VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

APLIKACE V ASM: BLUDIŠTĚ

AUTOR JAKUB SVOBODA

AUTHOR

Úvod

Tento dokument slouží jako dokumentace k projektu do předmětu Pokročilé asemblery – Aplikace v ASM: Bludiště. Dané téma jsem si zvolil, protože mi tvorba aplikace byla bližší, než témata řešící optimalizaci a chtěl jsem se spíše zabývat přímo programováním v asembleru, než studiem algoritmů.

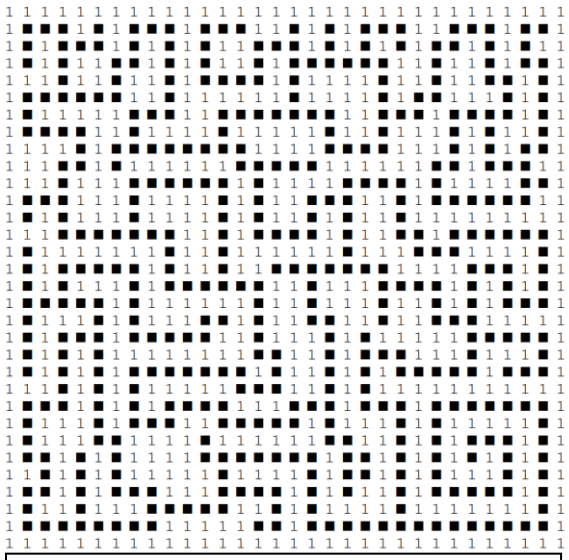
V dokumentaci popisuji, jakým způsobem je realizována reprezentace bludiště v kódu, algoritmus pro detekci stěn a výpočet možného pohybu. Dále popisuji problémy, se kterými jsem se při realizaci projektu setkal, stejně tak jako software a nástroje, které byly použity.

Realizace projektu

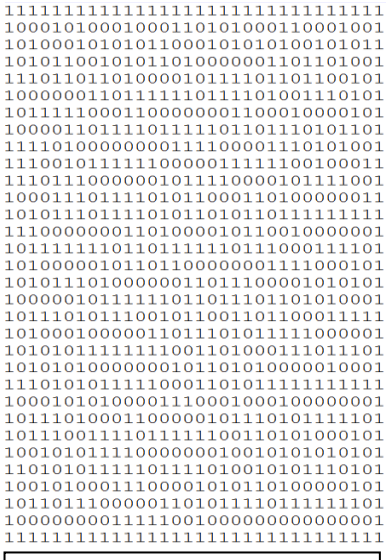
Reprezentace mapy v kódu

Při volbě hrací plochy jsem se rozhodl pro mřížku s rozměry 32x32 hracích políček, kde hráč, cílové pole i šířka zdi je reprezentována blokem 1x1 a pohyb je možný pouze po celých polích. Tento rozměr je dostatečný pro ověření funkčnosti hry a zároveň je bludiště i dostatečnou výzvou pro hráče.

Každý řádek mapy byl zapsán jako 32 bitové číslo, kde jednička na indikovala stěnu a nula prázdné místo. Celá hrací plocha potom tedy byla reprezentována jako pole 32 položek, každá o velikosti 4 Bytů. Tímto způsobem je samotná logika hry nezávislá na její grafické reprezentaci.



Návrh hrací mapy



Převod do bitové reprezentace

```
lines: dd 0xffffffff, 0x8A235189, 0xA2AC552B, 0xACAD0369, 0xEDA17B65, 0x81BF7A75, 0xBE301885, 0x86F7DBAD, 0xF403C3A9,0xE5F83F23,
0xEE05E179, 0x8EF58D03, 0xAEF5ADFF, 0xE0342C81, 0xBFB7EE3D, 0xA0B603C5, 0xAE81B855, 0x82FDBB51, 0xBAE59B1F, 0xA20DD7C1, 0xABFCD1DD,
0xAA02D411, 0xEAF8D7FF, 0x8A871101, 0xBA305D7D, 0xB9EFCD45, 0x95E02555, 0xD5F7A575, 0x9470AD05, 0xB706BDFD, 0x807C8001, 0xFFFFFFFF
```

Reprezentace v kódu po převodu do šestnáctkové soustavy pro jednodušší zápis

Do této mapy je následně umístěn cíl do pravého horního rohu a hráč do diagonálně opačného rohu. Poloha hráče je uchovávána ve dvou proměnných reprezentujících osu *x* a *y*.

Pohyb po mřížce

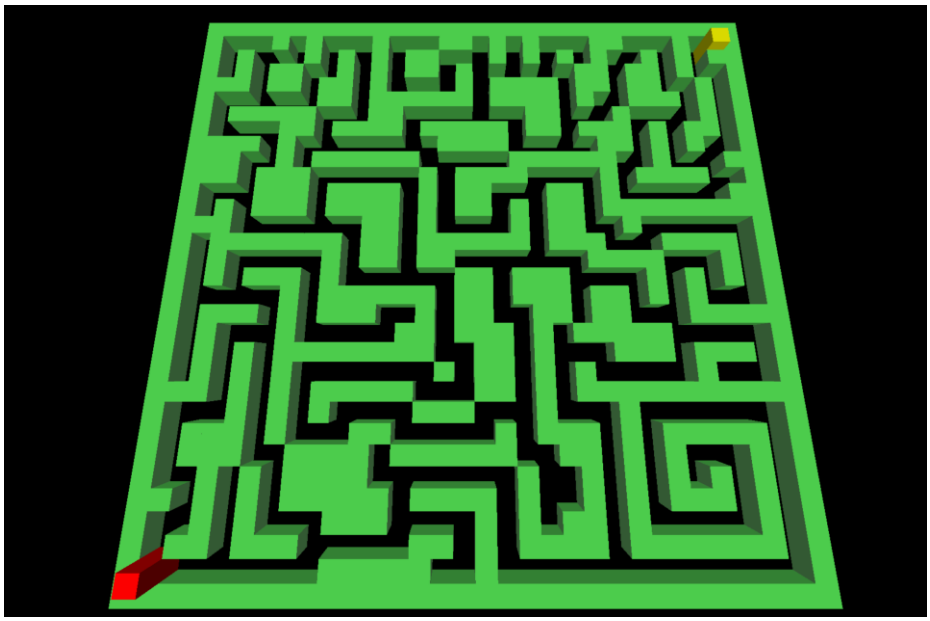
Při pohybu (zde pro demostraci pohyb vlevo) jsou přečteny souřadnice hráče, podle kterých je v poli mapy vyhledán příslušný řádek. Následně je ověřeno pomocí instrukce BT (bit test), zdali je na pozici o jedno pole vlevo od hráče jednička nebo nula. Při zjištění jedničky je pohyb neplatný a díky podmíněnému skoku není pohyb vykonán. Při platném pohybu se aktualizují souřadnice hráče, dojde k přepočtu souřadnic pro vykreslování bloku hráče a aktualizovaná mapa je překreslena. Následně je ještě ověřeno, zdali se shoduje pozice hráče s pozicí cíle a eventuálně nastane ukončení hry.

```
.goLeft:
    mov eax, dword[x_position]      ;Presun pozice hrace
    inc eax                         ;Inkrementace pro pozici o jedno pole vlevo
    mov ecx, dword[y_position]
    mov ebx, dword lines            ;Presun radku mapy
    mov ebx, dword [ebx+4*ecx]
    bt ebx, eax                     ;Bit test nastaví carry flag podle hodnoty na daném bytu
    jc .Paint
                  Vykreslovací instrukce
    mov dword[x_position],eax
```

Zjednodušená ukázka kódu vykonávajícího pohyb

Grafická reprezentace hry

Ačkoliv je hra samotná funkční i bez grafického znázornění, hráč potřebuje mí přehled o své pozici v mapě. K tomuto účelu by mohlo sloužit například vypsání znaku # konzole na místo stěny a mezery na volnou pozici. Já jsem se však rozhodl pro vykreslení pomocí funkcí z knihovny openGL. Při vykreslování jednoduchých objektů je práce s openGL relativně snadná a je možné dosáhnout mnohem atraktivnějšího grafického zobrazení.



Zobrazení pomocí openGL. Stěny jsou reprezentovány zelenými kvádry, hráč červeným a cílové pole žlutým.

Po spuštění hry se ve v kódu opakuje herní smyčka, která snímá přijaté stisky kláves. Registrované jsou klávesy W, S, A, D pro pohyb po mřížce a klávesa ESC pro vypnutí hry. Při stisku klávesy pro pohyb je proveden skok ke kódu pro ověření možnosti pohybu a je překreslena scéna.

Jako základní kostra byl použit ukázkový kód[1] dostupný pro studenty ze stránek předmětu, ukazující práci s openGL, který byl doplněn o kód samotné hry. Dále bylo nutné pro správné vykreslení scény především vypnout osvětlení, které znemožňovalo obarvení objektů. Samotný efekt osvětlení pak byl pouze simulován nastavením různého odstínu barev podle dané orientace stěny.

Pro vykreslení bylo využito volání funkce glVertex3f, pomocí které byl vždy vykreslen daný kvádr reprezentující stěnu, hráče nebo cílové pole. Jelikož je nutné zajistit přepočet souřadnic pro pohyb hráče, jsou souřadnice jeho kvádru ukládány do proměnných a dopočítávány při každém pohybu. Pro vykreslení mapy a cíle je funkce glVertex3f volána pouze s použitím konstant. To sice způsobuje nárůst délky kódu, ale není potřeba souřadnice dopočítávat, což je výpočetně i algoritmicky poměrně náročné, protože operace s hodnotami float je nutné provádět přes zásobník.

Použité nástroje a software

Při tvorbě projektu jsem využíval nástroje *SASM*. Tento nástroj se mi osvědčil především díky své naprosté jednoduchosti a taky díky debuggeru, který zobrazuje pozici v kódu spolu s obsahy registrů a proměnných. Podle mého názoru by byl SASM vhodný i pro výuku předmětu ISU na naší fakultě.

K samotnému překladu jsem využíval *NASM* a k linkování souborů potom *alink*. Pro překlad projektu je nutné do složce s projektem nakopírovat právě nasm.exe a alink.exe (z důvodu omezení velikosti při odevzdání nejsou součástí zkomprimovaného balíku). Překlad je pak možné provést pomocí přiloženého skriptu build.bat.

Projekt využívá funkce z knihoven win32.inc, opengl.inc a general.inc. Ty již odevzdaný projekt obsahuje.

Optimalizace a výkon

Rychlost aplikace nebyla u tohoto projektu klíčovým faktorem. K optimalizaci přispěl fakt, že při volání funkcí openGL je využíváno pevně daných konstant a souřadnice tak nemusí být dopočítávány. Další menší úspory je dosaženo vykreslováním pouze pěti stěn kvádrů. Při použití statické kamery je možné spodní stěny zanedbat.

Pro měření výkonu jsou použil program Fraps, pomocí kterého byl zjištěn počet snímků za sekundu. Při ponechání původního rozlišení 1200x1000 pixelů bylo dosaženo asi 1450FPS na výkonnějším PC (Intel Core i5-4590, 8GB RAM, AMD R9 280) a okolo 800FPS na notebooku (Intel Core i5-5200U, 8GB RAM, Intel HD Graphics 5500). Tyto hodnoty jsou daleko za hranicí obnovovací frekvence monitoru a další optimalizace by tedy byla zbytečná.

1. Autor projektu není autorem tohoto kódu.

Závěr

Při realizaci projektu byla vytvořena pomocí jazyka symbolických instrukcí jednoduchá hra na platformu Windows. Při svém studiu jsem se zatím nikdy s tvorbou okénkové aplikace ani s openGl nesetkal, proto jsem zadání vnímal jako výzvu. Požadavky ze zadání jsem podle mého názoru splnil a s konečným výsledkem jsem spokojený.