I C what you did last summer

Zombisch

avagy 2D-s játékmotor a gyakorlatban

Előadó:

Gazder Bence Gazben - 2. évfolyam/info

https://github.com/gazben/Space-Blaster---X/tree/Zombisch

Alapötlet

- Nyári elfoglaltság
- Mindenki mondja hogy hogyan kéne csinálni…
  - Ideje kipróbálni igazuk van-e?
- Nem egy main.cpp-ből álló programot kell írni

//a jelenlegi main.cpp:

```
#include "game.h"
#include "Globals.h"
int main(char** argc, int argv){
        Globals::INIT();
        Game newgame;
        newgame.Run();
        Globals::freeGlobals();
        return 0;
}
```

“Játékmenet”

A játéktól még messze van egy kicsit de már látszik az akarat :)

Mit látunk?

Minden egyes négyzethálónak külön textúra ezzel később lehet pályát generálni pl.: pályaszerkesztő

Játékosunk van.
Ellenfelek vannak.

Miről lesz szó?

- mit kellene tudnia ha lenne végtelen időm
- a motor működése
  - GameState rendszer
  - Csináljuk úgy mint a nagyok
  - Miért is működik?
- mire képes most

Mi is az a “motor”?

Röviden:

- Sokszor emlegetik de sokan nemtudják mit takar.
- Egy olyan “váz” amire ha felaggatsz különböző dolgokat (új játékos, új lejátszandó videó stb.) akkor azt rendezetten és viszonylag egyszerűen tudjad megtenni (ja és mellékesen működjön is) és ne kelljen változtatni a program struktúráján.
- FŐ ALAPELV: BŐVÍTHETŐSÉG, RENDEZETTSÉG

Akkor most hogyis?

Mire is van szükségünk?

- különböző játékállapotok kezelés
- fejlesztőt segítő lehetőségek (pl.: logolás)
- jól elkülönülő szintek (absztrakció)

Gamestate rendszer

Szükség van egy átlátható rendszerre amivel a játék állapotait kezeljük.

Az állapotok legyenek egymástól függetlenek és könnyen lehessen köztük váltogatni.

Megoldás: absztrakt osztály amiket bepakolunk egy tárolóba és ha váltanunk kell akkor mindig más elemet frissítünk.

A tároló és a váltás:

```
enum GState{MOVIE, MENU, INGAME};

GState currentState;
std::vector<GameState*> gamestates;
```

Megvalósítás: (gameState.h)

```
class GameState{
public:
        virtual void Update() = 0;
        virtual void Draw() = 0;
};
```

Mi történik minden frissítésnél?

Az aktuális játékállapotot frissítjük.

Minden osztálynak van egy külön Draw és Update függvénye amit az alábbi függvények hívnak meg:

```
gamestates[currentState]->Update();
gamestates[currentState]->Draw();
window.Getwindow().display();
```

Miért jó ez?

Mert a frissítés egy eléggé nagy probléma és így le tudjuk bontani kisebb elemekre.

Videó lejátszás

- Nem triviális :(
- Az SFML alapból nem támogatja mégegy library-t kellett használni.
  - sfeMovie
- Ajánlom mindenkinek!
  - jól használható és tényleg egyszerű

Videolejátszás - Gyakorlat

- Külön gameState van neki fenttartva.
- A frissítés függvény “tolja odébb a videót”.

Kirajzolás: game -> window.Getwindow().draw(*intro);

Továbblépés:

```
if( intro->getStatus() == sfe::Movie::Status::Stopped ){
        game->currentState = MENU;
    }
```

Miért volt ez gond?

- Linker hibák mindenhol.
- Kellett a game osztályra egy pointer amit a fordító nem talált kedvére valónak

További tervek:

Hálózati játék - gondok akadtak vele log szerint van szerver gyakorlatilag meg elvitte a cica

AI - Kell egy gráf ami készen van. Az útkereső algoritmust kellene implamentálni és kész.

Hogy vehetsz részt a projectben?

github: https://github.com/gazben/Space-Blaster---X/tree/Zombisch

Mi kell a fordításhoz?

- SFML 2.1 (2.0-val is működik)
- sfeMovie
- mymath (benne van a mappában, hasznos kis matek lib)

Windows:

A fenti libraryket kell hozzáadni és fordul.

Linux:

sudo add-apt-repository ppa:sonkun/sfml-development
sudo apt-get update
sudo apt-get install libsfml-dev
sudo apt-get install libsfemovie-dev
CB-ben hozzáadod a projekthez és fordul.

Köszönöm a figyelmet!4négy

Akik még mindig figyelnek azoknak itt egy cuki wombat: