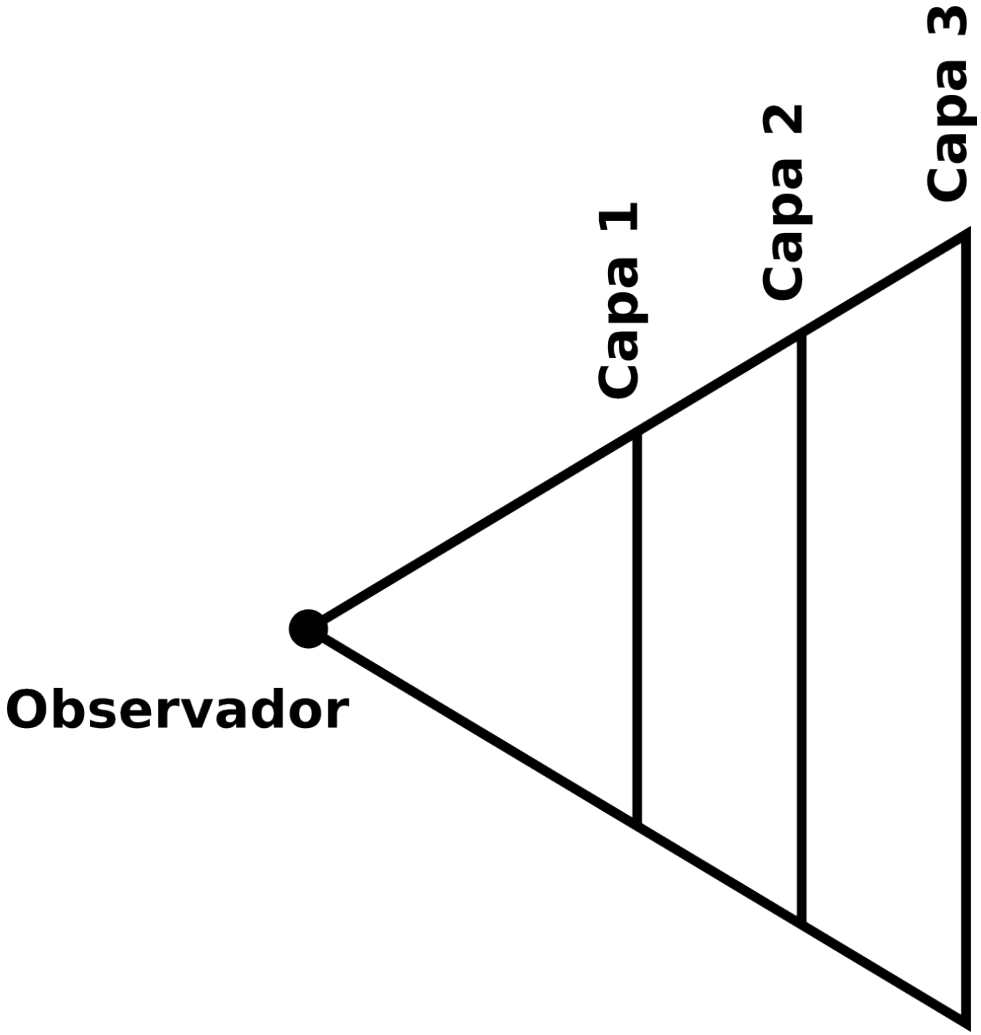
Jocs 2D

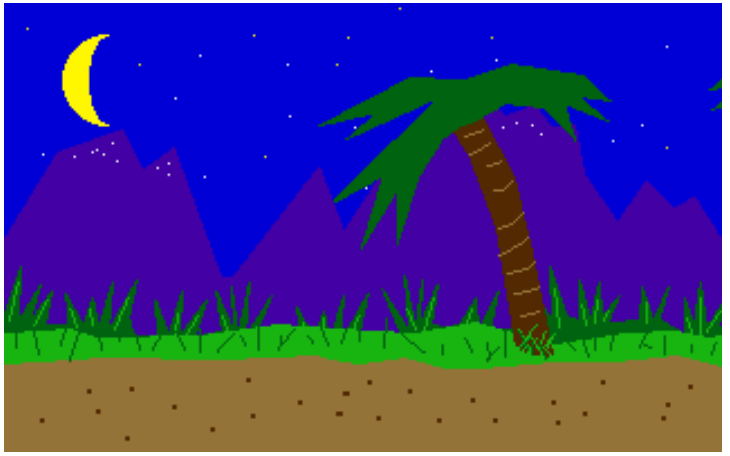
Professors de VJ

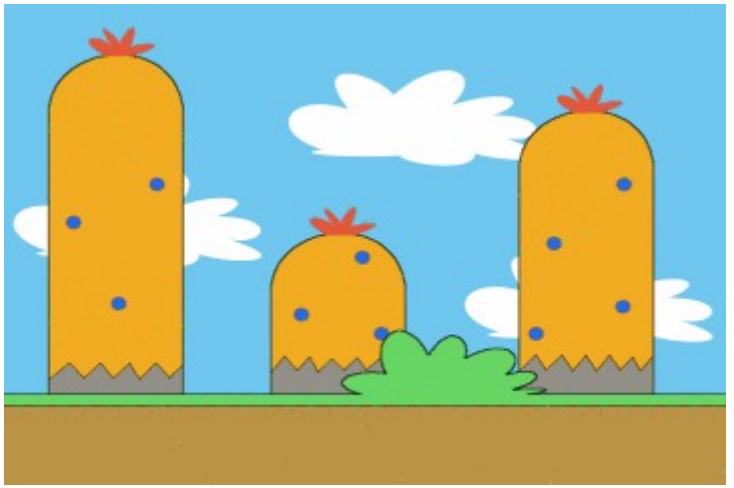
Parallax

- Parallax scrolling
  - Tenir diverses capes.
  - Es mouen a diferents velocitats.
- Resultat
  - Efecte de profunditat.
- Implementació
  - Adaptar tamany i velocitat a la distància.

Parallax



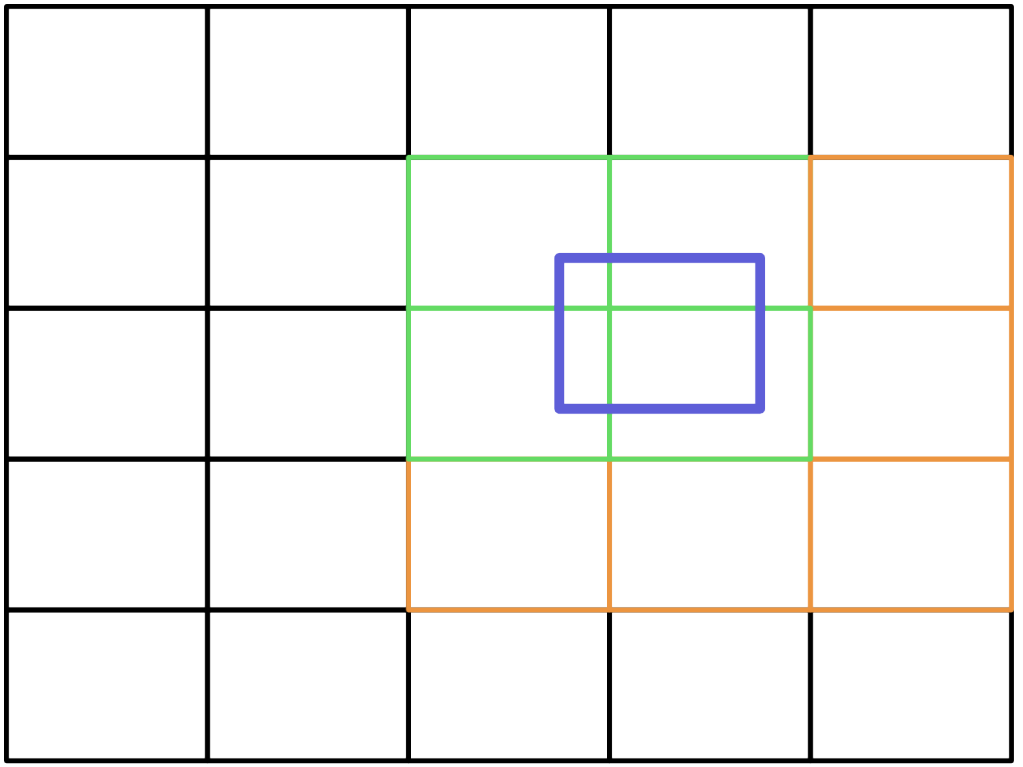




Page-Swap Scrolling

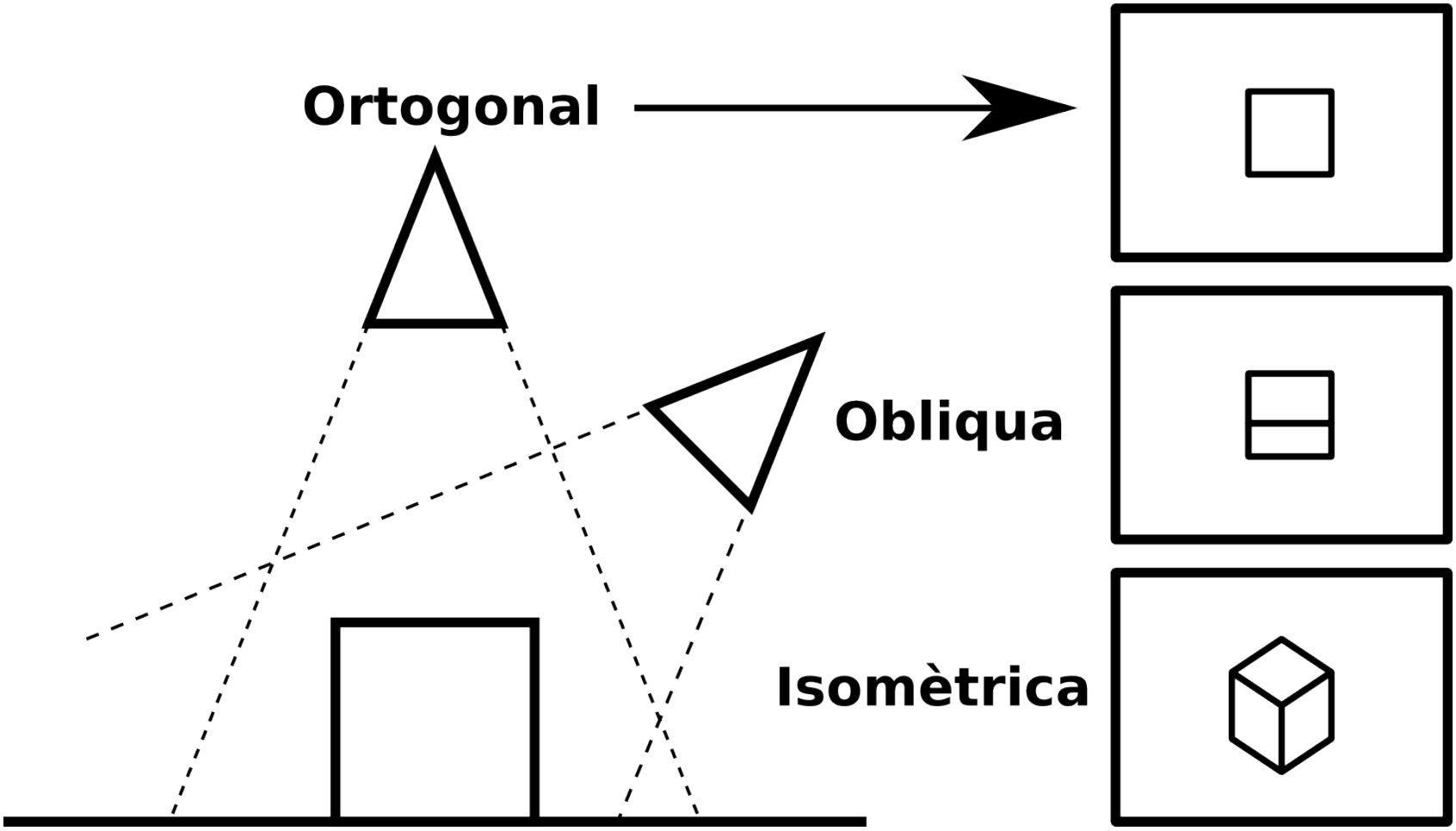
Si el mapa es tot únic:

- Cal carregar només el que es veu.

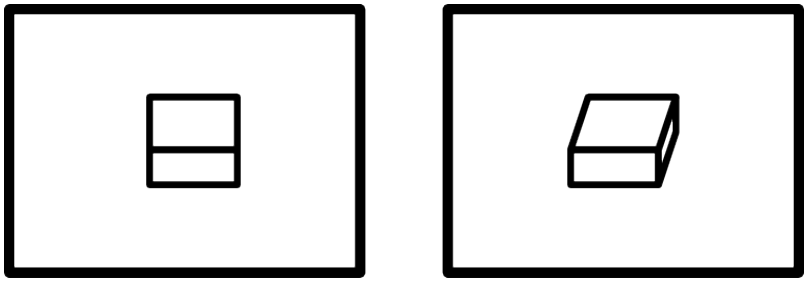


Projeccions

Es poden fer servir diferents projeccions:



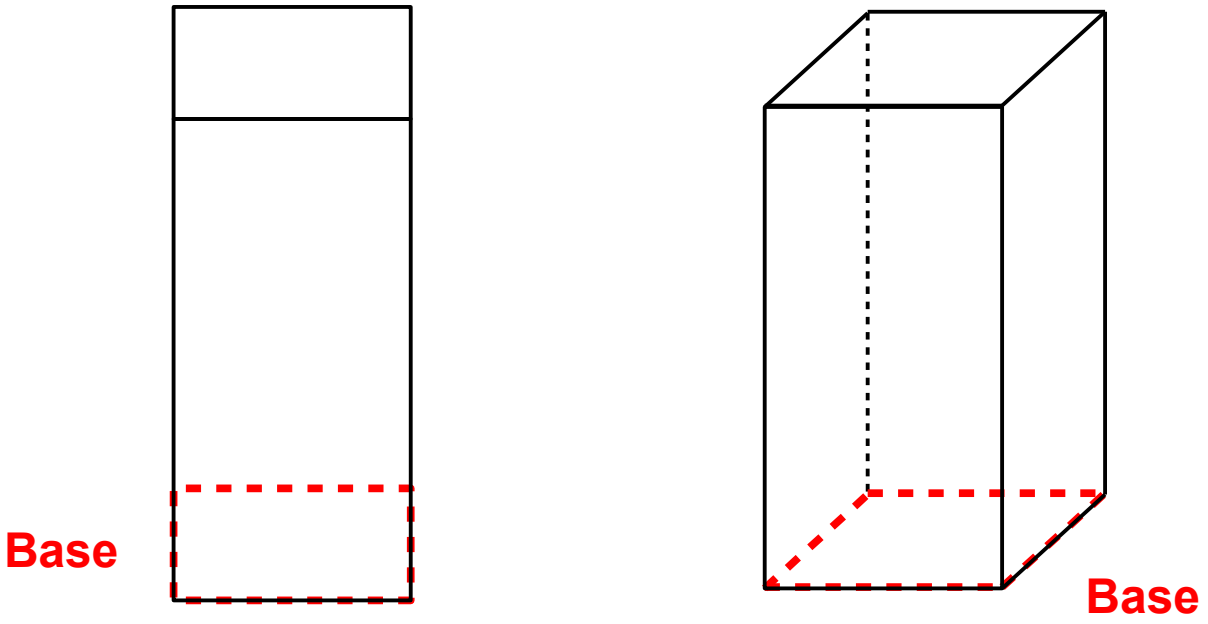
Obliqua



Obliqua

Els tiles ocupen més que la base.

Tiles



Obliqua

A tenir en compte:

- Cal visualitzar els tiles en un ordre específic.
- Algorisme del pintor:
  - Dibuixar els tiles de darrera a davant.
- Per decidir que dibuixar:
  - Cantonada inferior dreta
    - Ens serveix la base → Igual que ortogonal
  - Cantonada superior esquerra
    - Cal afegir la diferència de tamany entre base i tile.

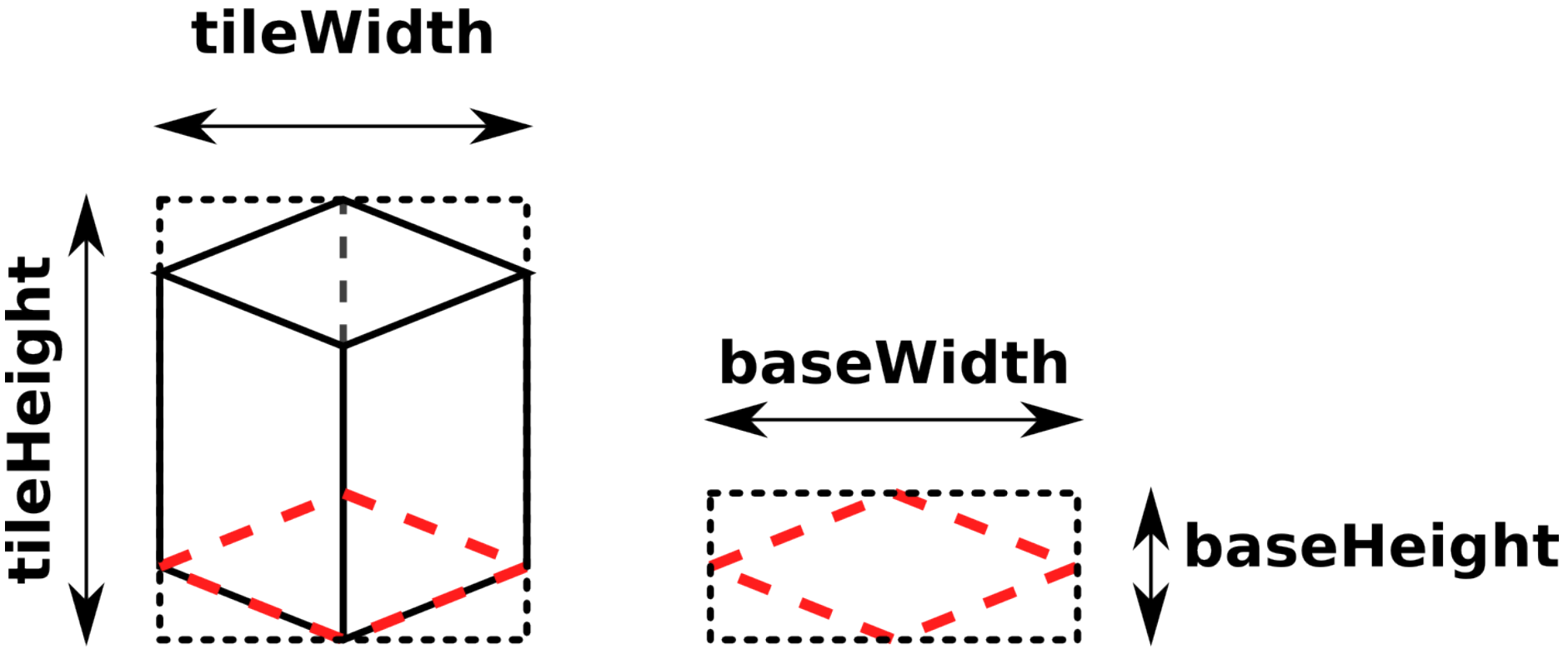
Isomètrica

Elevació de 30º i rotació de 45º.

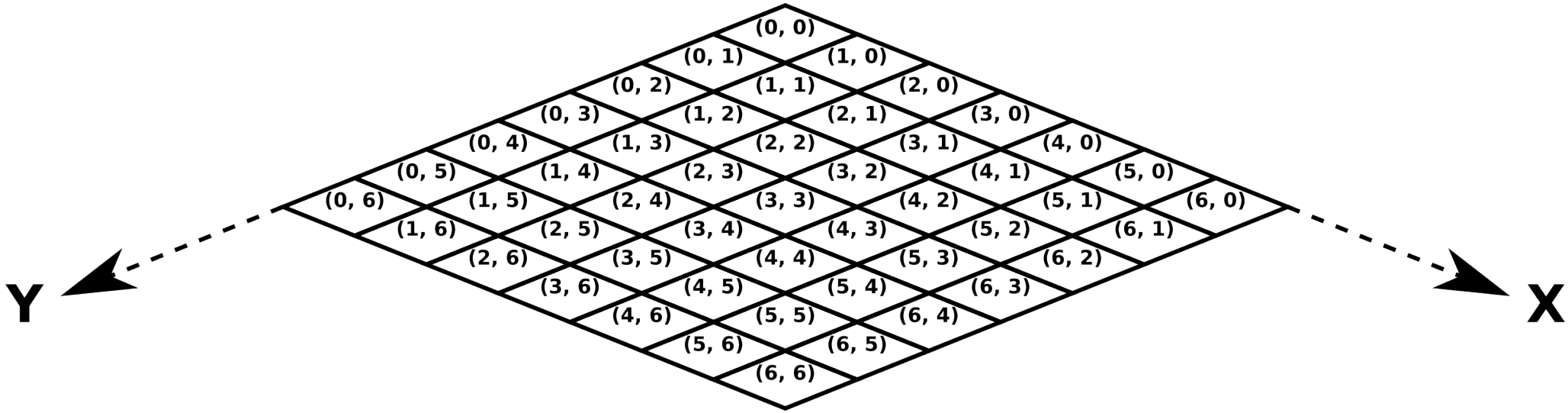
QBert

Diablo

Isomètrica



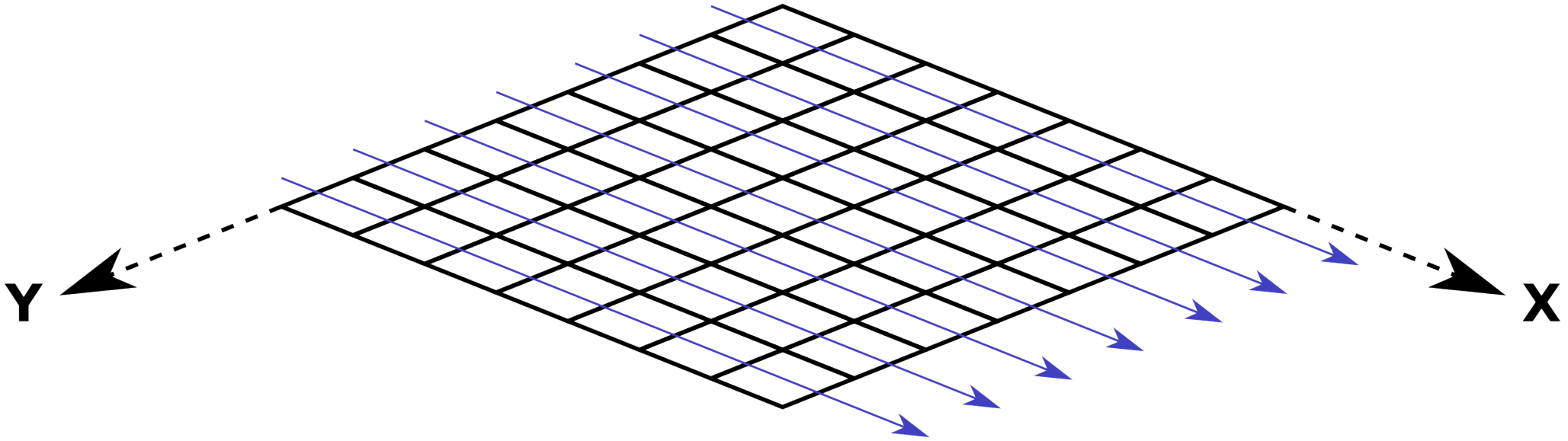
Isomètrica



(x, y) = (fila, columna)

Isomètrica

Dibuixat en diamant



Isomètrica

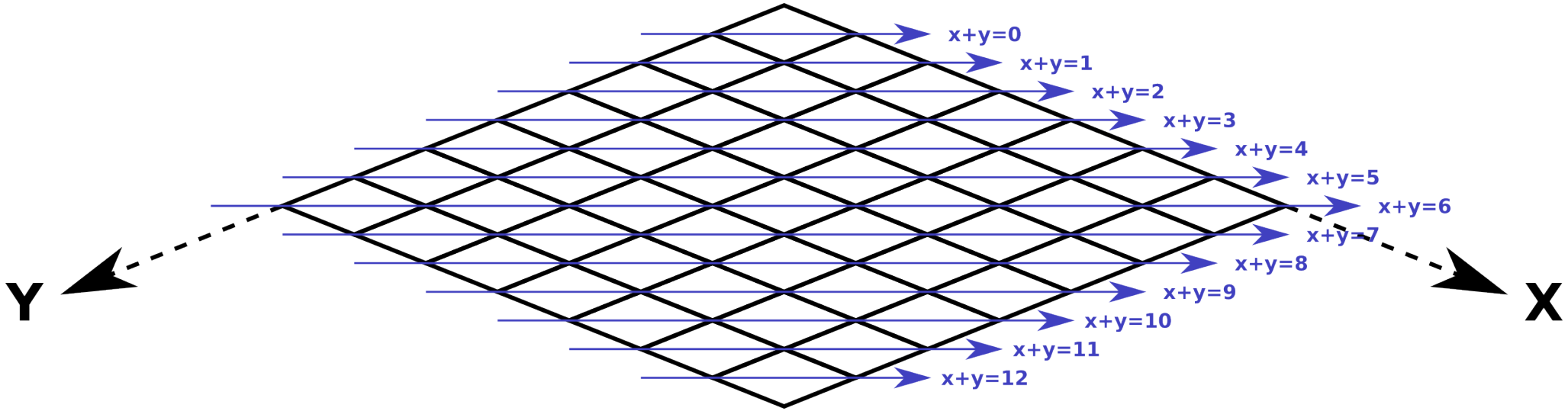
Dibuixat en diamant

```
void Render()
{
    int pos = 0;

    for(int y=0; y<mapHeight; y++)
        for(int x=0; x<mapWidth; x++, pos++)
            RenderTileIsometric(map[pos].index, x, y);
}
```

Isomètrica

Dibuixat en “zig-zag”.



Isomètrica

Mapejat a pantalla:

- Les diagonals es corresponen amb els eixos de la pantalla.
  - diagonalX = x - y
  - diagonalY = x + y
- Coordenades de pantalla pel tile (x, y):
  - Considerant que la base del tile (0, 0) es trobi al píxel (0, 0).
  - pixelX = (x – y) * baseWidth / 2
  - pixelY = (x + y) * baseHeight / 2 + baseHeight / 2
- Mapejat invers (Coordenades de pantalla → Tile):
  - Selecció.
  - Optimització: Renderitzar només el que es veu.

Isomètrica

Dibuixat en “zig-zag”.

```
void Render()
{
     int diagonal, pos;

     for(diagonalY=0; diagonalY<=mapWidth+mapHeight-2; diagonalY++)
          for(int x=max(0, diagonalY-mapHeight+1); x<=min(diagonalY, mapWidth-1); x++)
          {
               y = diagonalY - x;
               pos = y * mapWidth + x;
               RenderTileIsometric(map[pos].index, x, y);
          }
}

void Render2()
{
     int diagonalX, diagonalY, x, y;
     int min, max;

     for(diagonalY=0; diagonalY<=mapWidth+mapHeight-2; diagonalY++)
     {
          for(diagonalX=-mapHeight+1; diagonalX<=mapWidth-1; diagonalX++)
          {
               x = (diagonalX + diagonalY) / 2;
               y = (diagonalY - diagonalX) / 2;
               if((x >= 0) && (y >= 0) && (x < mapWidth) && (y < mapHeight))
                    RenderTileIsometric(map[pos].index, x, y);
          }
     }
}
```

Isomètrica

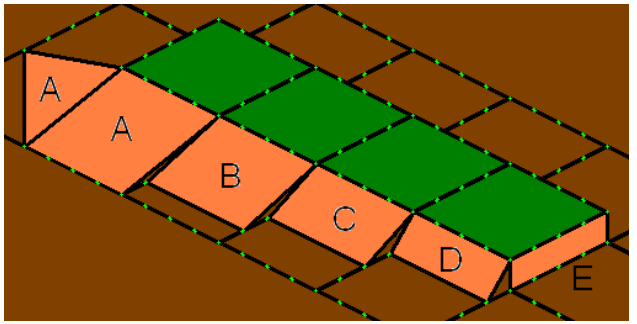
Tècniques avançades

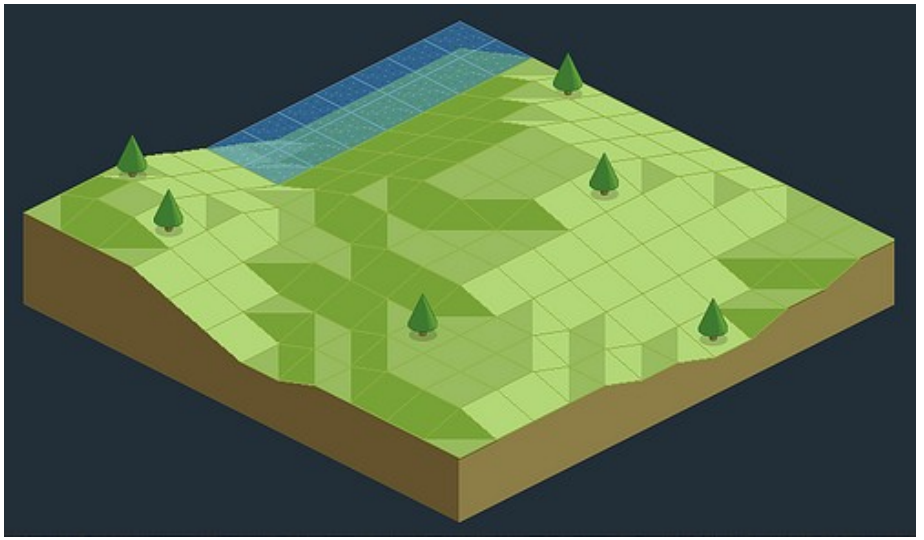
- Caching
  - Prerenderitzar parts estàtiques del mapa .
  - Problema: Combinar les capes dinàmiques.
  - Solució: Amb OpenGL es pot donar una profunditat (Z) a cada tile.
- Culling
  - Si un tile (degut a la seva altura) oculta a un altre, podem deixar de visualitzar el segon.

Isomètrica

Tècniques avançades

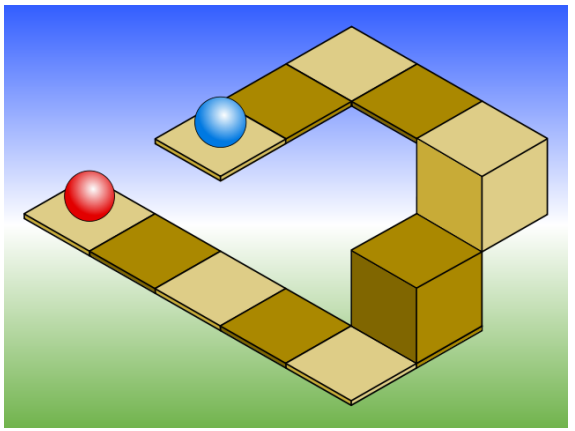
- Elevation layers
  - Les capes addicionals es poden posar de forma que encaixin en altura.
  - Cal desplaçar el mapejat dels tiles a pantalla.
  - És interessant tenir tiles de connexió.

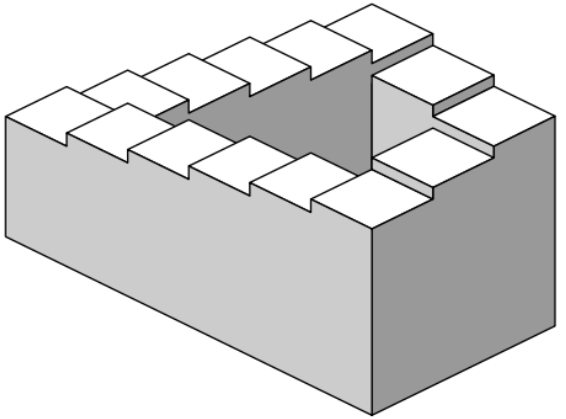




Inconvenients

- Problemes en la percepció de l'altura.
- Possibilitat d'escenes impossibles.





Solució

- Fer servir projecció perspectiva.
  - Gràfics 3D.
  - Encara que lògica 2D.

Sprites

Sèrie de imatges 2D per representar una entitat dinàmica del joc:

- Personatges, projectils, enemics, vehicles, ...

Sprites

Animació:

- Es fa actualitzant quin “frame” (quina imatge) es visualitza a cada moment.
  - Animació basada en frames.
- Està enllaçada a:
  - Moviment (en el cas de l'avatar del jugador).
  - Inteligència artificial.
  - Motor físic.
  - ...
- Cal tenir en compte la velocitat del main loop.
  - Velocitat de canvi de frame.
  - Cada frame té associat un temps que ha d'estar actiu.

Sprites

Animació:

- La animació pot ser fixa.
  - El sprite no es mou del seu lloc.
- Tenir un moviment extern
  - Causat per la simulació física.
- O tenir un moviment associat.
  - Per exemple, un personatge que camina.
  - En aquest cas cal moure el sprite en sincronia amb l'animació. Podem tenir:
    - Transformació fixa per frame.
    - Transformació diferent per a cada frame.
  - La transformació pot ser:
    - Simple traslació (en forma de vector).
    - Transformació completa, incloent traslació, rotació i escalat (en forma de matriu 3x3).

Sprites

Sprites “morts”

- Estan fora de pantalla i no han de tornar.
- Enemics derrotats.

Que fem amb els sprites “morts”?

- Eliminar per recuperar memòria.
- En el segon cas:
  - Posar en estat de mort durant un temps.
  - Després eliminar-lo gradualment.

Sprites

Sense desplaçament

Sprites

Sense desplaçament

Sprites

Sense desplaçament

Sprites

Sense desplaçament

Sprites

Sense desplaçament

Sprites

Sense desplaçament

Sprites

Amb desplaçament

Sprites

Amb desplaçament

Sprites

Amb desplaçament

Sprites

Amb desplaçament

Sprites

Amb desplaçament

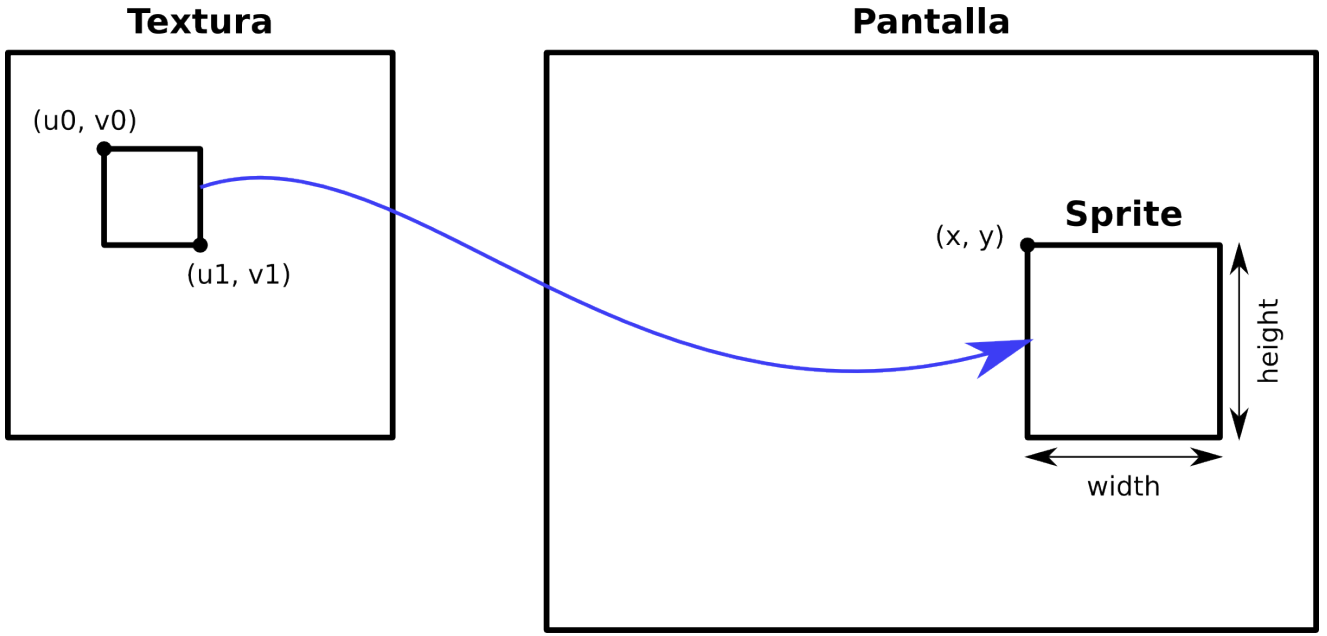
Sprites

Amb desplaçament

Sprites

Visualització

- Sistema àntic: Blitting
  - Còpia de blocs de memòria (que contenen la imatge).
- Amb OpenGL (o DirectX):
  - Dibuixem rectangles texturats.
  - Necessitem les coordenades de textura.



Sprites

```
struct Sprite
{
    float x, y, width, height;
    int animationId;
    int currentKeyframe;
    float currentAnimTime;
    SpriteAnimations *anims;
};

struct SpriteAnimations
{
    int texId;
    vector<Frame *> frames;
    vector<SpriteAnimation *> anims;
};

struct Frame
{
    float u, v, width, height;
};
```

```
struct SpriteAnimation
{
    vector<Keyframe *> keyframes;
};

struct Keyframe
{
    int frameId;
    float duration;
    float tx, ty;
};
```

Sprites

Altres coses a tenir en compte:

- Alguns frames poden estar en més d'una animació.
  - Cal poder aprofitar-los.
- Els frames es poden trobar en diverses textures.
- Un sprite pot tenir:
  - Posició genèrica.
  - Rotació arbitrària.
  - Escalat.

Il·luminació

Com que treballem amb OpenGL (o DirectX) podem:

- Modular el color de la textura amb els colors dels vèrtexs del quadrat.

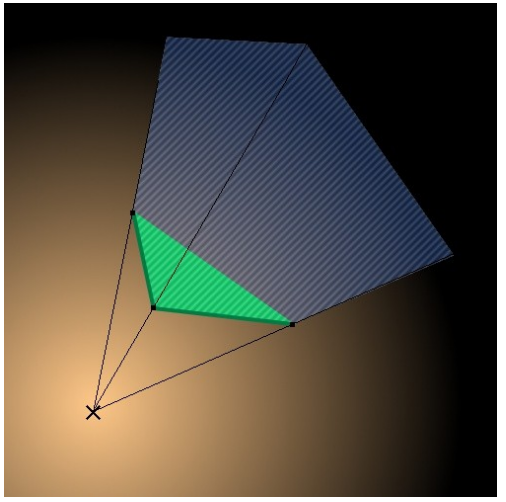
Això ens permet aplicar il·luminació bàsica.

- Definim fonts de llum donant la seva posició en coordenades de món (x, y).
- Modulem el color del quadrat en relació a la distància del tile o el sprite a la font.
  - Lineal: Atenuació = 1 / d
  - Quadràtic: Atenuació = $1 / d^2$

Il·luminació

Ombres simples

- Si la llum està adalt:
  - Podem fer servir un sprite pels elements dinàmics.
  - Tenir les ombres preintegrades als tiles.
- Si hi han elements més alts que la llum:
  - Caldrà calcular les ombres → Força complicat.
  - Difícil d'integrar amb el sistema anterior.
  - Solució més simple: 3D + Lògica 2D.
  - Alternativa: Traçar ombres.

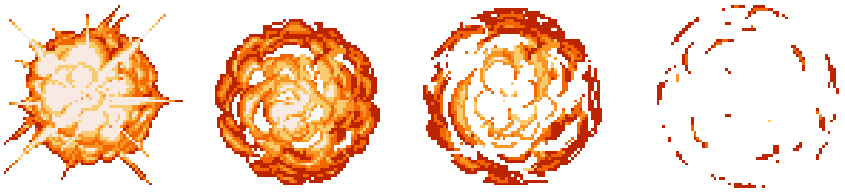


Altres efectes

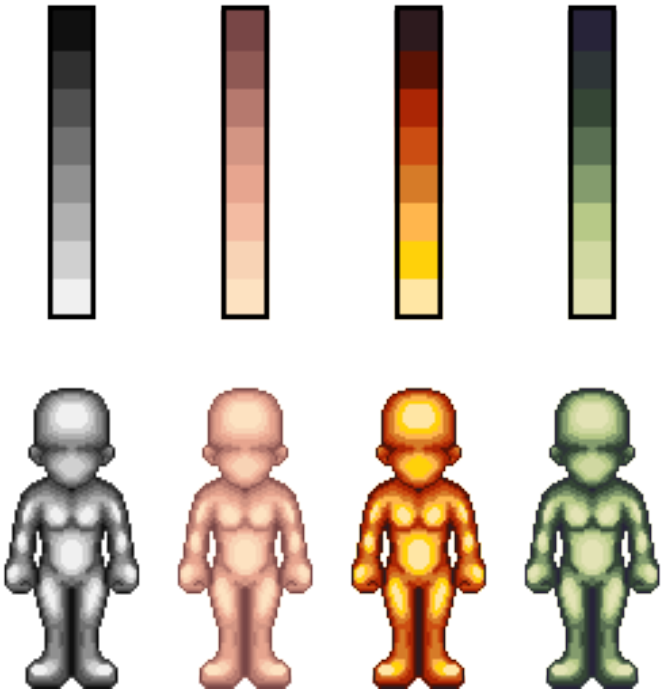
Reflexions

Ombres

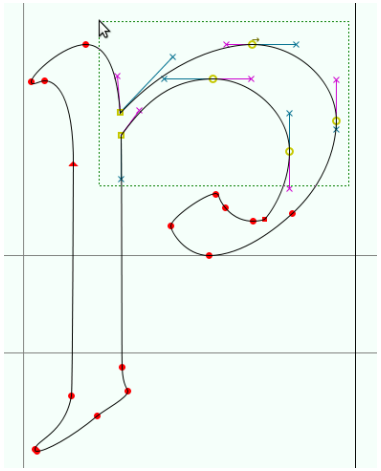
Explosions



Substitució de colors (Paletes)



Text



Visualització de text

- Arxius de fonts
  - Contenen descripció dels caràcters en segments i corbes de Bezier.
  - TTF – TrueType
  - OTF – OpenType
  - Fonts serif i sans-serif.

AaBbCc

- Conversió de fonts a bitmap (a una textura).
  - Cal tenir clar la diferència entre:
    - Monospace (amplada fixa).
    - Proportional typeface (amplada variable).

Text

Un cop tenim la textura i les posicions dels caràcters:

podem visualitzar-los:

- Fent un quadrat per caràcter.
- Prerenderitzant-los a una textura que farem servir després.

Text

També tenim l'opció de fer els nostres propis caràcters.

GUI

Un joc també pot necessitar una interfície gràfica (GUI – Graphical User Interface).

- Etiquetes, Botons, Entrada de text, Llistes i altres.
- Hi ha llibreries: CEGUI, ...
- Tenen dos components fonamentals:
  - Manegament d'events.
  - Visualització de components.

GUI

Visualització