Zápočtová práce z předmětu
Programování v C++ (NPRG041)
MFF UK

FakeInput

knihovna

Programátorská dokumentace

Autor:
Richard Jedlička

20. května 2011

Obsah

1 Úvod **2**

2 Důvod tvorby **2**

3 Sestavení **2**

4 Implementace **2**
4.1 Unix 2
4.2 Windows 3
4.3 Vstupní zařízení 3
4.3.1 Klávesnice 3
4.3.2 Myš 4
4.3.3 Systém - příkazy 4
4.4 Actions 4

1 Úvod

Knihovna FakeInput umožňuje velice snadno simulovat uživatelský vstup. Lze simulovat události jako stisk klávesy na klávesnici, pohyb myši atd. Vstupní zařízení, které knihovna podporuje (umí simulovat) jsou dvě - *klávesnice* a *myš*. Mimoto ji lze využít i pro automatizované spouštění libovolných *programů*. Knihovna je *multiplatformní*, lze ji tedy využít na *Unix-like* platformách a na platformě *Windows*.

2 Důvod tvorby

Důvod, proč tvořit tuto knihovnu, byl, že jsem nenašel nic podobného, co by splnilo moje požadavky. Nalezl jsem knihovnu InputManager. Ta by jistě splnila moje požadavky, problém je ale v tom, že je pouze pro *Windows*, přičemž moje hlavní preference bylo fungování na *Unix-like* platformách. Co se týče používání a možností tak je na tom FakeInput a InputManager vcelku podobně. InputManager tedy garantuje, že funguje správně s DirectX, což já zaručit nemohu, dokonce jsem neměl možnost to ani otestovat. Co se týče dalších knihoven tak zde asi můj výčet končí, nevylučuji, že něco existuje, já to akorát nenašel. Podařilo se mi ještě najít nějaké nástroje provádějící podobnou činnost, jedná se ale už o hotové aplikace nevyužitelné jako knihovna a navíc většinou nejsou zadarmo.

3 Sestavení

viz. Uživatelská dokumentace

4 Implementace

Popis rozhraní jednotlivých tříd je v API dokumentaci.

4.1 Unix

Na *Unixu* knihovna využívá knihovny **Xlib**, která se vyskytuje defaultně snad na každém *Unix-like* systému, který používá grafické prostředí. Zároveň s knihovnou Xlib se vyskytuje její rozšíření **XTest**, které právě slouží k simulování

vstupních událostí. Využívají se k tomu funkce jako `XTestFakeKeyEvent` pro klávesnici nebo `XTestFakeButtonEvent` pro myš atd.

4.2 Windows

Na *Windows* jsem použil možnotí rozhraní **Win32 API**. Obsahuje přímo vlatní funkci `SendInput` pro zasílání události do systému.

4.3 Vstupní zařízení

Bylo třeba se vypořádat s mírně rozdílnými přístupy k některým věcem na jednotlivých platformách. Např. na knihovna **Xlib** má zástupné kódy za tlačítka myši, kdežto **WinAPI** nemá kódy pro tlačítka, ale přímo pro události, jako „stisk levého tlačítka myši“ atd.

4.3.1 Klávesnice

Nejproblematičtější oblast je rozhodně klávesnice. Na klávesnici je obrovské množství možných událostí i když jsou jen jednoho ze dvou druhů - stisk nebo uvolnení klávesy. Jenže kláves je hodně, chovají se se různě při stisku modifikačních kláves a navíc jejich chování také ovlivňuje i nastavení rozložení a jazyka klávesnice v systému.

Problém je v reprezentaci kláves na jednotlivých platformách, na *Unixu* je reprezentují tzv. `KeySym`y což je číslo označující konkrétní klávesu. Pokrývají několik kódování, takže lze reprezentovat klávesu z různých jazyků a rozložení klávesnice. *Windows* má množinu reprezentujících kódů (`Virtual-Key codes`) mnohem menší, v podstatě obsahuje pouze klávesy z anglického rozložení klávesnice. Jednotlivé kódy na obou platformách reprezentují stejně klávesu na klávesnici ne znak atd., tzn. několik kódů může označovat jednu klávesu a ta se chová podle nasteveného rozložení klávesnice. Např. pokud zvolím na *Unixu* kód klávesy `XK_4` a `XK_ccaron` tak oba reprezentují klávesu s číslem ’4’ a písmenkem ’č’ na České klávesnici. Když pošlu stisk této klávesy do systému, tak nemám zaručeno zda se mi vytiskne znak ’4’ nebo ’č’ záleží to na nastaveném rozložení klávesnice. Tedy pokud chce uživatel knihovny využít možnosti pracovat s klávesami pomocí kódů jednotlivých platforem, což knihovna umožnuje, tak nemá zaručeno jaký bude výsledek zaslané akce, pouze že se simuloval stisk klávesy příslušející danému kódu.

Aby bylo možné pracovat s typy kláves (např. explicitně říct, chci aby se stiskla klávesa `Enter`) bez nutnosti používat platformě závislé kódy, definuje knihovna několik takových typů. Jedná se o běžně známé klávesy z anglické klávesnice, jedná se o množinů kódů která se mapuje do podmnožiny průniku kódů kláves na *Unixu* a na *Windows*. Všechny typy jsou uvedeny v souboru `types.hpp`.

Pokud chce mít uživatel knihovny větší jistotu, že zasílá stisk nebo uvolnění klávesy, kterou má opravdu na mysli, může využít možnosti získat klávesu z reálné události generované fyzickým stiskem klávesy na klávesnici. Toto je už ale platformně závislé, protože každá platforma používá jiné struktury pro události a je tedy potřeba s nimi jinak zacházet. Na *Unixu* je to `XEvent` na *Windows* `MSG`, z těchto ztruktur lze získat potřebné informace o klávese, tedy pokud se jedné o události týkající se kláves.

4.3.2 Myš

Narozdíl od klávesnice je myš skoro všude stejná až na nějaká speciální tlačítka navíc, které prostě knihovna podporovat nebude. Takže je podporována simulace stisků tří základních tlačítek (levé, pravé, prostřední/kolečko), rotace kolečkem a pohyb kurzorem. Ze jedinou komplikaci lze považovat, že na *Windows* je potřeba při nastavování pozice kurzoru souřadnice normalizovat, což ale není v podstatě žádný problém, pokud se ví jak na to :-). Po nějaké době hledání na internetu jsem to vyřešil, jak, to je možné najít v kódu `mouse_win.cpp`.

4.3.3 Systém - příkazy

Vykonávání příkazů pro příkazovou řádku se řeší velice snadno pomocí funkce `system`, která je dostupná na obou platformách. Je třeba ošetřit jednu věc na každé platformě. Na *Unixu* je potřeba dát na konec příkazu `&` aby se spustil na pozadí a neblokoval výpočet. Na *Windows* je zase potřeba, aby se příkaz správně provedl, dát před něj slovo `start`.

4.4 Actions

Aby se s knihovnou lépe pracovalo a aby bylo možné události k zasílání nějak uchovávat a vyvolávat podle potřeby obsahuje knihovna rozšíření *Actions*. Pracuje s ackemi reprezentujícími události určené k zaslání do systému na

rozdíl od základní knihovny, která v podstatě reprezentuje pouze vstupní zařízení ne události samotné. V kódu rozšíření již není použit žádný platformě závislý kód, je to čistě nadstavba nad základní knihovnou. Důležitou vlastností, kterou rozšíření přináší je možnost řetezit akce za sebou a pak je najednou provádět. Více o používání v Uživatelské dokumentaci.