

OpenGL grafické komponenty

Milan Ilavský

Obsah

0 Úvod	2
0.1 Účel dokumentu	2
0.2 Členenie dokumentu	2
1 Špecifikácia	3
1.1 Oboznámenie sa s problémom	3
1.2 Špecifikovanie aplikácie	4
1.3 Využité technológie	4
2 Analýza	5
2.1 OpenGL v Jave	5
2.2 Implementované komponenty	5
2.3 Návrh	6
3 Implementácia	7
3.1 Diagram tried	7
3.2 Popis tried	9
3.2.1 Triedy glComponentes	9
3.2.1.1 Trieda glGraphicsComponents	9
3.2.1.2 Trieda glCanvas	9
3.2.1.3 Trieda glLabel	9
3.2.1.4 Trieda glButton	9
3.2.1.5 Trieda glProgressBar	10
3.2.1.6 Trieda glTrackBar	10
3.2.1.7 Trieda glTimer	10

3.2.1.8	Trieda eventProc	11
3.2.2	Triedy demo aplikácie	11
3.2.2.1	Trieda gulicka	11
3.2.2.2	Trieda projekt	11
4	Záver	12

0 Úvod

0.1 Účel dokumentu

Dokument je dokumentáciou k semestrálnemu projektu na predmet *Vývoj programov pre platformu Java 2*. Obsahuje základný opis aplikácie a použité technológie. Taktiež hovorí o tom čo program robí a čo naopak nerobí.

0.2 Členenie dokumentu

Dokument je členený na štyri základné časti. Prvá je špecifikácia, kde je opísaný problém a sú vypísané využívané technológie. Za ním nasleduje analýza, ktorá rozoberá riešenie problému. Ďalšou časťou je implementácia, v ktorej je podrobne popísané riešenie, ako slovne tak aj graficky. Nachádza sa tam podrobný popis tried. Spomenuté sú aj dôležité časti kódu. Ako posledný je záver, v ktorom sú opísané poznámky k implementácii a vysvetlené prípadné rozdiely medzi analýzou a samotnou implementáciou.

Ako súčasť dokumentácie môžeme pokladať aj generovanú JavaDoc dokumentáciu a inštalačnú a užívateľskú príručku.

1 Špecifikácia

1.1 Oboznámenie sa s problémom

Ako výslednú aplikáciu pre tento predmet som si vybral tvorbu OpenGL grafických komponentov. Moja aplikácia obsahuje len základné komponenty, ktoré na prvý pohľad nie sú ničím výnimočným. Výnimočnou časťou je práve OpenGL reprezentácia v pozadí.

V súčasnej dobe sa začína odbúravať zataženie procesora a záťaž sa snaží rozložiť na viaceré zariadenia. Tento decentralizovaný prístup ponúka mnohé výhody (prečo sa ma o vykresľovanie starať procesor, keď grafická karta a jej GPU je na to určený). Dajú sa nájsť aj negatíva, ale tie pri súčasných multimediálne vybavených počítačoch strácajú opodstatnenie (každý novší počítač je v súčasnosti vybavený grafickou kartou s 3D akceleráciou). Najznámejším a asi aj najúspešnejším projektom postavením na tejto myšlienke, čo sa týka grafiky, je **XGL**.

Tento projekt je vyvíjaný ako opensource pre viac platforiem (Linux, Unix, atd.) a poskytuje možnosti OpenGL a tým pádom aj akcelerácie, pre bežné pracovné prostredie. Verejnosti sú prezentované rôzne grafické efekty, ktoré sú podľa môjho názoru samoúčelne a pri praktickej práci nepoužiteľné. Ja by som naopak vyzdvihol tú na prvý pohľad neviditeľnú časť tohoto projektu, ktorou je práve využitie OpenGL v samotnom správcovi okien. Ako jednu z najväčších výhod by som spomenul vykresľovanie programov, kde sa program nekreslí hneď na plochu, ako to štandardne býva, ale vykresľuje sa do textúry, čím jeho vzhľad ostáva uložený a pri pohybe okna po obrazovke nie je potrebné jeho opakované prekresľovanie. To isté platí aj pre vyberanie (focus) okien, prípadne stiahnutie aplikácie do lišty a v Linuxe aj pre prepínanie virtuálnych plôch.

1.2 Špecifikovanie aplikácie

Aplikácia sa dá špecifikovať ako grafické komponenty, využívajúce OpenGL na svoje vykresľovanie.

Aplikácia neposkytuje také možnosti ako vyššie spomínaný projekt, ide skôr o demonštráciu ako to funguje. Samozrejme je od nej možné odvodiť ďalší projekt, ktorý bude obsahovať zložitejšie komponenty, kde bude urýchlenie viac poznateľné.

Samotná aplikácia sú komponenty, ktoré nie sú samostatne nejak viditeľné, preto bude implementovaná aj demonštratívna ukážka, kde budú tieto komponenty využité. Táto ukážka bude iba ukážkou, nebude to žiaden použiteľný program.

1.3 Využité technológie

Aplikácia bude postavené na grafickom rozhraní **AWT**, pričom základné okno bude práve **AWT**. V tomto okne bude implementovaná technológia **OpenGL**. Využívať sa bude **I/O obrázkov** a práca s **textúrami**. Program bude využívať aj **multithreading** na komponent, ktorý nebude grafický, ale bude využitý pri vykresľovaní.

2 Analýza

2.1 OpenGL v Java

Ako implementáciu OpenGL v Java som si vybral projekt JOGL. Tento projekt implementuje OpenGL priamo do grafického rozhrania AWT. Pre každý komponent sa vytvára vlastný OpenGL Context, čiže sa prekresľuje vždy len ten komponent, ktorý treba.

2.2 Implementované komponenty

Nasleduje zoznam komponentov, ktoré budem implementovať. Nie je žiaden problém komponenty rozšíriť o ďalšie, vďaka objektovej architektúre.

- **Button** ide o klasické tlačítko, ktoré môže, okrem klasického textu, obsahovať obrázok, ktorý sa môže dynamicky meniť podľa stavu tlačítka. Taktiež je v ňom implementovaná priehľadnosť (resp. polo-priehľadnosť) pozadia,
- **Label** komponent zobrazujúci zadaný text.
- **ProgresBar** komponent zobrazujúci postup nejakej akcie (napríklad postup kopírovania súboru). Je možné použiť štandardné vykresľovanie, alebo zmeniť jeho vzhľad pomocou externých obrázkov.
- **TrackBar** komponent podobný ProgresBaru, s tým rozdielom, že je interaktívny a je ním možné plynulé nastavovanie nejakej hodnoty (napr. ovládanie hlasitosti)
- **Timer** jediný negrafický komponent. Pomocou neho je možné jednoducho spúšťať užívateľskú funkciu v pravidelných intervaloch.
- **Canvas** OpenGL canvas, je v ňom možné jednoducho vykresľovať OpenGL scénu.

2.3 Návrh

Základným komponentom na ktorom budú postavené ostatné grafické komponenty bude komponent **GraphicsComponent**. Tento komponent bude zdedený od klasického AWT komponentu **Canvas**. Bude zastrešovať klasické umiestnenie komponentu, vytváranie OpenGL Contextu a s ním volať metódy na prekresľovanie jednotlivých komponentov. Od triedy tohoto komponentu budú zdedené ostatné grafické komponenty, ktoré budú obsahovať metódu na prekresľovanie, prípadne ďalšie metódy, ktoré budú vyplývať z potrieb daného komponentu.

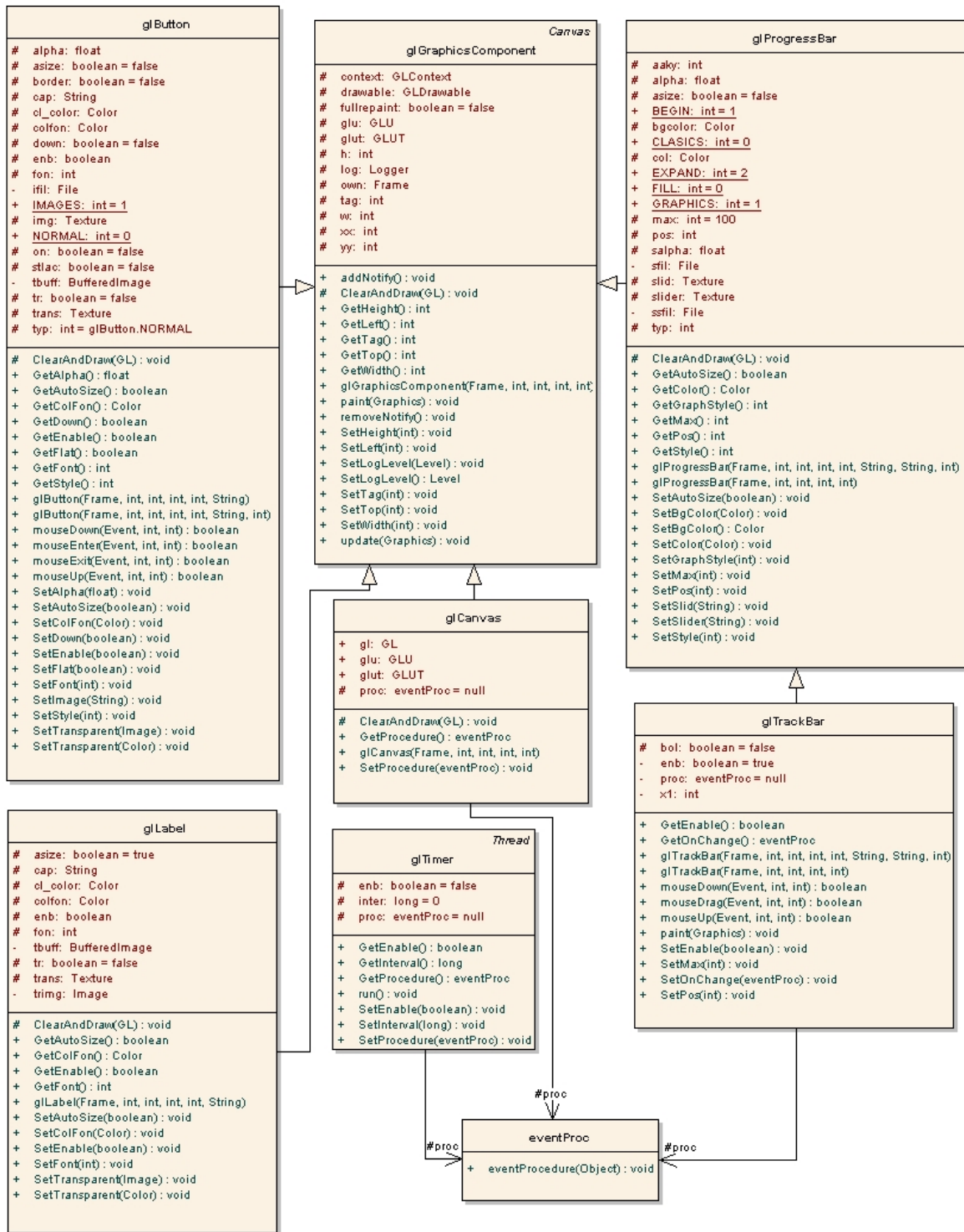
Od tohoto komponentu bude zdedený aj komponent **ProgresBar**. Nakoľko ďalší komponent **TrackBar**, je vo veľkej miere podobný **ProgresBaru**, bude od neho zdedený a v ňom bude implementované iba samotná interakcia užívateľa.

Negrafický komponent **Timer**, bude zdedený od triedy **threads**, kde sa bude vo vopred nastavených intervaloch volať vopred určená metóda. Volaná metóda bude z triedy **eventProc**, od ktorej sa pre každú ďalšiu metódu zdedí nová trieda a daná metóda sa prepíše.

3 Implementácia

3.1 Diagram tried

V tejto sekcii sa nachádza slovný aj grafický opis hierarchie tried využívanej v mojej aplikácii. Ako je jasne vidieť na obrázku 3.1, základnou triedou pre všetky grafické komponenty, je trieda **glGraphicsComponent**. Táto trieda je odvodená od komponentu **Canvas**, ktorý obsahuje **AWT**. Trieda **glTrackBar** vychádza s triedy **glProgressBar** a je v nej implementovaná iba samostatne interakcia. Uživatelské funkcie reagujúce na akcie komponentov rieši trieda **eventProc**.



Obrázok 3.1: Diagram tried

3.2 Popis tried

3.2.1 Triedy `glComponents`

Triedy ktoré obsahuje balík `glComponents`.

3.2.1.1 Trieda `glGraphicsComponents`

Komponent sa stará o umiestnenie komponentu v okne, čiže sa tu nastavuje pozícia a veľkosť. Okrem toho sa pri jeho vzniku vytvára **GL** objekt, ktorý sprístupňuje **OpenGL** funkcionality. S tým súvisí aj vytváranie grafického **glContextu** potrebného pre kreslenie. Ďalej je tu možné nastaviť úroveň logovanie, ktoré je štandardne nastavené na "level warning".

3.2.1.2 Trieda `glCanvas`

Táto trieda obsahuje primitívny **glCanvas** do ktorého je možné kresliť v ortogonálnej 2D projekcii. V prípade, že chce užívateľ použiť 3D projekciu, môže sa do nej prepnúť vo vykresľovacej metode, ktorú komponentu poskytne vo forme triedy **eventProc**. Pri tomto komponente je metóda `repaint`, ako `public`. Aby bolo zvonku možné vynútiť prekreslenie komponentu.

3.2.1.3 Trieda `glLabel`

Pomocou tohoto komponentu sa vykresľuje jednoduchý text. Je možné nastaviť písmo (z fontov ponúkaných knižnicou `glut`) a farbu písma. Taktiež je možné použiť komponentu priehľadné pozadie, kde je mu potrebné nastaviť obrázok/farbu pozadia. Veľkosť komponentu je možné nechať nastaviť automaticky, podľa veľkosti textu ktorý sa v ňom nachádza.

3.2.1.4 Trieda `glButton`

Komponent tlačidlo. Pri tomto tlačidle je možné použiť klasický štýl s nápisom bez obrázkov alebo je možné použiť obrázkové tlačidlo. Obrázkové tlačidlo môžeme rozlíšiť na dva typy. V prvom prípade sa jedna o klasický obrázok ktorá sa zobrazí namiesto

textu. V druhom prípade sa používa obrázok, ktorý sa v skutočnosti skladá z troch obrázkov umiestnených horizontálne za sebou. Prvá časť obrázku reprezentuje normálny stav, druhá časť je aktívny stav, čo je stav, keď sa nad tlačidlom nachádza kurzor myši a tretia časť obrázku, je stav tlačidla po zatlačení. Okrem toho je možné urobiť tlačidlo priehľadné tak ako pri **glLabel** spomínanom vyššie. Tlačidlo si dokáže v prípade potreby nastaviť veľkosť automaticky podľa vstupného obrázku.

3.2.1.5 Trieda **glProgressBar**

Tento komponent je taktiež možné použiť v dvoch typoch. Pri prvom type sa jedná o klasický vzhľad bez obrázkov. Je možné nastaviť farbu výplne a pozadia. Pri druhom type sa zadávajú dva obrázky. Jeden na pozadie a druhý ako výplň. Pri použití obrázkov je možné použiť 3 spôsoby naplňania výplne. Prvý je klasický, kedy sa z obrázku odrezáva a naplňa sa postupne od začiatku. Pri druhom spôsobe sa naplňa obrázok tak, že sa polovica posuvníka skopíruje z ľavej časti obrázku a druhá z pravej. Tretí spôsob obrázok posuvníka rozťahuje. Samozrejme je možné nastaviť maximálnu hodnotu na posuvníku a aktuálnu pozíciu posuvníka. Aj pri tomto komponente je možné nastaviť automatické zistenie veľkosti podľa obrázku.

3.2.1.6 Trieda **glTrackBar**

Komponent je potomkom komponentu **glProgressBar**. V princípe je to to isté, akurát v prípade **glTrackBaru** je možná interakcia užívateľa, ktorý môže hodnoty nastavovať myšou. Toto si vyžaduje nastaviť aj metodu, ktorá sa bude volať pri zmene tejto hodnoty. Na nastavenie sa využíva klasická metóda z triedy **eventProc**. Ostatné nastavenia majú zdedené od už vyššie spomínaného komponentu.

3.2.1.7 Trieda **glTimer**

Jediný negrafický komponent. Na svoju činnosť využíva paralelné vlákno. Je možné nastaviť interval v ktorom sa má spúšťať užívateľská funkcia a samozrejme aj samotnú užívateľskú funkciu prostredníctvom triedy **eventProc**. Časovač je možné pozastaviť a potom znovu spustiť.

3.2.1.8 Trieda `eventProc`

Táto trieda obsahuje metodu `eventProcedure`, ktorá ma byť prekrytá metódou definovanou užívateľom. Táto metóda sa napokon púšťa pri vykonaní nejakej akcie (niečo na spôsob callback funkcií). Metóda ma jeden parameter a tým je objekt ktorý metódu vyvolal.

3.2.2 Triedy demo aplikácie

Vyššie sú spomenuté všetky funkcie využívané komponentmi. Okrem týchto tried som implementoval ďalšie dve triedy ktoré demonštrujú funkcionality vyššie spomínaných komponentov.

3.2.2.1 Trieda `gulicka`

Trieda reprezentuje jeden bod, ktorý sa bude kresliť vo formulárovom okne. Obsahuje funkciu ktorá prepočíta jeho pozíciu, podľa aktuálneho smeru a rýchlosti. Trieda nesie aj informáciu o farbe bodu.

3.2.2.2 Trieda `projekt`

Táto trieda definuje formulárové okno v ktorom budú využívané vyššie spomínané komponenty. V pravej časti okna sa nachádza komponent `glCanvas` v ktorom bude vykreslovaný náhodný pohyb bodov. V ľavej hornej časti sa nachádza ovládanie animácie. Intuitívne vyzerajúcimi tlačidlami sa dá spustiť respektíve zastaviť náhodný pohyb bodov alebo pridať respektíve odobrať bod. Pod týmito tlačidlami sa nachádza posuvník, pomocou ktorého je možné meniť rýchlosť pohybu bodov. Ďalším posuvníkom sa dá ovládať komponent `glProgressBar` ktorý sa nachádza na spodku okna. Nad týmto komponentom sa nachádza opäť `glProgressBar`, ktorý v prípade spustenej animácie náhodne mení.

Zobrazované body sú objektom triedy `gulicka` ktorú som opísal vyššie. Body sú uložené v `collection`, presnejšie `List<gulicka>`. Spomínaný `collection` je synchronizovaný pre prácu vo viacerých vláknach.

4 Záver

Ako som už vyššie spomenul snaha odbúrať záťaž procesora a rozkladať ju na ostatné komponenty počítača začína byť čoraz badateľnejšia. Tento projekt nepatrí k rozsiahlym riešeniam a snaží sa viacmenej ukázať toho ako postupovať. Mnohé komponenty sú len v surovom stave a existuje ešte veľa vecí ktoré by bolo možné pridať ale vzhľadom k rozsahu a účelu projektu neboli implementované. Predkladanú knižnicu treba chápať ako začiatok, do ktorého nie je vďaka flexibilnému návrhu problém pridať nový komponent.

Pri testovaní implementácie som narazil na jeden problém. Keď beží aplikácia pod operačným systémom Windows, nastávajú problémy v zisťovaní veľkosti obrázku. Riešenie tohoto problému by mohlo byť v prechode na novšiu verziu knižnice JOGL.

Rozdiely medzi návrhom a implementáciou boli minimálne a aj to len v názvoch tried.

Program sa nainštaluje prostým rozbalením súboru *.zip a následne sa spustí z príkazového riadku príkazom `java -jar vppj.jar`. Pre spustenie je vyžadovaná knižnica s názvom **jogl.jar** a k nej sú potrebné priložené súbory *.dll. Adresár **images** obsahuje obrázky použité v demo aplikácií. V prípade že sa tam obrázky nebudú nachádzať zobrazia sa komponenty v štandardnom móde.

Pri dokumentácii je priložený obrázok obsahujúci class diagram. Kvôli lepšej čitateľnosti som obrázok priložil ako externý súbor **class.jpg**.