Knihovna Allegro a její využití

Allegro library and its applications

Bc. Boris Barák

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Boris BARÁK**
Osobní číslo:	**A08556**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Knihovna Allegro a její využití**

Zásady pro vypracování:

1. **Vytvořte literární rešerši na zadané téma.**
2. **Seznamte se s knihovnou Allegro. V teoretické části práce popište její vývoj, vlastnosti a oblast použití v aktuální verzi.**
3. **Knihovnu Allegro porovnejte s jinými knihovnami podobného typu (např. SDL).**
4. **Na základě dosažených znalostí vytvořte rozsáhlejší aplikaci založenou na této knihovně. Tuto aplikaci včetně zdrojových kódů umístěte do elektronické přílohy práce.**
5. **Vytvořte podrobný popis aplikace, který by usnadnil tvorbu jiných, obdobných aplikací založených na Allegru.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **HARBOUR, Jonathan S. Game Programming All in One. 2006th Third Edition edition. [s.l.] : Course Technology PTR , 2006. 832 s. ISBN 1598632892.**
2. **KERNIGHAM, Brian W. Programovací jazyk C 1. vyd. Praha : Computer Press, 2006. 80-251-0897-X s. ISBN 80-251-0897-X.**
3. **LIBERTY, Jesse. Naučte se C++ za 21 dní. 1. vyd. Praha : Computer Press, 2002. 80-7226-774-4 s., CD-ROM. ISBN 80-7226-774-4.**
4. **ALEXANDRESCU, Andrei. Moderní programování v C++. 1. vyd. Praha : Computer Press, 2004. 80-251-0370-6 s. ISBN 80-251-0370-6.**
5. **Allegro.cc : Game developing community network [online]. 1999-2009 [cit. 2009-01-30]. Dostupný z WWW: <http://www.allegro.cc/>.**
6. **C++ Game Programming [online]. [2005] [cit. 2009-01-30]. Dostupný z WWW: <http://www.cppgameprogramming.com/>.**
7. **Allegro - A game programming library [online]. [2009] [cit. 2009-01-30]. Dostupný z WWW: <http://www.talula.demon.co.uk/allegro/>.**
8. **Builder - Informacni server o programovani [online]. 1999-2003 [cit. 2009-01-30]. Dostupný z WWW: <http://www.builder.cz/>.**
9. **Allegro Wiki [online]. [2008] [cit. 2009-01-30]. Dostupný z WWW: <http://wiki.allegro.cc/>.**
10. **OpenLayer - Hardware Accelerated 2d Game Development [online]. [2005] [cit. 2009-01-30]. Dostupný z WWW: <http://openlayer.berlios.de/>.**

Vedoucí diplomové práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ Fakulta aplikované informatiky L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce je podrobně popsat multimediální knihovnu Allegro, která se již dlouhou řadu let používá ke tvorbě multimediálních aplikací, především ke tvorbě 2D počítačových her a ze získaných poznatků navrhnout, vytvořit a zdokumentovat vzorovou multimediální aplikaci, která by usnadnila začínajícím programátorům her tvorbu pomocí této knihovny. Aplikace by měla být také natolik univerzální, aby se dala lehce modifikovat a to nejen pomocí úprav zdrojového kódu, ale především pomocí externích datových souborů určujících vzhled a strukturu prostředí aplikace.

Klíčová slova: Allegro, grafická knihovna, 2D grafika, programování, C++, tvorba her

ABSTRACT

The aim of this thesis is in detail describe the multimedia library Allegro, which has for many years used to create multimedia applications, especially for make 2D computer games, and on the basis of obtained knowledge design, develop and document a standard multimedia applications, which would simplify create games to novice programmers using this library. Applications should also be so universal to be given not only by modifying source code changes, but mainly using external data files that specifies the appearance and structure of the application environment.

Keywords: Allegro, grafics library, 2D grafics, programing, C++, game development

Poděkování:

Mé díky patří Ing. Pavlu Pokornému, Ph.D. za cenné rady, které mi byly vydatnou pomocí při zpracovávání mé práce a také za čas věnovaný této práci.

Motto:

„Je vždy lehčí přizpůsobit své požadavky programu než naopak.“

Murphyho počítačové zákony - Osmý zákon programování

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně ……

podpis diplomanta

OBSAH

ÚVOD

U velkého množství začínajících a mírně pokročilých programátorů se objevují ambice ve tvorbě počítačových her a multimediálních aplikací. Často se však objevují v tomto úsilí překážky, které jim brání v dostatečném rozvoji nebo jej zpomalují.

Účelem této práce je snaha o poskytnutí dostatečného zázemí pro tvorbu počítačové hry pro mírně pokročilé programátory znalé jazyka C++, kteří mají základní znalosti objektového programování a chtějí své dovednosti rozšířit ve směru multimediálního využití.

Tato práce se zabývá především multimediální knihovnou Allegro a jejím rozšířením, které programátorům nabízí pohodlnou práci s potřebnou funkčností v oblasti 2D a 3D grafiky, zvuku a hudby, obsluhy vstupních zařízení (myš, klávesnice, joystick), časovačů a práce se soubory.

Nejprve bude představena knihovna Allegro, bude popsáno, jak vznikla a jaká je její blízká budoucnost. Poté budou podrobně rozebrány nejdůležitější funkce jejího API, především funkce grafické a práci se vstupními zařízeními. Dále bude následovat seznámení s nástroji a oficiálními rozšířeními této knihovny a její porovnání s alternativami. Tím bude zakončena část teoretická a bude následovat návrh vlastní aplikace.

Aplikaci bude vytvářena pod frameworkem OpenLayer ve vývojovém prostředí Code::Blocks a podrobně bude rozebrán její vývoj. Aplikace bude udělána co nejvíce rozšiřitelnou a to i bez znalostí programovacího jazyka C++.

Na závěr aplikace bude důkladně zdokumentována a to jak pro uživatele, tak pro případný další vývoj.

I. TEORETICKÁ ČÁST

1 VÝVOJ ALLEGRA

Podle Oxford Companion to Music, Allegro je italsky „rychlé, plné života, jasné“. Z toho byl vytvořen název, který je rekurzivním akronymem těchto slov: „**A**llegro **L**ow **LE**vel **G**ame **RO**utines“. [5]

V počátečním stádiu Allegro vytvořil Shawn Hargreaves pro Atari ST v roce 1990. Nicméně Shawn Hargreaves opustil Atari, když si uvědomil, že tato platforma zaniká a předělal knihovnu pod Borland C++ a DJGPP kompilátory v roce 1995. Podpora pro Borland C++ byla opuštěna ve verzi 2.0 a DJGPP byl jediný podporovaný kompilátor. DJGPP bylo kompilátorem pro operační systém DOS, takže všechny vzniklé aplikace (především hry) využívající Allegro byli spustitelné pouze pod tímto operačním systémem.

V roce 1998 bylo Allegro rozděleno do několika vývojových větví. Byl vytvořen port pro operační systém Microsoft Windows zvaný WinAllegro a současně byl vytvořen Unix port Allegra zvaný XwinAllegro. Tyto verze byly sjednoceny během vývoje Allegra ve verzi 3.9 WIP a verze 4.0 se stala první stabilní verzí Allegra s podporou více platforem. Současná verze Allegra podporuje operační systémy Unix (Linux, FreeBSD, Irix, Solaris, Darwin), Windows (kompilátory MSVC, MinGW, Cygwin, Borland C++), Mac OS X a verze 4.2 a vyšší podporují navíc BeOS, QNX a DOS (kompilátory DJGPP a Watcom). Ve verzi 4.9 (nestabilní verze) je taktéž vyvíjen port pro iPhone. Původní autor Shawn Hargreaves již nadále není zapojen do vývoje Allegra. [11]

Obrázek 1 Logo knihovny Allegro

2 SOUČASNÉ VLASTNOSTI ALLEGRA

2.1 Multiplatformní podpora

Allegro je od verze 4 jedna z mála multimediálních knihoven, které podporují více různých kompilátorů a více operačních systémů. Kód napsaný za pomocí Allegra lze tedy zkompilovat na více platformách. Knihovna je přeložitelná na platformách Unix, Windows, DOS, MacOS X, BeOS a QNX a v připravované verzi 5 je přidána také podpora pro iPhone. [1]

2.1.1 Linux konzole

Pod systémem Linux jsou k dispozici 2 různé systémové ovladače - jeden pro spuštění s použitím systému X-Window a druhý bez něj. Verzi s X-Window je možno použít i v jiných operačních systémech založených na platformě Unix, zatímco druhou verzi lze použít pouze ve spoluprácí s konzolí Linuxu. Obvykle je v Linuxu verze s X-Window preferována, pokud je X server dostupný. V případě, že je v operačním systému aktivní X-Window, bude se používat, jinak se bude využívat pouze konzolových funkcí. [1]

Pokud to není potřeba kvůli specifickým potřebám, je doporučeno raději používat verzi s X. Linuxový konzolový port však není ve výchozím nastavení povolen, protože nikdy nepracoval správně a nebyl o něj od vývojářů pracujících s Allegrem velký zájem. [5]

2.1.2 Unix X Window

Pro kompilaci a instalaci Allegra by měly být ve všech Unixových systémech nástroje již k dispozici. Je zapotřebí nástroj CMake ve verzi 2.6 nebo vyšší, většina distribucí Linuxu by jej však již měla obsahovat.

Při inicializaci se Allegro zkouší připojit k X serveru. V případě, že jej nenalezne, pokusí se nalézt některé jiné systémové ovladače, např. ovladač konzole, je-li povolen. Pokud se spojení podaří navázat, spustí se aplikace v prostředí X nebo lze nastavit proměnnou prostředí DISPLAY, aby určovala, jaké grafické prostředí chceme v aplikaci použít.

Poté zde existují dva různé grafické ovladače pod X-Window a to GFX_XWINDOWS, který používá standardní komunikaci s X serverem a GFX_XDGA2, který využívá rozšíření XFree86 DGA 2.0 (dodáváno s XFree86 verze 4.0.0 a vyšší) a umožňuje skrze

něj zapisovat přímo na zobrazovací surface a využívat hardwarovou akceleraci, je-li k dispozici. To je obyčejně rychlejší než běh po X-Window, ale je potřeba root oprávnění a nebude moci pracovat vzdáleně.

Pokud naše aplikace bude vyžadovat jinou barevnou hloubku, než které je nastavena v X-Window, bude požadovanou hloubku emulovat Allegro. Program však v tomto případě bude pracovat pomaleji než v případě, že bude zadána barevná hloubka totožně. Zda emulace probíhá, je možné zjistit ze struktury *gfx_driver* z položky *desc*. [5]

2.1.3 Windows

Na platformě Windows je možno s Allegrem tvořit GUI aplikace i konzolové aplikace. Je také možno kombinovat spolu s Allegrem funkce pro Win32 API nebo MFC. Lze to udělat tak, že se vloží místo hlavičkového souboru allego.h soubor winalleg.h a v případě potřeby použití MFC je možné jej zprovoznit definováním makra preprocesoru s názvem *ALLEGRO_AND_MFC*. Těmito kroky se však přijde o přenositelnost zdrojového kódu na jiné platformy. [5]

Aplikace zkompilované pod operačním systémem Windows vyžadují ke svému běhu DirectX verze 7 nebo vyšší. DirectX vyžaduje, aby před vykreslováním byly všechny bitmapy ze systémové a grafické paměti uzamknuty. Uzamykání se provádí automaticky, nicméně můžeme získat lepší výkon sami pomocí funkce *acquire_bitmap*.

Vzhledem k tomu, že hlavní dohled na přidělování paměti má DirectX, tak neexistuje způsob, jak zachovat obsah grafické paměti, když se uživatel přepne mimo náš program. Je potřeba pamatovat na skutečnost, že obsah obrazovky a obsah jakékoli bitmapy v paměti může být zničen v jakémkoliv bodě. Ke zjištění toho, kdy se to stane, lze použít funkci *set_display_switch_callback*.

Na platformě Windows jsou možné návratové hodnoty funkce *desktop_color_depth* pouze 8, 16, 24 a 32. To znamená, že 15-bitová a 16-bitová barevná hloubka plochy se nerozlišuje a obě jsou vráceny jako 16-bitová. Aplikace je možno spustit v okně nebo přes celou obrazovku. [5]

2.1.4 MacOS X

Pro operačním systémem MacOS X je Allegro dodáváno pouze ve formě zdrojových kódů, takže je nutné před použitím knihovnu zkompilovat. Ke zkompilování knihovny je potřeba mít MacOS X ve verzi 10.2.0 nebo vyšší a vývojové nástroje Apple Developer Tools (December 2002 nebo novější). K úspěšnému běhu aplikace však stačí MacOS X verze 10.1.0 nebo vyšší. [1]

Po spuštění vytvořené aplikace se inicializuje systémový ovladač a tím se připojí NSApplication objekt k Window Serveru. Přepínání oken je možné, ale pouze pokud je povolen režim SWITCH_BACKGROUND.

Grafický režim v MacOS X používá Cocoa okno se zobrazením pomocí Quartz QuickDraw. Všechny kombinace barevných hloubek okno / plocha jsou dostupné. Pokud je to potřeba, je automaticky provedena konverze barev transparentně pro uživatele.

V režimu celé obrazovky je použito CoreGraphics DirectDisplay API, které podporuje pouze 8, 15 a 32-bitovou barevnou hloubku. [5]

2.1.5 BeOS a QNX

V systému BeOS je jako grafický ovladač použito BWindowScreen pro zobrazování přes celou obrazovku a BDirectWindow pro zobrazování v okně. Pro zvuk jsou v systému BeOS podporována rozhraní BSoundPlayer a BMidiSynth. Jinak je běh v systémech BeOS a QNX založený na technologii Unixu X-Window. [5]

2.1.6 DOS

Pod operačním systémem DOS jsou k dispozici grafické ovladače VGA 13h (standardní VGA s 256ti barvami používající rozlišení 320x200 bez podpory hardwarového scrolování nebo nižší s jeho podporou), mode-X neboli 23 VGA (přidává spoustu dalších méně obvyklých rozlišení až do 400x600), SVGA (rozlišení až 800x600 v 8, 15, 16 a 24 bitech na pixel) a VESA ve verzích 1.x, 2.0 (v banked nebo lineálním framebuffer módu) a 3.0.

Také je zde možnost hardwarové akcelerace grafiky přes VBE/AF API, pokud je podporována a za pomoci projektu FreeBE/AF jsou k dispozici další rozšiřující grafické ovladače. [5]

2.2 Funkce API

2.2.1 Základní nastavení

2.2.1.1 Inicializace

Funkce *install_allegro* (parametry ***int*** *system_id*, ***int*** **errno_ptr*, ***int*** *(*atexit_ptr)()*) inicializuje knihovnu Allegro. Musí se zavolat buď tato nebo funkce *allegro_init* dříve, než se bude provádět cokoliv jiného, s výjimkou Unicode funkcí. Je-li potřeba použít jiné kódování textu než UTF-8, můžeme jej nastavit funkcí set_uformat dříve, než budeme inicializovat Allegro.

Parametr system_id se liší podle dané platformy, tudíž se doporučuje nastavit je na hodnotu SYSTEM_AUTODETECT. Případně se dá použít hodnota SYSTEM_NONE, která inicializuje Allegro očesané od všech platformě specifických funkcí včetně jakékoli komunikace s hardwarem. Toho se dá využít např. při práci pouze s pamětí bitmapy či volání funkcí rozhraní Windows GDI.

Parametr *errno_ptr* je ukazatel na proměnnou s návratovým kódem aplikace a parametr *atexit_ptr* je ukazatelem na funkci, která se vykovává při ukončení aplikace. Oba parametry jsou povinné, ale v případě, že parametr *atexit_ptr* nastavíme na hodnotu *NULL*, musíme poté sami před ukončením aplikace zavolat funkci *allegro_exit* (bez parametru). Pro obvyklé použití Allegra se však doporučuje zavolat funkci *allegro_init* (bez parametru), která zavolá funkci *install_allegro* s parametry *SYSTEM_AUTODETECT, &errno, atexit*. Všechny výše popsané funkce vrací datový typ int s hodnotou 0 v případě úspěch nebo 1 v případě chyby. [7]

2.2.1.2 Deinicializace

K ukončení práce s Allegrem, resp. k vyčištění paměti a ukončení práce se vstupním a výstupním hardwarem, který byl inicializován, slouží funkce *allegro_exit*. Běžně ji však volat nemusíme, pokud jsme inicializovali Allegro pomocí funkce *allegro_init*, která určí funkci atexit jako funkci pro deinicializaci. K jejímu volání slouží makro *END_OF_MAIN(),* které se stará mimo jiné i o správný postup deinicializace a také přepsání funkce *main* v závislosti na platformě. [7]

Doporučený zápis inicializace a deinicializace Allegra tedy vypadá následovně:

```
1 int main(void)
2 {
3    allegro_init();
4    /* zde se nachází další kód */
5    return 0;
6 }
7 END_OF_MAIN()
```

2.2.1.3 Výpisy a chybové hlášky

K výpisům zpráv je určena funkce *allegro_message* s nekonečným počtem parametrů a prvním parametrem **const char** **text_format*, který slouží k výpisu řetězce a případných proměnných dalších parametrů (volá se obdobně jako standardní ANSI C funkce *printf*, *sprintf* apod.). [7]

2.2.2 Práce s myší

Pro povolení práce s myší je nutno zavolat funkci *install_mouse*, která v případě, že již byla inicializována, vrátí hodnotu 0, v případě selhání inicializace, vrátí hodnotu -1 nebo v případě úspěšné inicializace vrátí počet použitelných tlačítek myši.

Zavoláním funkce *enable_hardware_cursor* (bez parametru) se aktivuje kurzor z operačního systému. V případě, že zavoláme funkci *disable_hardware_cursor* (také bez parametru), tak se o vykreslování kurzoru bude starat Allegro. Typ kurzoru poté se určí zavoláním funkce *select_mouse_cursor*, kde do parametru se zadá jedna z následujících hodnot:

- *MOUSE_CURSOR_NONE* - kurzor nebude viditelný
- *MOUSE_CURSOR_ALLEGRO* - vlastní kurzor z Allegra, ten se nastaví funkcí *set_mouse_sprite*
- *MOUSE_CURSOR_ARROW* - zobrazí defaultní systémový kurzor (šipka)
- *MOUSE_CURSOR_BUSY* - zobrazí systémový kurzor zaneprázdnění (přesýpací hodiny)

- *MOUSE_CURSOR_QUESTION* - zobrazí systémový kurzor pro nápovědu (šipka s otazníkem)
- *MOUSE_CURSOR_EDIT* - zobrazí systémový kurzor pro editaci textu (vertikální kurzor ve tvaru písmene I)

Funkcí *show_mouse* (parametrem je ukazatel na strukturu BITMAP) se nastaví zobrazování kurzoru na obrazovku, tedy parametr se určí jako *screen* (globální ukazatel na bitmapu obrazovky, viz níže). V případě potřeby vypnutí zobrazování kurzoru, se zavolá tato funkce s parametrem *NULL*.

K dočasnému skrytí kurzoru složí funkce *scare_mouse* (bez parametru). Pokud je potřeba skrývat myš jen v určité části obrazovky, lze zavolat funkci *scare_mouse_area* (parametry **int** *x*, **int** *y*, **int** *w*, **int** *h*), kde se v parametrech určí obdélníková oblast, ve které bude kurzor skrytý. Funkcí *unscare_mouse* (bez parametru) jde obnovit původní viditelnost kurzoru.

Pozici kurzoru lze nastavit funkcí *position_mouse* (parametry **int** *x*, **int** *y*), kde se v parametrech zadá absolutní souřadnice umístění na obrazovce, případně se může nastavit pozice kolečka myši funkcemi *position_mouse_z* (pro verzikální pozici) a *position_mouse_w* pro horizontální pozici kolečka.

Současná pozice kurzoru a informace o stisknutí tlačítek myši jsou přístupné z globálních proměnných a získávají se pomocí masek následujícím způsobem:

```
 8 if (mouse_b & 1)
 9    printf("Leve tlacitko je stisknuto\n");
10 if (!(mouse_b & 2))
11    printf("Prave tlacitko neni stisknuto \n");
12
13 int pos, x, y;
14 pos = mouse_pos;
15 x = pos >> 16;         /* horizontální pozice kurzoru */
16 y = pos & 0x0000ffff;  /* vertikální pozice kurzoru */
```

Zda je stisknuté tlačítko se zjistí z proměnné mouse_b tak, že se provede logický součin s číslem požadovaného tlačítka. Horizontální pozice se zjistí z horních 16ti bitů proměnné mouse_pos pomocí bitového posunu a vertikální pozice se zjistí ze spodních 16ti bitů

maskováním s číslem $FFFF_{hex}$. Pozice kurzoru se dá se rovněž získat z globálních proměnných mouse_x a mouse_y, ale tyto proměnné se aktualizují i v průběhu hlavního aktualizačního cyklu, což může u náročnějšího (resp. déle trvajícího) obsahu tohoto cyklu způsobit problém. Pozice kolečka myši se ukládá do globálních proměnných mouse_z (vertikální) a mouse_w (horizontální).

2.2.3 Práce s klávesnicí

Allegro inicializuje vstup z klávesnice funkcí *install_keyboard* (bez parametru), kterou je nutno zavolat před samotnou prací s klávesnicí. Jakmile dojde k této inicializaci, již se nedají použít funkce pro zpracování přerušení klávesnice z jiné knihovny nebo systému. Na některých platformách je také nutné před použitím funkcí pro práci s klávesnicí po této inicializaci také zavolat funkci *set_gfx_mode* (viz. dále) pro inicializaci grafického okna aplikace.

Pro zjištění stavů stisknutí jednotlivých kláves je určeno pole *key* o velikosti *KEY_MAX* (makro určující celkový počet použitelných kláves) v němž jednotlivé prvky (datového typu char) určují, zda je daná klávesa (hodnota *TRUE*) stisknuta či nikoli (hodnota *FALSE*). Také je možno zjistit, jestli je současně s klávesou zmáčknutá speciální klávesa jako Shift, Alt, Ctrl apod. maskováním proměnné *key_shifts* a maker z tabulky 2 následujícím způsobem:

```
17 if (key[KEY_W]) {
18    if (key_shifts & KB_SHIFT_FLAG) {
19       /* Uživatel zmáčkl Shift + W. */
20    } else {
21       /* Uživatel zmáčkl pouze W. */
22    }
23 }
```

2.2.3.1 Práce s joystickem

Podobně jako vstup z klávesnice a myši dokáže Allegro na většině platforem pracovat s přerušeními od joysticku. Inicializuje se funkcí *install_joystick* s parametrem *type*, který se obvykle nastaví na hodnotu *JOY_TYPE_AUTODETECT*, případně se zadá hodnota z manuálu k danému joysticku pro danou platformu.

Většinu joysticků je potřeba před použitím kalibrovat, na což jsou k dispozici funkce *calibrate_joystick* a calibrate_joystick_name**.** Doporučený postup kalibrace joysticků je následující:

```
int i;
for (i = 0; i < num_joysticks; i++) {
   while (joy[i].flags & JOYFLAG_CALIBRATE) {
      char *msg = calibrate_joystick_name(i);
      textprintf_ex(..., "%s, zmacknete lib. klavesu\n", msg);
      readkey();
      if (calibrate_joystick(i) != 0) {
         textprintf_ex(..., "Chyba!\n");
         readkey();
         exit(1);
      }
   }
}
```

Informace o stavu joysticku se ukládají do globálního pole joy, které se aktualizuje zavoláním funkce poll_joystick. Prvkem tohoto pole je struktura JOYSTICK_INFO, která je definována následovně:

```
typedef struct JOYSTICK_INFO
{
   int flags;                          /* status joysticku */
   int num_sticks;                     /* počet vstupů */
   int num_buttons;                    /* počet tlačítek */
   JOYSTICK_STICK_INFO stick[n];       /* info o vstupech */
   JOYSTICK_BUTTON_INFO button[n];     /* info o tlačítkách */
} JOYSTICK_INFO;
```

Struktura JOYSTICK_STICK_INFO zapouzdřující informace o tlačítkách joysticku je definovaná takto:

```
typedef struct JOYSTICK_BUTTON_INFO
{
   int b;                              /* stav on nebo off */
   char *name;                         /* popis tohoto tlačítka */
} JOYSTICK_BUTTON_INFO;
```

Vstupy joysticku zapouzdřuje struktura JOYSTICK_STICK_INFO a ta je definovaná následovně:

```
50 typedef struct JOYSTICK_STICK_INFO
51 {
52    int flags;                        /* stav tohoto vstupu */
53    int num_axis;                     /* počet os */
54    JOYSTICK_AXIS_INFO axis[n];       /* informace o osách */
55    char *name;                       /* popis tohoto vstupu */
56 } JOYSTICK_STICK_INFO;
```

Ta obsahuje ještě jednu strukturu a to *JOYSTICK_AXIS_INFO*, která obsahuje informace o ose joysticku:

```
57 typedef struct JOYSTICK_AXIS_INFO
58 {
59    int pos;                          /* analogová pozice */
60    int d1, d2;                       /* digitální pozice */
61    char *name;                       /* popis této osy */
62 } JOYSTICK_AXIS_INFO;
```

2.2.4 Grafický režim

Vzhledem k široké škále podporovaných platforem je grafický režim jediným způsobem, jak bezpečně komunikovat s uživatelem. Když bylo Allegro knihovnou pouze pro operační systém DOS (verze 3.x a starší), programátoři často používali pro komunikaci standardní C funkce jako *printf* před nastavením grafického režimu nebo *scanf* pro čtení vstupu od uživatele. To by však znamenalo problém pro aplikaci, která by běžela pod systému Windows, kde není standardní výstup na konzoli. I když by se aplikace úspěšně zkompilovala, byla by neovladatelná a to především v případě, že by zásadní informace byly v těchto textových zprávách. [7]

Allegro sice poskytuje funkci *allegro_message* pro zobrazování zpráv uživateli, ale ta není příliš uživatelsky přívětivá a je určena hlavně pro zobrazování menších chybových hlášek v případě, že není k dispozici grafický režim.

Nastavení grafického módu zahrnuje rozhodování o tom, jak rozdělit paměť grafické karty pro náš program. Na některých platformách to znamená vytvořit virtuální obrazovku větší,

než je fyzické rozlišení, udělat hardwarové scrolování nebo obracející stránky. Virtuální obrazovky můžou na první pohled způsobit mnoho problémů, ale jsou ve skutečnosti celkem jednoduché.

Grafickou paměť si lze představit jako obdélníkový kus papíru, který je viděn přes malý otvor v kartónu (náš monitor). Vzhledem k tomu, papír je větší než díra, jde vidět pouze její část v jedné konkrétní době, ale posunem kartónu kolem lze změnit viditelnou část obrázku.

Lze jednoduše nechat otvor v jedné poloze a ignorovat části video paměti, které nejsou viditelné. Je možné ale také získat nejrůznější užitečné účinky v neviditelné posuvné části kolem. Např. toho lze využít na kreslení obrázků v této skryté části video paměti a poté je pouze zobrazit, což je mnohem efektivnější než vytváření těchto obrázků do operační paměti a její následné kopírování. [1]

K aktivování grafického režimu a rozlišení slouží funkce *set_gfx_mode* (parametry **int** *card***, int** *w***, int** *h***, int** *v_w***, int** *v_h*). Parametry *w* a *h* slouží k nastavení šířky a výšky okna a parametry *v_w* a *v_h* jsou určeny k nastavení výše popsané virtuální vykreslovací plochy (celkové grafické paměti). Jako parametr *card* se zadává obvykle zadává makro jednoho z následujících režimů:

2.2.4.1 Režim GFX_AUTODETECT

Allegro zkusí nastavit zadané rozlišení s momentálně systémem nastavenou barevnou hloubkou na celou obrazovku. V případě selhání zkusí nastavit stejné rozlišení v systémovém okně. Pokud volání funkce *set_gfx_mode* uspěje, je tímto garantováno specifikované rozlišení se současnou barevnou hloubkou a jestli aplikace běží v okně nebo přes celou obrazovku. [7]

2.2.4.2 Režim GFX_AUTODETECT_FULLSCREEN

Allegro vyzkouší nastavit specifikované rozlišení se současnou barevnou hloubkou na celou obrazovku. Pokud se to nepovede, funkce vrátí hodnotu *FALSE*. [1]

2.2.4.3 Režim GFX_AUTODETECT_WINDOWED

Allegro zkusí nastavit specifikované rozlišení s momentální barevnou hloubkou v systémovém okně. Pokud se to nepovede, vrátí hodnotu FALSE. Pokud se to naopak povede, tak specifikované rozlišení bude znamenat samotnou grafickou oblast pro vykreslování programu, tj. mimo rámečku a dekorací okna. V případě, že by nastaly problémy kvůli konfliktu barevné hloubky nastavené v operačním systému a v naší aplikaci, se může tato situace vyřešit zavoláním *desktop_color_depth*, která tento problém vyřeší. [1]

2.2.4.4 Režim GFX_SAFE:

Použitím tohoto režimu Allegro garantuje, že grafický režim bude nastaven úspěšně. Pokusí se nastavit požadované rozlišení a v případě selhání se pokusí nastavit nejjistější rozlišení pro danou platformu. Tím je pod systémem DOS rozlišení VGA 320x200, pod Windows rozlišení 640x480, pod Linuxem aktuální rozlišení framebufferu (tj. rozlišení konzole), pokud je podporováno atd. Pouze v případě, že se ani tímto nepodaří úspěšně nastavit grafický režim, vrátí funkce hodnotu FALSE. [7]

2.2.4.5 Režim GFX_TEXT:

V tomto nastavení dojde k zavření případného předchozího grafického režimu, znemožní se používání globální proměnné *screen* určené pro vykreslování grafiky na obrazovku a aktivuje se čistě textový režim (rozlišení obvykle 80x25, záleží však na platformě). Při nastavování tohoto režimu se ignoruje nastavení rozlišení ve funkci *set_gfx_mode*. [7]

Seznam podporovaných grafických rozlišení u konkrétních grafických režimů je dostupný funkcí *get_gfx_mode_list*, která vrátí ukazatel na strukturu *GFX_MODE_LIST*. Ta má následující uspořádání:

```
63 typedef struct GFX_MODE_LIST
64 {
65    int num_modes;
66    GFX_MODE *mode;
67 } GFX_MODE_LIST;
```

Je vidět, že tato struktura obsahuje seznam grafických režimů definovaných strukturou GFX_MODE. Ta je definovaná následovně:

```
68 typedef struct GFX_MODE
69 {
70    int width, height, bpp;
71 } GFX_MODE;
```

Obsahuje tedy rozlišení (počet pixelů na šířku a výšku) a barevnou hloubku.

Dále je možné před nastavením grafického režimu a rozlišení nastavit barevnou hloubku funkcí *set_color_depth*, kde do parametru se zadá pouze počet bitů barevné hloubky. Také je možné vynutit obnovovací frekvenci monitoru funkcí *request_refresh_rate*, kde do parametru zadá se počet zobrazených obrázků za sekundu.

2.2.5 Práce s bitmapami

Po nastavení grafického režimu je možno používat globální proměnnou *screen* což je ukazatel na strukturu *BITMAP*. Struktura *BITMAP* je definovaná následovně:

```
1 typedef struct BITMAP
2 {
3    int w, h;              /* rozměry bitmapy v pixelech */
4    int clip;              /* nenulové pokud je otočení zapnuto */
5    int cl, cr, ct, cb;    /* otočení vlevo, vpravo, nahoru a dolů */
6    unsigned char *line[]; /* ukazatel na začátek každého řádku */
7 };
```

2.2.5.1 Načítání obrázků ze souboru

V Allegru lze v základu načítat obrázky ze souborů formátu BMP, LBM, PCX a TGA. Načítání ze souboru se prování funkcí *load_bitmap* (parametry **const char** **filename*, *RGB* **pal*), která automaticky rozpozná typ souboru. V prvním parametru se zadá název souboru a ve druhém barevná paleta. [1]

Podporu pro další formáty je možno naprogramovat použitím funkce *register_bitmap_file_type* (parametry **const char** **ext*, *BITMAP* *(**load*)(**const char** **filename*, *RGB* **pal*), **int** (**save*)(**const char** **filename*, *BITMAP* **bmp*, **const** *RGB*

**pal*)), kde do prvního parametru se zadá přípona souborů formátu, do druhého pak ukazatel na funkci, ve které se definuje algoritmus nahrávání do struktury *BITMAP* a do posledního parametru se zadá ukazatel na funkci pro uložení obsahu souboru do bitmapy.

Po načtení do struktury *BITMAP* ještě lze provést konverzi barevné palety funkcí *set_color_conversion*, kde do parametru se dá jedno z maker určující typ konverze z manuálu. [5]

2.2.5.2 Transformace a zobrazování bitmap

Kvůli problémům na některých platformách je doporučeno před zobrazením bitmapy nejprve tuto bitmapu uzamknout. To se může udělat funkcí *acquire_bitmap*, kde se do parametru zadá konkrétní bitmapa. Odemknutí pak lze provést funkcí *release_bitmap*. [5]

Samotné vykreslení bitmapy, tj. zkopírování její obdélníkové části nebo celé bitmapy do globální proměnné *screen* představující výsledný obraz, je možno udělat několika způsoby.

Prvním z nich je funkce *blit* (parametry *BITMAP *source, BITMAP *dest,* **int** *source_x,* **int** *source_y,* **int** *dest_x,* **int** *dest_y,* **int** *width,* **int** *height*), kde do prvního parametru dá se bitmapa, kterou lze vykreslit, do druhého pak bitmapu, do které se má vykreslovat, většinou tedy globální proměnnou *screen*. Parametry source_x a source_y určují souřadnice levého horního rohu obdélníku ze zdrojové bitmapy, další dva parametry určují souřadnice stejného rohu u cílové bitmapy. Zbývající dva parametry určují rozměry zkopírovaného obdélníku v cílové bitmapě.

Funkcí *stretch_blit* (parametry (*BITMAP *source, BITMAP *dest,* **int** *source_x, source_y, source_width, source_height, int dest_x, dest_y, dest_width, dest_height*) se může vytvořit efekt roztažení (resp. smrštění) původní bitmapy. Jsou zde navíc totiž parametry *source_width, source_height* do kterých zadáme rozměry zdrojového obrázku a *dest_width, dest_height*, do kterých se zadájí cílové rozměry.

Pak je zde k dispozici funkce *masked_blit,* která pracuje stejně jako funkce *blit* s tím rozdílem, že ignoruje průhledné pixely. K té je také alternativa s roztažením (resp. smrštěním) a to funkce *masked_stretch_blit*.

2.2.5.3 Transformace a zobrazování spritů

Sprite (někdy též sprajt) je označení používané v počítačové grafice pro většinou malý dvourozměrný obrázek. U interaktivních prvků, které se mají hýbat, každý sprite definuje jednu animační fázi prvku. Jejich obměňováním je vytvářena iluze pohybu.

Obsah spritu je (jako u každého rastrového obrázku) definován uvnitř obdélníku, ale pro naprostou většinu pohyblivých objektů se tento obdélník nevykresluje celý – jeden barevný atribut byl rezervován jako tzv. alfa kanál („průhledná barva“) – při vykreslování spritu do grafické paměti oblasti s touto barvou nepřepisuje její původní hodnoty. [14]

Funkcí *draw_sprite* (parametry *BITMAP *bmp, BITMAP *sprite,* **int** *x,* **int** *y*) se vykreslí kopie zdrojové bitmapy *sprite* do cílové bitmapy *bmp* na pozici specifikovanou souřadnicemi v posledních dvou parametrech. Funkce pracuje skoro stejně jako funkce *blit* s tím rozdílem, že používá při vykreslování masku, která ignoruje transparentní pixely. Místo toho jako transparentní barvu určuje v 256 barevné bitmapě bajty s hodnotou 0 a v true color barvách růžovou barvu (maximum červená a modrá, nulová zelená).

K roztažení nebo smrštění se může použít funkce *stretch_sprite* (parametry *BITMAP *bmp, BITMAP *sprite,* **int** *x,* **int** *y,* **int** *w,* **int** *h*), kde se zadávají navíc cílové rozměry spritu. Dále lze sprite vykreslit vertikálně převrácený funkcí *draw_sprite_v_flip*, horizontálně převrácený funkcí *draw_sprite_h_flip* nebo převrácený v obou osách funkcí *draw_sprite_vh_flip*.

Funkce *draw_trans_sprite* zase slouží k míchání barev ze spritu s definovanými barvami. K tomu používá globální tabulku barev pro míchání barev při režimu 256ti barev nebo mixovací funkci při vyšším počtu barev. Tuto tabulku lze předtím vytvořit funkcí *create_trans_table*. Takto se může napsat do kódu:

```
 8 COLOR_MAP global_trans_table;
 9 create_trans_table(&global_trans_table, my_palette,
10                    128, 128, 128, NULL);
11 /* nějaký kód mezi tím */
12 if (get_color_depth() == 8)
13    color_map = &global_trans_table;
14 else
15    set_trans_blender(128, 128, 128, 128);
16
17 draw_trans_sprite(buffer, ghost_sprite, x, y);
```

Funkcí *draw_lit_sprite* se zase může vykreslit sprite s efektem osvětlení. Funguje to podobně jako v předchozím případě s tím rozdílem, že hodnotu stupně osvětlení lze zadát do posledního parametru. Tedy v 256 barevném režimu se definuje pro efekt globální tabulku (v tomto případě funkcí *create_light_table*), při vyšším počtu barev pak mixovací funkci. Toto se provede např. následovně:

```
18 COLOR_MAP global_light_table;
19 create_trans_table(&global_trans_table, my_palette,
20                    128, 128, 128, NULL);
21 /* nějaký kód mezi tím */
22
23 if (get_color_depth() == 8)
24    color_map = &global_trans_table;
25 else
26    set_trans_blender(40, 40, 255, 255);
27
28 draw_lit_sprite(buffer, colored_cape, x, y, 64);
```

Poslední z řady vykreslovacích funkcí se speciálním světelným efektem je funkce draw_gouraud_sprite, pomocí které na sprite použije se Gouraudovo stínování.

Princip tohoto stínování spočívá v tom, že pokud budou známi barevné odstíny všech vrcholů ploch tohoto objektu (tj. normál, ze kterých se pak dá barva dopočítat), pak lze tyto vrcholy propojit a barvy jednotlivých pixelů této úsečky pak vypočítat interpolací. Ve chvíli, kdy jsou obarvené všechny hrany, které ohraničují danou plochu, pak stačí vybrat jednu z os, po které se bude tato plocha procházet a lineární interpolací se zjistí barva

pixelů, které se nacházejí na úsečce ležící mezi dvěma body na hranách plochy. Tím je celá plocha pokryta barvou. [15]

Funkce má v parametrech oproti předchozí na konci místo jedné barvy 4, tedy každou pro jeden vrchol obdélníka spritu a její použití může vypadat následovně:

```
1  COLOR_MAP global_light_table;
2  create_trans_table(&global_trans_table, my_palette,
3                     128, 128, 128, NULL);
4  /* nějaký kód mezi tím */
5  if (get_color_depth() == 8)
6     color_map = &global_trans_table;
7  else
8     set_trans_blender(0, 0, 0, 128);
9
10 draw_gouraud_sprite(buffer, menacing_spy, x, y,
11                       light_strength_on_corner_1,
12                       light_strength_on_corner_2,
13                       light_strength_on_corner_3,
14                       light_strength_on_corner_4);
```

V Allegru jsou přítomny spousty funkcí na různé způsoby rotace spritů kolem svého středu. Funkce *rotate_sprite* slouží k otočení kolem o zadaný úhel, funkcí *rotate_sprite_v_flip* se může udělat totéž s tím rozdílem, že se sprite nejprve otočí okolo vertikální osy.

Funkcí *rotate_scaled_sprite* se dá s otočením změnit i měřítko a funkcí *rotate_scaled_sprite_v_flip* se provede to samé, jen se předtím obrátí sprite vertikálně.

Také lze rotovat kolem jiného bodu než kolem středu spritu a to funkcí pivot_sprite, které se zadá navíc souřadnice bodu, kolem kterého se má rotovat. K této funkci jsou rovněž alternativy stejné jako u funkcí *rotate_**.

2.2.5.4 Ukládání bitmapy do souboru

Bitmapa ze struktury *BITMAP* se může uložit do souboru funkcí *save_bitmap*, které se při volání zadají jako parametry název souboru, bitmapa, kterou chceme uložit a barevná paleta.

Může se např. takto vytvořit obrázek z aplikace funkcí create_sub_bitmap (zadá se bitmapa, kterou chceme "vyříznout" a rozměry obdélníku, který chceme vzít) a ten poté se může uložit do souboru. Kód by byl přesně následující:

```
15 BITMAP *bmp;
16 PALETTE pal;
17
18 get_palette(pal);
19 bmp = create_sub_bitmap(screen, 0, 0, SCREEN_W, SCREEN_H);
20 save_bitmap("dump.pcx", bmp, pal);
21 destroy_bitmap(bmp);
```

2.2.6 Vektorové kreslení:

- pixely, čáry, pravoúhelníky, kružnice, elipsy, oblouky, Bezierovy křivky
- vyplněné, s nebo bez vzorku
- mnohoúhelníky: ploché, Gouraud, texturované (3D) a průsvitné

2.2.7 Práce s textem

Allegro disponuje podporou fontů v GRX, 8x8 nebo 8x16 BIOS formátu s příponou *.ftn. Podporuje také fonty z bitmapových obrázků a samozřejmě také fonty zabalené utilitou Grabber (viz. níže) do datového souboru *.dat. K použití TrueType fontů je potřeba použít rozšíření (taktéž viz. níže). Výchozím kódováním textu je v Allegru UTF-8.

Fonty lze načítat funkcí *load_font*, kde zadáme v parametrech za sebou název souboru s fontem, dále barevnou paletu (jako pole struktur *RGB*) a do posledního parametru můžeme dát specifické informace o načítaném formátu. Obvykle se však do tohoto parametru předá pouze hodnota *NULL*.

Texty lze poté vypisovat funkcí *textout_ex* (parametry *BITMAP *bmp*, **const** *FONT *f*, **const char** *s*, **int** *x*, **int** *y*, **int** color, **int** *bg*), kde do parametru *bmp* se zadá cílová bitmapa (obvykle přímo *screen*), do které se bude vypisovat text, do dalšího parametru se předá načtený font a nekonec řetězec, který se vypíše. Následně se zadají souřadnice, na které je potřeba text vypsat, barvu textu a barvu pozadí textu.

Tato funkce vypíše text zarovnaný zleva. Může se však funkcí *textout_right_ex* vypsat text zarovnaný zprava, či funkcí *textout_centre_ex* zarovnaný na střed nebo dokonce funkcí *textout_justify_ex* text zarovnaný do bloku.

Také lze použít funkci *textprintf_ex* (parametry *BITMAP *bmp*, **const** *FONT *f*, **int** *x*, **int** *y*, **int** *color*, **int** *bg*, **const char** **fmt*, ...) s nekonečným počtem parametrů, které se zadají do parametru *fmt* jako formátovaný řetězec stejným stylem jako např. u standardní funkce *printf*. Tato funkce má stejné alternativní varianty pro zarovnávání textu jako funkce *textout_ex*.

2.2.8 Převody barevných formátů

Allegro disponuje funkcemi na převod mezi barevnými formáty RGB a HSV. Funkcí *hsv_to_rgb* se provede převod z formátu HSV do RGB a funkcí *rgb_to_hsv* opačně. Převod je možno uskutečnit následovně:

```
22 int r, g, b;
23 float hue, saturation, value;
24
25 /* Konverze z RGB do HSV formátu. */
26 rgb_to_hsv(255, 0, 128, &hue, &saturation, &value);
27
28 /* Konverze z HSV do RGB formátu. */
29 hsv_to_rgb(hue, saturation, value, &r, &g, &b);
```

2.2.9 Práce se zvukem

Zvuk se v Allegru inicializuje funkcí *install_sound* (parametry **int** *digi*, **int** *midi*, **const char** **cfg_path*). Do parametru digi se obvykle dává hodnota DIGI_AUTODETECT a do parametru midi hodnota MIDI_AUTODETECT, případná jiná hodnota se pak nasazuje pouze, pokud by byl problém s ovladači zvukové karty. Poslední parametr se ignoruje a je zde jen kvůli zpětné kompatibilitě se starší verzí Allegra.

Hlasitost zvuku můžeme nastavit funkcí *set_volume* zvlášť pro MIDI zvuk (druhý parametr) a ostatní (první parametr).

Zvuky můžeme nahrávat ze souboru funkcí *load_sample* (parametrem je název souboru) a to ve formátech WAV nebo VOC. K uložení zvuku do paměti se používá struktura SAMPLE, taká je definována následovně:

```
30 typedef struct SAMPLE
31 {
32    int bits;                   /* 8 nebo 16 bitů */
33    int stereo;                 /* mono nebo stereo */
34    int freq;                   /* frekvence zvuku */
35    int priority;               /* priorita 0 až 255 */
36    unsigned long len;          /* délka */
37    unsigned long loop_start;   /* pozice na začátku */
38    unsigned long loop_end;     /* pozice na konci */
39    void *data;                 /* data */
40 };
```

Další zvukový formát můžeme zaregistrovat funkcí *register_sample_file_type* (parametry **const char** **ext*, *SAMPLE* *(**load*)(**const char** **filename*), **int** (**save*)(**const char** **filename*, *SAMPLE* **spl*)) např. následovně:

```
41 SAMPLE *load_mp3(const char *filename)
42 {
43    /* kód s načtením formátu do struktury SAMPLE */
44 }
45
46 register_sample_file_type("mp3", load_mp3, NULL);
```

Přehrát zvuk se poté může přehrát funkcí *play_sample* (parametry **const** *SAMPLE *spl*, **int** *vol*, **int** *pan*, **int** *freq*, **int** *loop*), kde se zadá do parametru *spl* požadovaný zvuk, do parametru *vol* hlasitost a do parametru *pan* vyváženost stran pro stereo. V obou případech jsou hodnoty od 0 (ztlumeno/zcela vlevo) do 255 (maximum/zcela vpravo). Do parametru freq se předá frekvence a v parametru loop se určí, zda se má zvuk přehrávat opakovaně.

V případě, že se zadá opakované přehrávání zvuku, ukončí smyčku přehrávání funkce *stop_sample* (do parametru se dá pouze ukazatel na strukturu *SAMPLE*).

2.3 Rozšiřující knihovny

2.3.1 AllegroGL

AllegroGL je rozšíření Allegra, které umožňuje použití OpenGL API společně s kódem Allegra. Umožňuje používat OpenGL pro vykreslování obrazu a Allegro se přitom stará o vstupní zařízení, časovače, multiplatformní kompatibilitu, načítání dat a vykreslování textur.

AllegroGL také zpřístupní většinu ne-li všechna dostupná rozšíření OpenGL. To znamená, že se již nemusí ručně načítat, pokud je potřeba jej použít, management rozšíření to udělá totiž za programátora.

AllegroGL je rozšířením pro Allegro ve verzi 4.2.x až 4.4.x, ve verzi 5 však již nebude, protože zde bude OpenGL integrováno přímo ve standardním Allegru, takže jeho budoucnost nebude již příliš dlouhá. Jeho nejnovější verze vyšla 12. listopadu 2007 a nese označení 0.4.3. [16]

2.3.2 AllegTTF

AllegTTF je knihovna rozšiřující Allegro o funkce načítání TrueType fontů a jejich zobrazování s vyhlazením pomocí anti-aliasingu. Knihovna je založena na projektu Freetype. V případě 8bitového grafického módu používá AllegTTF 6 různých úrovní anti-aliasingu. V případě vyšší barevné hloubky je použito 256 úrovní vyhlazování. Bez knihovny Allegro je tato knihovna samostatně nepoužitelná. [17]

2.3.3 Glyph Keeper

Glyph Keeper je knihovna pro zobrazování textu. Pomáhá programu s načítáním fontu, zpracováním nejrůznějších glyfů a jejich zobrazení na cílovou bitmapu. Dokáže zobrazovat vyhlazené, průsvitné otočené či šikmé texty. Glyph Keeper je volně šiřitelný, přenosný, výkonný a snadno použitelný. [5]

Glyf je grafické znázornění grafému (písmena, číslice, znaku, piktogramu, interpunkčního a jiných znamének). Původ slova je ze starořeckého slova γλυφή (*glyphein* - tesat). Jeho vytvářením vzniká např. písmo. Pojem glyf je používán jako přípona pro označení různých druhů písem, které jsou založeny zejména na piktogramech (hieroglyf, petroglyf). []

2.4 Nástroje

Allegro není jen knihovna funkcí, ale obsahuje také několik užitečných nástrojů, které mohou velmi usnadnit tvorbu aplikací.

2.4.1 Makedoc

Nejen zkompilovaná knihovna je vytvořena ze zdrojových kódů Allegra. Dokumentace Allegra je z nich rovněž vytvořena a to nástrojem Makedoc, který zpracovává kód a dokumentační komentáře, ze kterých poté zpracuje výstup.

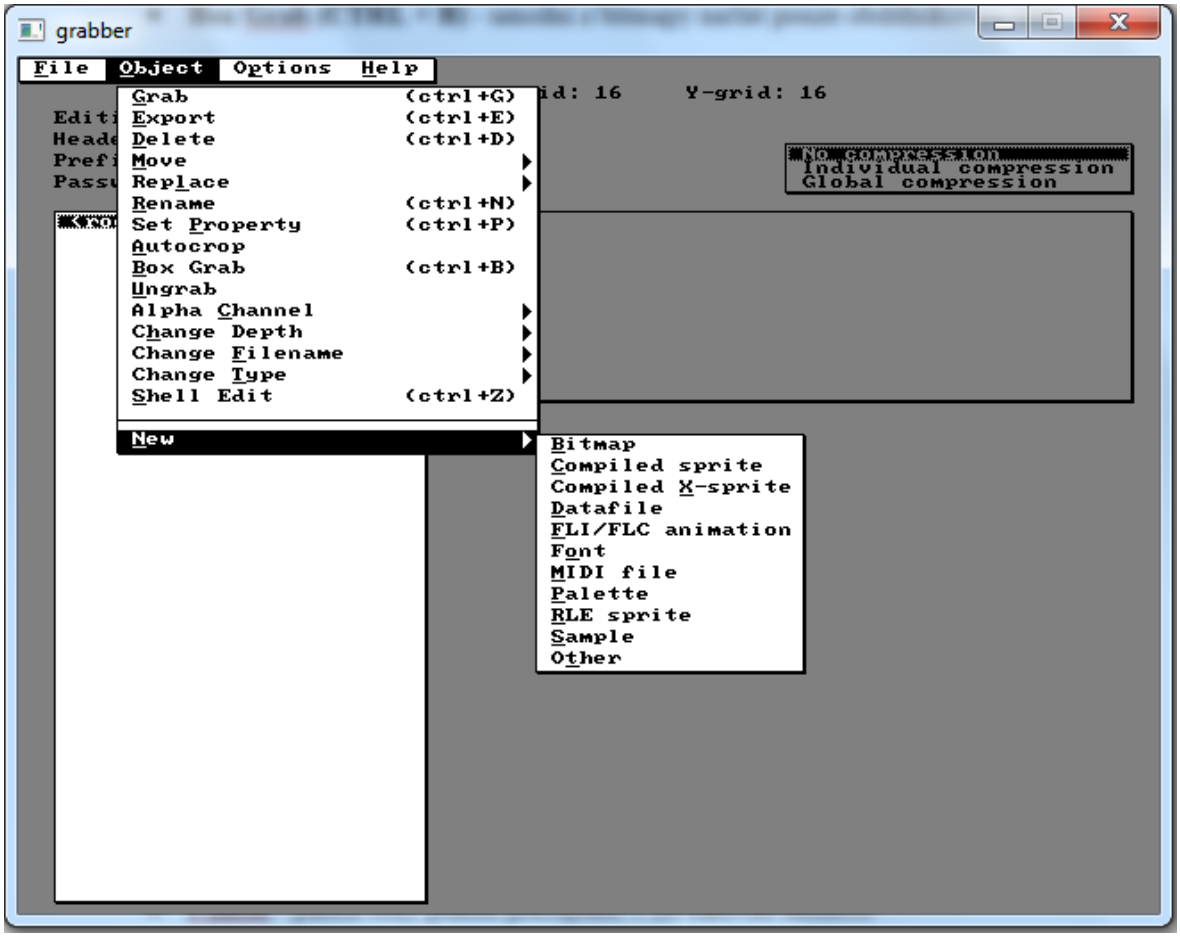
Jako výstupy jsou podporovány formáty TXT, RTF, HTML, 3 (Unixové manuálové stránky), TEXI, INFO, CHM, DVI (Device independent documentation file), PostScript, PDF, Devhelp (HTML + XML index format) a API (SciTE API formát). [5]

2.4.2 Grabber

V allegru je možné zpracovávat tzv. datafile. Jsou to datové soubory s příponou .dat a jsou jakousi obdobou archivů, takže do nich jdou zabalit nejrůznější data, která potřebuje aplikace načítat.

A ke tvorbě a editaci těchto souborů slouží právě nástroj Grabber, který je standardní součástí Allegra. Vývojové prostředí programu je uspořádáno následovně:

- Nabídka programu (vlevo nahoře)
- Seznam objektů (vlevo)
- Parametry (Informace) datového souboru (vlevo mezi seznamem a nabídkou)
- Nastavení programu (uprostřed nahoře)
- Informace objektů na kterém je pozice kurzoru v seznamu (uprostřed vpravo)
- Náhled objektu (ukázat obrázek, přehrát zvuk) (vpravo dole)



Obrázek 2 Nástroj na správu souborů *.dat Grabber

Nabídka obsahuje tři podmenu - File, Objekt a Help. Popíšeme si je postupně za sebou a pouze ty nejdůležitější funkce. [8]

2.4.2.1 Nabídka File

- **Load (CTRL + L)** - načte soubor se jménem datového souboru a pokud neexistuje, vytvoří nový soubor
- **Save (CTRL + S)** - uloží datový soubor
- **Save Stripped** - uloží pouze určitou část datového souboru podle výběru v dialogu, který se po kliknutí zobrazí
- **Merge** - přidá jiný datový soubor k tomu, který je již otevřený, tedy sloučí dva a více souborů dohromady
- **Update (CTRL + U)** - obnoví obsah datového souboru
- **Update Selection** - obnoví pouze jeden aktuální objekt, na kterém je kurzor

- **Read Bitmap (CTRL + R)** - načte do paměti obrázek, se kterým pak můžeme pracovat
- **Viev Bitmap (CTRL + V)** - umožní náhled obrázku, který jsme si načetli do paměti (pokud jsme nějaký načetli) nebo nic

2.4.2.2 Nabídka Object

- **Grab (CTRL + G)** - nahraje do nově vytvořeného objektu data jako obrázek, zvuk atd.
- **Export (CTRL + E)** - naopak z objektů, kde máme obrázek či zvuk, můžeme data uložit do nového souboru např. bmp, wav - tedy exportovat
- **Delete (CTRL + D)** - smaže objekt se seznamu a tedy i z datového souboru
- **Rename (CTRL + N)** - přejmenuje objekt
- **Box Grab (CTRL + B)** - umožní z bitmapy načíst pouze obdélníkovou část
- **Ungrab** - přesný opak načtení, tedy objekt v seznamu bude opět prázdný
- **Change Type** - změní typ objektu např. z bitmapy na sprite
- **Color Depth** - změní typ barevné hloubky (např. z 8bitů na 32bitů)

2.4.2.3 Nabídka New

- **Bitmap** - vytvoří objekt, do kterého pak můžeme nahrát bitmapu
- **Compiled Sprite** - vytvoří objekt, do kterého pak lze nahrát zkompilovaný sprite
- **Datafile** - do datového souboru může přidat další datový soubor jako objekt
- **Font** - vlastní písma, které jsou všechny na jednom obrázku
- **MIDI file** - zvuk ve formůtu MIDI
- **Palette** - paleta tedy pokud se pracuje v 8bitové barevné hloubce
- **Sample** - zvuk, např. wav
- **Other** - binární data v jiném formátu než výše uvedené

3 OBJEKTOVÉ FRAMEWORKY

Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovnu API, návrhové vzory nebo doporučené postupy při vývoji.

Cílem frameworku je převzetí typických problémů dané oblasti, čímž se usnadní vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání. Například, tým, který používá Apache Struts k vývoji webových stránek pro banku, se může zaměřit na to, jak se budou provádět bankovní operace a ne jak zajistit bezchybnou navigaci mezi jednotlivými stránkami.

Vyskytují se námitky, že použitím frameworku bude kód pomalý či jinak neefektivní a že čas, který se ušetří použitím cizího kódu, se musí věnovat nastudování frameworku. Nicméně při jeho opakovaném nasazení nebo ve velkém projektu dojde k výrazné úspoře času. Při odinstalování frameworku již nebude možno některé aplikace spustit.

Framework se skládá z tzv. frozen spots a hot spots. Frozen spots definují celkovou architekturu softwarové struktury, její základní komponenty a vztahy mezi nimi. Tyto části se nemění při žádném použití frameworku. Naproti tomu hot spots jsou komponenty, které spolu s kódem programátora vytvářejí zcela specifickou funkcionalitu, a proto jsou skoro pokaždé jiné.

V objektově orientovaném prostředí je framework tvořen abstraktními a klasickými (neabstraktními) třídami. Hot spots pak mohou být reprezentovány abstraktními třídami a vlastní kód se přidá implementací abstraktních metod. [12]

Pro Allegro bylo vytvořeno několik více či méně povedených frameworků jakožto objektových nadstaveb zapouzdřujících funkce Allegra do tříd v jazece C++ a zefektivnit tak vývoj aplikací nad tímto API. Allegro je napsáno procedurálně v jazyce C a obsahuje velké množství globálních proměnných a funkcí, což může znamenat zbytečné problémy při vývoji aplikace. Vybral jsem z nich několik nadstaveb, které jsem shledal jako oblíbené a používané [5] [9] [10] [12] [13] a stručně je popsal.

3.1 OpenLayer

Nejvíce programátory preferovaným frameworkem nad knihovnou Allegro se ukázal být OpenLayer. [13] OpenLayer je 2D grafická knihovna rozšiřující knihovnu Allegro a některé její doplňky. Specifikuje nové přepracované API, které k vykreslování 2D grafiky využívá hardwarovou akceleraci přes funkce rozhraní OpenGL, resp. přes standardní rozšíření AllegroGL, které tuto podporu zavání pro Allegro. Díky tomu dosahuje toto vykreslování vyšší rychlosti než konvenční vykreslování v obyčejném Allegru. [10]

Knihovna vyžaduje a také přímo v instalačním balíčku obsahuje GNU nástroje, OpenGL knihovnu, originální Allegro, rozšíření AllegroGL, Glyph Keeper (pro Allegro i AllegroGL), Freetype knihovnu pro podporu Freetype fontů, PNG knihovnu pro podporu tohoto formátu a knihovnu Zlib, pro podporu komprese a dekomprese formátu ZIP.

Nejnovější verze frameworku OpenLayer nese označení 2.1 a byla vydána 18. července 2007 pod licencí LGPL. [10]

3.2 Allegro Simplificator

Allegro Simplificator je C++ wrapper pro multimediální knihovnu Allegro. Zapouzdřuje mnoho běžně používaných funkcí Allegra do tříd a přináší také trochu nové funkcionality, především pak multiplatformní komunikaci po síti. Obsahuje také spoustu příkladů a rozsáhlou dokumentaci. [11]

Tento framework měl být původně využit pro tvorbu vlastní aplikace, avšak při vyhledávání informací o něm byl nakonec objeven [13] již výše zmíněný OpenLayer, který je řádově vyspělejší, i když u něj dokumentace je na trochu horší úrovni.

3.3 AXL Framework

AXL je soubor knihoven, který se skládá ze třech částí, které lze nezávisle přilinkovat. Modul GRANI slouží k práci s grafikou a animacemi, modul CONCUR je určen pro konfiguraci a inicializaci a FRAMS je kompletním frameworkem pro vytváření her. [5]

I když na první pohled vypadá zajímavě, byl pro další práci tento framework zavrhnut z důvodů nezvyklého, příliš jednostranně zaměřeného chování, neintuitivního a nečistého (míchání objektového s procedurálním programováním) uspořádání tříd a metod API.

4 VYUŽITÍ ALLEGRA

Allegro se využívá především ke tvorbě her. Na komunitních webových stránkách www.allegro.cc [5] je seznam oficiálně vytvořených her pod Allegrem. Aktuálně je mezi nimi 252 akčních her, 22 adventur, 216 arkádových her, 30 stolních her, 139 skládaček, 25 sportovních her, a 58 strategických her.

Je zde vytvořeno několik remaků, tj. nově přepracovaných starších her, které jsou známé, jakou je Dune II, Zelda, UFO: Enemy Unknown a několik variací na oblíbené arkádové hry Tetris, Asteroids, Pong, Pacman, Snake a Arcanoid. Jsou zde také deskové hry jako Scrabble, karetní hry atd. [5]

Obrázek 3 Dune II The Maker

Také zde bylo zveřejněn jeden titul ve 3D - bojová hra Mythic Blades. Z grafického hlediska jde zřejmě o nejpropracovanější hru, která byla na knihovně Allegro (resp. AllegroGL s podporou OpenGL) vytvořena. Z toho je také patrné, že Allegro je knihovna

určená na tvorbu jednodušších her především ve 2D a pro náročnější hry ve 3D a pro profesionální herní studia je výhodnější použití jiných technologií. [5]

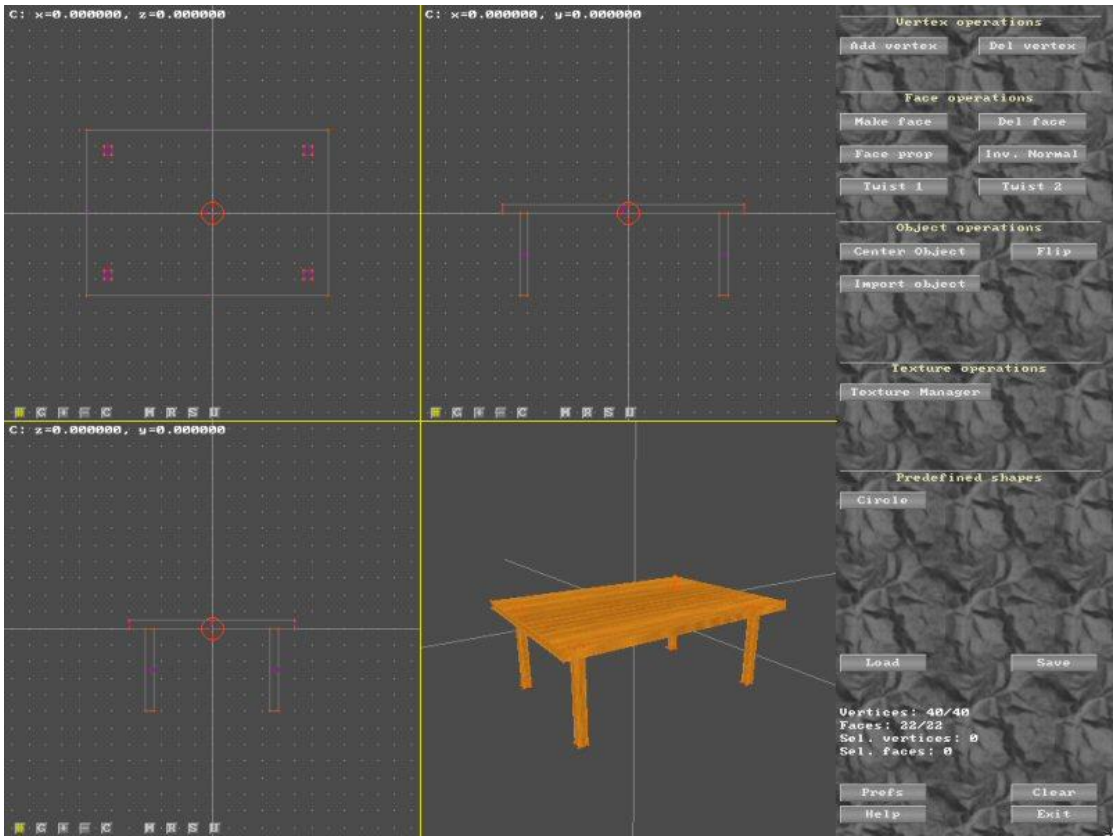
Obrázek 4 3D bojová hra Mythic Blades

Allegro a jeho rozšíření se však nepoužívají pouze ke tvorbě her. Uplatnění lze nalézt v široké škále multimediálních aplikací. Na komunitním webu je např. řada editorů. [5]

Jedním z nich je freeware 3D editor Escultor, který je určen pro tvorbu a úpravy modelů s nízkým počtem polygonů vhodných k použití ve hrách, kde je počet polygonů zásadní pro hardwarovou náročnost hry. Dokáže pracovat s modely ve formátech EMD a OBJ (formát Blenderu).

Dalším příkladem je Polygonize! 3D-Editor - jednoduchý freeware editor pro editaci jednoduchých modelů. The editor provides flat coloured, gouraud shaded and textured faces. Není limitován počtem vertexů, faců nebo textur. Editor má také jednoduchý management sekcí projektu. K editoru volně ke stažení k dispozici knihovna tříd v C++, které je určena k načítání modelů do tvořených aplikací, které by využívaly výstupy z tohoto editoru. Editor je možné využít ke tvorbě freeware i komerčních produktů, je však nutné se o jeho použití zmínit někde v tomto produktu.

Obrázek 5 editor modelů Escultor



Obrázek 6 Polygonize! 3D-Editor

V Allegru je také vytvořen Diablo II Map Editor. Jedná se o WYSIWYG editor prostředí do známé hry Diablo II a jejího rozšíření Lord of Destruction. K jeho používání je potřeba znalostí technologie Diablo II Moding.

Autor této aplikace Paul Siramy popisuje její vývoj jako nikdy nekončící úkol, který nikdy nebude dost dobrý. Především, když je stále několik neznámých položek v DS1 formátu. Není tedy bohužel možné kompletní layout úrovní, protože to vyžaduje zásah do DLL knihoven, jejichž struktura není dobře veřejně známa. I přesto jde o jeden z nejpropracovanějších projektů postavených na knihovně Allegro.

Nejnovější verze Diablo II Map Editoru nese označení 1.00b a byla vydána 24. ledna 2007. Je k dispozici ve 2 variantách a to jako stand alone, tedy samostatně provozuschopnou bez dalšího potřebného softwaru a jako variantu s vyžadovanou instalací hry Diablo II. [1]

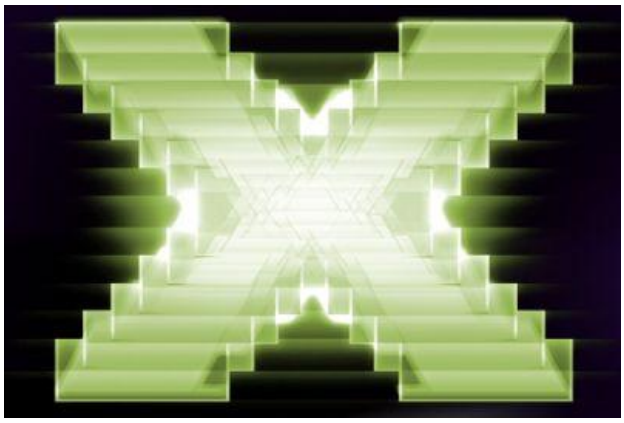
Obrázek 7 Diablo II Map Editor

5 ALTERNATIVNÍ MULTIMEDIÁLNÍ API

API (**Application Programming Interface**) označuje rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství. Jde o sbírku procedur, funkcí či tříd nějaké knihovny (ale třeba i jiného programu nebo jádra operačního systému), které může programátor využívat. API určuje, jakým způsobem se funkce knihovny volají ze zdrojového kódu programu.

5.1 DirectX

Microsoft DirectX je soubor rozhraní pro programování aplikací (API) pro manipulaci s úkoly související s multimedii, a to zejména pro programování her a video na platformách Microsoft. Původně jména těchto rozhraní API všechna začínala Direct, např. Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound, a tak dále. Jméno DirectX bylo vytvořeno jako zkrácený termín pro všechna tato API (X pro konkrétní jména API) a brzy se stalo názvem sbírky. Když Microsoft později vyvinul herní konzole, bylo X použito jako základ jména Xbox a konzole je založena na technologii DirectX. [1] X původní bylo použito na pojmenování API určených pro Xbox jako XInput a Cross-platform Audio Creation Tool (XACT), zatímco model DirectX byl i nadále pro Windows API, jako je Direct2D a DirectWrite.



Obrázek 8 Logo DirectX

Direct3D (3D grafika API v DirectX) je široce používáno ve vývoji videoher pro Microsoft Windows, Microsoft Xbox a Microsoft Xbox 360. Direct3D se používá také v jiných softwarových aplikacích pro vizualizaci a grafické úlohy, např. CAD / CAM aplikace ve

strojírenství. Direct3D je veřejně nejpopulárnější součástí DirectX, tudíž jsou často názvy "DirectX" a "Direct3D" zaměňovány.

Software DirectX Development Kit (SDK) se skládá z runtime knihoven v distribuční binární formě spolu s průvodní dokumentací a hlavičkovými soubory k dalšímu programování. Původně byly runtime knihovny nainstalovány pouze s hrami nebo explicitně uživatelem. Windows 95 původně nebyl vypuštěn s integrovaným DirectX, ale DirectX již byla součástí Windows 95 OEM Service Release 2 [2]. Windows 98 a Windows NT 4.0 byly oba dodávány s DirectX a po nich se všechny vydané verze Windows. SDK je k dispozici k bezplatnému stažení. Zatímco Runtimes jsou proprietární s uzavřeným zdrojovým kódem, zdrojový kód je k dispozici ve formě SDK (Software Development Kit) nástrojů.

Direct3D 9Ex, Direct3D 10 a Direct3D 11 jsou oficiálně k dispozici pouze pro Windows Vista a Windows 7, protože tyto nové verze jsou závislé na novém zobrazovacím systému Windows Driver Display Model, který byl zaveden pro systém Windows Vista. Nová grafická architektura Vista/WDDM zahrnuje novou správu grafické paměti, která podporuje virtualizaci grafického hardwaru pro více aplikací a služeb, jako je např. Desktop Window Manager.

5.1.1 Historie

Koncem roku 1994 byl Microsoft na pokraji uvolnění svého nového operačního systému Windows 95. Hlavním faktorem, který byl důležitý pro určení hodnoty uživatelů na jejich nového operačního systému bylo, jaké programy by být schopni na něm spustit. Tři zaměstnanci společnosti Microsoft - Craig Eisler, Alex St John, a Eric Engstrom - byli určeni k řešení tohoto problému, protože programátoři měli tendenci vidět předchozí operační systém Microsoftu, MS-DOS, jako lepší platformu pro programování her. Což znamenalo, že by bylo vyvinuto málo her pro Windows 95 a operační systém by neměl velký úspěch.

DOS umožňoval přímý přístup ke grafickým kartám, klávesnici, myši, zvukovým zařízením a všechny ostatní částem systému, zatímco Windows 95 a jeho chráněný paměťový model omezil přístup ke všem z nich a pracuje na mnohem standardizovanějším modelu. Microsoft potřeboval způsob, který bude pro programátory výhodný a potřebovalo ho rychle. Operační systém byl totiž pouhých několik měsíců od vydání. Eisler (lídr

vývoje), St John, a Engstrom (programový manažer), společně pracovali na vyřešení tohoto problému a jako řešení nakonec vzniklo DirectX.

První verze DirectX byla vydána v září 1995 jako Windows Games SDK. Tehdy bylo Win32 nahrazeno za DCI [3] a WinG API pro Windows 3.1. DirectX je ve všech verzích Microsoft Windows, počínaje Windows 95, zahrnuto jako vysoce výkonné multimediální řešení.

DirectX 2.0 se stalo součástí Windows samotných s verzí Windows 95 OSR2 a Windows NT 4.0 v polovině roku 1996. Jak Windows 95 byl sám o sobě stále nový a bylo pro něj vypuštěno málo her. Microsoft se zabýval těžkou podporou DirectX pro vývojáře, kteří byli většinou nedůvěřiví ke schopnosti Microsoftu vybudovat herní platformu Windows. Alex St John, který pracoval jako evangelista pro DirectX, představil v roce 1996 na koncerenci Computer Game Developers Conference spolu s herním vývojářem Jay Barnsonem prezentaci hry s tématem starověkého Říma, včetně skutečných lvů, oblečení a něco co připomínalo domácí karneval [5]. Microsoft nejprve představil Direct3D a DirectPlay a demonstroval multi-playerový MechWarrior 2, jak se hraje přes Internet.

Tým DirectX čelí náročným úkolům - testování každého vydání DirectX proti škále hardware a software. Různé grafické karty, zvukové karty, základní desky, procesory, vstupní zařízení, hry a jiné multimediální aplikace byly testovány s každou beta a finální verzí DirectX. Tým DirectX měl také vestavěné a distribuované testy, které umožnily hardwarovému průmyslu potvrdit, že nový hardwarové designy a ovladače budou kompatibilní s DirectX.

Před DirectX Microsoft zahrnoval OpenGL na své platformě Windows NT. V té době, OpenGL požadoval náročný hardware a byl zaměřen na strojírenství a CAD aplikace. Direct3D byl zamýšlen jako lehký partner k OpenGL, zaměřený na herní využití. Jak 3D hry rostly, OpenGL vyvinul lepší podporu programovacích technik pro interaktivní multimediální aplikace jako jsou hry, což dalo vývojářům možnost volby mezi využitím OpenGL a Direct3D jako 3D grafické API pro jejich aplikace. Na tom místě začalo soupeření mezi zastánci multiplatformního OpenGL a Direct3D určeného pouze pro Windows. Mimochodem OpenGL bylo podporováno DirectX týmem Microsoftu. Pokud se vývojáři rozhodli pro využití OpenGL jako 3D grafického API, ostatní API DirectX bylo

možno kombinovat s OpenGL v počítačových hrách, protože OpenGL nezahrnuje všechny funkce DirectX, jako např. zvukové rozhraní nebo rozhraní pro práci s joystickem.

Ve specifické konzolové verzi byl použit DirectX jako základ pro Xbox společnosti Microsoft a konzolového Xbox 360 API. API bylo vyvinuto společně firmou Microsoft a společností Nvidia, která vyvinula vlastní grafický hardware, který používá původní Xbox. Xbox API je podobné verzi DirectX 8.1, ale je neaktualizovatelný stejně jako ostatní konzolové technologie. Xbox byl kódově pojmenovaný jako DirectXbox, ale toto bylo zkráceno na Xbox jako obchodní jméno [6].

V roce 2002 vydala společnost Microsoft DirectX 9 s podporou pro použití mnohem více shader programů než dříve s pixel a vertex shadery verze 2.0. Microsoft pokračoval a aktualizoval DirectX dále, zavedl Shader Model 3.0 v DirectX 9.0c uvolněného v srpnu 2004.

V dubnu 2005 byl odstraněn DirectX DirectShow, který byl přesunut do Microsoft Platform SDK. DirectX SDK je však stále vyžadováno ke tvorbě DirectShow aplikací.

V roce 2006 se spolu s Vindows Vista (a pouze pro něj) objevilo DirectX 10 s podporou unifikovaných shaderů. Poté 4. února 2008 vyšla verze 10.1, která přidala několik efektů a optimalizovala práci s vyhlazováním a nakonec se objevila spolu se systémem Windows 7 (pro Windows 7 a Vista) dosud nejnovější verze 11, která přidala tesselaci, podporu multi-threadingu a Compute Shader. [7]

5.1.2 XNA framework

Pokud se například píše kód v DirectX, musí se hlídat i spousta dalších potencionálních problémů - pokud uživatel během hry minimalizuje okno a po čase jej zase obnoví, když se vrací do hry, je třeba do paměti grafické karty načíst znovu všechny textury a data, která z ní mezitím vypadla. Na tuto a spoustu dalších maličkostí, které ale mají velký význam, se musí bohužel myslet, když se hra programuje.

Obrázek 9 Logo XNA frameworku

Proto Microsoft vyvinul framework XNA. Dovoluje se soustředit pouze na naši herní logiku, tedy na to, co má hra dělat ve skutečnosti. Programátora nemusí zajímat, na jakém hardware hra poběží. Je jedno, jakou má hráč grafickou kartu a jestli tato karta umí to či ono. Může se plně soustředit na funkcionalitu hry. Hry napsané v XNA mohou běžet jak na PC s Windows XP a vyšším, tak i na herních konzolích XBox 360 (na to ale musí být zakoupené předplatné XNA Creators Club Subscription, které sice není drahé, ale nepodařilo se jej zakoupit z České republiky). Hry pro Windows se mohou v XNA vyvíjet zcela zdarma i komerčně.

Výhodou XNA je to, že spoustu funkcí, které by se v DirectX museli složitě psát, už obsahuje. Kromě podpory pro různé formáty obrázků a 3D modelů je zde i spousta funkcí pro počítání s maticemi a vektory, engine pro přehrávání zvuků a hudby, funkce pro komunikaci po síti atd. [19]

5.1.3 Srovnání DirectX s Allegrem

Na rozdíl od Allegra je DirectX od začátku psáno objektovým způsobem, zatímco Allegro je psáno strukturovaně a využívá pouze ANSI C bez jakýchkoliv rozšíření z C++. Bohužel je ale také natolik komplikované, že proti Allegru je jeho použití v "čisté" podobě pro začínající programátory téměř nemožné.

5.1.3.1 Výhody

- API obsahuje prakticky všechny potřebné funkce a většinou tedy není zapotřebí vyhledávat doplňující knihovny.
- Objektový přístup
- Díky podpoře nejmodernějších technologií jde relativně o nejlepší API pro programování profesionálních 3D her.
- Framework XNA, s jehož použitím prakticky odpadají níže zmíněné nedostatky

5.1.3.2 Nevýhody

- API je příliš komplikované pro začínající programátory a zabere hodně času se naučit s ním pracovat.

- Je potřeba napsat velké množství kódu, aby program vůbec něco dělal.
- API se často mění. A mění se tak, že není zpětně kompatibilní. Tedy nová verze znamená nové ponoření se do manuálů a přepisování programu. Je otázka, zda je vývoj opravdu tak rychlý, nebo je API tak nevhodně navržené, že se musí s každou novou verzí celé změnit.
- U verzí menších než 7 je velmi špatná dokumentace. U verzí novějších pro změnu chybí podpora 2D grafiky. Tedy i 2D hra potřebuje pro svůj běh 3D akceleraci, aby se hýbala rychle.
- Poslední nevýhoda je dostupnost pouze na Windows, DirectX není multiplatformní.

5.2 SDL

SDL je knihovna domovem v Linuxu, ale funguje i na operačních systémech Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS a OS/2. SDL je zkratka ze Simple Direct-media Layer. Tomu odpovídá i struktura knihovny. Knihovna sama je malá a obsahuje jen základní věci. Díky tomu je přehledná a nezahrnuje programátora gigantickým API. Například sama umí načítat obrázky pouze ve formátu BMP. K SDL pak existuje řada rozšiřujících knihoven. [20]

Mezi oficiální rozšíření patří:

- SDL_net pro tvorbu síťových her.
- SDL_image pro načítání obrázků rozličných formátů
- SDL_mixer pro lepší práci se zvukem
- SDL_ttf pro podporu TTF fontů
- SDL_rtf pro jednoduché renderování RTF
- SDL_bitmap pro lepší práci s bitmapami

Obrázek 10 Logo knihovny SDL

Simple DirectMedia Layer (SDL) je multiplatformní multimediální knihovna poskytující nízkoúrovňový přístup na audio, klávesnici, joystick, 2D a 3D počítačovou grafiku přes OpenGL. Napsaná je v jazyce C, nicméně díky tzv. language bindings je možné knihovnu použít také v C++, Javě, Ada, C#, D, Eiffel, Erlang, Euphoria, Guile, Haskell, Java, Lisp, Lua, ML, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk a Tcl. Současná verze 1.2.14 pro Microsoft Windows využívá knihovnu DirectX verze 7. Samotná knihovna obsahuje jen základní věci, na vše ostatní jsou potřebné doplňující knihovny (*SDL_image*, *SDL_ttf*, *SDL_net*, *SDL_Sound*, *SDL_Mixer*, *SMPEG* a další).

SDL má ve svém názvu slovo "layer" (vrstva) proto, že je to ve skutečnosti wrapperem nad funkcionalitou specifickou pro konkrétní operační systém. Na platformách s X11 používá SDL Xlib pro komunikaci s X11 systémem pro grafiku a události. Na Mac OS X používá SDL Quartz.

Knihovna SDL je propojitelná s téměř libovolným existujícím programovacím jazykem, od populárních C++, Perl, Python (skrze pygame), Pascal atd. k méně známým, jako například Euphoria či Pliant. Toto společně s faktem, že SDL je software s otevřeným zdrojovým kódem a licencované pod LGPL, činí SDL obvyklou volbou pro vývoj mnoha multimediálních aplikací.

SDL samotné je velmi jednoduché; funguje jako tenký multiplatformní wrapper poskytující podporu pro 2D operace s pixely, zvuk, přístup k souborům, zpracování událostí, časování, vlákna atd. Je často použito jako doplněk OpenGL k nastavení grafického výstupu a poskytnutí vstupu z klávesnice a myši, což je za předmětem zájmu OpenGL. [20]

5.2.1 Historie

Sam Lantinga vytvořil knihovnu během své práce pro Loki Software. Nápad dostal při portování Windows aplikace na Macintosh. SDL poté použil pro port Doomu na BeOS. Se SDL spolupracuje i několik dalších svobodných knihoven, jako například SMPEG a OpenAL. [20]

5.2.2 Srovnání s Allegrem

Obě knihovny, jak Allegro tak SDL, jsou multiplatformní knihovny určené pro 2D multimediální aplikace. Obě knihovny jsou napsané procedurálně v jazyce ANSI C a neobsahují objektové a ani žádné jiné prvky z C++, ale je možné je v C++ používat. Zatímco Allegro již většinu potřebných funkcí obsahuje, SDL je v základu velmi "štíhlá" knihovna a k dalším funkcím jsou potřeba instalovat rozšíření.

5.2.2.1 Výhody

- vysoká rychlost (až 14x podle benchmarku) zobrazování oproti klasickému Allegru bez hardwarové akcelerace [17]
- velmi malá velikost knihovny v základu
- pro začátečníky jednodušší volba než DirectX
- na rozdíl od Allegra nepoužívá tolik globálních proměnných

5.2.2.2 Nevýhody

- pro serióznější aplikace je potřeba hned několik doplňků, aby jsme dostali požadovanou funkčnost
- pro začátečníky (subjektivně) trochu obtížnější na naučení než u Allegra
- nepoužívá objektově orientované programování

II. PRAKTICKÁ ČÁST

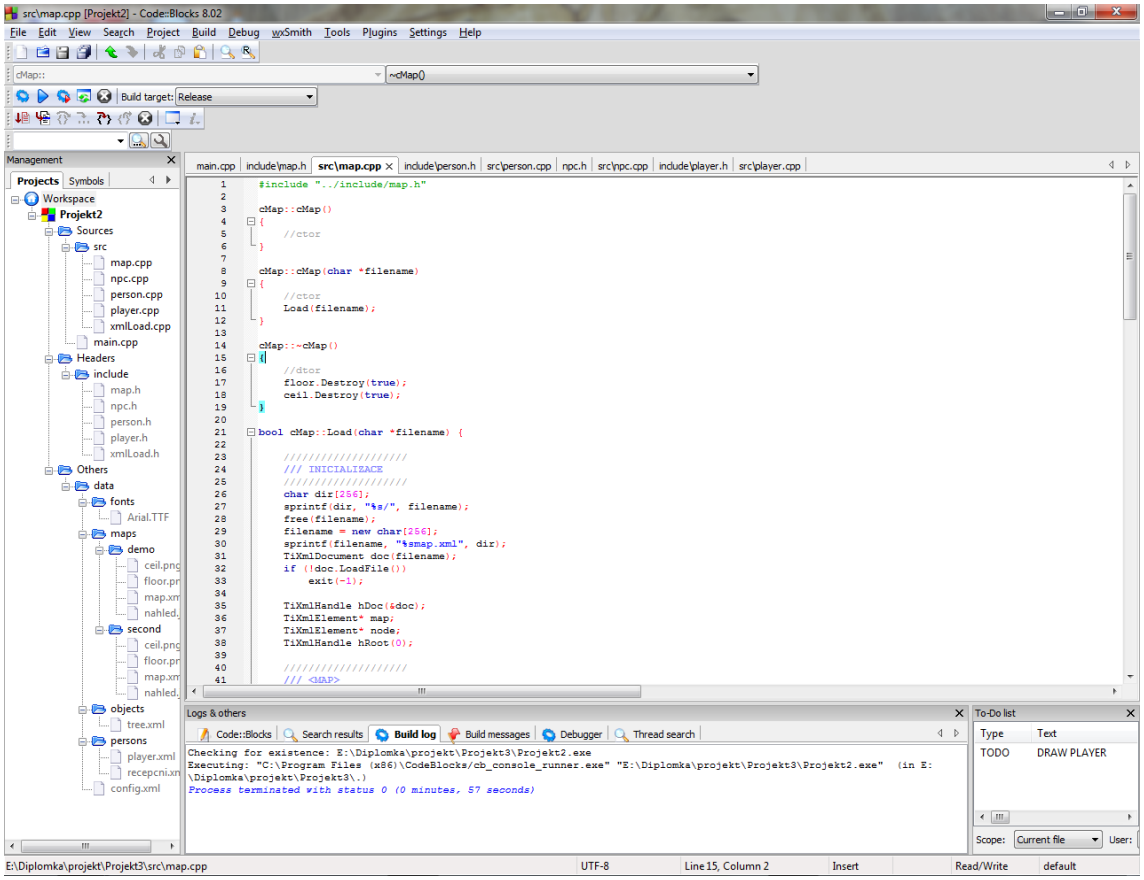
6 NÁVRH A TVORBA APLIKACE

6.1 Instalace a konfigurace vývojového prostředí

6.1.1 Volba a instalace IDE

Jako prostředí pro tvorbu aplikace bylo rozhodnuto použít multiplatformní IDE Code::Blocks, jelikož je volně šiřitelné (pod licencí GPL 3.0), tím pádem si jej může pořídit každý. Vývoj bude popisován na systému Windows, protože je to nejpoužívanější platforma. Jelikož se ale používají multiplatformní nástroje, vývoj na jiných platformách by se příliš nelišil.

V instalátoru pro Windows (které je dostupný volně ke stažení na oficiálním webu na adrese www.codeblocks.org) je přímo obsažen kompilátor MinGW ve verzi 3.4.5, takže instalace proběhne naprosto bez problémů jako jakákoliv jiná instalace běžné aplikace. Code::Blocks ale není závislé na kompilátoru, takže pokud by bylo potřeba použít jiný, např. MSVC, může se nastavit v menu Settings -> Compiler and debuger.



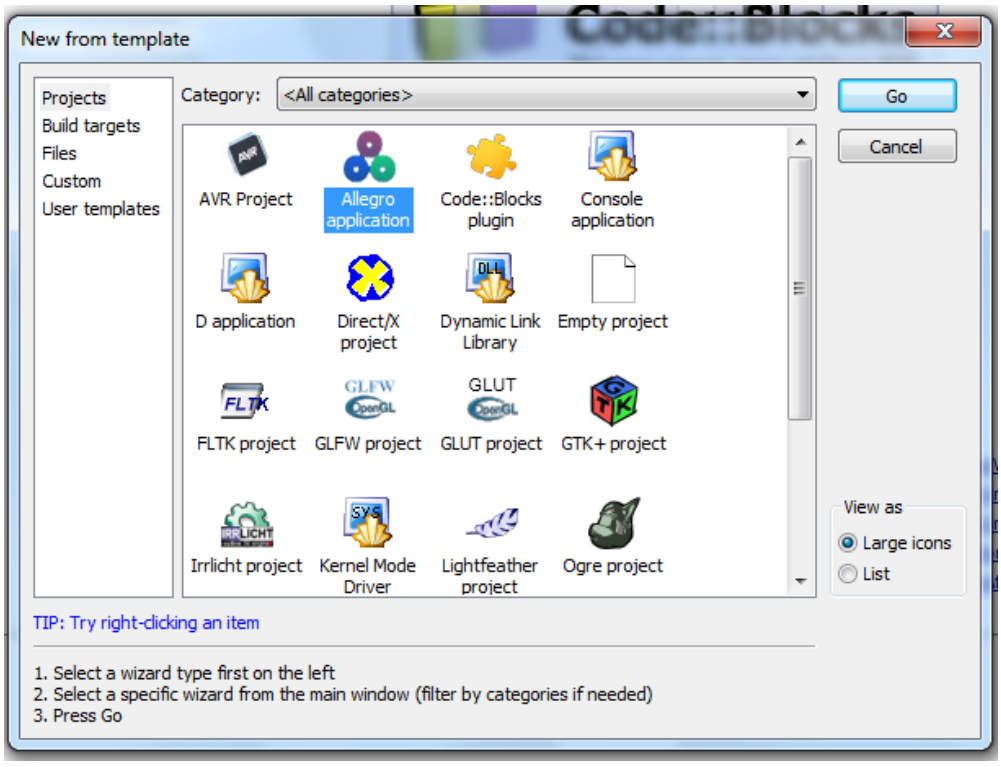
Obrázek 11 Vývojové prostředí Code::Blocks

6.1.2 Instalace Allegra a OpenLayeru

Nejprve je potřeba stáhnout OpenLayer z jeho oficiálního webu openlayer.berlios.de ve verzi 2.1, binární verzi pro MinGW a to verzi s knihovnou GlyphKeeper. Po stažení obsah archivu bude potřeba rozbalit do podadresáře MinGW adresáře, kde je nainstalovaný Code::Blocks. [10]

Poté se do Code::Blocks stáhne a nainstaluje (resp. rozbalí do adresáře programu) Allegro wizard z http://members.allegro.cc/h8or/files/allegro_wizard_codeblocks.zip.

Jakmile se pak otevře prostředí Code::Blocks a dá vytvořit nový projekt (File -> New -> Project), tak se mezi ikonami objeví na výběr i Allegro. Vytvoří se tedy nový projekt a zvolí projekt typu "Hello World!", který obsahuje jen základní inicializaci a výpis jednoduchého textu "Hello World!". [9]



Obrázek 12 Allegro projekt v prostředí Code::Blocks

Poté se ještě musí v projektu upravit nastavení linkeru, aby přilinkoval všechny potřebné knihovny zkompilované v binární podobě, které jsou součástí OpenLayeru. To se udělá tak, že v nabídce Project se zvolí položka Build options, v dialogovém okně se vybere záložka Linker settings a tam se pro režim Debug i Release vloží do textového pole Other linker options následující řádek:

-lopenlayer -lglyph-agl -lfreetype -lpng -lz -lagl -lalleg -luser32 -lgdi32 -lglu32 -lopengl32

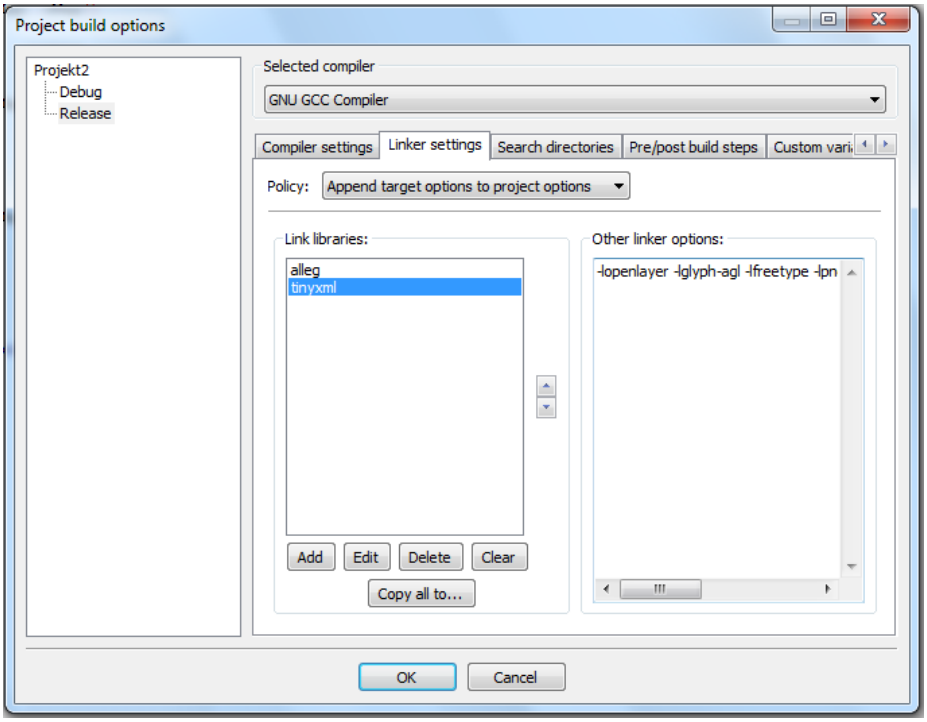
Tím se zajistí přilinkování knihoven OpenLayer, GlyphKeeper, Freetype, PNG, Zlib, Allegro, AllegroGL, GDI, GLUT a OpenGL, které jsou součástí celé knihovny OpenLayer.

6.1.3 Instalace knihovny TinyXML

U projektu bude potřeba načítat a zpracovával soubory ve formátu XML. K tomu je určena knihovna TinyXML, která se hodí především pro zpracovávání menších XML souborů. [4]

K jejímu použití bylo rozhodnuto z důvodu dobrých zkušeností ostatních programátorů [4][8] a také proto, že byla hledána spíše menší a jednodušší knihovna s dobrou dokumentací, což se u této knihovny našlo.

Knihovnu je možno stáhnout z oficiálních stránek www.grinninglizard.com/tinyxml a po rozbalení do podadresáře MinGW opět knihovnu přilinkovat a to tak, že v nastavení linkeru se zvolí v levé části Link libraries přidání knihovny tlačítkem Add a tam se napíše jen tinyxml a potvrdí.



Obrázek 13 Nastavení linkeru v prostředí Code::Blocks

6.2 Co by měla aplikace umět

Aplikací by měla být hra viděná z axonometrického pohledu shora. Mělo by se jednat o jednoduchou adventuru, ve které bude uživatelem ovládaná postava, se kterou budeme moci pohybovat uvnitř grafického prostředí a která bude moci komunikovat pomocí výběru rozhovorů s dalšími postavami rozmístěnými po tomto prostředí.

Aplikace by měla umět načítat obrázky prostředí ze souborů PNG s alfa kanálem s tím, že použití těchto obrázků bude měnitelné z XML souboru mapy. Pozadí budou tvořit 2 vrstvy, tedy 2 různé PNG obrázky. Nejprve se vykreslí půdorys, poté ostatní definované postavy včetně té ovládané uživatelem a nakonec vrstva se stěnami a dalšími vyššími dekoracemi, se kterými by měla uživatelova postava kolidovat, tzn. nebylo by možné přes ně procházet. V XML souboru prostředí také definované přímky ohraničující kolizní prostředí.

Aplikace by měla být postavená na frameworku OpenLayer, pro který jsem se rozhodl, protože jednak využívá hardwarově akcelerované vykreslování z AllegroGL a také zapouzdřuje nejdůležitější funkce a dodává k nim další užitečnou funkcionalitu.

Jednotlivé interaktivní postavy pak budou mít každá vlastní XML soubor, ve kterém budou specifikovány jednotlivé dialogy.

6.3 Adresářová struktura aplikace

Všechny datové soubory, se kterými aplikace pracuje jsou uloženy v adresáři data. Kromě něj je v kořenovém adresáři aplikace už jenom spustitelný soubor a DLL knihovny nutné pro spuštění aplikace.

V adresáři data jsou pak soubory tříděny do následujících podadresářů:

6.3.1 Adresář fonts

V adresáři fonts jsou uloženy soubory s fonty, které aplikace využívá ke zobrazování textů. Fonty jsou uloženy ve formátu TTF.

6.3.2 Adresář maps

Adresář maps obsahuje další podadresáře, které reprezentují jednotlivé mapy (resp. levely) hry. V každém podadresáři se pak nachází XML soubor s informacemi o mapě a potřebné PNG obrázky.

6.3.3 Adresář persons

V adresáři persons jsou umístěny XML soubory s definicemi vzhledu a dialogů jednotlivých postav. Jsou zde také PNG obrázky představující snímky postavy z jednotlivých stran a animační sekvence obrázků pro animovaný pohyb postav.

6.4 Struktura XML souborů

Soubory formátu XML jsou určeny k tomu, aby bylo možné dělat úpravy nastavení a vzhledu programu bez zásahů do zdrojového kódu aplikace. Jsou určeny ke konfiguraci inicializace aplikace, k nastavení prostředí map a postav v nich obsažených. Všechny XML soubory jsou uloženy v kódování znaků UTF-8.

6.4.1 XML soubor s konfigurací aplikace

Soubor config.xml v adresáři data je určen k nastavení programu před jeho spuštěním. Obsahuje informace o nastavení grafického režimu - rozlišení a zda má být program spuštěn přes celou obrazovku nebo v systémovém okně.

V souboru je kořenový element <config>, ve kterém je podřízený element <window> obsahující atributy s informacemi. V atributu width se nastaví šířka a v atributu height pak výška rozlišení, ve kterém má být program spouštěn. Do parametru autodetect se pak nastavuje buď hodnota windowed, pokud se program bude spouštět v oknech, nebo hodnota fullscreen, pokud se bude aplikace zobrazovat přes celou obrazovku.

Obsah souboru config.xml tedy vypadá např. následovně:

```
47 <?xml version="1.0" encoding="UTF-8" ?>
48 <config>
49   <window width="1024" height="768" autodetect="windowed" />
50 </config>
```

6.4.2 XML soubor mapy

V XML souboru v adresáři maps se nahází v každém podadresáři soubor s prostředím jednotlivé mapy hry. Ten obsahuje kořenový element <map> s parametry floor, který určuje soubor s obrázkem půdorysu a ceil, který určuje obrázek nejvyšší vykreslované vrstvy. U parametru ceil je vzhledem k jeho podstatě vhodné, aby byl obrázek s alfa kanálem. Adresy souborů jsou brány relativně vzhledem k adresáři s XML.

Dále jsou zde elementy <line> určující kolizní úsečky postavy s prostředím. Každý tento element má 4 parametry a to: x1 a y1 určující souřadnice počátečního bodu v pixelech a x2 a y2 určující souřadnice koncového bodu úsečky v pixelech.

Element <entry> určuje vstupní bod hráčovy postavy do mapy a směr pohledu postavy. Obsahuje tedy parametry x a y se souřadnicemi umístění v pixelech, parametr direction s hodnotou určující směr pohledu (může nabývat hodnot left, right, up a down). Ještě je zde parametr id, který identifikuje číselným pořadím (počítá se od nuly) vstup na mapě.

Naopak element <exit> určuje výstupní bod, pro přechod do jiné mapy. Parametry x a y opět určují souřadnice výstupního v pixelech, parametr dest obsahuje název mapy, do které se bude přecházet a parametr entry značí identifikační parametr id vstupního bodu v budoucí mapě.

Nakonec jsou po mapě rozesety postavy a to v XML elementech <npc>, které mají v parametrech souřadnice x a y dané postavy na mapě a parametrem src určíme XML soubor postavy z adresáře persons. Soubor mapy v XML formátu pak může vypadat např. takto:

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <map ceil="ceil.png" floor="floor.png">
3    <entry id="0" x="250" y="250" direction="left" />
4    <entry id="1" x="700" y="750" direction="left" />
5    <exit x="750" y="750" dest="demo2" entry="0" />
6    <npc x="250" y="250" src="cisnik.xml" />
7    <npc x="150" y="250" src="kuchar.xml" />
8    <npc x="250" y="300" src="host.xml" />
9    <line x1="0" y1="0" x2="1500" y2="0" />
10   <line x1="1500" y1="0" x2="0" y2="1500" />
11   <line x1="0" y1="1500" x2="0" y2="0" />
12 </map>
```

6.4.3 XML soubor postavy

Soubor s informacemi o herní postavě obsahuje kořenový XML element <person> u kterého jsou definovány pouze parametr name, kterým se určuje jméno postavy.

Tento element pak obsahuje elementy <stay>, <walk> a <dialog>.

6.4.3.1 Element <stay>

Element <stay> reprezentuje vzhled postavy ve stavu nehybném a sám nemá žádné parametry. Obsahuje elementy <img>, které mají parametry src a direction. Parametrem src se určí cesta k souboru s obrázkem. Doporučeno je používat obrázek s průhledným pozadím a tím pádem nejlépe ve formátu PNG, který OpenLayer umožňuje načítat.

Druhý parametr direction pak určuje směr pohledu postavy a může nabývat pouze následujících hodnot:

- left - pro postavu dívající se směrem doleva
- right - pro postavu dívající se směrem doprava
- up - pro postavu dívající se směrem nahoru
- down - pro postavu dívající se směrem dolů

6.4.3.2 Element <walk>

Dalším elementem v kořenovém elementu je element <walk>. Ten reprezentuje vzhled postavy ve stavu pohybu a obsahuje elementy <anim>, které obsahují informace o načítání animačních sekvencí obrázků. Jeden tento element tedy reprezentuje jednu animační sekvenci. Soubory v sekvenci musí být na konci názvu (před příponou) číslovány od nuly.

Element <anim> obsahuje parametry direction (se stejným významem jako u elementu <stay>), srcBegin, extension, num a speed. Do parametru srcBegin se zadává začátek cesty souborů bez čísla, do parametru extension se zadá přípona souborů bez tečky, do parametru num počet souborů v sekvenci a parametrem speed se určí rychlost animace.

6.4.3.3 Element <dialog>

Elementy <dialog> jsou určeny k textu dialogu postav. Obsahují dva hlavní elementy <question> a <answer>. Dialog <question> jak již název napovídá, obsahuje text otázky

hráče a element <answer> obsahuje textovou odpověď. Při zodpovězení dotazu hráčem ve hře se může odemknout další dialog.

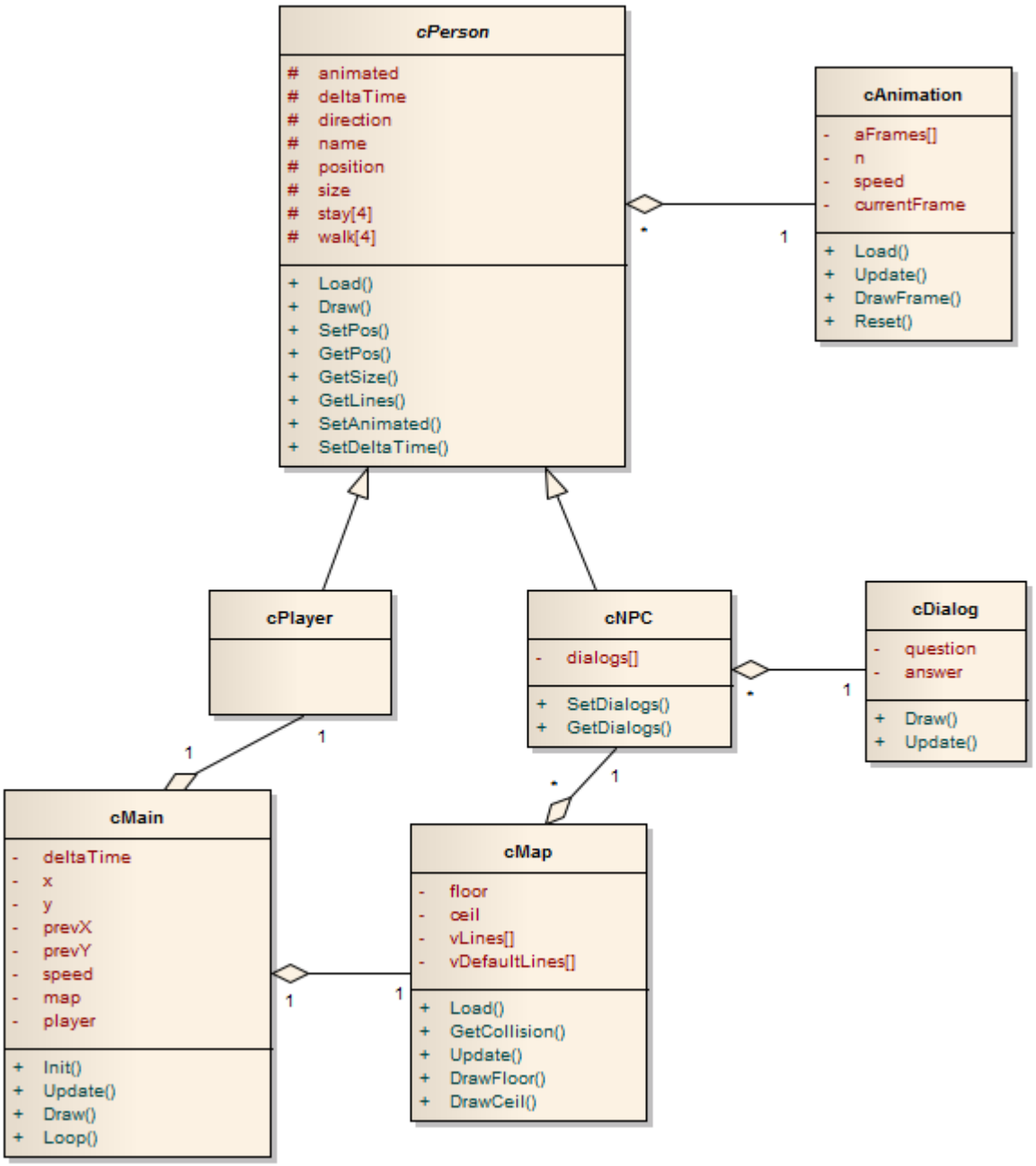
To je možné zařídit tak, že se v elementu <dialog> nadefinuje ještě element <set>, kterému zadáme v parametru var libovolný identifikátor, který převezme v herním prostředí proměnnou s hodnotou nastavenou na true. Dialog, který se má odemknout po nastavení této proměnné na hodnotu true, pak označíme tak, že do patřičného elementu <dialog> vložíme element <if>, kde do parametru var dáme název proměnné (identifikátor z elementu <set>, který tento dialog odemyká).

XML soubor s definicemi vlastností postavy může vypadat např. následovně:

```
13 <?xml version="1.0" encoding="UTF-8" ?>
14 <person name="Hrac">
15   <stay>
16     <img src="left1.png" direction="left" />
17     <img src="right1.png" direction="right" />
18     <img src="up1.png" direction="up" />
19     <img src="down1.png" direction="down" />
20   </stay>
21   <walk>
22     <anim direction="left" srcBegin="left" extension="png" num="4"
   speed="2" />
23     <anim direction="right" srcBegin="right" extension="png" num="4"
   speed="2" />
24     <anim direction="up" srcBegin="up" extension="png" num="4"
   speed="2" />
25     <anim direction="down" srcBegin="down" extension="png" num="4"
   speed="2" />
26   </walk>
27   <dialog>
28     <set var="promenna" />
29     <question>Nejaka otazka</question>
30     <answer>Nejaka odpoved</answer>
31   </dialog>
32   <dialog>
33     <if var="promenna" />
34     <question>Nejaka otazka navazujici podminena zobrazenim
   predchozi</question>
35     <answer>Nejaka jina odpoved</answer>
36   </dialog>
37 </person>
```

6.5 Struktura zdrojového kódu

Kód aplikace je psán objektově v programovacím jazyce C++ a skládá se z několika tříd, z nichž ty v této práci vytvářené budou v této části podrobněji rozebrány. Ke tvorbě tříd se dá v prostředí Code::Blocks využít velice šikovný plugin Class wizard, který vygeneruje hlavičkový a zdrojový soubor a vyplní do nich potřebný základní kód.



Obrázek 14 UML diagram hlavních tříd aplikace

6.5.1 Třída cMain

Třída *cMain* je hlavní třídou aplikace. Zapouzdřuje inicializaci grafického režimu a hlavní vykreslovací smyčku programu.

Metoda *Init* inicializuje počáteční mapu "default" skrze instanci třídy *cMap*, inicializuje třídu cPlayer starající se o hlavní postavu a také načítá a zpracovává pomocí knihovny TinyXML konfigurační XML soubor. Jakmile má načteny všechna potřebná data v proměnných, zavolá metodu SetupProgram ze třídy ol::Setup (*ol* je název namespacu zapouzdřujícího framework OpenLayer), čímž inicializuje Allegro. Poté zavolá metodu SetupScreen, které do parametru rozlišení a grafický mód a tím se inicializuje okno aplikace (případně celá obrazovka). Nakonec inicializuje čítač ve třídě ol::FpsCounter pro použití v animacích.

Metoda *Loop* obsahuje hlavní smyčku programu, ve které dokola volá aktualizační metodu *Update*, která aktualizuje data před vykreslováním a poté volá metodu *Draw*, která vykreslí grafické prvky. Smyčka probíhá tak dlouho, dokud uživatel nezmáčkne klávesu Esc. Kód tedy vypadá následovně:

```
38 while (!key[KEY_ESC]) {
39     Update();
40     Draw();
41 }
```

Metoda Update nejprve aktualizuje statickými metodami *NewFrameStarted* a *GetDeltaTime* časovač ze třídy *ol:: FpsCounter* a poté podle hodnot z pole *key* (stisky kláves) předá instancím třídy *cPlayer* a *cMap* aktuální souřadnice uživatelovy postavy.

Funkce *Draw* nakonec vyplní pozadí bílou barvou a postupně zavolá vykreslovací metody tříd *cMap* a *cPlayer*.

6.5.2 Třída cMap

Tato třída se stará o načtení dat o mapě z XML a přiložených obrázků a jejich správné zobrazení. Jako první metoda se ze třídy *cMap* volá *Load*, která parsuje zadaný XML

soubor pomocí tříd a metod knihovny *TinyXML*. Nejprve se získají obrázky z kořenového elementu, poté se projdou a uloží do vektoru (kontejner std::vector) kolizní úsečky, po nich vstupní a výstupní body mapy a nakonec NPC postavy.

Dále se volá metoda *GetCollision*, která zjistí, zda úsečky mapy nekolidují s úsečkami zadanými v parametru. To se zjišťuje metodou *Collides* u úsečky (třída *ol::Line*), které se zadá do parametru druhá úsečka a vrátí se, jestli koliduje (true) nebo ne (false). Poté se aktualizují hodnoty metodou *Update* na zadané souřadnice a poté se kreslí. Vykresluje se nejprve spodní vrstva metodou *DrawFloor*, poté se vykreslují herní postavy a nakonec horní část se stěnami metodou *DrawCeil*.

6.5.3 Třída cPerson a její potomci

Abstraktní třída *cPerson* obsahuje veškerou funkčnost společnou pro postavu hráče a pro NPC postavy ve hře. Má tedy na starost načítání a zpracování XML souborů z adresáře persons, jejich správné zobrazení a chování.

Metodou *Load* se provede přeparsování zadaného XML souboru. Nejprve se zjistí z kořenového elementu jméno, poté podle parametru direction přeparsují jednotlivé elementy <img> v elementu <stay>, ty se uloží do vektoru bitmap a poté se to samé udělá s elementy <anim> v elementu <walk>. Nakonec se přeparsují dialogy do vektoru instancí třídy cDialog.

Několika metodami jako *SetPos* (aktualizuje pozici postavy na mapě), *SetDirection* (aktualizuje směr pohledu postavy podle zmáčknutých šipek), *SetAnimated* (zapíná a vypíná animovaný pohyb) a *SetDirection* (předá aktuální čas od posledního vykreslení snímku) se aktualizují informace o postavě.

Následuje vykreslení postavy metodou *Draw*, která vykreslí aktuální obrázek metodou *Blit* ze třídy *ol::Bitmap*.

6.5.4 Třída cAnimation

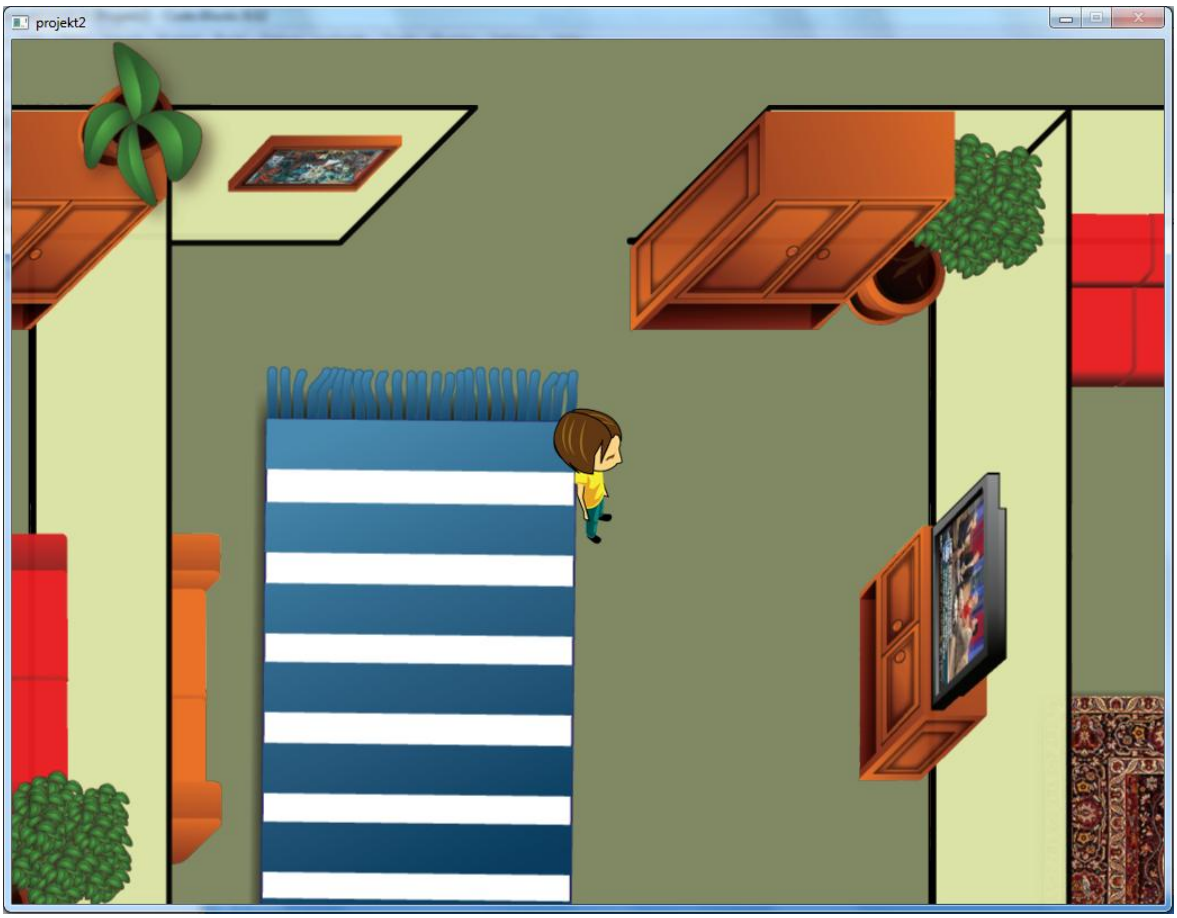
V knihovně OpenLayer je sice k dispozici třída ol::Animation pro správu animované sekvence obrázků, ta se ale ukázala jako nepoužitelná (program při její inicializaci z neznámého důvodu padal). Proto bylo nutno navrhnout třídu vlastní.

Metodou *Load* se načte sekvence obrázků do pole typu *ol::Bitmap*. Do parametrů se zadá rychlost animace, počáteční část názvu bez čísla, koncovku souborů a jejich počet.

Poté se metodou Update aktualizuje snímek sekvence podle zadané doby od poslední aktualizace a metodou *DrawFrame* se vykreslí aktuální obrázek. Případně se může animační sekvence resetovat metodou *Reset*.

6.5.5 Třída cDialog

Tato třída je určena k načtení a zobrazení dialogů postav. Obsahuje metody k jejímu nastavení *SetQuestion* (předá otázku), *SetAnswer* (předá odpověď), *SetIf* (předá název proměnné nutné k zobrazení dialogu v seznamu) a *SetVar* (předá název nastavené proměnné). Vykreslování podle splněných podmínek probíhá Vektor instancí této třídy je k dispozici ve třídě *cPlayer*.



Obrázek 15 Výsledná aplikace

7 UŽIVATELSKÝ MANUÁL APLIKACE

7.1 Požadavky na spuštění aplikace

Pro úspěšné spuštění a provoz aplikace je potřeba splňovat následující systémové požadavky:

- PC s operačním systémem Windows (95 nebo novější, není problém ani s Vista a 7)
- Nainstalované DirectX minimálně ve verzi 7
- Grafická karta s podporu hardwarové akcelerace přes OpenGL
- Klávesnice a myš

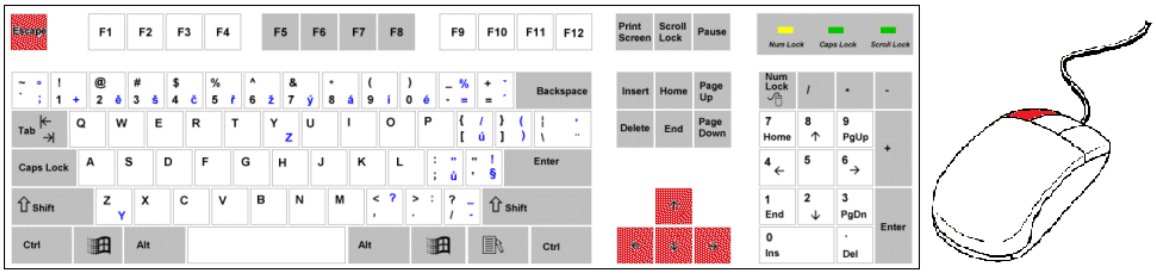
7.2 Konfigurace aplikace

Před spuštěním hry je dobré nastavit rozlišení hry na optimální pro daný monitor. To lze udělat tak, že v adresáři, kde je hra umístěna se půjde do adresáře data a v něj otevře v libovolném textovém editoru soubor config.xml.

V tomto souboru změníme u elementu <window> hodnoty v uvozovkách u parametrů width a height na vyžadované a taktéž se dá v případě potřeby změnit u atributu autodetect v uvozovkách, zda se má hra zobrazovat přes celou obrazovku (nastaví se hodnota "fullscreen") nebo v okně (v tom případě se tam nastaví hodnota "windowed").

7.3 Ovládání

Pohyb postavy po herním prostředí je možné ovládat šipkami a pro komunikaci s jinými postavami používat myš. Ukončení programu provedeme klávesou Esc.

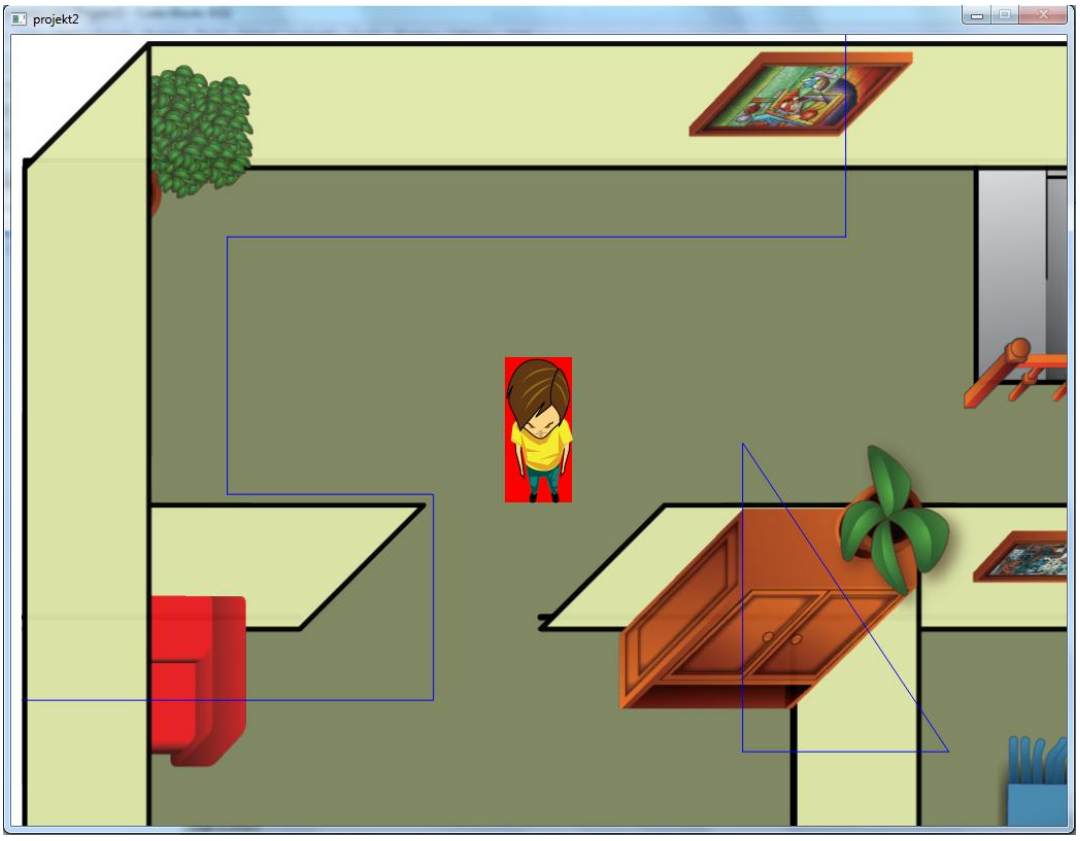


Obrázek 16 Ovládání aplikace klávesnicí a myší

8 MODIFIKOVATELNOST APLIKACE

Aplikaci je možno upravovat pomocí souborů v adresářové struktuře v adresáři data. Zde se nachází fonty, XML soubory a obrázky prostředí a herních postav. Pro úpravu herního prostředí aplikace, především pak pro tvorbu kolizních přímek v mapě je určena Debug verze aplikace. Tu je možno spustit samostatným spustitelným souborem debug.exe, který je přiložen nebo zkompilováním zdrojových kódů aplikace s tím, že v souboru main.h se definované přepíše hodnota makra preprocesoru DEBUG_MODE na true, čímž se aplikace zkompiluje v tomto režimu.

V této verzi aplikace se všechny kolizní přímky na mapě zobrazí o tloušťce jednoho pixelu modrou barvou a za herní postavou se objeví červené obdélníkové pozadí, které určuje jednak rozměry obrázku a současně kolizní obdélník této postavy.



Obrázek 17 Debug verze aplikace určená pro testování kolizních elementů

8.1 Editace map v XML souborech

Mapy, které se nacházejí v adresáři data/maps, je možné upravit co do vzhledu a rozložení prvků za pomoci výše zmíněného popisu XML formátu map. Také je samozřejmě možné

tvořit mapy zcela nové mapy a ty propojovat v návaznosti ke stávajícím mapám nebo vytvořit zcela novou kolekci map.

Mapy jsou jednoduše propojitelné pomocí kombinace vstupů a výstupů, kterých se v jednom XML souboru dá definovat i několik a je tak možné poskládat z jednotlivých map i velmi komplexní herní prostředí.

8.2 Editace herních postav v XML souborech

Postavám v XML souborech je možno upravovat vzhled ze všech 4 stran a to jak v nehybné poloze, tak animačním sekvencím pro pohyb. Také je samozřejmě možné tvořit postavy zcela nové s jiným vzhledem. Díky XML souborům můžeme vytvořit více postav se stejnou grafikou či její částí bez nutnosti tato grafická data duplikovat.

8.3 Modifikovatelnost aplikace změnou zdrojového kódu

Aplikaci je samozřejmě možné modifikovat i úpravou zdrojového kódu, pokud je potřeba dělat hlubší změny než ty, které se dají udělat úpravou datových souborů. Zdrojový kód je psán objektově a je tedy možné vytvořené třídy rozšířit mnohem jednodušeji, než pokud by byla aplikace tvořena klasickým procedurálním programováním.

Nabízí se kupříkladu možnost rozšířit třídu cPerson, resp. jejího potomka cNPC metodami, které by definovali např. umělou inteligenci herních postav ovlivněnou různými vstupy od uživatele nebo jiných postav.

Dalším příkladem rozšíření může být podpora komplexnějšího tvaru podmínek v XML souborech či přidání inventáře a možnosti sbírání předmětů po mapě nebo od jiných postav rozšířením třídy cPerson či jejího potomka cPlayer.

Také je možné hru upravit i na jiný žánr, např. akční, strategickou čí RPG hru. Základní kolize prostředí a postav jsou zde pro to již připraveny, takže tento herní systém by i takto zaměřeným tvůrcům her měl usnadnit práci.

Škála různých možných rozšíření aplikace je tedy prakticky neomezená a záleží pouze na fantazii potenciálního tvůrce, jak se rozhodne aplikaci přizpůsobit k obrazu svému.

ZÁVĚR

V diplomové práci byla rozebrána po teoretické stránce knihovna Allegro, byla popsána její historie, současné vlastnosti a blízká budoucnost. Rozebrány byly jak funkce API, tak nástroje dodávané s touto knihovnou. Dále byly zevrubně rozebráni největší konkurenti této knihovny, tedy SDL a DirectX a Allegro s nimi bylo teoreticky srovnáno.

Z nabitých poznatků lze tvrdit, že Allegro je knihovna určená především na tvorbu her, čemuž odpovídá celkem dost obsáhlé API, které poskytuje pro aplikace tohoto typu dostatečnou funkčnost. Bohužel však současná stabilní verze nedisponuje řadou moderních datových formátů a je tudíž pro serióznější tvorbu doinstalovat podpůrné knihovny. Navíc Allegro v současné verzi nevyužívá při vykreslování grafiky hardwarovou akceleraci. Z těchto důvodů tato verze zaostává za současnou konkurencí a i někteří její tvůrci přecházejí na konkurenční řešení, dokonce i původní autor Allegra Shawn Hargreaves, který přešel na DirectX framework XNA. [20]

Nicméně se již delší dobu připravuje verze 5, která se blíží do závěrečné fáze vývoje a nabídne odstranění těchto nedostatků a přidá podporu pro iPhone. Navíc je vyvíjen i nástroj, který převede kód pro Allegro 4 na kód Allegra 5. [5]

V praktické části byla navrhnuta aplikace (2D hra) založená na Allegru a frameworku OpenLayer. Aplikace byla naprogramována v programovacím jazyce C++ ve vývojovém prostředí Code::Blocks a její tvorba byla byla důkladně zdokumentována včetně popisu zdrojových kódů. K aplikaci byl nakonec sepsán uživatelský manuál a několik tipů, jak lze aplikaci dále rozšířit.

Z důvodů výše popsaných problémů s nepodporou moderních formátů a chybějící hardwarovou akcelerací byl použit framework OpenLayer, který tyto nedostatky odstraňuje a přidává velmi přehledné objektové API se spoustou ulehčení práce pro vývojáře. Aplikace také využívá z velké části pro definici herního prostředí soubory XML, takže není nutno zasahovat do zdrojových kódů.

Tímto tedy byly splněny všechny body zadání této diplomové práce. Tato práce by mohla pomoci začínajícím vývojářům multimediálních aplikací a přinést jim inspiraci ke zkvalitnění a zefektivnění tvorby s knihovnou Allegro.

ZÁVĚR V ANGLIČTINĚ

In the thesis was analyzed the Allegro library theory, described history of Allegro, current characteristics and near future. In this thesis were analyzed as a functions of the API, so tools that come with this library. Have also been thoroughly analyzed the main competitors of the library, namely SDL and DirectX and Allegro with them was theoretically compared.

Based on the lessons learned can be said that Allegro is a library designed especially for creating games, giving a total pretty extensive API that provides sufficient functionality to this type of applications. Unfortunately, the current stable version does not have many modern data formats and it is therefore necessary for the more serious creation install support libraries. Moreover, the Allegro in the current version does not use hardware acceleration for rendering. For these reasons, current version lags behind the competition and even some of its creators are moving to a competitive solution, even the original author of Allegro, Shawn Hargreaves, who moved to the XNA Framework, which is an extension of DirectX.

However, it has longer time to prepare version 5, which is approaching the final stage of development and offers close these gaps and adds support for iPhone. Moreover, it also developed a tool that converts code for Allegro 4 to Allegro 5 code.

In practical part was designed application (2D game) based on Allegro and OpenLayer framework. The application was programmed in C + + by development environment Code:: Blocks and its output has been thoroughly documented, including a description of the source code. For application was eventually written user manual and some tips on how to further extend the application.

Because of the above described problems about not supporting modern formats and missing hardware acceleration was used OpenLayer framework, which eliminates these shortcomings and adds a very easy object-oriented API with features for developers which make development easy. The application also used in most cases XML files for the gaming environment definition, so there is no need to intervene in the source code.

This therefore met all the entry points of this thesis. This work could help novice developers of multimedia applications and bring them inspiration to improve and streamline the creation of the Allegro library.

SEZNAM POUŽITÉ LITERATURY

[1] HARBOUR, Jonathan S. *Game Programming All in One*. 2006th Third Edition edition. [s.l.] : Course Technology PTR , 2006. 832 s. ISBN 1598632892.

[2] KERNIGHAM, Brian W. *Programovací jazyk C* 1. vyd. Praha : Computer Press, 2006. 80-251-0897-X s. ISBN 80-251-0897-X.

[3] LIBERTY, Jesse. Naučte se C++ za 21 dní. 1. vyd. Praha : Computer Press, 2002. 80-7226-774-4 s., CD-ROM. ISBN 80-7226-774-4.

[4] ALEXANDRESCU, Andrei. Moderní programování v C++. 1. vyd. Praha : Computer Press, 2004. 80-251-0370-6 s. ISBN 80-251-0370-6.

[5] Allegro.cc : Game developing community network [online]. 1999-2009 [cit. 2009-01-30]. Dostupný z WWW: ‹http://www.allegro.cc/›.

[6] C++ Game Programming [online]. [2005] [cit. 2009-01-30]. Dostupný z WWW: ‹http://www.cppgameprogramming.com/›.

[7] Allegro - A game programming library [online]. [2009] [cit. 2009-01-30]. Dostupný z WWW: ‹http://www.talula.demon.co.uk/allegro/›.

[8] Builder - Informacni server o programovani [online]. 1999-2003 [cit. 2009-01-30]. Dostupný z WWW: ‹http://www.builder.cz/›.

[9] Allegro Wiki [online]. [2008] [cit. 2009-01-30]. Dostupný z WWW: ‹http://wiki.allegro.cc/›.

[10] OpenLayer - Hardware Accelerated 2d Game Development [online]. [2005] [cit. 2009-01-30]. Dostupný z WWW: ‹http://openlayer.berlios.de/›.

[11] Allegro library - Wikipedia, the free encyclopedia [online]. [2010] [cit. 2010-03-02]. Dostupný z WWW: ‹http://en.wikipedia.org/wiki/Allegro_library›.

[12] Framework - Wikipedia, the free encyclopedia [online]. [2010] [cit. 2010-04-09]. Dostupný z WWW: ‹http://cs.wikipedia.org/wiki/Framework›.

[13] Forum Allegro.cc - Allegro Simplificator [online]. [2010] [cit. 2010-04-09]. Dostupný z WWW: ‹http://www.allegro.cc/forums/thread/589484›.

[14] Sprite (počítačová grafika) - Wikipedia, the free encyclopedia [online]. [2010] [cit. 2010-06-01]. Dostupný z WWW: ‹http://cs.wikipedia.org/wiki/Sprite_%28po%C4%8D%C3%ADta%C4%8Dov%C3%A1_grafika%29›.

[15] Geometrie/Stínování - Wikiknihy [online]. [2010] [cit. 2010-06-01]. Dostupný z WWW: ‹http://cs.wikibooks.org/wiki/Geometrie/St%C3%ADnov%C3%A1n%C3%AD›.

[16] AllegroGL [online]. [2010] [cit. 2010-06-01]. Dostupný z WWW: ‹http://allegrogl.sourceforge.net/›.

[17] Benchmark: Allegro and SDL [online]. [2010] [cit. 2010-06-01]. Dostupný z WWW: ‹http://www.allegro.cc/forums/thread/592874›.

[18] Glyf - Wikipedia, the free encyclopedia [online]. [2010] [cit. 2010-06-04]. Dostupný z WWW: ‹http://cs.wikipedia.org/wiki/Glyf›.

[19] Seznámení s XNA frameworkem - VB.net [online]. [2010] [cit. 2010-06-04]. Dostupný z WWW: ‹http://www.vbnet.cz/clanek--75-xna_2_0_ve_vb_net_dil_1_seznameni_s_xna_frameworkem.aspx›.

[20] SDL - official website [online]. [2010] [cit. 2010-06-04]. Dostupný z WWW: ‹http://www.libsdl.org/›.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

C	Programovací jazyk ANSI C.
C++	Programovací jazyk ISO C++ vzešlý z jazyka ANSI C.
OpenGL	Open Graphics Library
Allegro	„**A**tari **L**ow-**Le**vel **G**ame **Ro**utines“ – přenositelná knihovna, poskytuje přímý přístup k funkcím zvuku, klávesnice, myši, joysticku, 3D akcelerátorům (přes OpenGL) a 2D video bufferu.
SDL	„Simple Directmedia Layer“ – přenositelná knihovna, poskytuje přímý přístup k funkcím zvuku, klávesnice, myši, joysticku, 3D akcelerátorům (přes OpenGL) a 2D video bufferu.
OpenLayer	„Simple Directmedia Layer“ – přenositelná knihovna, poskytuje přímý přístup k funkcím zvuku, klávesnice, myši, joysticku, 3D akcelerátorům (přes OpenGL) a 2D video bufferu.
LGPL	„GNU Lesser General Public License“ – licence svobodného softwaru, publikovaná Free Software Foundation. Byla navržena jako kompromis mezi silně copyleftovou licencí GNU General Public License (GPL) a permisivními licencemi, jako jsou BSD licence nebo MIT Licence.

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH