Visão Isométrica & Tilemap Isométrico

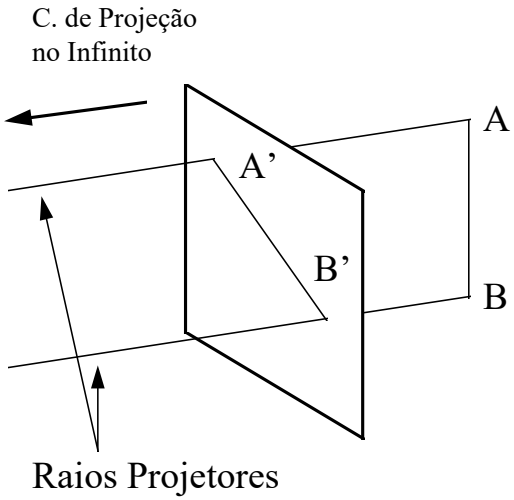
por Rossana B Queiroz

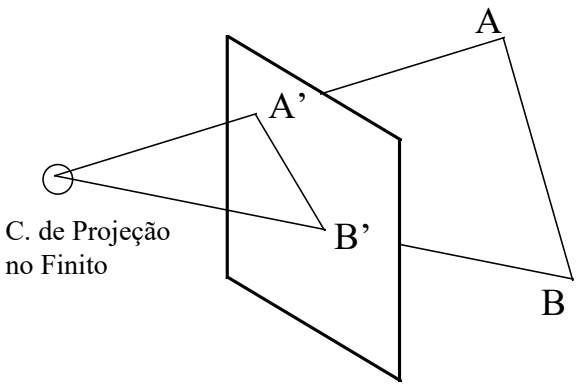
Projeções Paralelas e Perspectivas

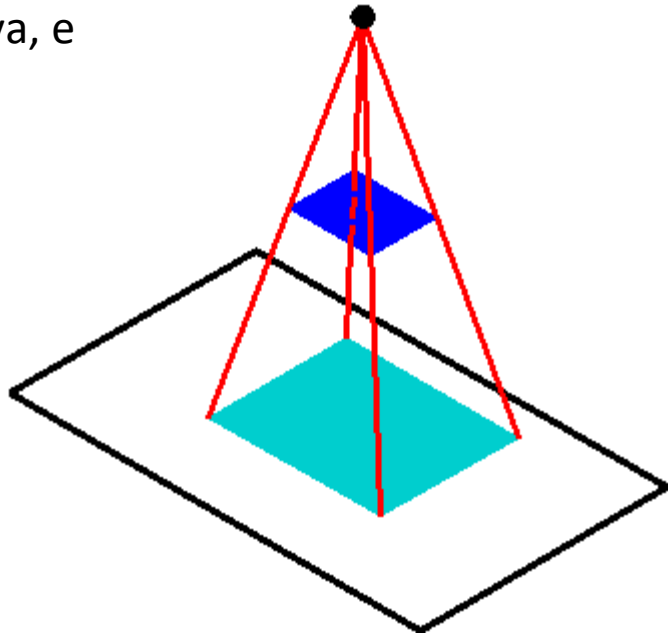
(recapitulando)

As projeções planares paralelas e perspectivas diferem com relação a distância do plano de projeção ao centro de projeção:

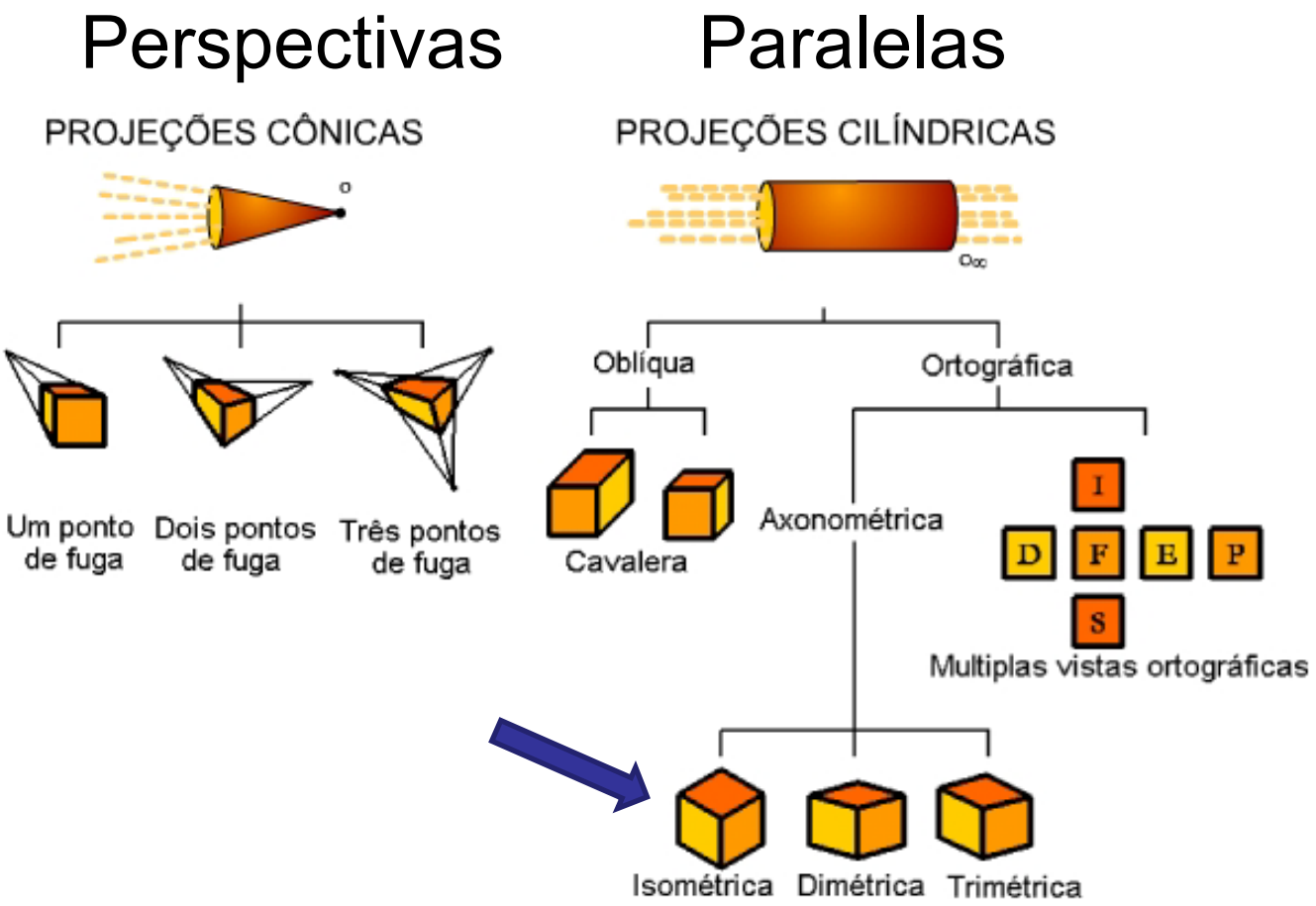
- se a distância é finita, a projeção é perspectiva, e
- se a distância é infinita, a projeção é paralela







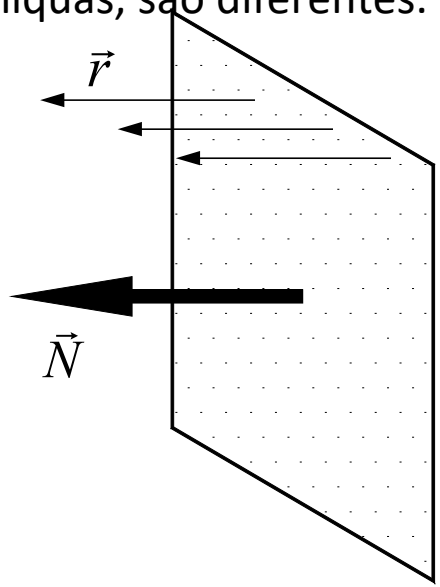
Projeções Paralelas e Perspectivas



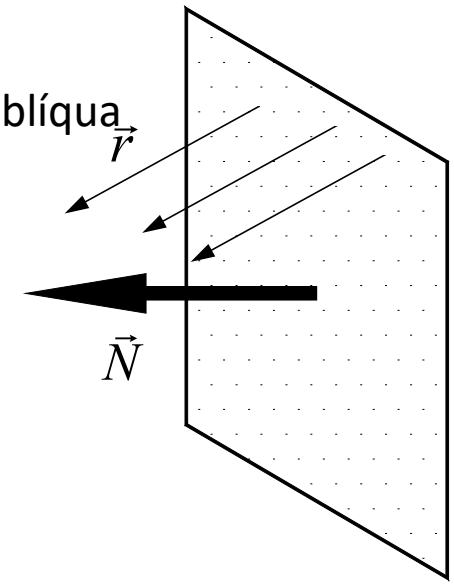
Projeções planares paralelas

- Projeções planares paralelas são subclassificadas em ortográficas e oblíquas dependendo da relação entre a direção dos raios projetores e a normal ao plano de projeção.
  - projeções ortográficas, as direções são as mesmas (raios perpendiculares ao plano de projeção).
  - projeções oblíquas, são diferentes.

(a) Ortográfica



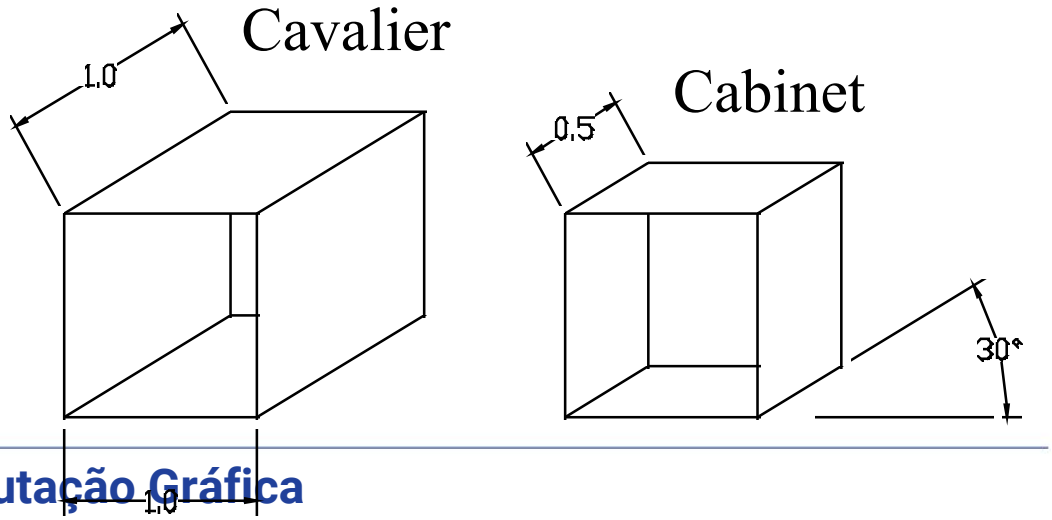
(b) oblíqua



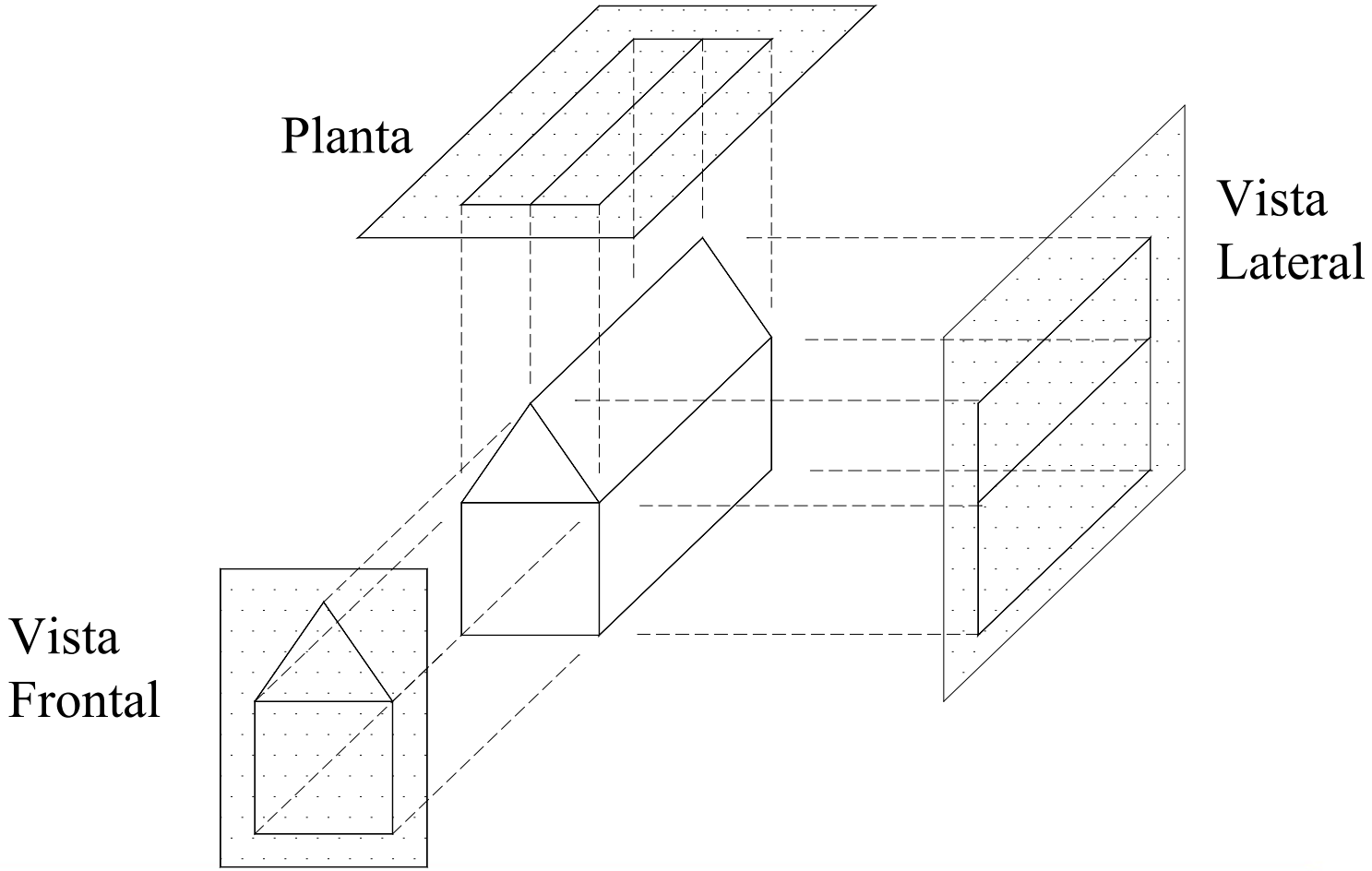
Projeções Oblíquas

- Cavaleira vs. Cabinet
  - Alguns tipos

Na *cabinet* há um encolhimento na dimensão do versor perpendicular ao plano de projeção para corrigir a ilusão de que o objeto exibido é maior na direção deste versor.



Projeções ortográficas: vistas lateral, frontal e planta



Projeções Axonométricas

- Ocorrem quando o plano de projeção não é ortogonal a algum eixo principal do sistema.

Projeções Axonométricas

- Possuem as seguintes características:
  1. A projeção no espaço 2D não possui “ponto de fuga”
  2. Linhas paralelas no espaço 3D continuam paralelas no espaço 2D
  3. Objetos que estão distantes possuem o mesmo tamanho de objetos que estão próximos

Projeções ortográficas axonométricas

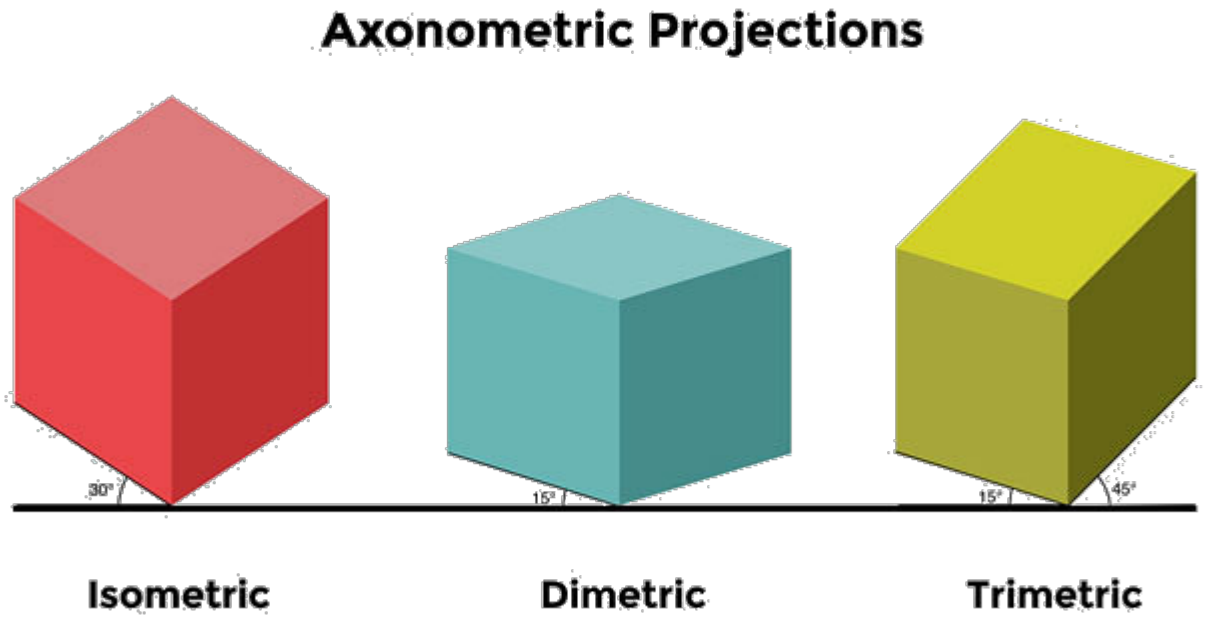
- Projeções axonométricas distorcem os objetos, alterando as relações de ângulos e dimensões de lados dos objetos, no entanto, mantém as relações de paralelismo entre eles.

- A alteração da dimensão dos lados é relacionada com a alteração da dimensão dos versores (vetores unitários) em cada um dos eixos **x**, **y** e **z**, quando projetados no plano.

- Projeções axonométricas se subdividem em:
  - **dimétricas**, quando dois versores variam a dimensão igualmente quando projetados no plano;
  - **isométricas**, quando três versores variam na mesma proporção; e
  - **trimétricas**, os três versores variam de forma diferenciada.

Projeções ortográficas axonométricas

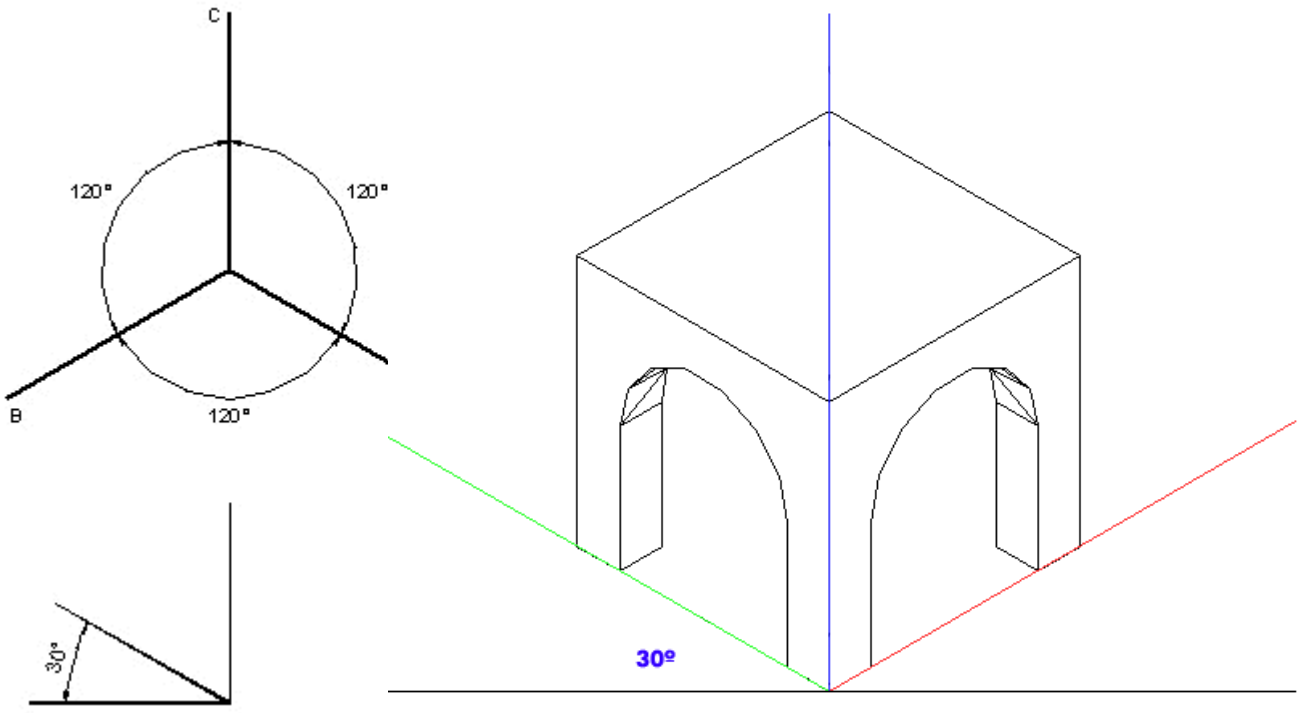
- Projeções Trimétricas, Dimétricas e Isométricas



Isometria

- Caso especial em que o plano de projeção forma o mesmo ângulo com os três eixos principais. As projeções dos três vetores unitários canônicos formam ângulos de 120º entre si.
- Isto permite que as medições feitas na projeção em cada eixo utilize a mesma escala

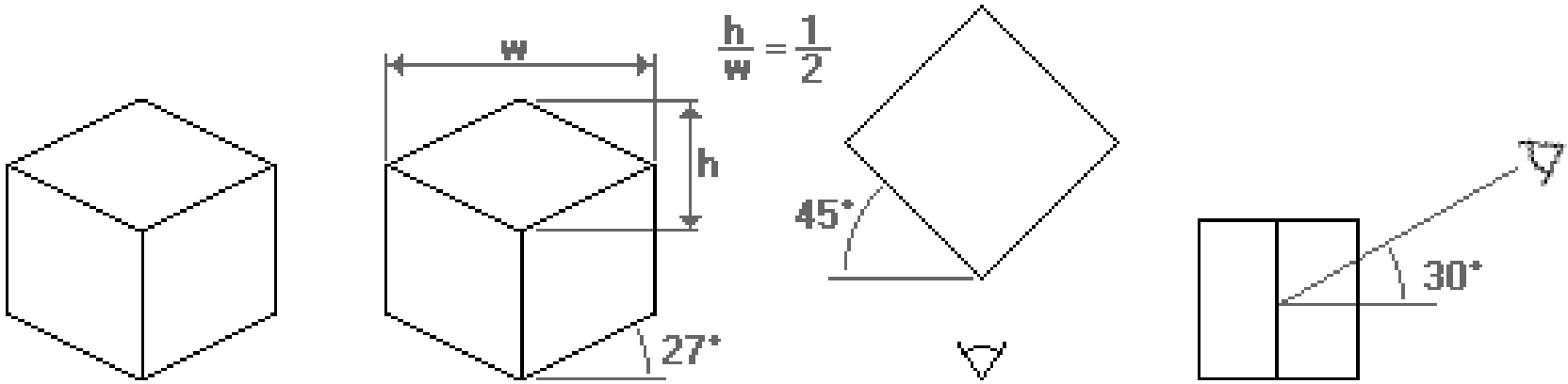
Isometria



Tipos de Isometria

- Existem várias projeções isométricas possíveis. Entretanto, os jogos de computadores isométricos são geralmente baseados em tiles
- É necessário então fazer com que os tiles casem para poder formar um mapa de tiles.
- Por isso, geralmente a projeção isométrica utilizada é a conhecida 1:2
  - altura e o comprimento do tile possuem uma razão de 1 para 2
- Os tamanhos de tiles mais usados nos jogos de computadores são os de 16 pixels por 32 e o 32 pixels por 64.

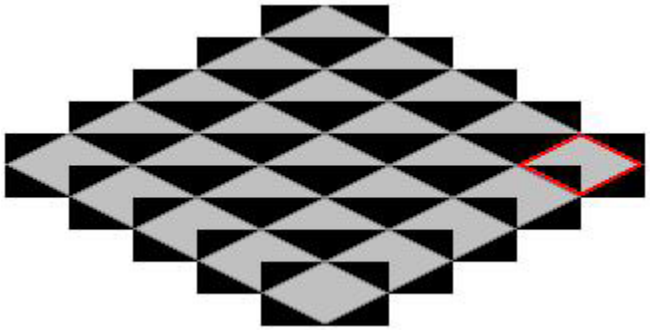
1:2

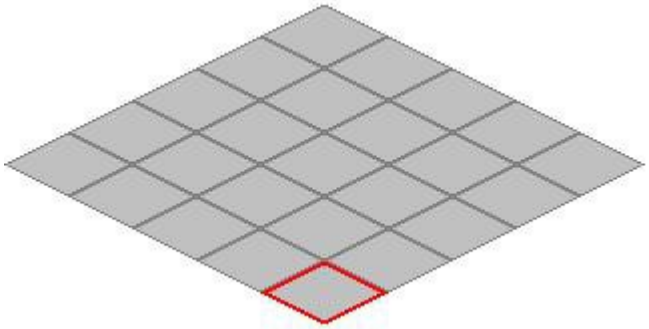


Visão em Perspectiva | Visão Superior | Visão Lateral

Mapas com Tiles Isométricos

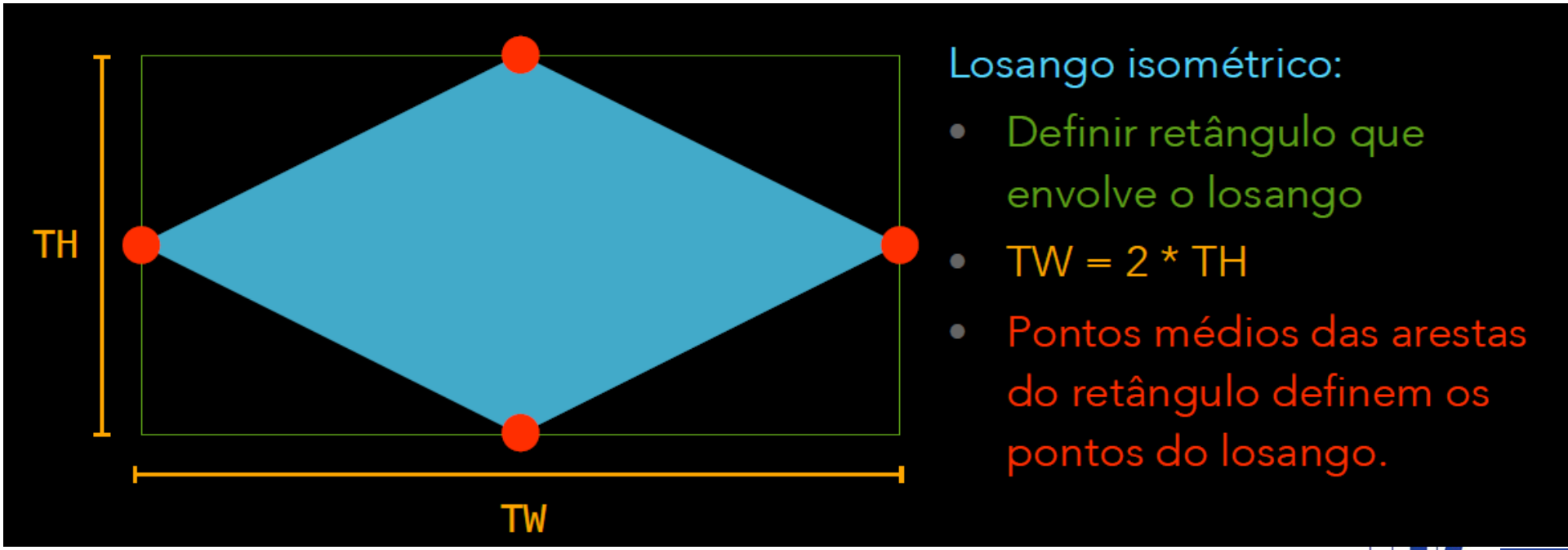
- Para conseguir exibir um *tile isométrico,* que não possui uma forma retangular, na representação gráfica bidimensional que utiliza a transferência de mapa de bits retangulares, é necessário o uso da técnica de transparência.





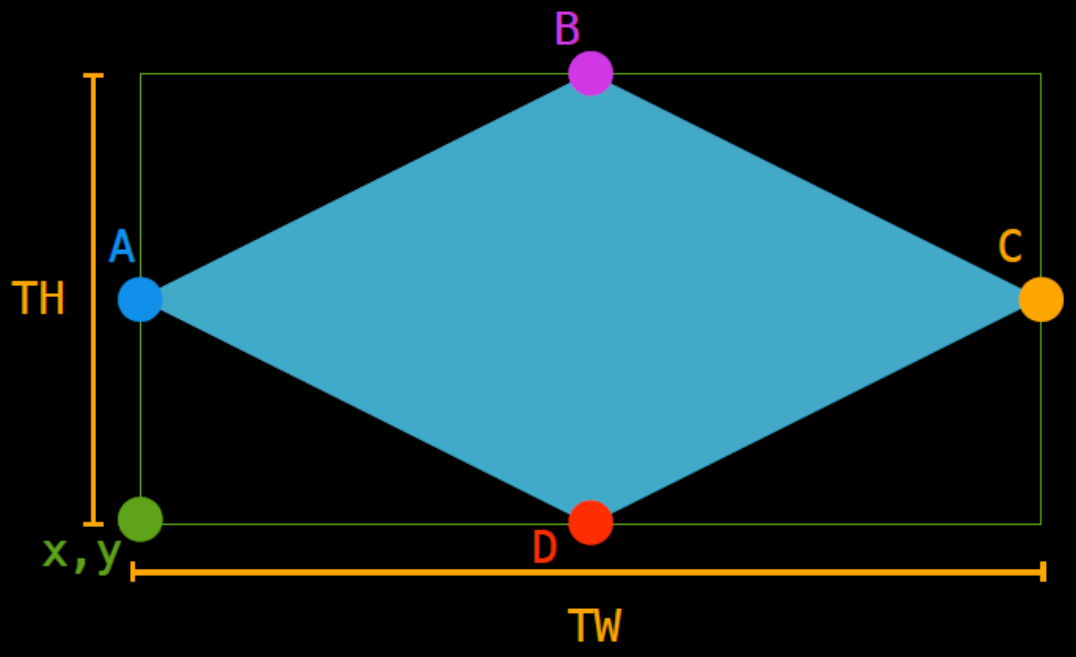
Tile Isométrico

- O tile isométrico pode ser representado por um losango, justamente porque o ponto de vista do mundo é rotacionando em 30º.



Tile Isométrico - Desenho

```
void drawIsoPolygon(float x, float y, float tw, float th){
  // com base no canto esquerdo inferior:
  A = {x, y+th/2};
  B = {x+tw/2, y+th};
  C = {x+tw, y+th/2};
  D = {x+tw/2, y};
}
```

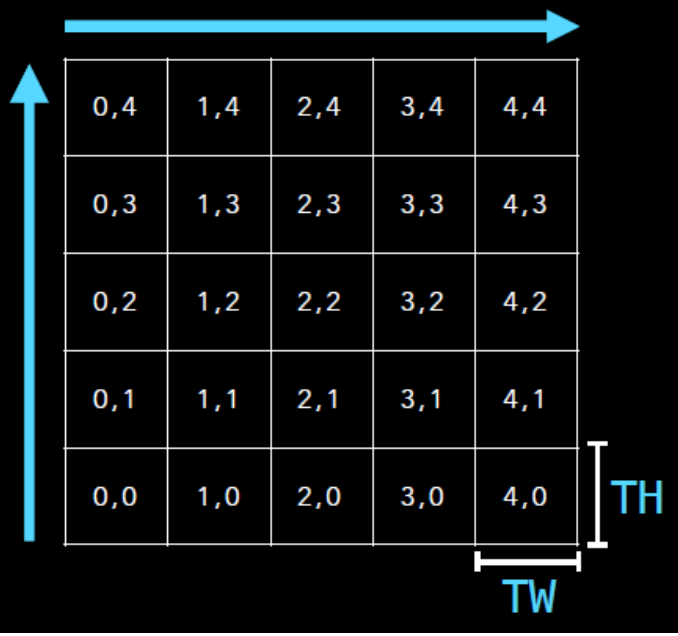


Questões de Implementação

- Para implementar mapas isométricos, deve-se ter atenção:
  - Representação de objetos e do próprio na matriz de tiles (estrutura e sua manipulação) e desenho do cenário (como mostrar a matriz na tela): **calculoPosicaoDesenho()**
  - Movimentação de cursor/personagem pelo cenário e na matriz (estrutura): **tileWalking()**
  - Clique do mouse sobre tiles: identificar tile clicado e selecioná-lo: **mouseMap()**

Exemplo de base: Tilemap Regular

- Tiles são retangulares e o método de desenho é trivial. Segue algortimo raster-scanorder (de cima para baixo, da esquerda para direita)

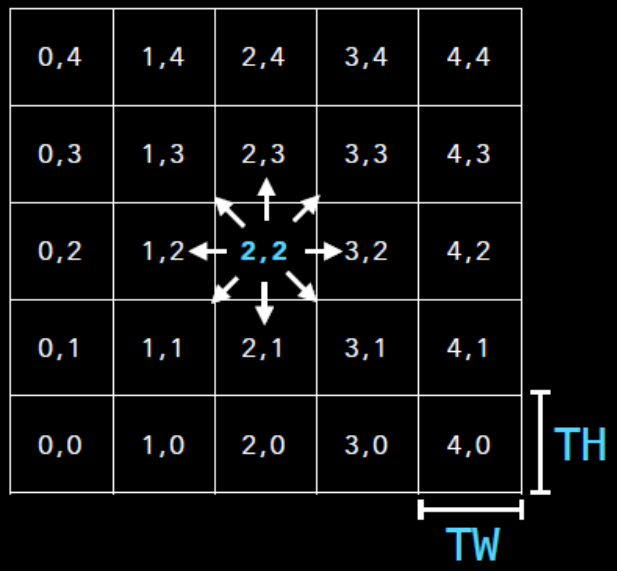



```
calculoPosicaoDesenho()

Para cada linha do tilemap
  py = linha * TH
  Para cada coluna do tilemap
    px = coluna * TW
    tile = tilemap[col][lin]
    desenha(tile, px, py);
```

Exemplo de base: Tilemap Regular

- Navegação pelo mapa (*tile walking*) também é trivial



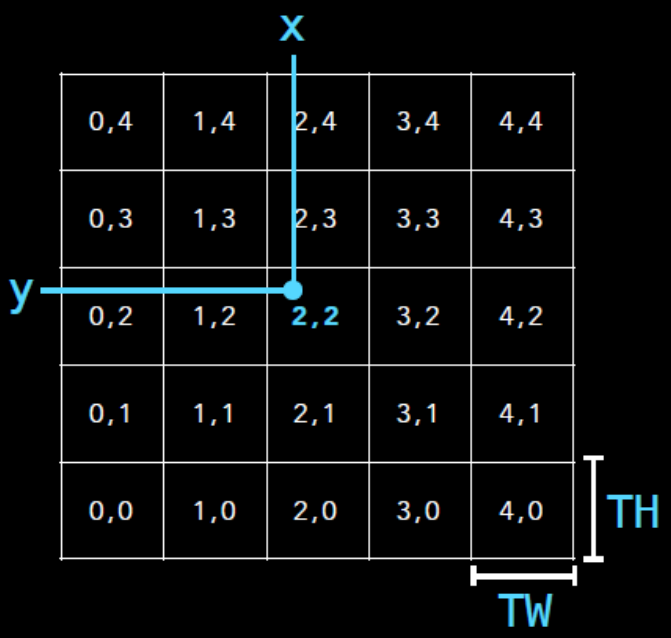

```
tileWalking()

NORTH: r++;
SOUTH: r--;
WEST : c--;
EAST : c++;
NORTHEAST: c++, r++;
SOUTHEAST: c++, r--;
NORTHWEST: c--, r++;
SOUTHWEST: c--, r--;
```

NOTE QUE AS MATRIZ ESTÁ ESPELHADA VERTICALMENTE, POIS Y (LINHA DA MATRIZ) CRESCE PARA BAIXO E Y (UNIVERSO) CRESCE PARA CIMA!

Exemplo de base: Tilemap Regular

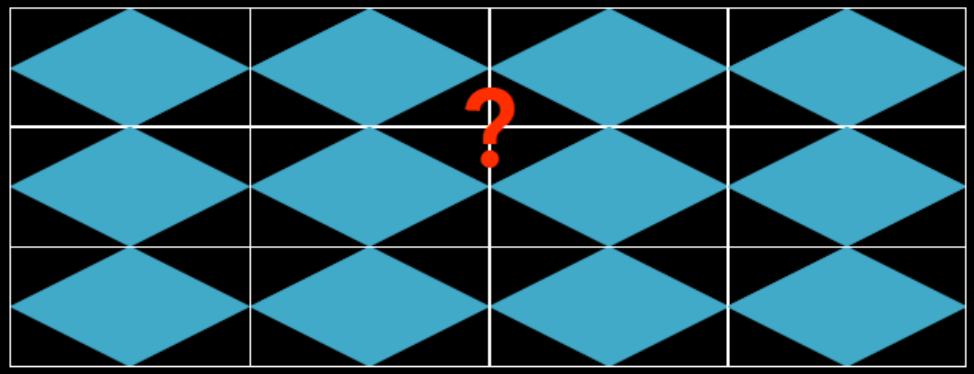
- Clique do mouse




```
mouseMap()

// para saber a coluna:
c = x / TW;
// para saber a linha:
r = y / TH;
```

Tilemap Isométrico - Desenho



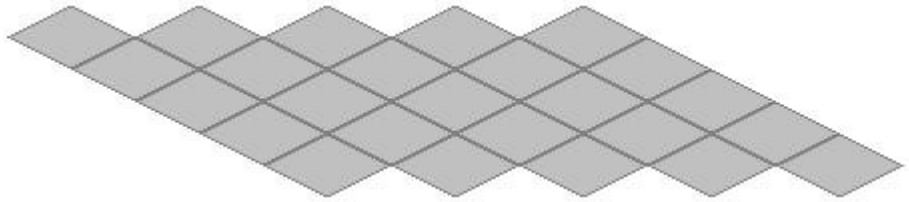
O que fazer com as regiões não preenchidas?

Tipos de Mapas Isométricos

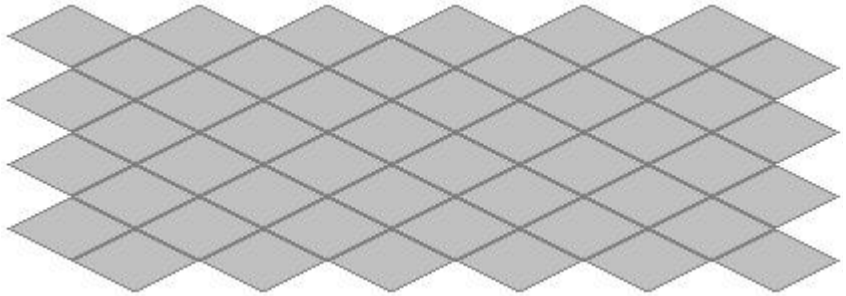
- Principais tipos:
  - Slide Maps
  - Staggered Maps
  - Diamond Maps

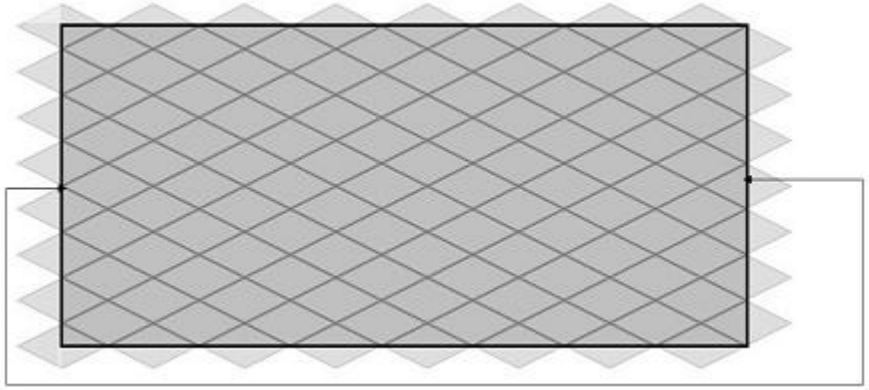
Slide Maps

- Mais fácil dos mapas isométricos de se navegar e de renderizar
- No entanto, ele possui uma aplicação prática limitada por ocupar um espaço muito grande na tela, e por isso poucos jogos utilizam esse tipo de mapa
- Contudo, por ser um mapa fácil de lidar, ele é um estudo de casos perfeito para aprender a estabelecer um sistema de coordenadas, a movimentar unidades nos mapas isométricos e de descobrir a posição de um *tile na tela.*



Staggered Maps



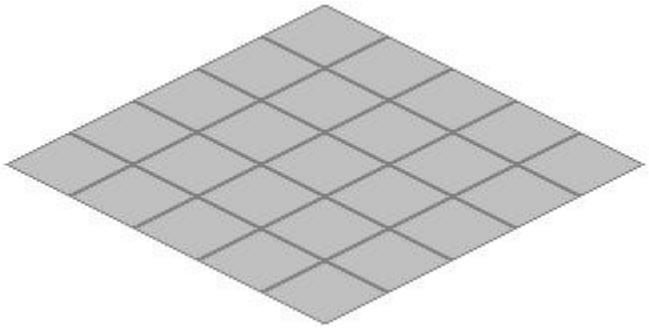


Staggered Maps

- Os Staggered Maps são mapas isométricos bastante utilizados nos jogos para PC. Jogos como Civilization II, Alpha Centauri e Civilization: Call to Power utilizam este tipo de mapa.
- Os mapas do tipo Staggered são um dos mais complicados de manipular. No entanto, algumas características os tornam bastante atrativos:
  - Pelo formato quase retangular, esse tipo de mapa é o que menos desperdiça espaço na tela. Existe ainda a possibilidade de cortar as “arestas” do mapa fazendo com que ele tome um formato totalmente retangular.
  - Aproveitando o formato retangular, é possível fazer com que o scroll do mapa na tela seja contínuo, onde a linha/coluna de tiles mais à direita/cima leva para a mais à esquerda/baixo e vice-versa, dando a impressão de um mapa cilíndrico.
  - Uma boa aplicação para os mapas cilíndricos é uma representação da Terra, como usado em Civilization II.

Diamond Maps

- Os mapas do tipo *Diamod* são um dos mapas isométricos mais utilizados, principalmente pelos jogos de estratégia em tempo real.
  - Exemplos: jogos clássicos como *Age of Empires, Sim City 2000/3000* e *The Sims*



Principais Problemas

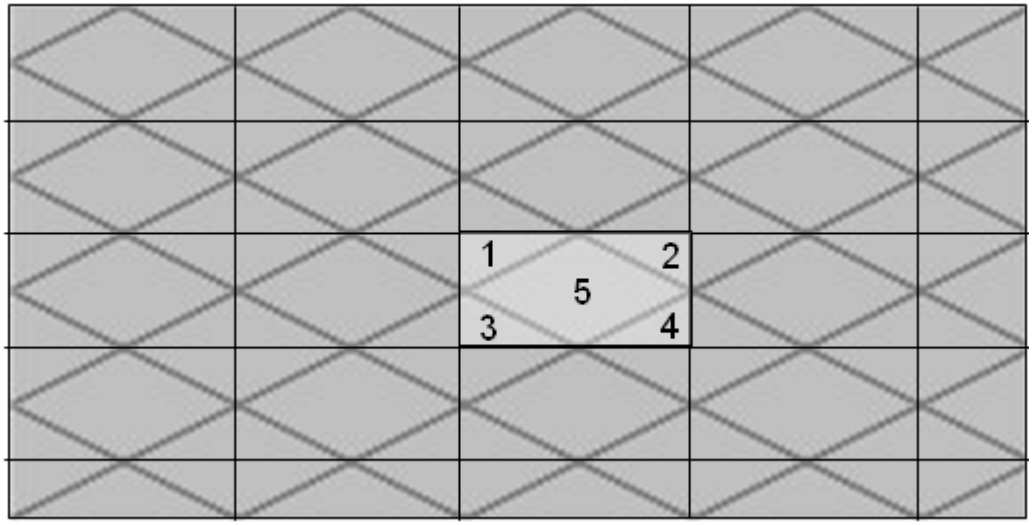
- Ordem de desenho
  - *Tiles* precisam ser renderizados de forma que nenhum tile seja plotado após outro que está “à frente” dele
  - Se uma pequena porção da tela for atualizada, é necessário atualizar os *tiles* modificados e todos os vizinhos, obedecendo à regra anterior

Principais Problemas

- Mapear um ponto na tela para uma posição no *TileMap,* ou seja, dado um ponto na tela, a que tile ele pertence.
  - Cálculos matemáticos ou...
  - Verificar em que retângulo de uma grade retangular um ponto está contido
    - Divide-se o mundo em retângulos
    - Descobrindo o retângulo em que está o ponto na tela
    - Conhecendo como andar no *tile* (de acordo com o tipo de mapa)
      - Descobrir o tile central do retângulo (índice)
      - Descobrir em qual das 5 regiões formadas está o ponto

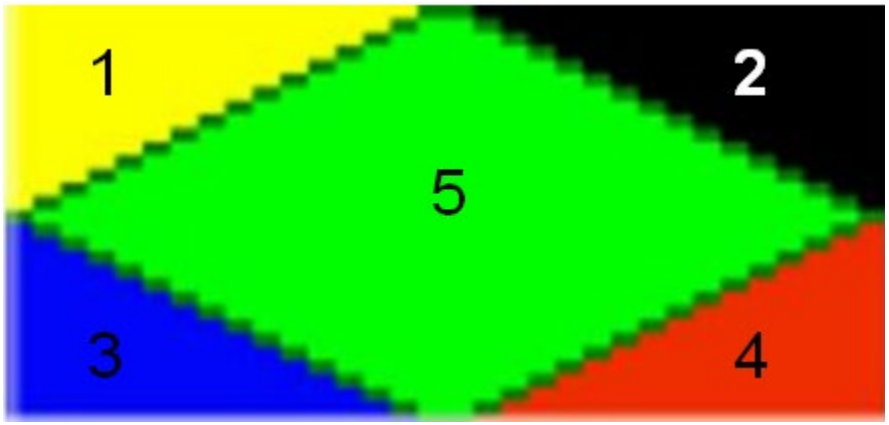
Principais Problemas

- Dividindo o mundo em retângulos



Clique do Mouse – Máscara de Textura

- Como os tiles possuem o mesmo tamanho, é possível construir uma figura externa do mesmo tamanho do retângulo construído anteriormente onde cada tile possua uma cor diferente
- Assim, para descobrir a que tile o ponto pertence, basta mapear o ponto nas coordenadas do retângulo na figura e pegar a cor existente



Clique do Mouse – Coords Baricêntricas

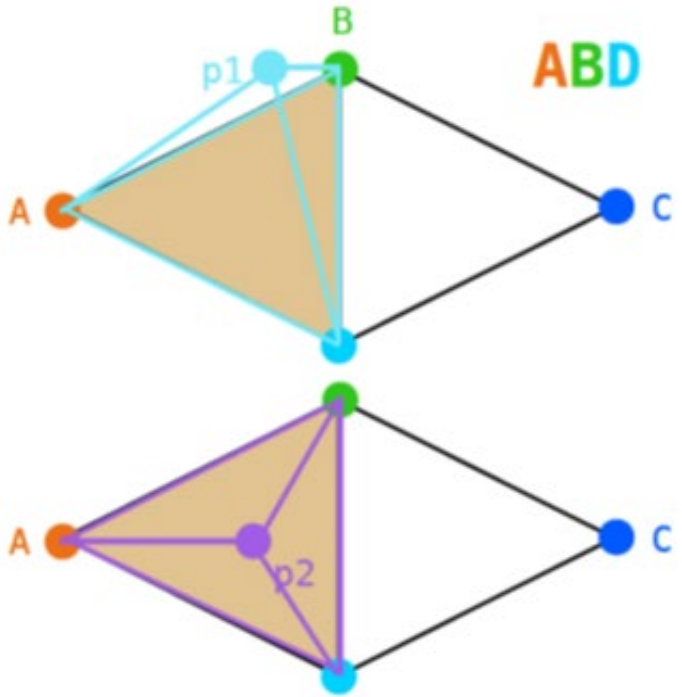
- Uma alternativa interessante é fazer o cálculo de área de triângulo para verificar a colisão:

  Se area(ABD) == area(ApB)+area(pBD)+area(ApD)

  então tem colisão!

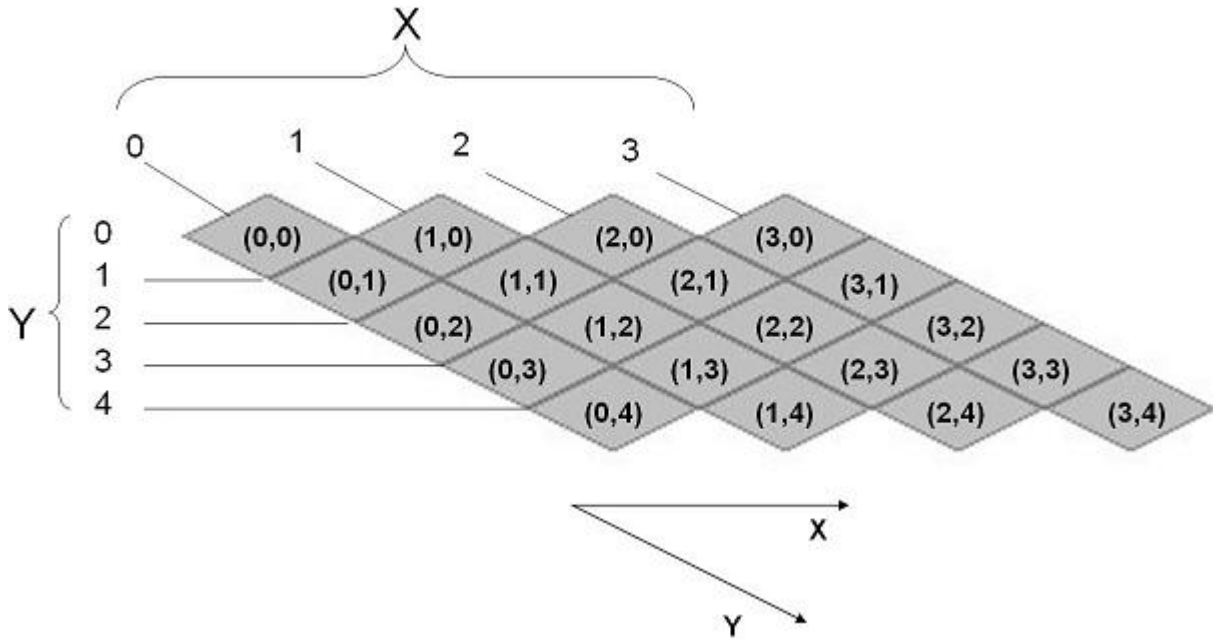
```
Cálculo de área do triângulo:
 abs(
   (
       (V2x - V1x) * (V3y - V1y)
     - (V3x - V1x) * (V2y - V1y)
   ) / 2
 )
```

CRIE DUAS FUNÇÕES:
AREA(TRIANGULO)
COLISAO(TRIANGULO, PONTO)



Slide Maps

- Adotar um sistema de coordenadas
  - *x* cresce para o leste e o *y cresce para sudeste*



- Uma característica importante deste sistema de coordenadas é que ele facilita a manutenção da ordem de renderização dos *tiles,* uma vez que o ponto (0,0) do *TileMap* está na linha superior do mapa.

34

Slide Maps

- Mapeamento

Pixel	*Incremento em 1 unidade de X do TileMap*	*Incremento em 1 unidade de Y do TileMap*	*Equação*
X	+TileWidth	+TileWidth/2	MapX*TileWidth + MapY*TileWidth/2
Y	0	+TileHeight/2	MapY*TileHeight/2

Tabela 4-1 Mapeando uma Coordenada do *TileMap* para a Tela no *Slide Map*.

Movimentação dos objetos na tela

- 8 direções

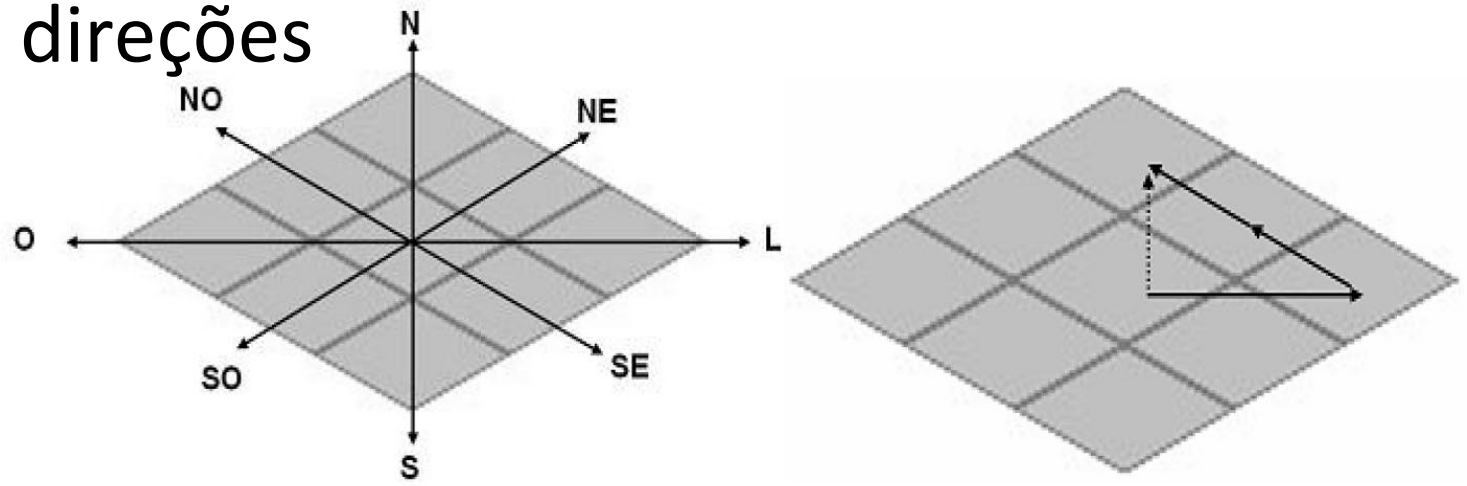


Figura 4-10 Direções Regulares nos Mapas Isométricos (figura à esquerda) e a Direção Norte sendo composta de Direções conhecidas (figura à direita).

Supondo que exista uma unidade no *tile (1,2) da Figura do slide anterior,* e se deseje movê-la para o norte. Para que posição do *TileMap deve-se movê-la?*

36

Movimentação dos objetos na tela

- Slide Map

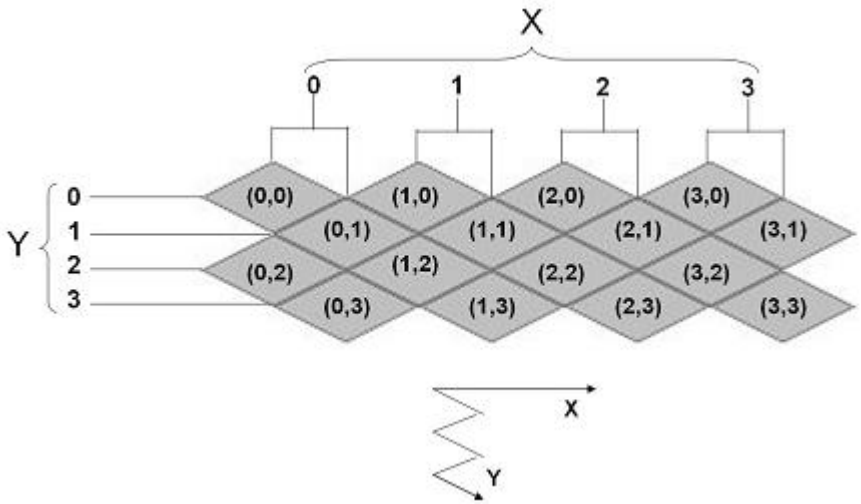
Direção	*Variação no X do TileMap*	*Variação no Y do TileMap*
Norte	*+1*	*-2*
Sul	*-1*	*+2*
Leste	*+1*	*0*
Oeste	*-1*	*0*
Nordeste	*+1*	*-1*
Noroeste	*0*	*-1*
Sudeste	*0*	*+1*
Sudoeste	*-1*	*+1*

Tabela 4-2 Variação na Coordenada de um *tile* nos *Slide Maps* segundo uma Orientação.

Staggered Map

- Sistema de coordenadas

Se MapY for par → PixelX = MapX*TileWidth
Se MapY for ímpar → PixelX = MapX*TileWidth + TileWidth/2
PixelY = MapY*TileHeight/2



Movimentação dos Objetos na tela

Direção	*Paridade do Y*	*Incremento em X*	*Incremento em Y*
Leste	-	*1*	*0*
Oeste	-	*-1*	*0*
Norte	*Par*	*0*	*-2*
Sul	*Par*	*0*	*2*
Nordeste	*Par*	*0*	*-1*
Noroeste	*Par*	*-1*	*-1*
Sudeste	*Par*	*0*	*1*
Sudoeste	*Par*	*-1*	*1*
Norte	*Ímpar*	*0*	*-2*
Sul	*Ímpar*	*0*	*2*
Nordeste	*Ímpar*	*1*	*-1*
Noroeste	*Ímpar*	*0*	*-1*
Sudeste	*Ímpar*	*1*	*1*
Sudoeste	*Ímpar*	*0*	*1*

Tabela 4-4 Variação nas Coordenadas do *tile* nos *Staggered Maps* seguindo uma Orientação.

Diamond Maps

- PixelX = (MapX-MapY)*TileWidth/2
- PixelY = (MapX+MapY)*TileHeight/2

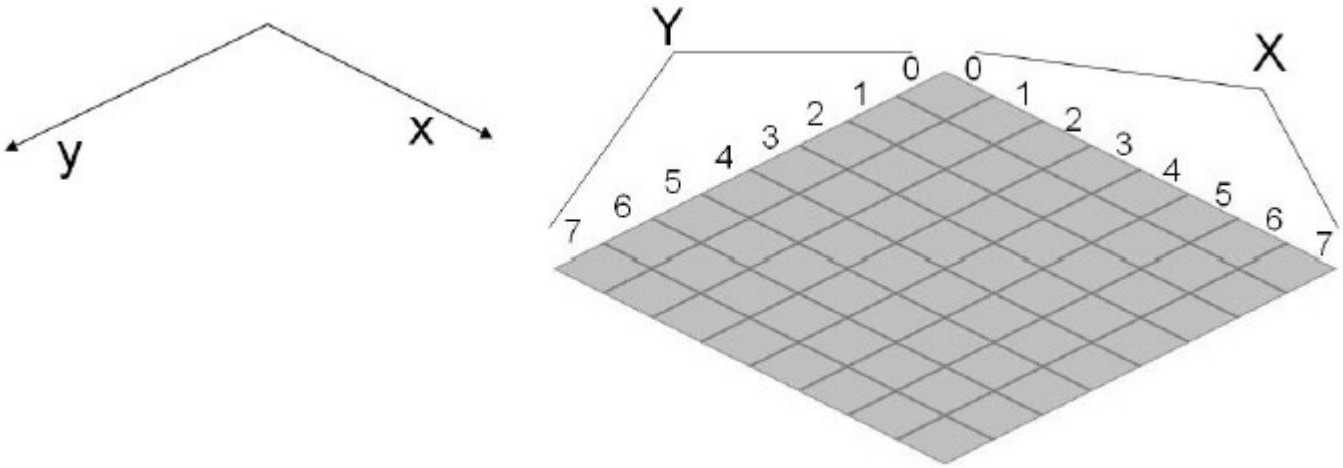


Figura 4-14 Sistema de Coordenadas do Mapa do tipo Diamond

40

Movimentação dos Objetos na Tela

Direção	*Variação em X*	*Variação em Y*
Sudeste	1	0
Sudoeste	0	1
Noroeste	-1	0
Nordeste	0	-1
Norte	-1	-1
Sul	1	1
Leste	1	-1
Oeste	-1	1

Tabela 4-5 Variação nas Coordenadas do *Tile* nos *Diamond Maps* seguindo uma orientação

Leitura Obrigatória:

- Dissertação de Mestrado: "*Forge 16V: Um Framework para Desenvolvimento de Jogos Isométricos*"
  - Capítulo 4
  - Disponibilizado no Canvas ou
    - https://repositorio.ufpe.br/bitstream/123456789/2531/1/arquivo4819_1.pdf

Referências

- José Torres Sampaio, Eduardo; Lisboa Ramalho, Geber. *Forge 16V : um framework para o desenvolvimento de jogos isométricos*. 2003. Dissertação (Mestrado). Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife, 2003.

- Os materiais de Processamento Gráfico/Fundamentos de CG são preparados pelo grupo de professores que ministra (ou ministrou) a Atividade. Créditos para:
  - Leandro Tonietto
  - João Ricardo Bittencourt
  - Luiz Gonzaga da Silveira Jr.
  - Rossana Baptista Queiroz