Inovace přednášky byla podpořena v rámci projektu OPPA CZ.2.17/3.1.00/33274 financovaného Evropským sociálním fondem a rozpočtem hlavního města Prahy.

Praha & EU: investujeme do Vaší budoucnosti

Programování počítačových her

Práce v týmu, rozdělení rolí, práce na větším projektu

Obsah

- Kdo je to programátor?
- Nástroje pro týmovou práci
- Specifika většího projektu
- Programování v praxi
- Middleware
- Skriptovací jazyky

Kdo je to programátor?

aneb Není to jen o renderování.

Role

- Vedoucí
- Architekt
- Senior
- Junior

- Programátor vs. Kodér

Specifika

- Rozmanitost
- Složitost

- Matematika
- Datové struktury
- Algoritmy

- Důraz na bezpečnost a rychlost kódu

Náplň práce

- Návrh řešení problémů
  - Analýza
- Implementace
  - Vlastní psaní kódu
- Code review
  - Nikdo není neomylný
- Opravy chyb / Optimalizace
  - Debugging

Oblasti práce

- Jádro
  - Každý chce pracovat na grafice.
- Simulace
  - Skoro každý chce pracovat na fyzice.
- Gameplay
  - Nikdo nechce dělat práci designéra.
- Nástroje
  - Je to vůbec ještě o hrách?

Výsledek práce

- Renderer
- Datové struktury
  - interakce
- Playground pro designéry
- Nástroje pro grafiky

- Kdo má poslední slovo?
  - Programátor vs. Designér

Nástroje pro týmový vývoj

Co používají profesionální týmy

Nástroje

- IDE
- Compiler a linker
- Profiler
- Správa verzí (Version control)
- Správa dat (Asset management)
- Bugtracker
- Správa úkolů
- Dokumentace/Wiki

IDE

- Editor, compiler, debugger
- MS Visual Studio
  - nejrozšířenější
  - užitečné pluginy
  - nejlepší debugger
  - používá se i pro Xbox i pro PS3
  - na druhou stranu podporuje závislost na kompilaci
- Linuxové editory
- Ostatní (Eclipse, Bloodshed Dev-C++)

Profiling a tuning

- Měření rychlosti
- Kontrola paměti
- Komerční projekty
  - VTune, Gprof, Telemetry
  - Memory Validator, BoundsChecker, Valgrind
- Nejlepší je vlastní
  - Profiler: Sean Barrett, Game Developer 08/04
  - Vlastní memory manager

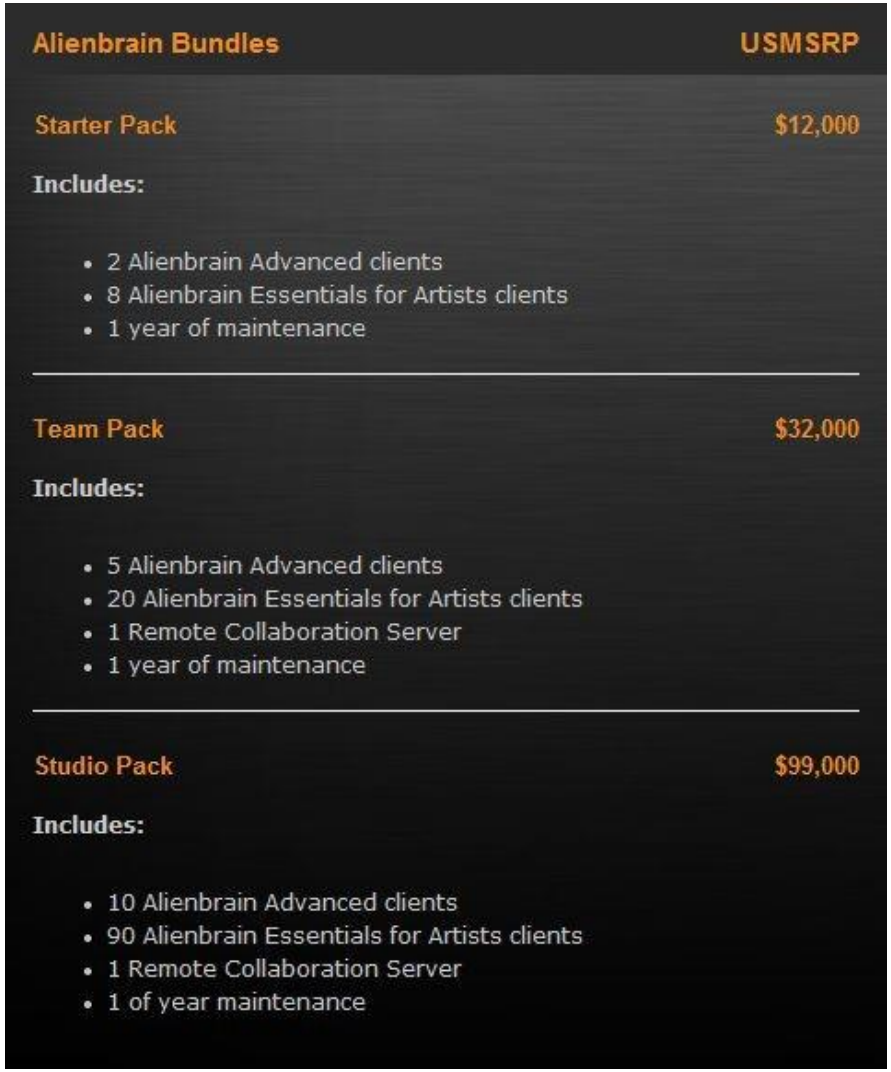
Správa verzí

- Řešení
  - CVS, SVN, git, Mercurial, SourceSafe
- Aspekty
  - zamykání souborů (check in/out)
  - podpora binárních dat
  - srovnání lokálně nebo na serveru
  - databáze nebo soubory

Správa dat

- Data (assety)
  - Modely, textury, zvuky, animace, skripty
  - Zdrojová data
    - 3DS Max, Photoshop, Sound Forge, ...
  - Nativní data
    - 'čitelná' pro používaný engine
    - nedají se přímo editovat
- Speciální, drahé systémy
  - Alien Brain, Perforce
- Systémy správy verzí

Správa dat - Alienbrain



Alienbrain Bundles	USMSRP
Starter Pack	**$12,000**
Includes:	
• 2 Alienbrain Advanced clients • 8 Alienbrain Essentials for Artists clients • 1 year of maintenance	
Team Pack	**$32,000**
Includes:	
• 5 Alienbrain Advanced clients • 20 Alienbrain Essentials for Artists clients • 1 Remote Collaboration Server • 1 year of maintenance	
Studio Pack	**$99,000**
Includes:	
• 10 Alienbrain Advanced clients • 90 Alienbrain Essentials for Artists clients • 1 Remote Collaboration Server • 1 of year maintenance	

Správa dat - Perforce

Number of Users	Per User License Fee	Per User License Fee + Standard Support ($160 per user)
Users 1-20	$740 per user	$900 per user
Users 21-50	$690 per user	$850 per user
Users 51-100	$640 per user	$800 per user
Users 101-250	$590 per user	$750 per user
Users 251-500	$540 per user	$700 per user
Users 501-1,000	$480 per user	$640 per user
Users 1,001-2,500	$420 per user	$580 per user
Users 2,501-5,000	$360 per user	$520 per user
Users 5,001-10,000	$300 per user	$460 per user
Users 10,001+	$270 per user	$430 per user

Ostatní

- Bugtracker
  - Mantis, Bugzilla
- Dokumentace
  - Doxygen, Sandcastle
- Wiki
  - Twiki
- Integrované řešení
  - Trac
  - JIRA

Specifika většího projektu

Velikost je rozdíl kvality

Specifické oblasti

- Povšechně jde o zcela subjektivní výběr
  - vývojové prostředí
  - psaní kódu
  - projekt a projekt v MSVC
  - stavový automat
  - memory manager
- Všeho s mírou

Vývojové prostředí (1)

- Integrated Development Environment
  - Microsoft Visual Studio
  - Rational Software Architect
  - WxDev-C++
  - C++Builder
  - Eclipse
  - NetBeans
  - Flash Developer
- Neměnit během vývoje projektu

Vývojové prostředí (2)

- Co největší důraz na usnadnění práce
- Užitečné vlastnosti
  - uživatelský příjemný debugger
  - syntax highlight
  - kontextová nápověda
  - prohledávání/nahrazování
  - rychlý a kvalitní kompilátor
    - multiprocesorová kompilace
  - možnost tvořit konfigurace projektu
    - debug/release · alpha/beta/gold
  - pre/post build kroky

Psaní kódu

- Lidé si musí být schopni číst navzájem kód
  - dodržování Coding standardu
  - sdílení a šíření informací
  - Odhalování chyb
- Wrappery kolem systémových funkcí
  - systémové funkce (např. strcpy) nepoužívat přímo
  - zjednodušuje portování
- Standardní makra
  - délka pole, preprocesorová makra, aserty, profiler

Dokumentace

- Skoro nikdy nepíšete kód sami pro sebe...
  - … a když ano, za 2 měsíce nevíte, co jste napsali
- Komentovat? Komentovat!
  - předpokládejte inteligenci čtenáře
  - složitější konstrukce si žádají komentář
  - doxygen
- Externí dokumentace
  - analýza a popis složitých implementačních celků
  - popis řešení
  - popis použití

Projekt jako projekt v MSVC (1)

- Čas na build projektu
  - Distribuovaná kompilace
    - Incredibuild ($500)
  - Inkrementální linkování
    - Chyba ve VS2005
- Méně *.cpp
- Rozdělení na *.lib
  - Má svá pro i proti
    - Umožňuje např. jinou utility library pro nástroje a pro engine
  - Komplikuje linkování

Projekt jako projekt v MSVC (2)

- Organizace souborů
  - přehlednost
  - snadná orientace
  - of> filename.cpp
- Addony
  - Visual Assist
    - hledání souborů
    - syntax highlight
    - doplňování
  - Ankh SVN
    - annotate

Hlavičkové soubory

- Problém:
  - Jednoprůchodový překladač → forward deklarace → olbřímní textový soubor
  - Deklarace třídy na jednom místě → private funkce a proměnné ve stejném souboru jako public → změna private části vynutí kompilace všude

Jak includovat

- Includy v *.h souborech?
  - Vysoce praktické
  - Includovat jen to, co skutečně potřebuješ
  - Je lepší mít .h menší, i když bude includovat jiná .h
- Jiné tipy
  - Jeden class může být implementován ve více souborech
    - přepínání mentálního kontextu programátora je drahé
    - může používat jiná .h
  - Include guard vs. #pragma once

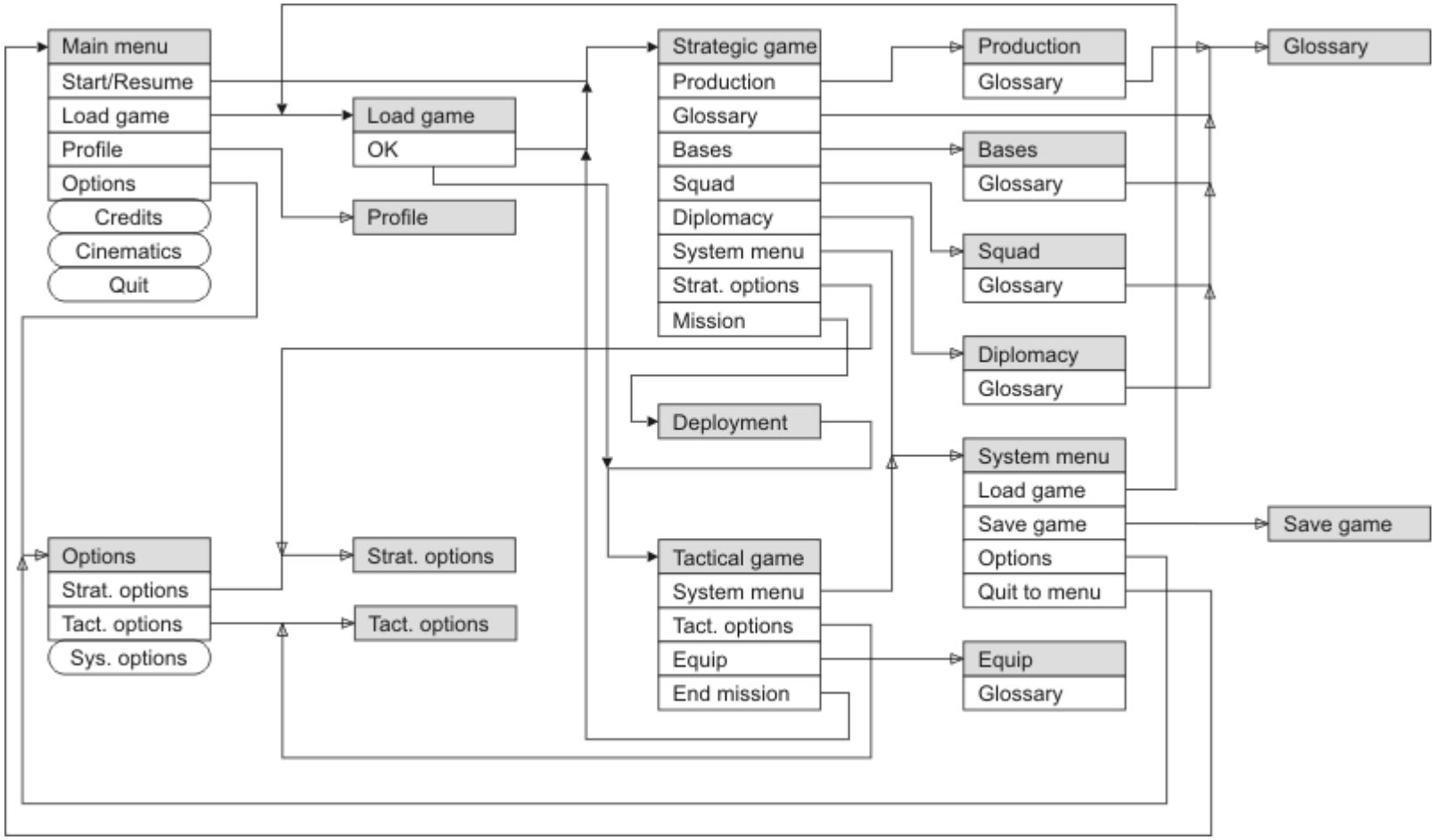
Propojení .h → .cpp

- Předkompilované headery
  - Co do nich?
  - Pomůže to?

Stavový automat

- Aplikace je vlastně stavový automat
- Na začátku projít možné stavy a definovat přechody
- Neřídit aplikaci okny ale okna aplikací
  - Stav aplikace by neměl záviset na tom, které okno je zrovna nahoře
- Jednodušeji se rozšiřuje a adaptuje na zvláštní situace

Stavový automat – příklad



Memory manager

- Několik paradoxů:
  - Konzole a Windows: jiná motivace, stejný výsledek
  - Zásadní rozhodnutí, 'neviditelné' důsledky
  - Nemoderní, ale populární
- Výhody vlastního memory manageru:
  - Je lepší alokovat staticky
  - Vlastní 'sesypávání' paměti
  - Možnost plnit paměť vlastním vzorkem
  - Kontrola memory leaků

Části hry

- Core
  - Grafický engine
  - Fyzikální engine
  - Zvukový engine
- Hra
  - Entity
  - AI
  - UI

Programování v praxi

Co je dobré vědět

Obecné schopnosti

- Programátorský způsob myšlení
  - Analytický přístup k řešení problémů
  - Schopnost vybrat optimální řešení
  - Znalost syntaxe C++ není vše
  - Vysoká škola
- Programátorská rutina
  - Intuitivní psaní kódu
  - Rychlá orientace
  - Zkušenosti z praxe

Základní znalosti herního vývojáře

- Datové struktury
- Objektově orientované programování
- Algoritmy
- Lineární algebra
- Problematika velké 3D scény
- Problematika síťového hraní

Datové struktury

- Co jsou zač
  - Linked list/Array
  - Binární strom
  - Mapa
- Kdy se hodí
  - Overhead
- Složitost
  - Základy
  - Nemluvíme o milisekundách či FPS
- Vybrat správně je umění

Objektově orientované programování (1)

- Dědičnost
  - Construtor/Destructor
    - Pořadí volání
  - virtual
  - Vícenásobná dědičnost
- Public/Protected/Private
  - Udržovat pořádek
- Šablony
  - STL nestačí
  - Debugging

Objektově orientované programování (2)

- #define vs const
- Statické proměnné
- Konstantní metody

```
class Person
{
  public:
    ...
    char* GetName() //kolikrát patří 'const' na tento řádek?
    {
        return m_szName;
    };
  private:
    char* m_szName;
};
```

Algoritmy

- Třídění
  - Nemluvíme o funkci sort()
  - Jaké jsou třídící algoritmy
    - Teorie
  - Jaký algoritmus jste implementovali
    - Praxe
- A*
- Genetické algoritmy

Lineární algebra

- Pracujeme v prostoru
- Dot product
  - Skalární součin
- Cross product
  - Vektorový součin
- Transformační matice
  - Inverzní matice

Problematika velké 3D scény

- Optimalizace scény
  - Terén
  - Modely
  - Simulace
- Velké rozměry
  - Problémy s floaty…
    - … a dokonce s doubly

Problematika síťového hraní

- Server/Client
- Dedicated server
- Optimalizace posílaných dat
- Dead reckoning
  - Interpolace

Middleware a smrt algoritmů

Všechno už někdo naprogramoval

Co je to middleware

- Cizí kód v naší aplikaci
- Klasifikace
  - Podle stupně integrace (od zdrojáků po OS)
  - Podle účelu (fyzika, AI, ...)
- Middleware je outsourcing pro programátory

Middleware reloaded

- Pole vlastního programování se zmenšuje...
  - Abstrakce hardwaru
  - Delegace odpovědnosti
    - toto je opravdový 'middleware'
  - Integrace
- ...a současně prohlubuje
  - Zvětšuje se podíl tvůrčí práce
  - Komplexnější mechanismy, sofistikovanější světy
- Totální middleware?

Fyzikální middleware 1/2

- Využití fyziky
  - ragdoll, animace smrti
  - ničení objektů, což obnáší další zásadní problémy
  - animace vody a kapalin
  - brodění se krabicemi
- Kritéria pro hodnocení fyz. mw
  - rychlost, přesnost (stabilita), API
  - rozsah – tuhá tělesa, kapaliny, ragdoll atd.

Fyzikální middleware 2/2

- Problémy s fyzikou
  - HW akcelerace dává výsledky z minulého snímku, navíc vyžaduje dvojí data
  - Musí se simulovat kontinuálně i na místech, které hráč nevidí
  - Často je jen fyzika pro fyziku, herní význam je nejasný nebo žádný
- Není jasné, zda je to spíše grafický nebo spíše designový efekt

Fyzikální mw – příklady

- Havoc (Intel)
  - drahý?
- Nvidia PhysX
  - není Meqon
  - HW akcelerace – konkurence od graf. karet a multicorů
- ODE
  - takřka použitelné
  - nemá sweep test (CCD)
- Bullet
  - původně z Blenderu
  - má CCD a používá ODE solver

AI middleware

- Path-finding, steering, konečné automaty
  - PathEngine
- Komerční = drahé
  - AI.Implant (Presagis)
  - Kynapse (Autodesk)
  - xaitment
    - xaitMap, xaitControl
- Akademické
  - ScriptEase (Java, Never Winter Nights)

Engine jako middleware 1/3

- Co je to engine
  - Správa scény
    - Hierarchie objektů, světel, kamer, render targetů
    - Viditelnost, kolize
    - Materiály, shadery
  - Řešení osvětlení a stínů
  - Animační systém
  - Podpora skriptování
  - Formát dat
  - …

Engine jako middleware 2/3

- Všechna studia v ČR používají vlastní engine
- Proč mít vlastní engine:
  - je to naše (cena)
  - je to naše (vyznáme se v tom a dokážeme to upravit a vylepšovat)
  - je to naše (nejvhodnější pro náš typ hry)
- Vývoj vlastního enginu trvá dlouho
  - 2 – 4 roky
  - 1 – 6 programátorů

Engine jako middleware 3/3

- Kritéria pro výběr enginu
  - Komerční vs. OS
  - Workflow a nástroje pro něj
    - nástroje jsou klíčové (editor, pluginy)
  - API
  - Podpora skriptování (integrace s herním kódem)
    - může být i jen skriptování
  - Multiplatformnost
  - Výkon
    - multithreading
    - velikost scény

Engine jako middleware 4/4

- Výhody
  - Multiplatformnost
  - Zbavení se starostí s grafickou kartou
  - Ušetření času a peněz
- Nevýhody
  - Originální koncepce
    - vnucování typu hry
  - Dokumentace
  - Podpora

Engine mw – příklady

- Komerční enginy
  - Doom3, Half Life, Unreal…
- OS enginy
  - Ogre: rozumný kompromis mezi složitostí a funkčností
  - Irrlicht: hezký menší, ale funkční
  - CrystalSpace: příliš velký pro své vlastní dobro
  - XEngine, atd…

Ostatní middleware

- Audio a Video
  - Bink/Miles
  - Ogg/Theora/Vorbis/FLAC
  - OpenAL
- Grafika
  - FreeImage
  - FreeType
- Ostatní
  - zLib

Jak zvolit middleware?

- Čím blíže standardu, tím lépe
  - Stabilní, neměnná specifikace
  - Popularita, tj. počet uživatelů
- Opatrně na závislost na proprietárním řešení
  - Open Source není automaticky správná odpověď
- Postupujte racionálně:
  - Nebojte se cizího kódu
  - Nebojte se delegovat zodpovědnost

Skriptovací jazyky

V čem se píší hry

Skriptování obecně 1/3

- Překlad do bajtkódu
  - Virtuální procesor
    - může se zrychlit generátorem assembleru podle platformy
    - výhoda virtuálního procesoru je vlastní scheduling, správa stacků
  - Překladač je součástí jazyka
    - Pro vlastní jazyk lze použít OS lexer a parser (lex, yacc, bison)

Skriptování obecně 2/3

- Multitasking
  - Kooperativní – coroutines (Lua)

```
int sound_handle = GetBigFish().Say("thing1");
if(sound_handle > 0)
    coroutine.yield("sound_handle", sound_handle);
```

    - jednodušší pro programátory, zodpovědnější pro skriptaře
  - Preemptivní
    - Vlastní thread manager
    - Těžší naprogramovat, skriptování není jednodušší
    - Aplikace je odolnější

Skriptování obecně 3/3

- Propojení s aplikací
  - Voláním nativního kódu aplikace
    - jména funkcí se při překladu vyhledají v tabulce

```
if (!IsInArea(soldier1,area1)
{
    SetCommandMoveToArea(soldier1,area1);
    yield(1);
}
```

  - Voláním eventů
    - pro určité eventy se zaregistrují funkce, které se mají volat

```
SetCommandMoveToArea(soldier1,area1);
OnEvent UnitInArea(soldier1,area1) { ... }
```

Skriptovací příklady 1/2

- Lua
  - Jednoduchá syntaxe
  - Není debugger
  - Rozporuplná rychlost
- Python
  - Nemáme praktické zkušenosti
  - Expresivnější syntaxe
  - Prý pomalejší než Lua

Skriptovací příklady 2/2

- AngelScript, GameMonkey
  - OpenSource
  - Integrovatelné do C++, C++ podobná syntaxe
  - Nemáme praktické zkušenosti
- Unreal script, Enforce script
  - (Enforce engine = Black Element Studio)
  - 2,5 MB zdrojáků pro ES se zkompiluje za 0,5s
  - C++ podobná syntaxe
  - Rychlejší interpretace

Výhody skriptování 1/2

- Bezpečnější
  - runtime kontrola herního kódu
  - kontrola všech operací
- Jednodušší ukládání hry
  - ukládají se všechny objekty skriptu
  - handle pointer (místo C-pointeru)
- Debugování
  - v případě pádu je kompletní informace o stacku, stavu atd. i v release verzi

Výhody skriptování 2/2

- Čistší design, rychlejší vývoj
  - herní programátoři jsou oddělení od enginu
  - hru není nutné kompilovat a linkovat s enginem
  - i v releasu jsou dostupné všechny debugovací informace
  - viz kapitola o middlewaru
- Jednodušší a lepší multitasking
  - virtuální stroj může skript libovolně přerušit a pouštět jiné thready aplikace