Specifikace projektu Ocerus

Tým

Vedoucí:
Ondřej Sýkora (ondrasej@centrum.cz)

Členové:
Michal Čevora (macjariel@gmail.com)
Lukáš Hermann (lukas.hermann@seznam.cz)
Ondřej Mocný (hardwire@volny.cz)
Tomáš Svoboda (svoboda77@volny.cz)

Úvod

Průmysl počítačových her od svého vzniku stále roste a v posledních letech již dosahuje mohutnosti toho filmového. Dnes tento trh ovládají především ,,velkorozpočtové" 3D akční hry, zaměřené především na pravidelné hráče. Ale s tím, jak se počítačové hry dostávají do povědomí širší veřejnosti, roste zájem o jednodušší hry, určené především svátečním hráčům – tzv. ,,casual hry". Tyto hry jsou obvykle vytvářeny menšími vývojářskými studii s nižšími rozpočty. Vzhledem k počtu potenciálních hráčů však jde o velmi lukrativní část trhu, která je objemem srovnatelná s prodeji her pro pravidelné hráče.

Hry pro sváteční hráče jsou ve srovnání s hrami pro pravidelné hráče jednodušší a menší nejen pokud jde o rozsah, komplexnost a celkovou náročnost hry, velmi odlišné jsou také nároky na technické zpracování. Oproti 3D zobrazení obvyklému u velkorozpočtových her zde tvůrci častěji využívají 2D nebo 2.5D zobrazení, které je pro nezkušené hráče přehlednější. Pojem 2.5D znamená, že herní svět je v principu dvourozměrný a také většina herní grafiky je 2D, nicméně některé herní objekty mohou byt renderovány jako

3D objekty. Herní svět je zobrazený buď přímo z boku, nebo přímo z vrchu.

V současné době na trhu sice existují kvalitní a snadno dostupné nástroje pro vývoj 3D her, ale pro velkou část casual her je přirozenější 2D nebo 2,5D zobrazení. Navíc vývoj takových her je snazší, a tedy i levnější. Tyto nástroje nejsou tedy pro casual hry velmi vhodné.

Popis projektu

Cílem tohoto projektu je navrhnout a implementovat herní engine pro 2.5D počítačové hry pro PC. Engine bude navržen tak, aby ho v případě zájmu vývojářských studií bylo možné snadno přenést i na jiné platformy, především na současné herní konzole. Těžiště projektu je ve vytvoření kvalitních a snadno použitelných editačních nástrojů, které budou přímo integrované v herním engine. V tom spočívá pravděpodobně i hlavní odlišnost od jiných podobných nástrojů.

Výsledný produkt má v sobě spojovat engine pohánějící hru s editačními nástroji. Pokud je program spuštěn v editačním režimu, uživatel může měnit herní svět a umisťovat do něj herní objekty, zároveň ale lze snadno přepnout do herního režimu a provedené změny testovat. Herní objekty lze vytvářet pouze z předpřipravených bloků (komponent). Editací herního objektu se pak rozumí změna vlastností komponent objektu. Uživatel může přidávat vlastní funkčnost pomocí zabudovaného skriptovacího jazyka. Editor skriptů není součástí tohoto projektu.

Herní engine obsahuje nástroje pro automatickou správu dat jako jsou grafické nebo zvukové soubory. Přímo v editoru je možné procházet soubory s daty a používat je v komponentách, které ke své práci data vyžadují (např. přiřazovat obrázky postavičkám v herním světě). Engine se sám postará o načtení dat ve chvíli, kdy budou potřeba, a jejich opětovné uvolnění, pokud bude docházet paměť.

Části aplikace

Engine se bude skládat z následujících částí.

Správa aplikace

Zajišťuje správu stavu aplikace, přepínání mezi herním a editačním módem a hlavní smyčku programu. Uvnitř této části dochází k aktivaci dalších částí aplikace (aktualizace stavu herních objektů, renderování grafiky, apod.).

Renderer

Jedná se o vrstvu zajišťující renderování grafiky a poskytující platformově nezávislé nástroje pro vykreslování ostatním částem engine. Renderer bude implementován celý v rámci projektu, s využitím rozhraní OpenGL pro interakci s grafickým hardware a knihovny SDL pro interakci s operačním systémem. Rozhraní rendereru bude navrženo tak, aby umožnilo případnou změnu těchto technologií.

Správa herních objektů

Herní objekty se skládají z komponent, ke kterým se přistupuje pomocí vlastností. Tato část bude zprostředkovávat komunikaci mezi herními objekty navzájem a mezi herními objekty a zbytkem systému. Zároveň se bude starat o vytváření a rušení objektů a jejich komponent. Součástí bude i sada předpřipravených komponent pro vytváření vlastních objektů (komponenty pro reprezentaci spritu, fyzikálního těla, ...).

Při implementaci komponent budou použity knihovny poskytující požadovanou funkcionalitu, např. Box2D pro komponentu zajišťující fyzikální reprezentaci, AngelScript pro skriptovou komponentu, ...

GUI

Engine bude obsahovat jednak uživatelské rozhraní editoru a jednak bude poskytovat prostředky pro vytvoření uživatelského rozhraní ve vytvářených hrách. Pro implementaci obou variant uživatelského rozhraní bude využita knihovna CEGUI, která ovšem pracuje na velmi nízké úrovni. Cílem vrstvy uživatelského rozhraní je poskytnout vhodné abstrakce pro práci s objekty knihovny CEGUI, např. ze skriptovacího jazyka.

Editor

Základem projektu je editor pro vytváření her – ten umožní editaci herního světa a objektů v něm obsažených – jejich pozice, komponent a vlastností. Uživatelské rozhraní editoru bude využívat z velké části založené na funkcích poskytovaných vrstvou uživatelského rozhraní.

Správa scény

Pro usnadnění editace bude mít uživatel možnost uspořádat si herní objekty ve scéně do stromové struktury. Toto bude implementováno jako vhodná datová struktura v rámci editoru.

Skripty

Skripty jsou procedury kompilovatelné a spustitelné za běhu aplikace a jde o základní způsob implementace herní logiky v engine. Tato část bude implementována jako nadstavba nad knihovnou AngelScript. Její hlavní funkcí bude spravovat skriptové moduly a umožnit jim přistupovat k nativním částem systému (binding tříd a funkcí).

Správa vstupních zařízení

Pro správu vstupních zařízení bude použita knihovna OIS, nad kterou bude vytvořena vrstva zprostředkující tuto funkčnost ostatním částem systému. Prací se vstupním zařízením rozumíme reakce na události (stisknutí klávesy, pohyb myši, apod.) a dotazování na aktuální stav zařízení (seznam stisknutých kláves, ...).

Správa paměti

Hry mají specifické nároky na správu paměti, protože s ní často pracují nestandardním způsobem. Proto je dobré mít pod kontrolou veškerou dynamickou alokaci a dealokaci paměti pro možnost její kontroly. Zároveň jsou vhodné speciální alokátory, např. pool alokátor pro malé objekty.

Ladicí prostředky

V případě vydání hotové hry se může stát, že na cílovém počítači nebude správně fungovat. V takovém případě se hodí mít možnost vygenerovat ladicí výstup, který pak vývojář může dekódovat. V tomto případě se rozumí stack trace a logy. Stack trace zajistí knihovna DbgLib.

Pro vyladění výkonu systému je nutné mít možnost kód profilovat. Protože hry jsou real-time aplikace, bude nutné použít profiler, který je možné aktivovat/deaktivovat při běhu aplikace. Proto bude použita knihovna RTH-Profiler, která toto umožňuje.

Testování bude probíhat také pomocí automatických Unit testů. Ty budou implementovány pomocí knihovny UnitTest++.

Cílová platforma

Vzhledem k vývoji trhu s nízkorozpočtovými hrami v posledních letech je nutné počítat s možnou konverzí enginu na next-gen herní konzole. Z tohoto důvodu musí být kód projektu napsaný v C/C++ a renderer musí být připraven na podporu jak OpenGL, tak DirectX. Všechny použité knihovny musí být na konzole přenositelné.

Knihovny

Pro urychlení vývoje je vhodné při implementaci projektu použít co nejvíce hotového kódu, ať už ve formě celých knihoven nebo jen jejich částí (podle licenčních podmínek). Zde je uveden předběžný seznam knihoven a způsobů jejich využití v projektu:

- AngelScript – práce se skriptovacím jazykem AngelScript
- Boost – pomocné datové třídy, algoritmy a C++ utility
- Box2D – 2D real-time fyzika
- DbgLib – nástroje pro real-time ladění a crash reporty
- Expat – XML parser
- FreeType – zobrazování freetype fontů

- OIS – zpracovávání událostí ze vstupních zařízení
- PCRE – parsování regulárních výrazů
- RTHProfiler – interaktivní real-time profilování kódu
- SDL – usnadnění renderování grafiky
- SILLY – načítání textur různých formátů
- SOIL – načítání textur různých formátů
- UnitTest++ – framework pro unit testy