

Gymnázium Uherské Hradiště

Maturitní projekt z informatiky

Multiplatformní knihovna Halcyon

1 Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně pouze s využitím uvedené literatury a pramenů. Všechny doslovné citace jsou v textu řádně označeny s uvedením zdroje.

V Uherském Hradišti dne 2. 4. 2024

.....

Petr Šácha

2 Anotace

Halcyon je multiplatformní knihovna pro vývoj her, založena na SDL. Aplikuje výhody C++ v obalové vrstvě, která neztrácí rychlost, přičemž zvyšuje přehlednost a bezpečnost. Má také zajímavé konfigurační možnosti.

Obsah

1 Prohlášení.....	2
2 Anotace	3
3 Teoretická část	5
3.1 Funkce	5
3.2 SDL	5
4 Zásady.....	6
4.1 Rychlost	6
4.2 Stabilní struktura.....	6
4.2.1 Projekt	6
4.2.2 Adresáře	6
4.3 Bezpečnost.....	6
4.3.1 Ladění.....	6
4.3.2 Správa paměti pomocí RAI	7
4.3.3 Typově bezpečné výčtové typy	7
4.3.4 Zajištění potřebných knihoven	7
5 Uživatelská příručka.....	9
5.1 Instalace.....	9
5.2 Zahrnutí souborů	9
5.3 Výčet systémů.....	9
5.4 Načítání souborů.....	10
5.5 Příkladový program.....	10
6 Seznam literatury	10

3 Teoretická část

3.1 Funkce

Halcyon je multiplatformní C++ knihovna, která obaluje knihovnu SDL, napsanou v jazyce C. Obsahuje sadu nástrojů pro vývoj her a obecně GUI aplikací na nízké úrovni. Dokáže tvořit okna, načítat a kreslit textury a zpracovávat zprávy a události od operačního systému a uživatelský vstup.

Je napsaná idiomatically a nabízí rozsáhlý systém předcházení chyb nejen za běhu, ale i za kompilace. Toto vše funguje se skoro nulovým zpomalením v porovnání s SDL. Zkrátka využívá mnohá řešení pro problémy jazyka C, které nabízí C++.

Pár z výhod činí:

- automatická (de)alokace objektů (RAII)
- nepotřebnost práce se směrnicí
- bezpečnostní prostředky za kompilace
- zabudované, konfigurovatelné ladění
- typově bezpečné výčtové typy

3.2 SDL

Simple DirectMedia Layer je již dlouhá léta vyvíjená a aktivně udržovaná knihovna napsaná v jazyce C, jejíž popularita vyplývá z možnosti vývoje nezávisle na platformě. Shrnuje mnohé nízkoúrovňové prvky, které nejsou standardizované, pod jednotné API – například grafika, zvuk, vlákna¹ apod.

Knihovna má široké uplatnění. Užívá ji například Valve v jejich herním enginu Source, přičemž také značně investoval do jejího dalšího vývoje.

Jazyk C je ovšem proslulý dvěma vlastnostmi: rychlostí a nebezpečností. V rukách zkušeného vývojáře je možné vytěžit maximum z daného systému – ale zároveň bylo skoro každé vážné nedávné bezpečnostní riziko nalezeno v knihovnách v něm napsaných. Dlouhá léta představovalo tedy vysoké riziko vykompenzované snad nejvyšší možnou rychlostí.

¹ C++ sice od standardu C++11 knihovnu pro vlákna má, ale C zatím ne.

4 Zásady

4.1 Rychlost

C++ je jazyk známý jeho rychlostí. Rozhodl jsem se pro design, který tuto hodnotu upřednostňuje nade všechny jiné. Systém ladění zároveň zajistí, že to není na úkor bezpečnosti.

Vybíral jsem čistě z tzv. *zero-cost abstractions*, tj. snížení množství práce při zachování výkonu. Kompilátor díky tomu dokáže zachytit množství chyb, které by se jinak mohly projevit až za běhu a je mnohem těžší vytvořit objekt, aniž by byly načteny potřebné knihovny. Nechtěl jsem nikde procesoru přidat nadbytečnou práci, aniž bych usoudil, že to opravdu stojí za zjednodušení práce koncového uživatele.

Halcyon má vlastní definice pro aritmetické typy – např. 16-bitové celé číslo je `hal::i16` a 64-bitový float je `hal::f64`. Ve výchozí formě se mapují přímo na `std::[u]intN_t` typy, kde N je počet bitů, ovšem zahrnuta je možnost tyto definice převést na `std::[u]int_fastN_t` typy, které jsou vhodnější pro procesor².

4.2 Stabilní struktura

4.2.1 Projekt

Struktura projektu je navržena pro rozšiřitelnost, pro což využívá výhody objektově orientovaného programování. Souhrnná funkčnost mnoha objektů z SDL je sjednocena ve třídě `sdl::object`. Dodatečné knihovny mají poté v projektu jejich vlastní namespace za účelem přehlednosti.

Věci vyžadující systémy mají taky namespace, a vše určené pro interní využití má vlastní namespace `sdl` a `detail`.

4.2.2 Adresáře

Funkčnost patřící pod systém se nachází v podadresáři a na nejvyšší úrovni je vždy hlavičkový soubor s korespondujícím jménem, který vše z této složky zahrnuje – pro video systém se zahrne `<halcyon/video.hpp>`, pro události zase `<halcyon/event.hpp>`.

Na nejvyšší úrovni je tedy funkčnost, která funguje samostatně nebo má vlastní kontext; příklady jsou `surface.hpp` nebo dodatečné knihovny `image.hpp` pro načítání obrázků a `ttf.hpp` pro fonty.

4.3 Bezpečnost

4.3.1 Ladění

Valná většina funkcí v SDL vrací hodnotu, která komunikuje, zda uspěla, nebo ne. Uživatel například může poskytnout neplatný soubor/objekt, nebo systém nemusí podporovat jistou funkci. Mnozí s tímto designem pracují následovně:

```
// nula značí úspěch
if (SDL_Init(...) != 0) {
    std::cout << "Inicializace SDL selhala: " << SDL_GetError() << '\n';
    std::exit(EXIT_FAILURE);
}
```

Toto je vykonáno za běhu a přidává do konečného programu nepotřebné instrukce.

² Na Windows jsou mi například 16-bitové čísla převedena na 32-bitová, Linux mění 16- i 32-bitové hodnoty na 64-bitové a macOS (na architektuře AArch64) ponechává vše stejné.

Součástí mé “filozofie” při vývoji je názor, že základní funkce tohoto rázu fungovat za běhu budou nebo ne a je povinností vývojáře to zkontrolovat. Halcyon nabízí poměrně jednoduchý, ovšem robustní systém pro kontrolu předpokladů využívající makra, který může být ve finálním, optimalizovaném programu kompletně odstraněn pro maximální rychlost.

Makro	Funkce
HAL_PANIC(...)	Vypíše, kde se způsobila panika spolu s poskytnutými informacemi.
HAL_ASSERT(cond, ...)	Pokud se cond == false, způsobí paniku, které předává poskytnuté informace.
HAL_WARN_IF(cond, ...)	

Pokud je definováno NDEBUG, ladění je automaticky aktivované v pokročilém režimu³. V opačném případě lze stále definovat HAL_DEBUG_ENABLED, čímž se zprovozní základní režim ladění. Jednoduše řečeno, uživatel má volbu; ví-li, že bude vše v pořádku, může se velmi jednoduše zbavit nepotřebných kontrol správnosti.

4.3.2 Správa paměti pomocí RAI

Kdybych měl říct můj názor na nejznámější C++ idiom, přidělil bych tento titul tzv. *Resource Acquisition Is Initialization* (RAII). Nebojte se složitěho názvu – je to řešení známého problému C, kde musí vývojář manuálně alokovat a vracet dynamickou paměť známou kombinací malloc/free.

V C++ se dá funkce na získání paměti dát do konstruktoru, a opak do destrukturu třídy. Zde přebírá kormidlo kompilátor, který ví, kdy budou objekty zničeny a zajistí patřičné kroky. Tím vzniká tento idiom, díky kterému kódujeme nejen rychleji, ale i bezpečněji.

Jak jste si mohli domyslet, Halcyon tohoto využívá v plné míře. Objekt stačí vytvořit, a o zbytek je postaráno.

4.3.3 Typově bezpečné výčtové typy

Výčtové typy – enumy – jsou základní nástroj k přehlednosti kódu. Jsou to v podstatě jen celá čísla, a enum jednoho typu se může bez problému přiřadit k jinému, přestože nemusí ani existovat korespondující hodnota. C++ pro toto má řešení, které z nich vytvoří unikátní typ, kterému se naopak tyto konvertující funkce musí dodatečně vytvořit. Uživatel tedy nemůže jen tak tvrdit, že se proměnná reprezentující klávesu na klávesnici rovná tlačítku na myši.

```
#include <halcyon/event/keyboard.hpp>
```

```
int main(int argc, char* argv[]) {
    // Toto je “v pořádku”
    SDL_Scancode s = SDL_KEYCODE_A;

    // Toto ne
    hal::keyboard::button b = hal::keyboard::key::A;
}
```

4.3.4 Zajištění potřebných knihoven

SDL vyžaduje před využitím jistých věcí načíst patřičné systémy, ale může se stát, že se na to zapomene, což se zjistí až za běhu. Halcyon toto nedovolí; okno se nedá vytvořit bez video

³ V pokročilém režimu je zaznamenán i čas hlášení a vše se zrcadlí v dodatečném výstupním souboru. Je aktivován definicí HAL_DEBUG_ADVANCED.

systému, video systém se zase neobejde bez kontextu. Konstruktory ověřují existenci potřebných kontextů či systému, takže uživatel nedostane přístup k nepřipraveným funkcím. Toto “vedení za ručičku” před během aplikace zajišťuje jen správnost programu a nepřidá ani jednu instrukci.

Příklad:

```
#include <halcyon/video.hpp>

int main(int argc, char* argv[]) {
    hal::context ctx;

    // Toto kompilátor přijme.
    hal::video::system vid1 { ctx };

    // Toto ne; destrukce kontextu by systém ihned vypla.
    hal::video::system vid2 { hal::context {} };

    // Toto kompilátor přijme.
    hal::video::window wnd1 { vid1, "Test", { 640, 480 } };

    // Toto ne; systém by se ihned vypl po konstrukci.
    hal::video::window wnd2 { hal::video::system { ctx }, "Test", { 640, 480 } };
}
```


5 Uživatelská příručka

5.1 Instalace

Instalace využívá sestavovací systém CMake a má jen pár kroků:

1. Nainstalujte přes vašeho správce balíčků knihovny *SDL2*, *SDL2_image* a *SDL2_ttf*.
2. Stáhněte si zdrojový kód knihovny a vložte složku *halcyon* do složky vašeho projektu.

Nyní by struktura vašeho projektu mohla vypadat následovně:

- project/
 - [zdrojový kód...]
 - halcyon/
 - CMakeLists.txt

3. V *CMakeLists.txt* přidejte příkaz `include(halcyon/config.cmake)`.

CMake zkontroluje, zda jsou nainstalovány potřebné knihovny, nakonfiguruje je a vytvoří patřičné proměnné.





Název	Funkce
HALCYON_INCLUDE_DIRS	Složky obsahující hlavičkové soubory potřebné pro knihovnu.
HALCYON_SOURCES	Implementační soubory, které musí být kompilovány společně s vaším projektem.
HALCYON_LIBRARIES	Dodatečné knihovny, které musí být dány linkeru.

5.2 Zahrnutí souborů

Jak již bylo zmíněno, funkčnost patřící pod systém má svou vlastní složku a na nejvyšší úrovni je vždy hlavičkový soubor s korespondujícím jménem – pro video systém se zahrne `<halcyon/video.hpp>`, pro události zase `<halcyon/event.hpp>`.

Zahrnuté soubory budou zpravidla mít patřičný namespace, ale ne vždy. Pro přehlednost struktury složka `video/` zahrnuje třeba `driver.hpp`, který funguje samostatně, přestože je s GPU spjatý, a `event/` zahrnuje ve stejném duchu `keyboard.hpp` a `mouse.hpp`.

5.3 Výčet systémů

Název	Hlavičkový soubor	Potřebuje kontext?	Popis
Halcyon Video	<code>video.hpp</code>		Okna, textury a vše co využívá GPU.
Halcyon Events	<code>event.hpp</code>		Sbírání událostí od operačního systému.
Image Loading	<code>image.hpp</code>	 (má vlastní)	Načítání souborů PNG, JPEG apod.
TrueType Fonts	<code>ttf.hpp</code>	 (má vlastní)	Načítání TTF souborů a renderování textu s nimi.

V budoucnu plánuji také přidat audio systém.

5.4 Načítání souborů

Funkce či konstruktory očekávající soubor nepřijímají řetězec jakožto jméno souboru, ale `hal::accessor`. Toto umožňuje načítání nejen soubory z úložiště, ale i přímo z paměti. Vytvořte ho pomocí funkcí `hal::load`.

5.5 Příkladový program

```
#include <halcyon/event.hpp>
#include <halcyon/video.hpp>

int main(int argc, char* argv[]) {
    hal::context ctx;
    hal::video::system vid { ctx };

    hal::video::window wnd { vid, "Test", { 640, 480 } };

    hal::event::handler evt { vid.events };

    while (true) {
        while (evt.poll()) {
            using enum hal::event::type;

            switch (evt.event_type()) {
                case quit_requested:
                    return EXIT_SUCCESS;

                case key_pressed:
                    if (const auto btn = evt.keyboard().button(); btn != hal::keyboard::button::esc)
                        HAL_PRINT(hal::to_string(btn), " pressed");
                    else
                        return EXIT_SUCCESS;

                default:
                    break;
            }
        }
    }
}
```

Tento program vytvoří okno, přičemž vypíše jakýkoliv stisk klávesy přitom co je aktivní. Stisknutí klávesy Q nebo systémová žádost o konec typu Alt-F4 program ukončí. Objekty se postupně tvoří s odkazy na podpůrné systémy, čímž se minimalizuje míra chyb.

V event smyčce se specifikuje, pro co program bude naslouchat a jak zpracuje patřičné data.

6 Seznam literatury

- SDL Community (duben 2024). *Simple DirectMedia Layer - Homepage*. <https://www.libsdl.org/index.php>
- eerorika (duben 2024). *c++ - Why are the fast integer types faster than the other integer types?*, <https://stackoverflow.com/a/59593614/17814437>