Dr. Mileff Péter

GRAFIKA PROGRAMOZÁSA

GYAKORLATI 2D GRAFIKA III. RÉSZ

Miskolci Egyetem
Általános Informatikai Tanszék

TILEMAP ALAPÚ MEGJELENÍTÉS…

Általános áttekintés

- A „**Tile-Map**” alapú megjelenítési technika:
  - széles körben elterjedt a két dimenziós számítógépes játékok világában.
  - Magyar elnevezése nincs, fordítása nem lenne túl sikeres.
- Az eljárás a korai számítógépes időkből származik:
  - a gépek teljesítménye még alacsony volt,
  - a bennük lévő memória mennyisége pedig kevés.
  - A tile alapú megvalósítás ezekhez a lehetőségekhez alkalmazkodva fejlődött ki
- Ma a mobil platformok terjedésének köszönhetően ismét nagy népszerűségnek örvend.

A módszer lényege

- Szeretnénk egy olyan alkalmazást készíteni:
  - amely megjelenítési területe nagyobb, mint egy képernyő mérete
  - a képernyő görgethető valamilyen irányban.
- **Probléma:** **A nagy területhez nagyobb memória szükséges**
  - A képernyők nem tárolhatók külön képként!
  - Régen nem volt lehetséges, ma is gond
- Tipikusan ilyen kialakításúak a platformer és a felülnézeti játékok
  - Ahol valamilyen területet lehet bejárni, sok az ismétlődő elem
- Több ezer ilyen játék készült és készül manapság is:
  - Pl. Giana Sisters (C64), Super Mario (Nintendo), Sonic (SegaM), Prince of Persia, Droid Assault (PC), Gish (PC), stb

Példa Tile-Map alapú játékra (Frogatto - 2010)

A módszer lényege

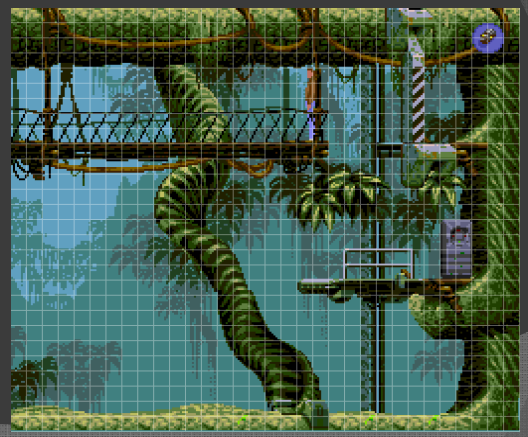
- **A módosított vizualizáció:**
  - A bejárható területet és a képernyőt virtuálisan Tile-okra bontjuk fel
    - pl. 128x128, 64x64, 32x32 pixel méretűre.
  - A Tile-ok ismétlődnek a terület (háttér) elkészítésekor,
  - Ezért elég őket egyszer betölteni, így csak egy Tile méretnyi területet foglalnak a memóriában.
  - A térképhez egy leíró mátrix/adathalmaz készül.
- A korai játéknál jól megfigyelhető a kevés Tile-ból való építkezés.
- A mai programok azonban már sokkal részletesebb grafikai megvalósítással rendelkeznek.

A módszer lényege

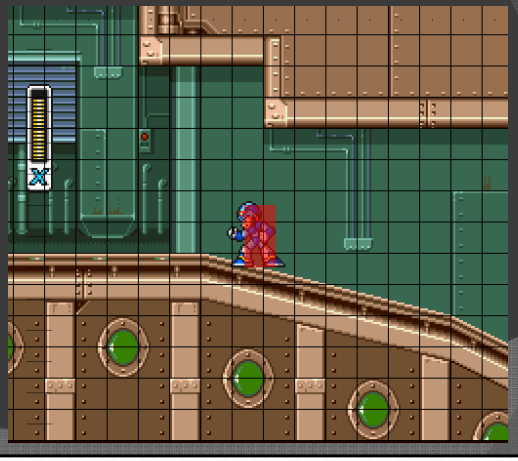
- A bejárandó területet valamilyen szerkesztő szoftverrel hozzák létre
  - legtöbbször belső fejlesztésű
- **A tárolás megoldása:**
  - Egy terület leíró fájlt készítenek.
  - a pálya Tile egységeit, azok indexeit egy N x M-es mátrixban tárolják el
  - esetlegesen hozzájuk kapcsolódó egyéb információkkal
    - Pl. átjárhatók-e vagy sem

The Great Giana Sisters C64 - (1987)

Flashback - (1992)



Mega Man X- (1993)



A módszer jellemzője

- **Nehézség:**
  - jelentős plusz munkát kell befektetni a grafikus tervezői oldalról
  - a világ szinte minden elemét Tile határra kell megrajzolni
  - és azokat külön-külön kivágni.
- Vannak a világhálón elérhető segédprogramok:
  - Segítenek a szerkesztésben. Pl. rácsháló
  - képesek egy képet megadott méretű Tile-okra szétdarabolni.
- A megrajzolt és szétdarabolt Tile-ok halmaza a **Tileset**
  - általában egy külön képben tárolják

Super Mario 3 TileSet

Tile-Map megvalósítások jellemzői

- **Legtöbbször rögzített, azonos méretű tile-ok**
  - Könnyű kezelhetőség
  - Kissé rugalmatlan vizualizáció
    - Minden azonos méretű, nehezebb rajzolás
- **Változó méretű tile-ok**
  - **Nehéz kezelhetőség**
  - **Magasabb memória igény**
  - **Rugalmas vizualizáció**

13

EGY TIPIKUS TILEMAP MEGVALÓSÍTÁS…

14

Példa Tile-Map megvalósítás

- Egy kezdeti TileMap implementáció nem igényel bonyolult algoritmusokat.
- Szükség van egy CTile, és egy CTileMap osztályra.

```
class CTile{
   CTexture *m_pTileTexture;          // Texture of the tile
   unsigned int m_uiTextureIndex;     // Texture index from the tileSet
   CVector2 m_pVerts[4];              // Vertices
   CVector2 m_pTexcoords[4];          // Texture coordinates
  unsigned short index_i;             // horizontal index in the MAP
   unsigned short index_j;            // vertical index in the MAP
   bool m_bEmpty;                     // is Tile empty or not
   bool m_bCollide;                   // can player collide or not
public:
...
};
```

15

Példa Tile-Map megvalósítás

```
/// Tile Map base class
class CTileMap{
   CTile ***m_pTilemap;               // TileMap
   unsigned int m_uiSizex;            // Map horizontal size
   unsigned int m_uiSizey;            // Map vertical size

   vector<CTexture*> m_vTileTextures; // TileSet textures
   unsigned int      m_uiTileSize;    // size of the tiles
   float m_fScrollX;                  // vertical scroll
   float m_fScrollY;                  // horizontal scroll

public:
...
};
```

16

TILEMAP ELŐNYEI…

A módszer előnyei

- A megvalósítási forma számos előnnyel rendelkezik a memóriatakarékosság mellett.
- Egyik ilyen pozitívum az **ütközésvizsgálatok viszonylag egyszerű megvalósítása**.
- A Tile alapú megközelítés szintén a befoglaló dobozok technikáját használja az ütközések detektálására.
- Egy Tile pont egy box-nak felel meg egyszerűbb esetben.
  - Box-box ütközés
- Bonyolultabb programoknál pixel szintű és egyéb megközelítés is szükséges lehet.

Tile-Map BB ütközésvizsgálat

```
bool CheckTileBoundingBoxCollision(CBoundingBox2D *box){

for (int i = 0; i < m_uiSizex; i++){
      for (int j = 0; j < m_uiSizey; j++){
         if (m_pTilemap[i][j]->isEmpty() == false) {
            if (m_pTilemap[i][j]->isCollidable() == false){
               continue;
            }

            CVector2 *tilePos = m_pTilemap[i][j]->GetPosition();
            if (box->maxpoint->x < tilePos->x ||
               box->minpoint->x > tilePos->x + m_uiTileSize){
               continue;
            }
```


Tile-Map BB ütközésvizsgálat

```
            if (box->maxpoint->y < tilePos->y ||
                box->minpoint-> y > tilePos->y + m_uiTileSize){
                     continue;
                }
                return true; // Hit !!!
            }
         }
      }
      return false;
}
```

A módszer előnyei

- További előnye a **gyors útkeresés algoritmikus** megvalósítása.
- **Oka:**
  - egy Tile nem egy pixelt jelent, hanem nagyobb egységet.
  - Lehetőség van a Tile alapú útkeresésre valós időben
- **Példa:**
  - Ez tette lehetővé a korai stratégiai játékok helyes működését
  - nagy távolságokra a számítógép rögtön képes volt megtalálni a legrövidebb utat és elnavigálta az objektumot.
  - pl. Warcraft 1

21

SZÖVEGEK MEGJELENÍTÉSE…

22

Szövegek megjelenítése

- A szövegek kirajzolása a képernyőn alapvető követelmény.
- Nem grafikus alkalmazások esetében egyszerű:
  - használhatjuk az operációs rendszer karakterkészletét és megjelenítő rutinjait
- Hardveresen gyorsított szoftverek esetében ez már nehezebb
  - Mind az OpenGL mind a DirectX esetén megoldható
    - **true type** betűkészlet is
  - DirectX **- D3DXFont** beépített megoldás
  - OpenGL – nem egyszerű, nincs beépített rutin

23

Szövegek megjelenítése

- **Az OS karakterkészlet használatának előnye:**
  - a karakterek típusa, a kiírt szöveg bármikor változtatható
  - a szöveg kiírása viszonylag sok erőforrást vesz igénybe,
  - csak olyan betűtípust használhatunk, amely az operációs rendszerben jelen van.
- A mai játékszoftverek számára ez nem kielégítő!
  - Nem lehet tetszőleges formájú betűket kirajzolni
  - legtöbb esetben az úgynevezett **bitkép (bitmap fonts) alapú szövegkiíró megoldás**t választják.

24

Szövegek megjelenítése

- **Bitmap font:**
  - olyan textúra, amely tartalmazza a kiírandó betűk, illetve egyéb jelek képi megfelelőit.
    - A grafikus készíti el
  - Az elkészített képet fix méretű blokkokra bontja,
    - minden blokk egy karakternek felel meg.

25

Szövegek megjelenítése

- A megoldással tetszőleges stílusú karakterek ábrázolhatók.
- **Megszorítás:**
  - csakis olyan karaktereket tud értelmezni, amelyek szerepelnek a textúrában.
- **Használata:**
  - a képet betöltéskor szét kell darabolni az egységes karakterméret alapján
  - Szükséges egy összerendelés:
    - melyik textúra valójában melyik betű megfelelője lesz.

26

Példa összerendelésre (Android)

```
// Map to associate a bitmap to each character
private Map<Character, Bitmap> glyphs = new HashMap<Character,
Bitmap>(62);
private int width;   // width in pixels of one character
private int height; // height in pixels of one character

// the characters in the English alphabet
private char[] charactersL = new char[] { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };

private char[] charactersU = new char[] { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };

private char[] numbers = new char[] { '1', '2', '3', '4', '5', '6', '7', '8', '9', '0' };
```

27

Példa összerendelésre (Android)

```
public void drawString(Canvas canvas, String text, int x, int y) {

  for (int i = 0; i < text.length(); i++) {
     Character ch = text.charAt(i);
     if (glyphs.get(ch) != null) {
       canvas.drawBitmap(glyphs.get(ch), x + (i * width), y, null);
     }
   }
}
```

28

Szövegek megjelenítése

- A karaktereket tartalmazó textúra létrehozása plusz munkát jelent a grafikus számára
- A világhálón elérhetők szoftverek, amelyek segítségével a fontkészlet generálható.
- Általában a karakter betöltő és kirajzoló rutin saját fejlesztés egy adott szoftver fejlesztői számára
  - vannak lehetőségek, kész megoldások is.
- Pl. Freetype 2, a GLUT függvénykönyvtárak

29