Multiplatform szoftverfejlesztés

Bevezetés

Tárgy adminisztráció

https://portal.vik.bme.hu/kepzes/targyak/VIAUMA04/

- Előadó: Rajacsics Tamás (raja@aut.bme.hu)
- Előadás/gyakorlat péntek és páros csütörtök
  - Csütörtök második héten lesz
  - Gyakorlatot lehet követni saját géppel
    - VS2019
    - Visual Studio Code
    - Qt Creator

Tárgy adminisztráció

- ZH
  - Minimum elégséges az aláíráshoz
  - Teljes C++ anyag (JS nincs benne)
  - Mindig az előadás idejében
  - Amikor odaérünk az anyaggal: kb. április vége
  - PótZH: ZH után 2 héttel
  - PótPótZH: pótlási héten
    - Ez egy elővizsga is egyben, akinek megvan az aláírás
- Vizsga minimum elégséges
- Végső jegy: ZH * 0,4 + Vizsga * 0,6

Tartalom

- Multiplatform célok
- Programozási környezetek
- Megoldások
  - HTML, CSS, JS
  - C++
  - Java
  - .NET
  - Xamarin
  - MonoGame
  - Unity, Unreal, CryEngine

Multiplatform célok

- Emberi erőforrás kezelés
  - Nem kell minden célplatformhoz fejlesztő
  - Könnyebb cserélni/pótolni az embereket
  - A teljes gárda jobban egyben tartható
    - Ez lehet közösségi cél is
  - Egy dologhoz érteni jobban lehet, a cég/csapat erősebb kompetenciát tud építeni

- Célközönség elérése
  - Olyan platformokra is ki lehet adni az appot, ami nincs benne a kiírásban (pl. Windows Phone…)

Multiplatform célok

- Költségek csökkentése
  - Minden multiplatform technológia olcsóbban hoz ki 2 platformot, mintha megírnánk mindkettőre
- Gyorsabb fejlesztés
- Támogatás
  - Hibát csak egyszer kell javítani, és mindegyikben megjavul
  - A platformok változása esetén a technológia változik alattunk
    - Esélyes, hogy nem kell hozzányúlni a kódhoz új platform verzió esetén
  - Új verzió esetén könnyebb zökkenőmentesen frissíteni

Multiplatform célok

- Újrafelhasználható kód
  - Idő és pénz
- Egységes dizájn
  - Ha nem cél, akkor meg lehet csinálni többfélére is
- Korszerű
  - Egyre nagyobb a multiplatform nyelvek/technológiák aránya

Multiplatform problémák

- Funkcionalitás hiánya
  - Nem tud mindenhez hozzáférni, amihez natív társa igen
    - Ezen valamikor lehet segíteni, de körülményes
- Technológia gyenge pontjai és hibái
  - Nem tudunk segíteni rajta általában

Mire jók a programozási környezetek?

- Hardverközeli programozás problémás
  - CPU, gépikód, assembly
  - I/O portok, eszközök, időzítés
- Cél
  - Absztrakciós szint növelése
    - Magasabb szintű fogalmak bevezetése (nyelvi és könyvtárbeli)
    - Szöveg, grid, rekord, reláció, lambda, stb.
  - Produktivitás növelése
    - Nem azonos az absztrakciós szinttel

Nyelvi konstrukciók

- Imperatív, vagy deklaratív megközelítés
- Metódusok / függvények
  - Változó paraméter lista
  - Névtelen, beágyazott
- Lokális változók (capture)
- Modulok – láthatóság
- OO koncepciók
  - Öröklés, polimorfizmus, virtuális metódusok
- Rekurzió, lista kezelés, stb.

Nyelvi konstrukciók implementációja

- Adatok ábrázolása a memóriában
  - Egész szám, még egy nyelven belül sem mindig azonos a jelentés (16 bit / 32 bit / 64 bit)
  - Szöveg (Karakter kódolás? Hossz/végjel?)
  - Lebegő pontos számok
- Típusok
  - Szerződés/specifikáció a különböző részek között
- OO koncepciók ábrázolása a memóriában
  - Virtuális függvénytábla
  - Futásidejű típus információ (RTTI, reflection, typeof)

Futásidejű konstrukciók

- „this” pointer – JavaScript vs. C++/C#
- Ki takarítja a vermet?
  - Pascal vagy C hívási konvenció
- Kivételkezelés
- Memória kezelés
  - Manuális vagy sem? Pinning?
- Debuggolás, Edit&Continue, …
  - Hot/Live reload nem Edit&Continue
  - Hot Module Replacement az

Megvalósítások 1

- A nyelvi szint megvalósítása „összenő” az infrastruktúrával a gépi kódot generáló fordítók esetében (C, C++, pascal, prolog, stb.)
  - Kód generálásakor meghatározza, hogy milyen hívási konvencióval, szám ábrázolással stb. dolgozik
- A köztes nyelvet használó megoldásoknál (Java, .NET, VB stb.) a futtató környezet határozza meg a konvenciókat
  - Interpretált
  - JIT fordító

Megvalósítások 2

- A megvalósítások általában erősen kötődnek egy adott operációs rendszerhez
  - Tipikusan C API hívások
  - Bootstrapping más
  - Paraméterezés más
  - Koncepciók különbözhetnek (thread, fork)

Nincs együttműködés

- A programozási környezetek zárt rendszerek
- Se a környezetek se a programnyelvek nincsenek felkészítve az együttműködésre
  - Általában egy C jellegű külső hívási megoldást támogatnak (pl. Java)

- Miért?
  - Elvi nehézségek (nyelvi koncepciók)
  - Technikai kihívások (pinning)
  - Motiváció hiánya

Multiplatform (Cross-platform)

- Nyelv
- Platform támogatás
- Eszközök
- Dokumentáció, Tutorial, Fórum, stb.
- Alapkönyvtárak (STL, BCL, stb.)
- UI
- Egyéb könyvtárak

Java – 1995

- Modern koncepciók
  - Automatikus memória kezelés (lisp, ‘60)
  - Egyszerű OO (egyszeres öröklés, COM)
  - C jellegű tömör szintaktika
  - Nincs szabvány
- „Írd meg egyszer, futtasd mindenhol.”
  - Szép cél és szinte sikerült
- A nyelv és a futtató környezet összenőtt
  - Kb. 300 nyelv készült hozzá kutatási jelleggel
  - Nehéz más nyelvi környezetekkel összeilleszteni

Miért született meg a .NET?

- A ’90-es évek vége: VB és C++ nem elég
  - Nincs produktív, modern nyelvi megoldás
- J++: MS saját Java implementációja
  - Új konstrukciók (COM): például események, tulajdonságok
  - Nyílt levelek, pereskedés – zsákutca
- Internet térnyerése
  - Minden mindennel össze lesz kötve
  - Nem asztali számítógép eszközök megjelenése

.NET – 2002

- Modern futtatókörnyezet
  - Automatikus szemétgyűjtés, biztonság stb.
- Új, korszerű, produktív nyelv
  - C#: kompromisszumok nélkül
- Több nyelv támogatása
  - Kompatibilitás: VB, C++, COM, …
- Platformfüggetlenség
  - Ekkor még csak cél

Szkript nyelvek

- Ha az interpreter elérhető más platformon, akkor általában a szkriptek is futtathatóak ott
  - Egy értelmező elkészítése egy új platformra kisebb költségű, mint egy fordító
- JavaScript, ActionScript, python, perl, tcl, php, ruby, …
- Gyors fejlesztés, kis kódbázis, prototipizálás
- Kiterjesztés: játékok (WebGL), tervezőprogramok stb.

Xamarin

- .NET, Mono (legújabb verzió hamar bekerül)
  - Saját CLR, JIT compiler
- Platform: Android, iOS, UWP
- Közös UI: Xamarin.Forms
- Ingyenes (2016 március 31-től)
- iOS fejlesztéshez kell Mac (Compile, Deploy, Debug)
- Szolgáltatások: Test Cloud, Crash Monitor

MonoGame

- XNA 4 API
  - Microsoft XNA kifutott
    - nincs már rá támogatás
- Az egyik legmagasabb szintű SDK
- Nem játékmotor, azt meg kell írni benne

MonoGame

- Platform támogatás
  - iOS, Mac OS, PlayStation Mobile, Android, Linux, BSD, Ouya, Windows Phone 8, Windows Store, Windows Desktop
  - iOS, Android: Xamarin
  - Windows, Windows Phone, Windows Store: SharpDX (ingyenes DirectX C# API)
  - OpenGL és OpenGL ES máshol (OpenTK)

Unity 3D

- Játékmotor (nem sima keretrendszer)
- Scriptelhető (programozható)
  - C#
  - .NET 4.7 (C# 6)
  - Játéklogika és sok minden más
- Plugin
  - C++ (vagy bármi, ami natív kódot hoz létre)
  - Scriptből hívható
  - Renderbe is bele lehet avatkozni
  - Platformonként meg kell írni, 32/64 bitre is

Unity 3D

- Ingyenes
  - Pro: $125/hónap
- Platform támogatás
  - Windows Phone, BlackBerry, Android, iOS, PlayStation, Web, Linux, Mac OS, Windows, Windows Store, Xbox, WiiU, PS Vita, PS Mobile
- Magas szintű támogatás

CryEngine/Unreal Engine

- Játékmotorok
- Nem ugyanaz megy mobilon
- CryEngine
  - Platform támogatás: Windows, Linux, PlayStation, Xbox, WiiU, iOS, Android
  - Ár: Ingyenes
- Unreal
  - Platform támogatás: Windows, Linux, PlayStation, Xbox, iOS, Android, Oculus Rift, OS X, HTML 5
  - Ár: 5% a teljes bevételből $1M felett

Kérdések?