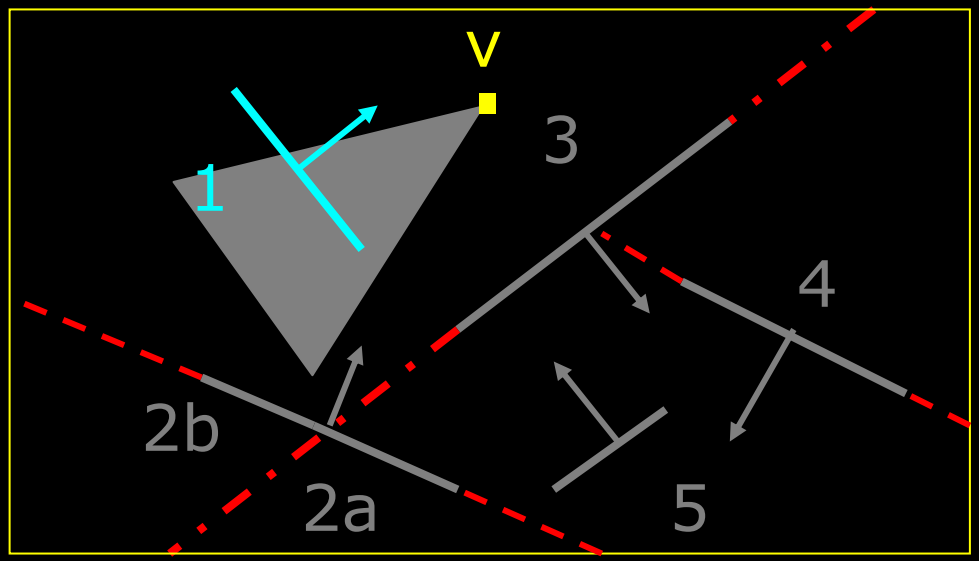


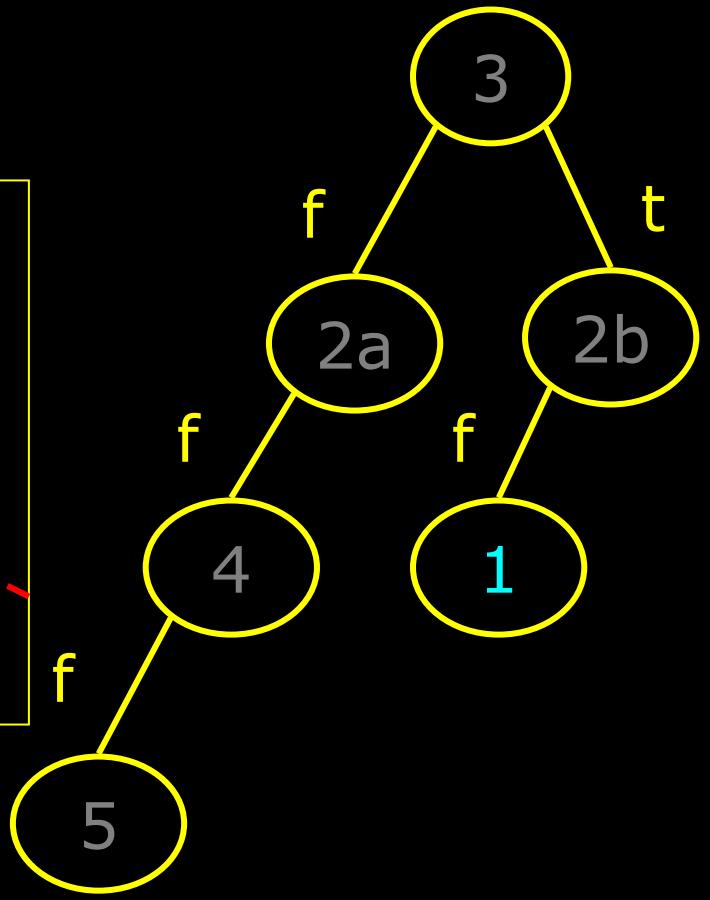


Árvores *BSP*

Otimizações

View-frustum culling

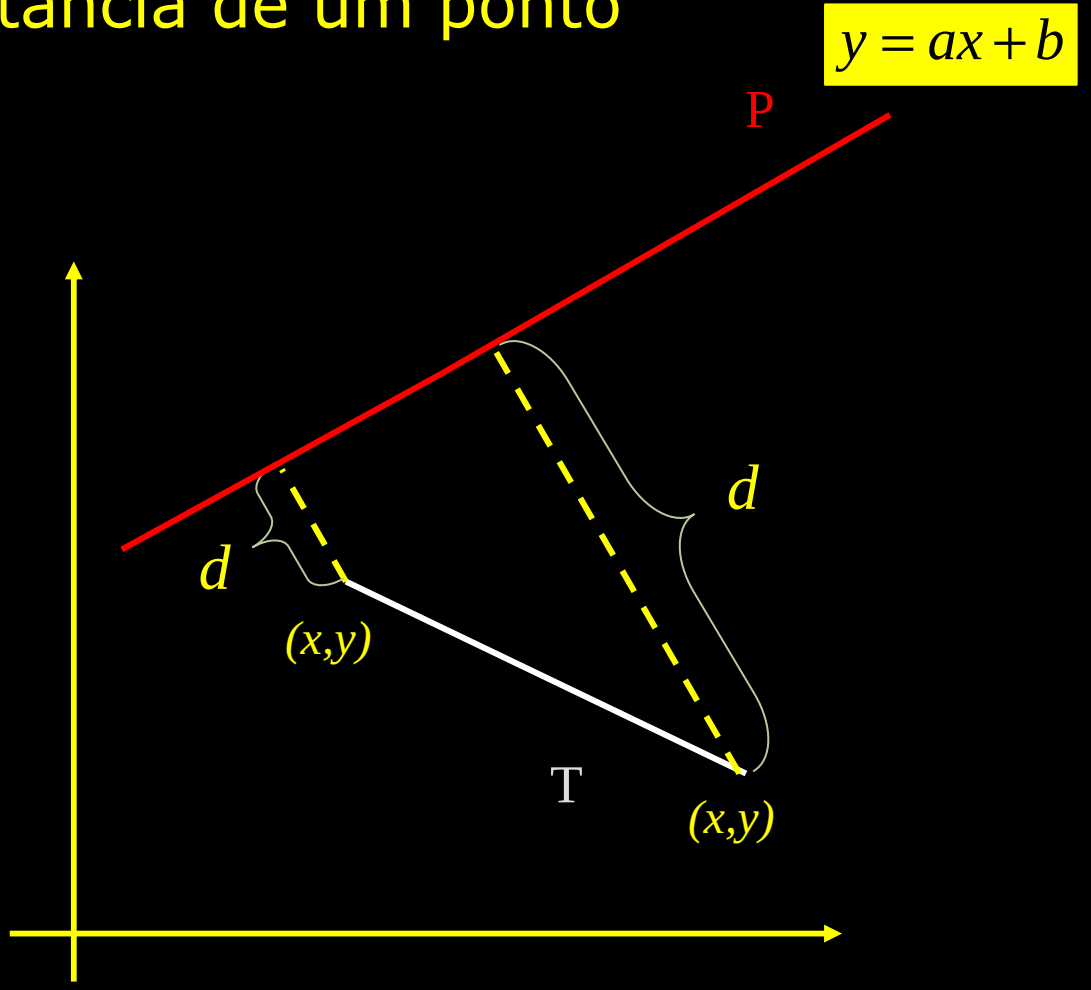




Árvores *BSP*

Cálculo da distância de um ponto à reta

$$d = \frac{ax - y + b}{\sqrt{a^2 + 1}}$$



Plataforma *Playstation*

Playstation - 1994

PSOne - 2000

Plataforma *Playstation*

Ambientes de desenvolvimento

- *Net Yaroze*
  - Características
  - Limitações
- PSy-Q
  - Ferramentas
  - Bibliotecas

Plataforma *Playstation*

Ambientes de desenvolvimento

- Acessórios
  - *PC Comms link*
  - *Action Replay / Game Shark*
  - *Xplorer / Xploder*
  - *Free Wing*
  - *Skywalker*

Plataforma *Playstation*

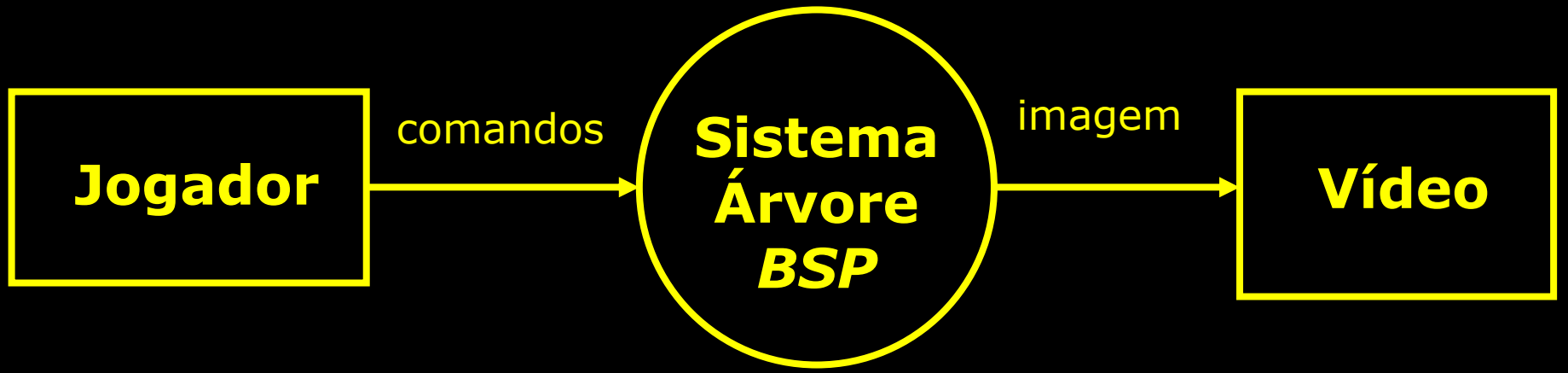
PSEmu Pro

Protótipo

- Requisitos
  - Demonstrar árvores *BSP*
  - Determinação de superfícies visíveis
  - Cenário 2D
  - Interatividade

Protótipo

- Diagrama de Fluxo de Dados (DFD)



Protótipo

Fluxograma

Protótipo

- Implementação

Protótipo

- Implementação

Conclusões

Estudo de uma técnica de computação gráfica

Protótipo de jogo para a plataforma *Playstation*

Problemas superados

Extensões

- Cenários 3D
- Renderização 3D
- Detecção de colisões em 3D e 2D
- Animação
- *Ray Tracing*