GFG

Uživatelské rozhraní pro hry

Bc. Michal Jirouš
V Praze dne
10.10.2008

Obsah

1. Úvod.....3
2. Předpoklady vzniku.....3
3. Obecně.....4
4. Komponenty.....5
Tlačítka.....6
Popisky.....6
Kontejnery.....7
Ostatní.....7
5. Organizace komponent.....9
6. Kořenová komponenta.....9
7. Události.....12
Vykreslování.....13
Běh.....13
Vstupní kontrolér.....14
Parametry zpráv vstupního kontroléru.....18
Nastavení dočasného zakázání komponenty.....18
Nastavení povolení.....19
Nastavení alfa multiplikátoru.....20
Ztráta fokusu následovníka.....20
Dej mi fokus.....20
„isFocusable“.....20
Závěr kapitoly.....21
8. Callback funkce.....21
9. Implementace komponent.....21
Skupina tlačítka.....22
Skupina popisky.....23
Skupina kontejnery.....24
Skupina ostatní.....27
Hlavní okno (Třída CMainWindow).....32
10. Shrnutí a závěr.....32
11. Zdroje a literatura.....33

GFG

1. Úvod

Zkratka GFG znamená Graphical user interface For Games. Jedná se o uživatelské rozhraní vytvořené přímo na míru pro potřeby her, které chtějí využívat okenní systém. Důvod, proč to využít, je větší přehlednost různých nabídek a menu, které se pomocí tohoto nástroje dají vytvářet. Jsme přeci jen zvyklí na uživatelské rozhraní operačních systémů, které možností oken také využívají. Primárním účelem uživatelského rozhraní je interakce s uživatelem, proto každé takové řešení musí obsahovat rozmanité množství ovládacích prvků, pomocí nichž tato komunikace probíhá. Všechny tyto prvky musejí mít definovaný určitý vzhled, který bude co možná graficky nejpřijatelnější pro většinu lidí. Jako doplněk jsou vhodné animace a průhlednost některých komponent rozhraní, které celkový vzhled dále vylepší. Tyto vlastnosti jsou v tomto rozhraní implementovány a i díky tomu jsou pro hry vhodné, protože grafické prvky jsou v tomto oboru velice důležité. Z technického hlediska je tento systém postaven na grafické knihovně OpenGL, která se stará o vykreslování prvků a všechny implementace se vytvářely v programovacím jazyce C++. I díky tomu lze rozhraní využívat na různých systémech. Nová okna se ale nevytváří v operačním systému, na kterém běží, ale vytvářejí se nové grafické objekty (okna) v hlavním okně OpenGL. Jelikož rozhraní samo o sobě žádná specifická uspořádání oken nevytváří, protože jde pouze o nástroj, který se pro jeho tvorbu používá, je potřeba, aby bylo použití co nejsnadnější. Proto byl při implementaci kladen důraz na přehlednost a jednoduchost , aby bylo použití snadné. K tomu jsem využil objektového návrhu, hierarchie ovládacích prvků a vhodného pojmenování funkcí.

2. Předpoklady vzniku

Proč vytvářet uživatelské rozhraní čistě jen pro hry? Důvodem je to, že by nebylo vůbec jednoduché vytvořit systém, který by byl vhodný pro klasické aplikace jako jsou například kancelářské balíky, multimediální přehrávače a podobné, ale i pro hry. Právě tento obor je svým způsobem jedinečný a každý, kdo už někdy hrál alespoň několik her, mi dá za pravdu, že uživatelské rozhraní se snaží vypadat jedinečně, aby se od ostatních her odlišovalo. Uživatelským rozhraním se ve hrách myslí hlavně hlavní menu, ale i různé inventáře, nabídky apod. Tyto prvky se od neherních liší hlavně vzhledem, někdy i různou funkčností. Pro obyčejného uživatele je ale pravděpodobně nejlepší, držet se z hlediska funkčnosti zaběhnutých standardů, takže i v tomto systému jsem se snažil tyto základy neměnit. A nyní ten hlavní rozdíl mezi herním a neherním uživatelským rozhraním.

- Neherní aplikace nebo velice jednoduchá herní aplikace využívá rozhraní operačního systému a může si vytvářet nová okna jeho prostřednictvím. Pro tyto aplikace je to normální, mohou být vícevláknové či víceprocesové. Není to žádný problém, uživatele to nerozptyluje. Skrz rozhraní se program přímo ovládá.
- V herním prostředí (mám na mysli složitější grafické hry) je toto uživatelské rozhraní jen součástí interakce a neovládá přímo hru. Běží v jejím okně a bývá vykreslováno pomocí nějakého grafického API jako například OpenGL nebo DirectX společně s hlavní scénou. Hry ve většině případů nemají příliš možností využit více procesorů. V budoucnu se ale tento problém pravděpodobně vyřeší.

Takže hlavním rozdílem je způsob vykreslování a umístění a kvůli těmto rozdílům je toto rozhraní určeno jen pro hry, protože vykreslování probíhá pomocí OpenGL a běží v okně hry, což jsou vlastně dvě podmínky, které musí být splněny.
Proč vytvářet úplně novou implementaci? Ano, existuje celá řada uživatelských rozhraní. Několik

jich zde popíši.

- GTK+ - je to velice bohaté prostředí pro tvorbu GUI a pracuje se s ním jednoduše. Aplikace postavené na tomto rozhraní jsou platformě nezávislé a lze je vytvářet v různých programovacích jazycích. Primárně v jazyce C, ale je ho možné použít i v C++, Python a v C#. Je licencováno pod GNU LGPL 2.1, což ho dovoluje používat zcela zdarma. Z vlastní zkušenosti vím, jak je tento nástroj silný, ale pro účely her se spíše nehodí. Běží na rozhraní X-Window systém a bylo by asi velice těžké, snad i nemožné ho zakomponovat do hry jako hlavní menu. GTK+ je známé z grafického programu GIMP a z prostředí GNOME.
- QT – spolu s GTK+ jsou neoblíbenějšími nástroji pro tvorbu uživatelského rozhraní. Je též multiplatformní. Podporuje jazyk C++, ale i Javu, je to grafická knihovna, ale nyní zprostředkovává i tvorbu GUI. Co se týče licence, tak existuje verze zdarma s otevřeným zdrojovým kódem, ale i placené verze. Navíc umožňuje integrovat editor GUI do vývojového nástroje Visual Studio a Eclipse. Nicméně pro vytvoření GUI pro hry opět není vhodné.
- GUICHAN – tato knihovna byla vytvořena pro potřeby her jako tento projekt. Neimplementuje žádné vykreslování a tím je nezávislá na grafickém API. Součástí je pouze několik základních ovládacích prvků a každý uživatel si musí sám vytvořit své nové komponenty a implementovat vykreslování. Důvodů proč jsem nevyužil právě tohoto nástroje i přesto, co všechno poskytuje, je hned několik. Za prvé jsem chtěl poznat a pochopit principy tvorby uživatelského rozhraní a proto provést implementaci téměř od začátku. Druhým důvodem byly velice specifické nároky na výsledek, které znemožňovaly použití externím zdrojů. Výsledné rozhraní mělo být použito v hlavním projektu ChronoPhobia, kde mělo zastávat úlohu hlavního menu, načítacích oken i prostředí HUD (Head Up Display) a umožňovat mnohdy téměř přímý přístup k datům.

Takhle bych mohl dál jmenovat další API jako například LibUFO, GLOW, GLUI, Agar atd. Všechny tyto knihovny, stejně tak jako GTK+ a QT, se snaží vytvářet uživatelské rozhraní pro neherní aplikace a hry mohou běžet uvnitř okna. Tím nesplňují druhou podmínku, kde je řečeno, že GUI musí běžet v okně hry. Vykreslování probíhá u každé implementace jinak. Některé používají X-Window systém, některým stačí OpenGL.

Závěrem lze říci, že každá rozsáhlejší hra má své unikátní uživatelské rozhraní (hlavně hlavní menu) jedinečné a vytvořit obecné rozhraní není jednoduché. GUICHAN může být dobrou volbou pokud není čas a chuť vytvářet nové rozhraní.

Z těchto důvodů jsem vytvořil nové rozhraní, díky kterému budu moci ve své hře vytvořit GUI, které bude přesně podle mých představ.

3. Obecně

Rozhraní GFG není knihovnou. Ke své funkci potřebuje nějaké jiné API, od kterého bude přijímat události a reagovat na ně a kontext pro vykreslování pomocí OpenGL, takže není samostatná. Díky tomuto kroku je sice těžší porozumění a použití, ale zato získáme nástroj, který je flexibilní a půjde ho nasadit všude, kde se používá OpenGL API k vykreslování. GFG je okenní uživatelské rozhraní, což je velice rozšířená forma v neherních aplikacích. Ve hrách se objevují různé koncepty. Nejrozšířenější je asi menu, kde se vyskytuje několik tlačítek vertikálně a při kliknutí se zobrazí určité okno, přičemž otevřené je vždy pouze jediné. Ale například ve hře Half-Life 2, která běží na platformě Steam, se vyskytuje okenní systém jako uživatelské rozhraní menu. Právě zde jsem se inspiroval, protože mám z toho dojem přehlednosti a jednoduchosti na použití a proto je GFG svým stylem podobné, nikoliv však konečnou podobou.

Tento systém umožňuje zpracovávat a vykreslovat naráz více oken, která se však mohou vyskytovat pouze v hlavním okně aplikace. Tyto okna jsou vlastně pouze grafickými elementy, které grafická knihovna zobrazí. Každý element je zde označován jako komponenta, což můžou být kromě okna standardní prvky uživatelských rozhraní jako například tlačítka, popisky, textové pole, seznam, obrázky. Rozhraní ale neobsahuje všechny známé prvky, ale pouze ty, které mohou být ve hře, pro kterou je určeno, užitečné. Vytvoření nového prvku ale není příliš složitá věc, stačí porozumět základní myšlence implementace. GFG umožňuje sdružovat jednotlivé komponenty do skupin označovaných jako kontejnery, kterých existuje několik typů a výběr záleží pouze na programátorovi, jak chce, aby výsledné okno vypadalo. Podrobný popis bude uveden v další kapitole. Kromě komponent se zde ještě vyskytuje objekt hlavní okno, které je prostředníkem mezi aplikací a samotným uživatelským rozhraním. Přijímá události, posílá je komponentám a vykresluje je. Aby rozhraní umožňovalo interakci s uživatelem, bylo potřeba zajistit způsob reakce na určité události vyvolané uživatelem. Komponenty samotné si obsluhují události jako například pohyb myší, kliknutí. Navíc u tlačítek lze zaregistrovat callback funkci, která se zavolá, když na něj uživatel klikne myší a nebo stiskne klávesu enter, když je tlačítko označeno jako aktivní (vlastní fokus). K funkci lze ještě připojit určitá data, která se při volání funkce předají jako parametr. Tím se obsluha značně zjednoduší (tento způsob je v jiných rozhraních velice oblíbený). Jak už bylo řečeno, komponenty se mohou stát aktivními, ale v jeden okamžik pouze jeden. Když je aktivní, tak přijímá události vyvolané klávesnicí, takže když textové pole získá fokus, je v něm možné zapisovat nebo upravovat text. Mezi prvky, které se mohou stát aktivními je možné přepínat.

4. Komponenty

Systém je navržen objektově, takže využívá hierarchie komponent. To značně zjednodušuje tvorbu nových komponent, které se liší jen nepatrně od nějaké již vytvořené. Komponenty dohromady tvoří strom, kde kořenem je společný předek, který definuje společné a povinné funkce, základní inicializace a proměnné. Díky dědičnosti tak všechny komponenty tyto vlastnosti získají též. Některé komponenty lze považovat za základní, protože se pak dále využívají k tvorbě jiných, které se z nich skládají, ale neodvozují. Například rolovací panel, který se využívá k posouvání pohledu, používá tři tlačítka, a nebo obyčejné okno, které využívá přepínací tlačítka. Základními komponentami je tlačítko, obrázek ale i rolovací panel. Při implementaci pak bylo nutné tyto komponenty nadefinovat co nejdříve, aby se dali následně používat. Všechny komponenty lze rozdělit do čtyř skupin:

- Tlačítka
- Popisky
- Kontejnery
- Ostatní

Strom hierarchie je vyobrazen na obrázku 1 .

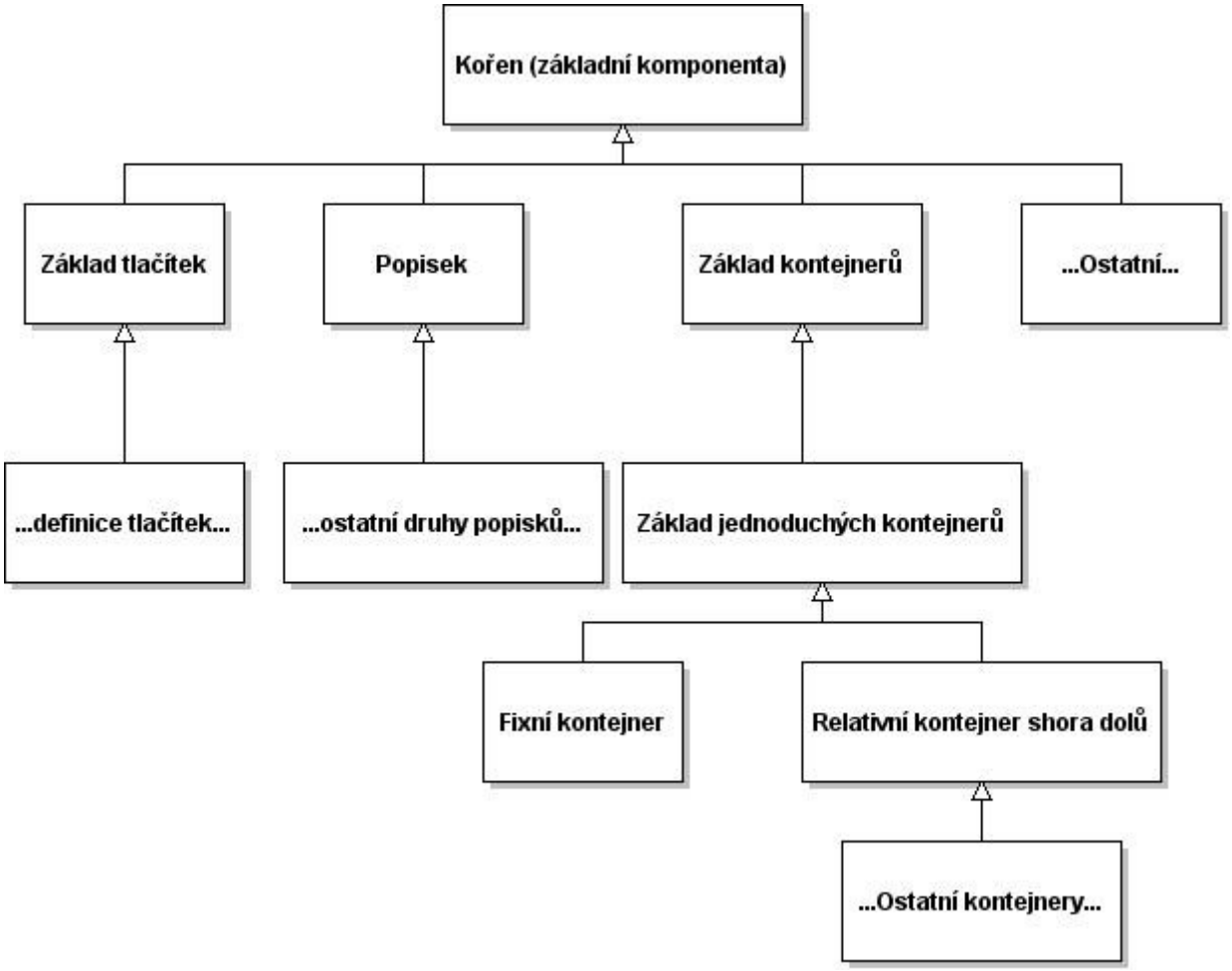
Komponenty ve skupině tlačítka jsou odvozeny od společného základního kořene tlačítek. Je to z toho důvodu, že tlačítka jsou si velice podobná. Ve skupině popisky je jako základ popisek. Ostatní komponenty se od něho jen nepatrně liší. U kontejnerů je situace složitější. Kromě základu kontejnerů dále existuje ještě základ jednoduchých kontejnerů, ze kterého pak vznikají samotné konkrétní implementace. To vzniklo už v počátcích vývoje, kdy se předpokládaly ještě další typy kontejnerů, které se od skupiny, která je definována jednoduchými kontejnery, liší. Tím měl být například kontejner definovaný tabulkou. V budoucnu bude možná implementován. Všechny ostatní komponenty mají základ v kořeni celé hierachie.

Následuje stručný popis jednotlivých komponent.

Tlačítka

Komponenty v této skupině mají několik společných vlastností:

- Mohou se nacházet ve dvou stavech: v klidu a stisknuté. Pro každý stav mají přidělenu jednu texturu.
- Při pohybu myší nad oblastí, která je jim přidělena nastává změna barvy
- Při kliknutí se mění stav a nastává reakce, kdy se volá příslušná callback funkce
- Mohou získat fokus a reagovat na události z klávesnice



Obrázek 1: Strom hierarchie komponent

Prvky, které do této skupiny patří jsou:

- Tlačítko – základní komponenta. Po stisknutí tlačítka myši se přepne do stavu stisknuté a po uvolnění myši zpět a zároveň se zavolá callback funkce.
- Zatržítko – Má jiný vzhled než tlačítko a při každém kliknutí se mění zaškrtnutí
- Přepínač – Stejné jako tlačítko, při uvolnění myši se nevrací do původního stavu. Stav se tím pádem při každém kliknutí přepne.
- Tlačítko volby – většinou se sdružují do skupin a dávají uživateli na výběr mezi několika možnostmi

Popisky

Sem patří komponenty, které zobrazují text. Systém obsahuje čtyři typy popisků, které se liší svým zobrazením a zarovnáním textu. Popisky nemohou získat fokus a kromě rolovacího popisku ani nereagují na pohyb či kliknutí myši.

- Popisek – je základní, zobrazuje rámeček a text zarovnává vlevo nahoru
- Jednoduchý popisek – nezobrazuje rámeček text zarovnává vertikálně na střed a horizontálně vlevo.
- Popisek bez rámečku – stejný jako popisek, ale bez rámečku
- Rolovací popisek – stejný jako popisek, ale obsahuje ještě navíc rolovací panel, který dovoluje posouvat text vertikálně a tím dovolit zobrazení více informací.

Kontejnery

Jsou to speciální komponenty, které slouží k uspořádání komponent v okně. Mohou zobrazovat rámeček i s popisem, ale primární nevykreslují nic. Události, které přijmou, předají dále svým komponentám a pokud ji ony nezpracují, tak na ni zareagují samy nebo ji ignorují. Kontejnery se dělí na tři podskupiny, které se liší uspořádáním komponent.

- Fixní – komponenty se umísťují podle zadaných souřadnic a při změně velikosti kontejneru tyto pozice zachovávají
- Relativní – dělí se na dva typy: shora dolů a zleva doprava. Komponentám se nastavuje priorita, kde čím vyšší číslo, tím větší část prostoru zabere. Pokud se jedná o typ shora dolů, tak komponenty mají vždy stejnou šířku a výšku si rozdělují a typu zleva doprava je to naopak.
- Semi-relativní – zde je to velice podobné relativnímu kontejneru, ale prvky zde nemají žádnou prioritu a uspořádají se postupně od shora resp. zleva dle nastavené velikosti. Při změně velikosti se pak výška resp. šířka nemění.

Uspořádání bude podrobně vysvětleno později. Kontejnery jsou jediné komponenty kromě hlavního okna, které mohou obsahovat více než jednu komponentu. Lze je vkládat do sebe a tím vytvářet komplexní uživatelské rozhraní.

Ostatní

Následující komponenty se řadí do této skupiny, jelikož nemají podobné vlastnosti.

- Okno – okno s horním panelem se dvěma kontrolními tlačítky, kde první okno zabalí jen do panelu a druhé okno zavře. Může obsahovat a kontrolovat jen jednu komponentu. Je to základní stavební kámen GUI.
- Pole tlačítek volby – komponenta umožňuje sdružovat tlačítka volby tak, aby dohromady vytvořila nabídku možností, z nichž si uživatel může vybrat právě jednu.
- Dělící čára – implementována je pouze horizontální.
- Šoupátko (slider) - používá se k nastavení hodnoty z určitého intervalu. Známý je z přehrávačů hudby nebo videa, kde určuje aktuální místo přehrávání. Tato komponenta se často používá pro určování citlivosti myši v mnoha hrách.
- Textové pole – klasické jednořádkové pole, do kterého lze zapisovat a slouží tak ke znakové interakci.
- Indikátor stavu operace (progress bar) – panel slouží ke grafickému zobrazení průběhu nějaké operace. Dále umožňuje i textové zobrazení ve formátu procent. Běžně se využívá v

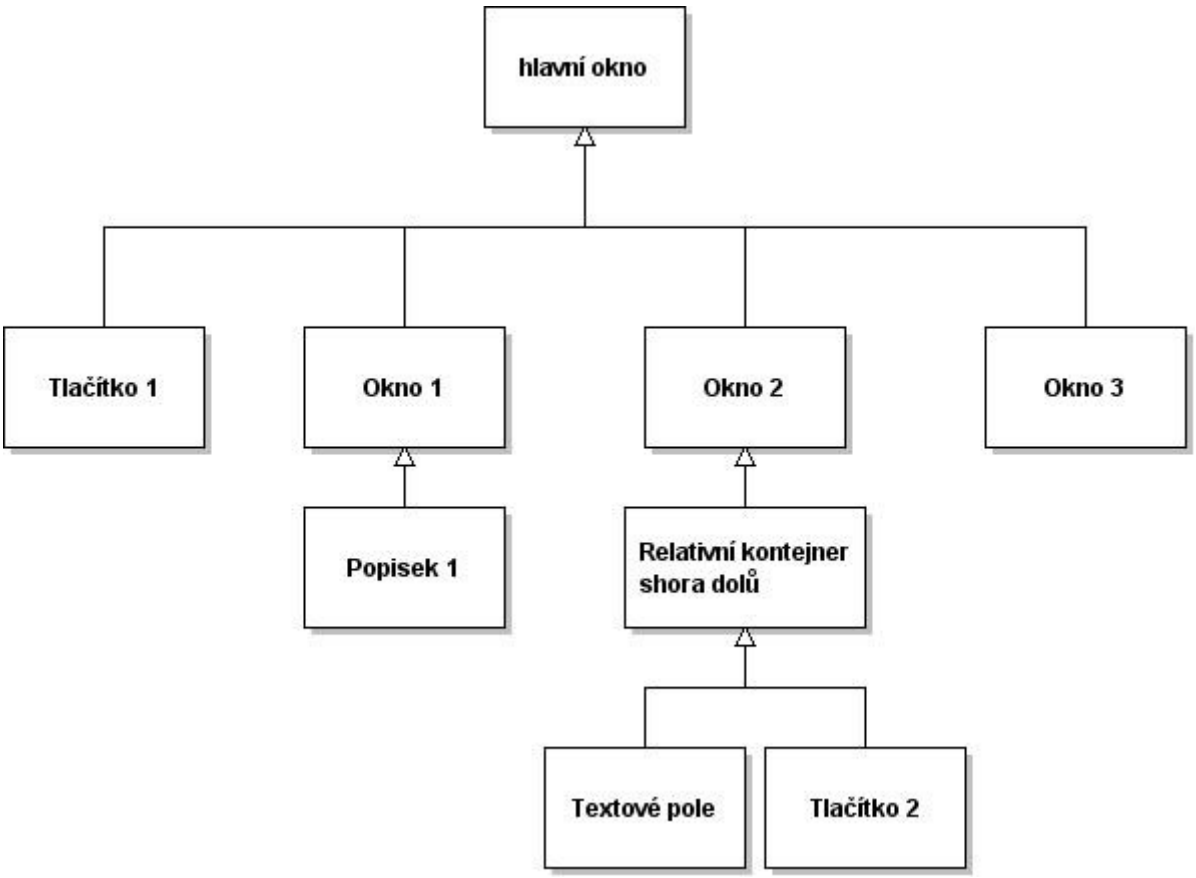
delších procesech, kdy programátor vyžaduje určitou formu ukazatele průběhu.

- Záložky – klasická komponenta, která umožňuje přepínat mezi různými obsahy v jednom okně. Využívá se v případech, kdy je potřeba zobrazit více informací než lze v jednom okně zobrazit. Pak nastávají dvě možnosti: využít rolovací panel a tím zvětšit možný zobrazovací prostor a nebo využít právě záložek a rozdělit data mezi více bloků.
- Rolovací panel – obsahuje lištu s jezdcem, který se využívá k posouvání obsahu nějakého pole (okna). Dále obsahuje dvě tlačítka, z nichž jedno posouvá obsah dolů a druhý nahoru. Jedná se tedy o vertikální panel.
- Seznam – tato komponenta umožňuje do sebe vkládat panely, které mohou obsahovat různé komponenty. Tyto panely jsou uspořádány vertikálně a uživatel má možnost vybrat právě jeden. Ten se pak stane aktivním a lze ho ovládat jako jinou komponentu. Využívá rolovací panel k posouvání obsahu, když je větší než výška komponenty.
- Dialog (Message box) – běžně se používá k rychlému získání odpovědi od uživatele. Při vyvolání se ostatní okna zablokují a čeká se, dokud uživatel nevybere jednu z možností, které dialog nabízí. Message box může ovládat pouze jednu komponentu, takže se využívá kontejnerů, do kterých se celý obsah vkládá. V implementaci je globální funkce, která automaticky vytváří dialog s jedním obrázkem, jedním textem, který je většinou otázkou a dvěma tlačítky. Nastavení těchto komponent probíhá formou předání parametrů oné funkci. Předdefinovány jsou čtyři možné obrázky dle významu dialogu:
  - informativní
  - otázka
  - upozornění
  - chyba

Komponenty budou podrobně rozepsány včetně obrázků v kapitole o implementaci komponent.

5. Organizace komponent

Pro pochopení funkčnosti celého systému je třeba vysvětlit, jak se komponenty, které jsou už v hotovém uspořádání v oknech, organizují. V kapitole 6 Kořenová komponenta bude řečeno, že každá komponenta zná svého předchůdce. Takže vytvářejí strom, kde kořenem je hlavní okno. Aby byl ale systém funkční, tak některé komponenty mohou mít potomky. Většinou maximálně jednoho, ale jak už bylo řečeno, tak kontejnery a hlavní okno jich mohou mít více. Tato organizace tak umožňuje přizpůsobovat pozice a velikosti komponent svým následovníkům. Většinou se velikost následovníka odvíjí od velikosti vlastní komponenty kromě kontejnerů, které jsou speciální, a samozřejmě hlavního okna, kam se prvky umísťují s pevně danou pozicí i velikostí a dále se nemění. Na obrázku 2 je příklad takového stromu:



Obrázek 2: Organizace komponent

Na obrázku je vidět, že hlavní okno ovládá tři okna a jedno tlačítko. Okno 1 má jako následovníka pouze popisek 1. Okno 2 už je složitější, protože jako jeho následovník je Relativní kontejner, který dále obsahuje textové pole a tlačítko 2. Okno 3 žádného následovníka nemá. Tento příklad byl velice jednoduchý, ale názorný.

6. Kořenová komponenta

Nyní už známe organizaci struktur ale než přejdu k událostem, které uživatel svojí interakcí se systémem vytvoří, je nejprve potřeba znát základní vlastnosti všech komponent. Jelikož všechny komponenty dědí vlastnosti kořene, tak právě zde jsou definovány společné vlastnosti.

- Název – jedná se o řetězec znaků a je na každé komponentě, jak ho využije.
- Typ vstupních událostí – pokud komponenta využívá standardní obsluhy událostí, tak lze

nastavit na které typu bude reagovat (myš, klávesnice).
- Typ objektu – identifikační číslo daného typu komponenty.
- Identifikační číslo – slouží především u tlačítek k jejich rozlišení při volání callback funkce.
- Ořezávací box – vymezuje hranice komponenty, kde se může vykreslovat.
- Barva pozadí – barva, se kterou může každá komponenta naložit jinak.
- Alfa multiplikátor – hodnota z rozsahu 0-1, která určuje průhlednost komponenty.
- Viditelnost – logická hodnota, která určuje, zda je komponenta vidět nebo ne.
- Indikátor kurzoru myši nad komponentou – logická hodnota.
- Povolení – určuje, zda komponenta reaguje na události od uživatele.
- Aktivita – udává, jestli aktuálně vlastní fokus.
- Dočasné zakázání – pokud je hodnota nastavena na „pravda“, tak některá z rodičovských. komponent není povolena.
- Rodič – odkaz na rodiče.
- Výška, šířka, x-ová a y-ová souřadnice.
- Callback – struktura, která uchovává callback funkci a připojená data.

Všem těmto vlastnostem v implementaci odpovídají proměnné, ke kterým patří příslušné funkce, které jejich hodnoty nastavují resp. vracejí (setter a getter).

Každá komponenta má definovaný okraj (padding), jehož výchozí hodnota je jeden pixel. Tento okraj se definuje pomocí určitých funkcí, které mohou nastavit najednou všechny čtyři okraje nebo každý zvlášť. A samozřejmě k nim existují opačné funkce, které tyto hodnoty vracejí. Jistě si lze všimnout, že vlastnost okraj není výše v proměnných uvedena. Tyto informace jsou totiž uchovávány v objektu ořezávací box, který se při jakékoliv změně pozice, šířky nebo výšky musí vždy aktualizovat.

Dále obsahuje dvě funkce, které se používají ke zjištění, zda se kurzor myši nachází v oblasti komponenty. První funkce testuje celou oblast (výška a šířka komponenty) a druhá oblast ořezávacího boxu. Každá se používá při jiné příležitosti, kde té druhé využívají většinou pouze komponenty, které mohou mít potomky, protože ti mají přidělen pouze prostor ořezávacího boxu.

Nyní uvedu funkce, které jsou důležité pro zpracování událostí a u každé komponenty může být její tělo jiné, a tak mohou provádět něco jiného. Tyto funkce jsou tzv. *dopředné* (forward; viz kapitola o událostech).
- Vykreslování – to je také událost, kterou vyvolá samotná aplikace.
- Běh – funkce se volá v určitém intervalu (30ms) a umožňuje tak nejrůznější animace.
- Vstupní kontrolér – zpracovává události klávesnice a myši.
- Nastavení dočasného zakázání komponenty – komponenty, které mohou mít následovníky, mají tělo funkce odlišné od těch, které je nemají.
- Nastavení povolení – stejný rozdíl jako u předchozí funkce.
- Nastavení alfa multiplikátoru – stejný rozdíl jako u předchozí funkce.

Následují funkce, které se starají o ovládání fokusu, jsou označovány jako *zpětné* (backward):
- Ztráta fokusu následovníka – funkce vrací logickou hodnotu, která určuje, zda by se měl následovník dobrovolně vzdát fokusu. U komponent, které nemají předchůdce (například okno), je výsledkem funkce hodnota FALSE – nepravda, protože existuje pouze jeden následovník, takže komponenta si svůj fokus ponechá.
- Dej mi fokus – jak už název napovídá, funkce předá fokus vybrané komponentě.

Další zde zmíněnou funkcí je: „isFocusable“ – vrací TRUE (pravda), pokud se jedná o komponentu,

která fokus získat může (například tlačítko).

Následují dva typy funkcí, které jsou reakcí na určité události. Tyto funkce si každá komponenta implementuje sama a hlavně ty z druhé části jsou důležité pro komponenty s více následovníky, protože ony jim musí upravovat pozici, výšku a šířku při jakékoliv změně pozice či velikosti.

1. Reakce vyvolané myší nebo klávesnicí. Zde použiji přímo názvů funkcí
   - onMouseOver – vyvolá se, pokud se kurzor myši pohybuje v oblasti komponenty
   - onMouseButtonUp – spustí se při puštění tlačítka myši
   - onMouseButtonDown - spustí se při stisknutí tlačítka myši nad oblastí komponenty
   - onKeyboardDown – vyvolá se při stisknutí klávesy a komponenta má fokus
   - onSpecialKeyDown - vyvolá se při stisknutí speciální klávesy a komponenta má fokus
2. Reakce vyvolané změnou pozice nebo velikosti komponenty a ořezávacího boxu.
   - onComponentTranslate – vyvolá se při změně pozice celé komponenty
   - onComponentResize – vyvolá se při změně velikosti celé komponenty
   - onScissorBoxTranslate – vyvolá se při změně pozice ořezávacího boxu
   - onScissorBoxResize – vyvolá se při změně velikosti ořezávacího boxu

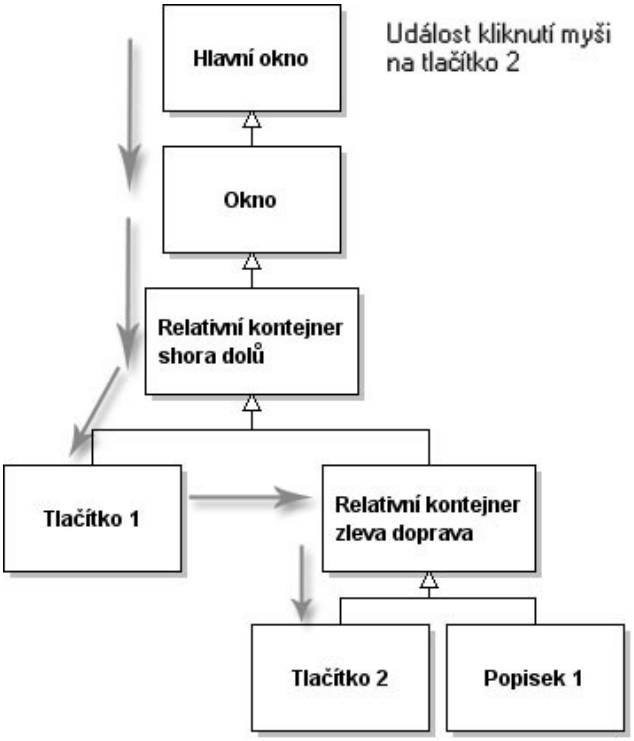
V této kapitole jsem ukázal, jaké vlastnosti má kořenová komponenta a tím pádem i všichni potomci. Kromě proměnných a jejich přidružených funkcí (setter a getter) jsem uvedl také funkce, které se starají o obsluhu událostí a budou podrobně rozebírány v kapitole o událostech.

7. Události

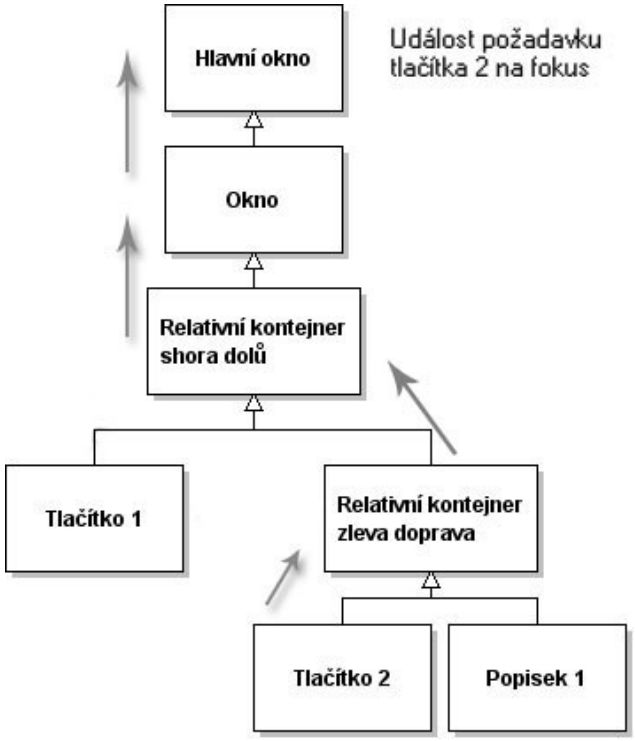
Bez událostí by nebyla žádná interakce, proto jsou velice důležitou součástí systému. Pokaždé, když uživatel klikne myší, stiskne nějakou klávesu, se nějaká vyvolá. Pro pochopení této kapitoly je důležité znát organizaci komponent a jejich funkce, které se k událostem používají. Ty jsem rozdělil na dopředné a zpětné. Volání těchto funkcí označím jako předávání zpráv a nyní vysvětlím rozdíl mezi nimi.

- Dopředné – Princip těchto událostí ukážu na jednoduchém příkladu. Budu mít jednoduché okno s jedním kontejnerem. V něm bude tlačítko 1 a další kontejner, kde bude popisek a tlačítko 2. Vyvolám událost kliknutí myši a kurzor budu mít na tlačítku 2. Výsledkem bude posílání zprávy o této události shora dolů (od kořene - hlavního okna) až po tlačítko 2. Tuto situaci vyobrazuje obrázek 3, kde šipka zobrazuje postup předávání zprávy.
- Zpětné – Na stejném příkladu ukážu posílání zpráv při požadavku tlačítka 2 na fokus. V tomto případě bude situace zcela opačná. Zpráva se bude posílat od tlačítka 2 směrem vzhůru až ke kořeni, kterým je hlavní okno. Problém popisuje obrázek 4

To je rozdíl mezi těmito zprávami a nyní přejdu k popisu jednotlivých typů událostí, které mohou nastat.



Obrázek 3: Dopředné zprávy událostí

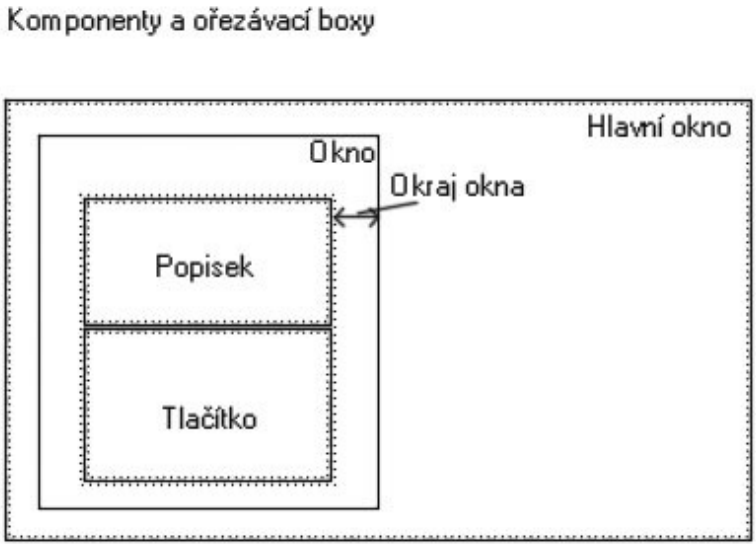


Obrázek 4: Zpětné zprávy událostí

Nejprve se budu zabývat dopřednými událostmi. Ty obsluhují vykreslování, ovládání, animace a změny pozic a velikostí. První tři vyvolává sama aplikace, tu poslední vyvolá reakce na změnu velikosti nebo pozice nějaké komponenty. Vyvolat ji může posun okna, posun seznamu pomocí rolovacího panelu nebo i změna okraje komponenty, která má následovníka.

Vykreslování

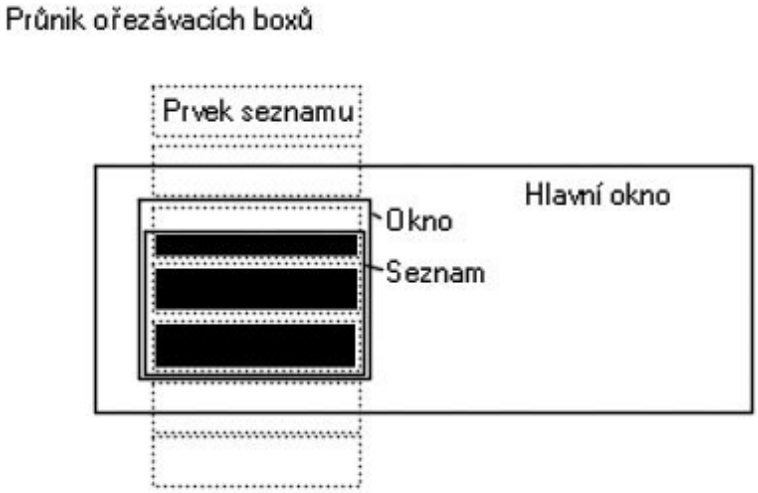
Tato zpráva komponenty informuje o tom, že se mají vykreslit a generuje ji pouze aplikace. Tato operace probíhá pouze ve dvourozměrném prostředí a je nutné dodržet hlavně pořadí. Samozřejmě se jedná o dopředný typ, takže každá komponenta, která má následovníka, vykreslí nejdříve sama sebe a teprve poté všechny komponenty, které obsluhuje. Toto pořadí je z toho důvodu, že pokud by se komponenta vykreslila až po svých následovnících, tak by je překreslila. Každá komponenta má uložený svůj ořezávací box, který přesně určuje, jaký prostor je určen následovníkům k vykreslování. Příklad ukazuje obrázek 5, kde plná čára představuje hranice komponenty a přerušovaná ořezávací box a okno má navíc nastavený větší okraj než je standardní hodnota. Informaci o ořezávacím boxu je nutné následovníkům předat při každém vykreslování, neboť se mění při každé změně velikosti i přesunu. Komponenty pak musí provést průnik předaného a vlastního, protože může nastat situace, že ořezávací box komponenty nebude ležet uvnitř předchůdcova. Pouze hlavní okno, jako kořen organizace, předává svůj ořezávací box, komponenty musí vždy provést nejdříve průnik a pak teprve informaci předat. Na obrázku 6 jsou vyobrazeny pouze ořezávací boxy a nachází se tam okno se seznamem, který má prvky (vyobrazené přerušovaně), jež neleží v jeho ořezávacím boxu. Plná část pak ukazuje jejich průnik, což symbolizuje části prvků seznamu, které se vykreslí. Takže komponenty mohou při vykreslování použít souřadnice, které nepatří do ořezávacího boxu, protože nakonec záleží na průniku obou boxů s následným využitím ořezávací funkce grafické knihovny OpenGL glScissor, která zpracuje jen plochu danou výsledkem průniku, což může být i prázdná množina. Komponenta, která není viditelná nevykresluje sebe ani své následovníky a ani jim nepředává svůj ořezávací box. Nakonec ještě nutno dodat, že při vykreslování se používá proměnná alfa multiplikátor, která násobí alfa hodnotu nastavované barvy a tím určuje průhlednost celé komponenty. Pokud je jeho hodnota 0, tak je výsledek vykreslování zcela průhledný.



Obrázek 5: Ořezávací boxy

Běh

Bez této zprávy by nefungovaly žádné animace. Generátorem této události je aplikace, která tento systém využívá. Zasílá se postupně všem komponentám prefixově dle stromu uspořádání; zde na pořadí nezáleží. Interval je 30ms a komponenty si pokaždé provedou krok animace či jiné operace. Těchto zpráv se využívá například při animaci zavírání okna, kdy se zvětšuje průhlednost, při rolování seznamu nebo i posunu rolovacího panelu pomocí tlačítek.



Obrázek 6: Průnik ořezávacích boxů

Vstupní kontrolér

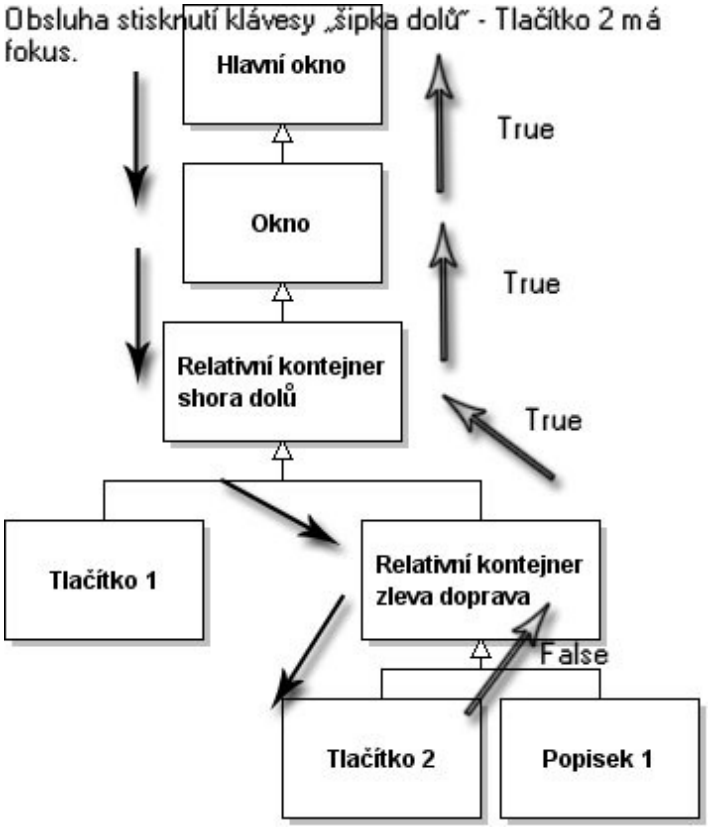
Zprávy, které sem patří jsou generovány aplikací a to událostmi, které vyvolá uživatel svojí interakcí. Jedná se o ovládání myši a klávesnice, ale i získání a ztrátu fokusu. Tento typ zpráv a jejich předávání je jednou z nejsložitějších věcí na tomto systému. V kapitole o kořenové komponentě jsem uvedl několik funkcí, které se vyvolávají po interakci klávesnicí nebo myši. Ale aby se tyto funkce vůbec zavolaly, tak při implementaci se musely řádně nastavit reakce na tyto zprávy. V rozhraní existují tyto typy zpráv (budu uvádět anglické názvy ve zkratce, které jsou využity i v implementaci):

- KEYBOARD_UP – uvolnění klávesy.
- KEYBOARD_DOWN – stisknutí klávesy.
- SPECIAL_KEY_UP – uvolnění speciální klávesy (kurzorové šipky, klávesy HOME apod.).
- SPECIAL_KEY_DOWN – stisknutí speciální klávesy .
- MOUSE_BUTTON – tlačítko myši.
- MOUSE_MOTION – pohyb myši se stisknutým tlačítkem.
- MOUSE_OVER – pohyb myší.
- LOST_FOCUS – ztráta fokusu.
- GOT_FOCUS – získání fokusu.

Ovládání klávesnice

Do této skupiny patří první čtyři zpráv a jejich ovládání je dosti podobné. Přidruženými daty této zprávy je číslo klávesy. Princip spočívá v předávání zpráv komponentě, která má fokus. Tímto se zpráva dostane vždy až ke komponentě, která nemá následovníka. Pokud zprávu nezpracuje, tak vrací hodnotu FALSE svému předchůdci a ten se ji pokusí zpracovat. Pokud ale zprávu zpracuje, tak je návrat funkce TRUE a zprávu už nikdo obsluhovat nebude. Obrázek 7 ilustruje takovou situaci: Předpokládejme, že tlačítko 2 má fokus a uživatel stiskne klávesu – kurzorová šipka dolů. Zpráva se bude šířit od hlavního okna až po tlačítko 2, protože má fokus, ale tuto zprávu neumí obsloužit, takže vrací hodnotu False a tím říká, že zpráva dosud nebyla zpracována. Relativní

kontejner už ale tuto informaci zpracovat dokáže (přehodí fokus na svého dalšího následovníka), takže poté vrací hodnotu True, což znamená, že zpráva byla obsloužena.



Obrázek 7: Obsluha klávesnice

Ovládání tlačítka myši

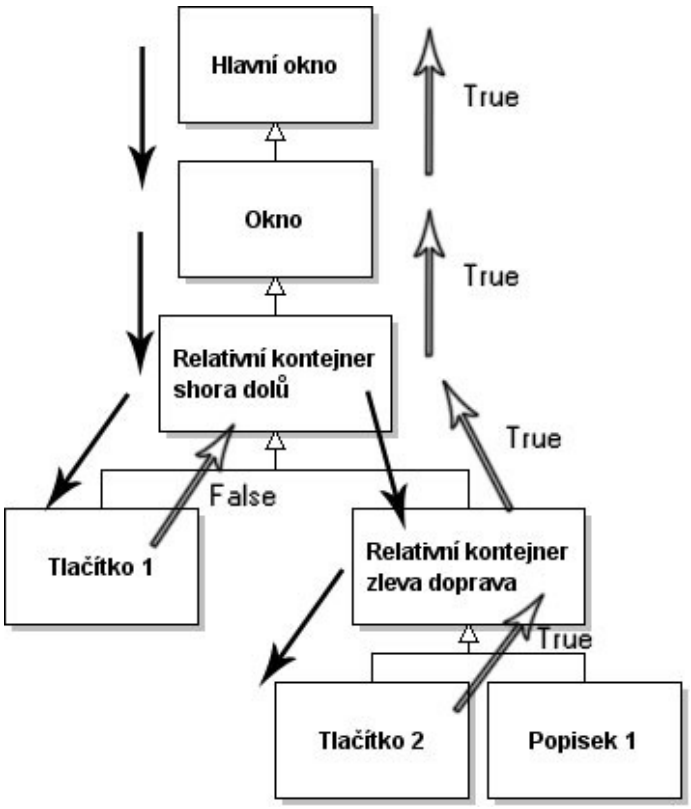
Zpráva MOUSE_BUTTON. Při ovládání této zprávy velice záleží na pořadí komponent. Na tuto zprávu je nutné nejdříve testovat ty komponenty, které jsou co nejvíce navrchu. Více následovníků může mít vlastně jen kontejner a hlavní okno. Oba prvky si uchovávají spojový seznam všech komponent, které obsluhují. Pokaždé když dojde ke změně fokusu, tak nově aktivní prvek se dostane na konec tohoto seznamu a je tím pádem na vrchu. Tímto je zajištěno, že pořadí v seznamu odpovídá pořadí zdola nahoru, což se využívá při vykreslování, a při obsluze událostí ovládání kteréhokoliv typu myši se využívá pořadí odzadu spojového seznamu (shora dolů).

Tato zpráva má důležitá přidružená data. Kromě souřadnic kurzoru myši a typu tlačítka, což v systému nerozlišuji, se tam vyskytuje zda bylo tlačítko stisknuto, nebo uvolněno. Tyto dva typy se liší obsluhou.

Stisk tlačítka

Při obsluze se prochází všechny komponenty shora dolů. Pokud má komponenta pouze jednoho následovníka, tak mu předá zprávu, pokud jich může mít více, prochází spojový seznam komponent odzadu, aby je procházel graficky shora dolů. Až se dojde ke komponentě, která nemá žádného následovníka, tak se provede test, jestli kurzor myši leží v její oblasti (ohraničená šířkou a výškou). Pokud tam leží, provede se operace stisku tlačítka a návratová hodnota se nastaví na TRUE a obsluha je ukončena. Pokud v oblasti neleží, vrací hodnotu FALSE a o obsluhu se snaží další komponenta v pořadí, pokud taková existuje, jinak zpracování končí. Na obrázku 8 je příklad stisku tlačítka 2. Předpokládejme, že tlačítko 1 má fokus a je tedy na konci spojového seznamu svého

předchůdce (kontejneru) a stejně tak tlačítko 2 u svého předchůdce. Zpráva o stisku tlačítka myši nejprve doputuje k tlačítku 1, kde ale neproběhne úspěšně test, zda kurzor leží v jeho oblasti. Proto vrátí svému předchůdci hodnotu FALSE a Relativní kontejner shora dolů předá zprávu svému dalšímu následovníkovi a ten ji předá tlačítku 2, kde test kurzoru proběhne úspěšně, takže vrací hodnotu TRUE a obsluha může být ukončena.



Obrázek 8: Stisk tlačítka myši

Uvolnění tlačítka

Zde je obsluha jednodušší. Tato zpráva se posílá všem komponentám a nezáleží na pořadí. To, že se posílá všem, je nutné kvůli komponentám, které jsou po stisku tlačítka ve stavu stisknutí. Kdyby se obsluha někde zastavila, tak by se nemohli vrátit do klidového stavu. Nyní by mohlo někoho napadnout, jak funguje kliknutí na tlačítko, když reakce se spouští právě při uvolnění. Aby se spustila reakce, tak musí být tlačítko ve stavu – stisknuté –, a to funguje pouze, pokud se u něj zastavila zpráva o stisku tlačítka myši (viz předchozí odstavec).

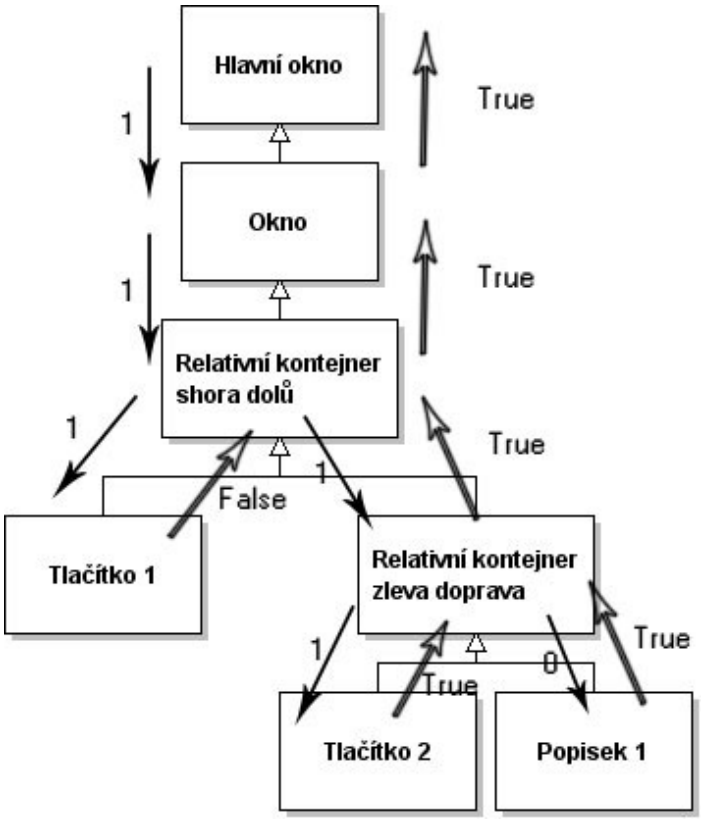
Pohyb myší se stisknutým tlačítkem

Zpráva MOUSE_MOTION. Obsluha je velice jednoduchá, protože se jedná pouze o informativní zprávu, která v sobě nese souřadnice kurzoru. Posílá se všem komponentám a na pořadí nezáleží a je pouze na nich, jak ji zpracují. Využívá se například při posunu okna, pohybu jezdcem u rolovacího panelu nebo u šoupátka (slider).

Pohyb myší

Zpráva MOUSE_OVER. Této události se využívá při zvýraznění komponenty, na kterou právě ukazuje kurzor myši. Musí se posílat všem komponentám, ale s jedním důležitým parametrem, který říká, zda už ji někdo neobsloužil a tím pádem už se nikdo jiný nemůže zvýraznit. Komponenty

se zde musí procházet stejně jako u obsluhy stisknutí tlačítka myši, tj. Od konce seznamu následovníků s parametrem hodnoty 1, což znamená, že ještě nikdo zprávu neobsloužil. Vždy, když se dojde ke komponentě, která nemá žádného následovníka, se provede test, zda na ni neukazuje kurzor myši. Pokud ano, tak vrací svému předchůdci hodnotu TRUE a ten nastaví parametr na hodnotu 0 a pokračuje s průchodem dalších následovníků, pokud existují. Nakonec vrátí hodnotu TRUE svému předchůdci, který opět svůj parametr nastaví na 0 a takto proces pokračuje dokud se neprojdou všechny komponenty. Pokud komponenta vrátí hodnotu FALSE, tak se parametr nemění. Obrázek 9 ilustruje takovou situaci. Kurzor myši ukazuje na tlačítko 2 a tlačítko 1 má fokus.



Obrázek 9: Pohyb myši

Směrem dolů se předává speciální parametr a vzhůru návratová hodnota. Tlačítko 1 má fokus, takže je první na řadě. Kurzor myši na něj neukazuje, a tak vrací hodnotu FALSE. Následuje tlačítko 2, na které kurzor ukazuje, a proto vrací TRUE. Kontejner nastaví parametr na 0 a posílá zprávu popisku, který pak vrací TRUE, protože přijal hodnotu parametru 0.

Ztráta fokusu

Zpráva LOST_FOCUS, oznamuje komponentě, že má nastavit hodnotu fokusu na FALSE, čímž se stane neaktivní. Pokud obsahuje následovníky, tak tuto zprávu pošle tomu, co má fokus. Může se vyvolat na libovolnou komponentu.

Zisk fokusu

Zpráva GOT_FOCUS. Může se vyvolat na libovolnou komponentu a pokud má následovníky, tak ji jednomu předá dál. Výběr se provádí dle parametru této zprávy, čímž je klávesa, kterou byla změna fokusu inicializována. Tato funkce se týká pouze kontejnerů a bude podrobně vysvětlena v kapitole o jejich implementaci. Komponenta, která tuto zprávu obdrží se nastaví jako aktivní (má fokus).

Parametry zpráv vstupního kontroléru

Zpráva		**Parametry**					
Vstupní kontrolér	Klávesnice	KEYBOARD*	x	y	Klávesa	
	Tlačítko myši	MOUSE_BUTTON	x	y	Tlačítko	Stav**
	Pohyb myší	MOUSE_OVER	x	y		
	Pohyb myší se stisknutým tlačítkem	MOUSE_MOTION	x	y		
	Ztráta fokusu	LOST_FOCUS				
	Zisk fokusu	GOT_FOCUS			Klávesa	

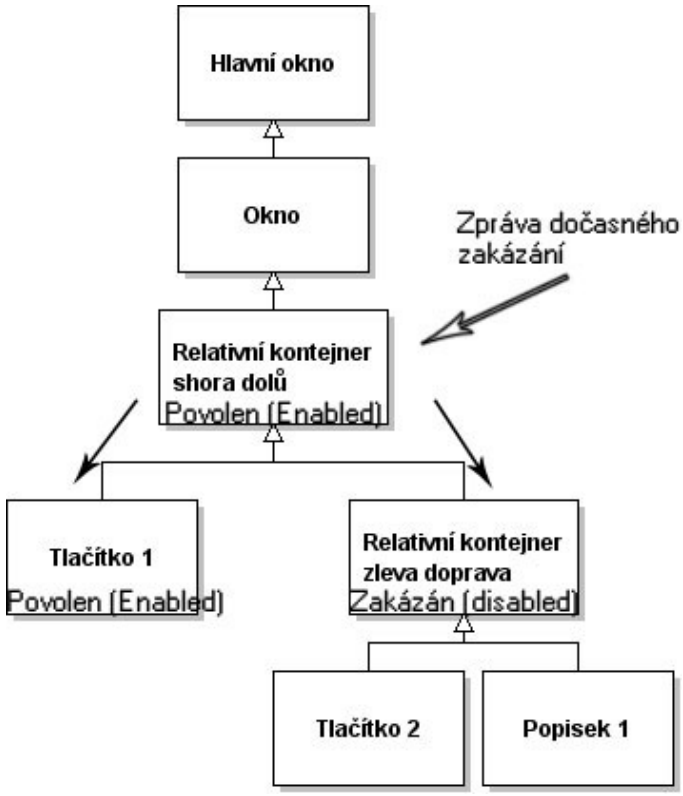
*Typy klávesnice mohou být: KEYBOARD_UP, KEYBOARD_DOWN, SPECIAL_UP, SPECIAL_DOWN.

**Stav může být UP (uvolněno) nebo DOWN (stisknuto).

Pozn. Souřadnice y je vzdálenost od spodního okraje

Nastavení dočasného zakázání komponenty

Komponenty, které se dostaly do tohoto stavu, mají nějakého předchůdce, který byl zakázán. Pokud se tato zpráva předá nějaké komponentě, tak ta ji pošle dále svým následovníkům, pokud není zakázaná. Ti pak provádějí stejnou operaci. Situaci ilustruje obrázek 10. Vyvolaná zpráva se posílá Relativnímu kontejneru shora dolů, který je povolen, takže dále předá tuto zprávu. Tlačítko 1 se nastaví, ale relativní kontejner zleva doprava je zakázán, takže u něj tato zpráva skončí.



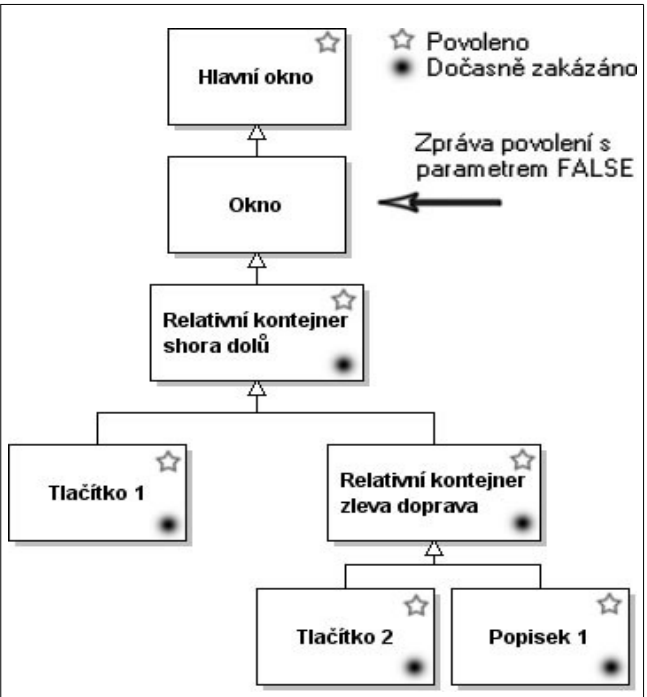
Obrázek 10: Dočasné zakázání komponenty

To, že je určitá komponenta dočasně zakázaná, znamená úplně to samé, jako kdyby byla zakázaná. Rozdíl bude pouze v jejich povolení, kdy všichni následovníci, kteří jsou dočasně zakázaní se povolí, což je přesně opačná situace než ukazuje obrázek 10.

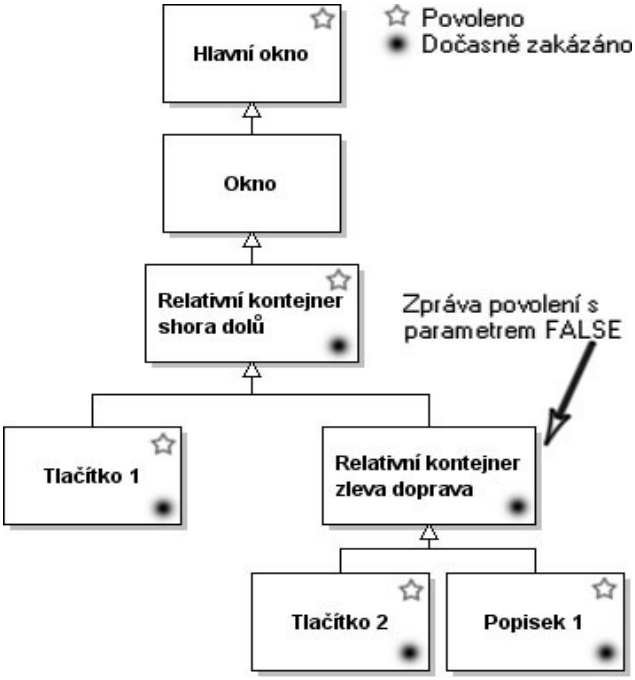
Nastavení povolení

Situace, kterou zobrazuje obrázek 10, vznikla tak, že okno bylo zakázáno, což znamená, že obdrželo tuto zprávu s parametrem FALSE. Tato zpráva totiž vyvolává u komponent, jenž mají následovníky, vznik zprávy o dočasném zakázání. Komponenta, která obdrží zprávu o povolení, nastaví hodnotu proměnné povolení dle parametru zprávy (TRUE nebo FALSE).

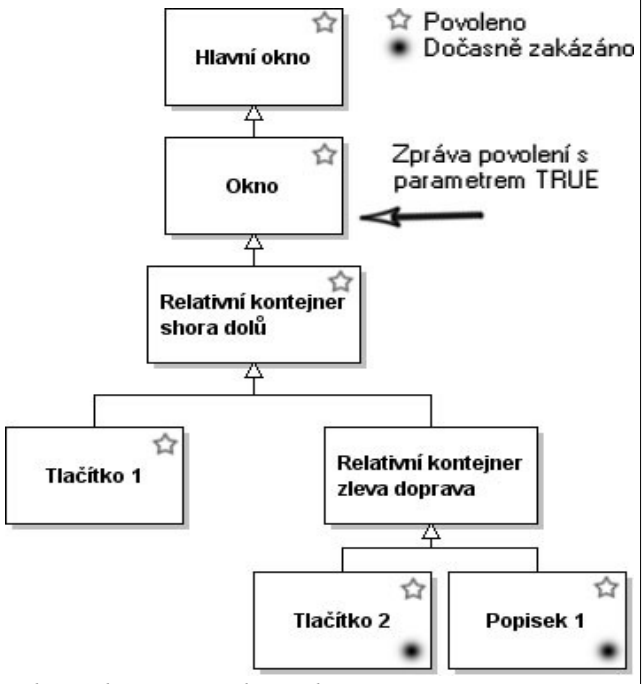
Obrázek 11: Výchozí stav - všichni povoleni



Obrázek 12: Zakázání okna



Obrázek 13: Zakázání kontejneru



Obrázek 14: Povolení okna

Pro objasnění ukážu jeden příklad, kde se toho využívá (obrázky 11-14).

Obrázek 11: Výchozí stav, kdy jsou všechny komponenty povoleny.

Obrázek 12: Nejprve se zakáže okno (zpráva povolení s parametrem FALSE). Tím se vyvolá zpráva o dočasném zakázání všech následovníků s parametrem TRUE – dočasně zakázat.

Obrázek 13: Zakáže se relativní kontejner zleva doprava a vyvolá se zpráva o dočasném zakázání všech následovníků, které už tento stav mají, tak se u nich nic nezmění.

Obrázek 14: Povolí se okno, což vyvolá zprávu o dočasném zakázání s parametrem TRUE (povolení), která se předá všem následovníkům. Zastaví se ale na relativním kontejneru zleva doprava, protože je zakázán.

Výsledkem je, že komponentám lze měnit povolení i když jsou dočasně zakázány, čímž se změní výsledek po opětovném povolení.

Nastavení alfa multiplikátoru

Tuto zprávu vyvolává pouze okno, které mění svou průhlednost díky otevírání/zavírání, kdy se provádí příslušná animace. Posílá se všem komponentám, které jsou následovníky tohoto okna.

Ztráta fokusu následovníka

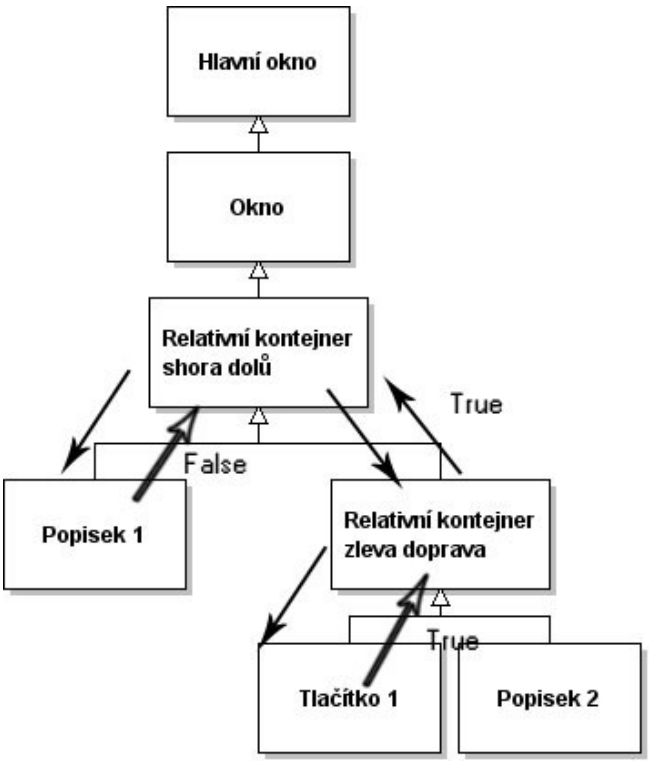
Tato zpráva je zpětná, takže se posílá zdola nahoru. Používá se v situacích, kdy by komponenta měla ztratit fokus kvůli jiné. Například leží ve stejném kontejneru. V tom případě se ptá svého předchůdce (kontejneru), jestli se ho má vzdát ve prospěch nějaké jiné. U většiny komponent se vyvolává tato zpráva na dalšího předchůdce tak dlouho, dokud někdo hodnotu nevrátí. Například okno vrací vždy FALSE, protože nemá žádného jiného následovníka (není komu předat fokus) a ani předchůdce (není už koho se ptát). U kontejneru je situace jiná. Pokud má kontejner více než jednu komponentu, tak vrací okamžitě TRUE, protože lze předat fokus jinému následníkovi. Pokud obsahuje pouze jednu nebo žádnou, tak se ptá předchůdce.

Dej mi fokus

Zprávu vyvolá komponenta, kterému byl násilně nastaven fokus. Důsledkem toho je, že musí postupně oslovovat všechny své předchůdce, aby si nastavili tuto komponentu jako aktivní. Zejména kontejnery ji musejí najít ve svém seznamu a přesunou na jeho konec.

„isFocusable“

Kontejnery umožňují přesunou fokus na jiného následovníka pomocí kurzorových šipek a klávesy TAB. V systému je umožněno nastavit aktivní pouze komponenty, které mohou fokus získat, a proto je tu tato zpráva. Komponenty obvykle vracejí TRUE nebo FALSE podle toho, jestli mohou být aktivní. Ale kontejnery samy o sobě aktivní být nemůžou, proto se musí ptát svých následovníků. Jakmile naleznou jeden, kterému lze fokus přiřadit, tak operace končí. Na obrázku 15 je příklad této operace. Na relativní kontejner shora dolů byl vznesena otázka, zda může získat fokus. Ten se musí zeptat svých následovníků, takže se táže nejdříve popisku 1, který nemůže být aktivní, takže vrací FALSE. Poté se ptá relativního kontejneru zleva doprava, který posílá dotazy dále svým následovníkům. Tlačítko 2 fokus mít může, takže vrací TRUE a operace končí.



Obrázek 15: Test fokusování

Závěr kapitoly

Z popisu výše zmíněných typů zpráv je vidět, jak důležité jsou pro chod celého systému. Umožňují vykreslovat a ovládat všechny komponenty, které pomocí nich mezi sebou komunikují. Tato kapitola byla pro pochopení chodu rozhraní pravděpodobně nejdůležitější.

8. Callback funkce

Pro definice reakcí na některé události (v tomto systému zatím jen kliknutí na tlačítko) se používají callback funkce. V kapitole o kořenové komponentě jsem zmínil proměnnou „Callback“. Je to objekt, který uchovává ukazatel na funkci a také na přidružená data. Při implementaci se předpokládá, že jedna funkce se bude používat pro několik komponent, proto se vždy definuje globální callback funkce a ta se pak automaticky předává nově vytvořeným komponentám.

Callback funkce musí mít tento předpis:

*void name(CBaseComponent *, void *)*

9. Implementace komponent

V této kapitole se budu věnovat podrobně vlastnostem komponent. Budu postupovat podle pořadí, které jsem uvedl na začátku textu o rozhraní GFG, takže podle skupin. Každá z nich vychází z kořenové komponenty, které jsem se už věnoval, takže nyní se budu zabývat pouze odlišnostmi a novými vlastnostmi. Pro upřesnění vždy uvedu obrázek pro inspiraci. Nejprve tedy začnu skupinou tlačítek, dále popisků, komponent a nakonec všech ostatních. Důležité bude též sledovat reakce na události a přizpůsobování změnám umístění a velikosti.

Skupina tlačítka

Sem patří:

- Základ tlačítek
- Tlačítko
- Přepínač
- Zatržítko
- Tlačítko volby

Základ tlačítek (Třída ButtonBase)

Dle obrázku 1 mají tlačítka společný základ v této komponentě. Zde se poprvé objevuje proměnná m_bPushed, která indikuje stav tlačítka (stisknuté uvolněné), a je důležitá pro chod všech komponent v této skupině. Základní princip vykreslování tlačítka závisí na této proměnné, kdy na každý stav připadá jedna textura (dohromady dvě). Ke kontrole událostí od myši a klávesnice používá standardní funkce definované v kořenové komponentě. Pokud uživatel stiskne tlačítko myši a kurzor je nad tlačítkem, tak se přepne do stavu „stisknuté“. Pokud se kurzor dostane mimo tlačítko, tak se nastaví stav „uvolněné“. Pokud je při uvolnění myši tlačítko ve stavu „stisknuté“, zavolá se callback funkce. Tato reakce nastane též, pokud má tlačítko fokus a uživatel stiskne klávesu ENTER. Třída nic nevykresluje, je jen předpisem pro komponenty, které z ní vycházejí.

Tlačítko (Třída Button)

Tato třída je již samotnou realizací základního typu tlačítek. Od společného základu se liší pouze vykreslováním. K tomu si při inicializaci nadefinuje barvy a načte textury. Samotné vykreslování se pak provádí dle několika pravidel. Liší se podle toho, v jakém je tlačítko stavu a jestli na něj ukazuje kurzor myši. V předchozím odstavci jsem zmínil dvě textury, které tlačítko má k dispozici. Právě zde se využívají. Pokud je stisknuté, tak se vykresluje jiná textura než v klidovém stavu. Pokud na tlačítko ukazuje kurzor, použijí se k vykreslení světlejší barva, což dá uživateli najevo, že na tlačítko ukazuje. Dále se zde využívá název komponenty (textový řetězec), který se vykresluje doprostřed a text lze samozřejmě měnit. Pokud je tlačítko aktivní (má fokus), tak se navíc zobrazí jednoduchý obdélník okolo této komponenty pro indikaci aktivity.

Přepínač (Třída SwitchedButton)

Toto tlačítko dědí vlastnosti od předchozí (Button). Využívá stejné způsoby vykreslování, pouze používá jiné textury. Od svého předka se liší pouze tím, že při puštění tlačítka myši, kdy se obyčejné tlačítko vrací do klidového stavu, zůstává ve stavu „stisknuté“ a vrací až po opětovném kliknutí.

Zatržítko (Třída CheckButton)

Tato třída je složitější než předchozí dva typy. Ve svém základě je to obyčejné tlačítko, kterému dodává navíc proměnnou m_bSwitched, která u této komponenty indikuje stav zaškrtnutí. Pokud je tlačítko v tomto stavu, tak zobrazuje obrázek, který o tomto stavu uživatele informuje. S obrázkem je prováděna jednoduchá animace rotace.

U této komponenty se poprvé používá jiná komponenta jako funkční část. Jedná se o onen obrázek, který bude popsán později. Tím, že ho zatržítko používá, mu musí také předávat některé zprávy, které samo přijalo. Jedná se o alfa multiplikátor, povolení či dočasné zakázání. Důležité ale je, měnit mu pozici a velikost v případech, kdy se změní tyto vlastnosti u vlastního tlačítka.

Při inicializaci se vytvoří nový objekt obrázku a nastaví se mu zdroj ze souboru. Obrázek má vždy šířku a výšku rovnou výšce ořezávacího boxu zatržítka (pokud je šířka menší než výška, tak se použije šířka). Dále se musí nastavit textury a barvy. Reakce na kliknutí je naprosto stejná jako u obyčejného tlačítka, ale při každé této události se přepne hodnota proměnné, která indikuje zaškrtnutí. Při vykreslování se pak zobrazuje obrázek s animací rotace pouze, pokud je tlačítko zatrhnuté. Drobný rozdíl oproti minulým třídám je v zarovnání zobrazovaného textu, které bylo pozměněno na levou stranu.

Tlačítko volby (Třída RadioButton)

Komponenta je odvozena od zatržítka a liší se pouze použitými texturami a jiným obrázkem, jinak jsou to naprosto stejné třídy.

Skupina popisky

Sem patří:

- Popisek
- Jednoduchý popisek
- Popisek bez ohraničení
- Rolovací popisek

Popisek (Třída Label)

Tato komponenta je společným základem pro všechny ostatní popisky. Je implementačně velice jednoduchá. Jedná se v podstatě jen o vykreslení několika prvků, kterými jsou:

- Text, který je zarovnaný doleva nahoru.
- Ohraničení celé komponenty.

Důležitou vlastností je zalamování textu. Pokaždé, když se nastaví nový text nebo dojde ke změně velikosti ořezávacího boxu popisku, se musí přepočítat rozložení textu.

Jednoduchý popisek (Třída SimpleLabel)

Třída je odvozena od popisku a liší se tím, že nevykresluje ohraničení, text nezalamuje a zarovnává ho vertikálně na střed.

Popisek bez ohraničení (Třída LabelNoOutline)

Toto je základní popisek bez vykreslení ohraničení. Jinak jsou naprosto identické.

Rolovací popisek (Třída ScrollLabel)

Jedná se o nejsložitější typ popisků. Používá komponentu rolovací panel k tomu, aby uživatel mohl posouvat text vertikálně, pokud je ho víc než se vejde do okna popisku. Tím se musí začít tento popisek starat i o události (obyčejný popisek některé ignoruje) a předávat je rolovacímu panelu, který si je obslouží. Jsou to:

- Vstupní kontrolér
- Nastavení alfa multiplikátoru
- Nastavení povolení

- Nastavení dočasného zakázání
- Běh

Další důležitou vlastností je předávání parametrů popisku rolovacímu panelu, aby mohl správně pracovat. Ten potřebuje znát pouze celkovou výšku textu a výšku zobrazovacího pole. Pak při každém vykreslování textu se vezme výsledná hodnota (0-1) rolovacího panelu a podle ní se posune text, takže uživatel může při posunu vidět i text, který nebyl vidět.

Ke správně funkčnosti je tedy zapotřebí, aby se při každé změně velikosti textu nebo celé komponenty nastavily parametry rolovacího panelu.

Pozn.: Více o hodnotách rolovacího panelu v odstavci o rolovacím panelu.

Skupina kontejnery

Sem patří:

- Základ kontejnerů
- Jednoduchý základ kontejnerů
- Fixní
- Relativní shora dolů
- Relativní zleva doprava
- Semi-relativní shora dolů
- Semi-relativní zleva doprava

Tato skupina komponent je důležitá pro uspořádání ve výsledném uživatelském rozhraní. Existuje několik typů, které se však liší minimálně, ale mají společný základ, který implementuje téměř všechny funkce a vlastnosti.

Základ kontejnerů (Třída CBaseContainer)

Jelikož ovládají více než jednu komponentu, tak potřebují spojový seznam, kde budou uloženy. Potřebují však více informací než jen odkaz na komponentu, takže jsem implementoval objekt, který nese tyto důležité informace:

- Odkaz na komponentu – díky této proměnné ji lze jednoduše ovládat a získávat informace
- Dodatečné informace – tato proměnná se používá například k určení priority
- Horní – předchozí přidaná komponenta
- Dolní – další přidaná komponenta nebo úplně první

Díky tomuto objektu pak ve výsledku existují vlastně dva spojové seznamy. Prvním je ten základní, který obsahuje tyto objekty a pořadí se mění při každé změně fokusu. Druhý si vytvářejí tyto objekty mezi sebou pomocí proměnných „horní“ a „dolní“. Vzniká při vkládání komponent do kontejneru stejně jako ten první, je cyklický, takže má spojený konec se začátkem a pořadí se nikdy nemění. Používá se při změnách fokusu pomocí kurzorových šipek, protože pokud se komponenty přidávají například shora dolů, tak toto pořadí zůstává zachováno a šipkou nahoru se dostanu na horní komponentu té aktuální.

Každý kontejner umožňuje i vykreslování rámečku s popisem, čímž vznikne tzv. Frame (rámec), díky kterému se ostatní komponenty oddělí od těch co leží v tomto kontejneru. Zda se má zobrazit určuje proměnná m_bDraw.

V tomto kořenovém kontejneru není definováno vkládání nových komponent ani jejich uspořádání a ani reakce na změny velikosti, předávání všech událostí obsaženým komponentám a to podle

principů uvedených v kapitole o událostech. Například u zprávy Dej mi fokus je třeba vyhledat komponentu v parametru a té dát fokus. Při změně pozice se posunou i všechny komponenty. Zpráva „isFocusable“ byla též vysvětlena v kapitole o událostech, kde byl i příklad.

Nicméně je zde implementována jedna funkce, která je důležitá pro změnu fokusu. Hledá první komponentu, která může získat fokus, podle parametru směru (dopředu dle dolní a dozadu dle horní komponenty v druhém seznamu. Aby mohla získat fokus, tak musí splnit následující podmínky:

- Fokusovatelnost
- Viditelnost
- Nesmí být dočasně zakázaná
- Musí být povolená

Takto se prochází celý druhý seznam, dokud se nenajde první komponenta, která splňuje podmínky nebo se nedojde k výchozí, což znamená, že žádný další fokusovatelný prvek neexistuje.

Dále je zde implementována individuální reakce na zprávy vstupního kontroléru. Pokud přijme zprávu o zisku fokusu, tak zkoumá i jeho parametr, čímž je klávesa, která ji vyvolala. Tím vyvolá funkci gotFocus(klavesa), která zde má pouze předpis a každý kontejner si ji může nadefinovat podle svého. Kontejner posílá ostatní zprávy vždy nejdříve komponentám, které obsahuje. Následně zkoumá jestli je obsloužily. Pokud nezpracují událost speciální klavesy nebo klávesy TAB, tak volá funkci changeFocusBySpecKey(klávesa), čímž se má změnit aktivní komponenta. Tato funkce tu též není implementována. Poslední je reakce na stisk tlačítka. Pokud zjistí, že některá komponenta ji obsloužila, tak ji nastaví jako aktivní (přesune na konec prvního seznamu a nastaví fokus).

Jednoduchý základ kontejnerů (Třída ContainerSimpleBase)

Tato třída definuje funkce pro změny fokusu, které měly předpis u předchozí třídy. Dějí všechny její vlastnosti a vytváří základ pro funkčnost jednoduchých kontejnerů. Jediné, co zde není definováno, je přidávání a uspořádání nových komponent, což je pro každý kontejner unikátní. Pro dokončení funkčnosti druhého spojového seznamu tu definuje dvě proměnné – začátek a konec.

Nyní lze definovat první ze dvou funkcí předepsaných základním kontejnerem:

changeFocusBySpecKey(int key)

Tato funkce předá fokus jiné komponentě dle klávesy v parametru. Tou klávesou může být kurzorová šipka vlevo, vpravo, nahoru a dolů a klávesa TAB. Klávesy vlevo a nahoru udávají směr nahoru ve druhém spojovém seznamu. Pokud je však aktivní komponenta na začátku, tak je třeba provést test, zda nemá kontejner ztratit fokus (nějaký předchůdce by ho chtěl předat jiné komponentě), proto se na to musí zeptat svého předchůdce. Pokud mu odpoví hodnotou TRUE, tak fokus ztrácí a funkce končí. V opačném případě se hledá první fokusovatelná komponenta dle funkce zmíněné u základu kontejnerů. Klávesy vpravo, dolů a TAB udávají směr dolů jinak je průběh funkce identický.

gotFocus(int type)

Funkce se zavolá při zisku fokusu. Je ale nutné rozhodnout, která komponenta fokus nakonec získá, protože to může být jen jedna. Parametr type je klávesa, kterou proces přehození fokusu vznikl. Pokud je to klávesa nahoru nebo vlevo, tak nastavím jako aktivní tu nejspodnější komponentu – poslední ve druhém seznamu, pokud klávesa vpravo, dolů nebo TAB, tak tu nahoře – první. Pokud je parametr roven nule, tak fokus získá komponenta, která ho měla naposledy.

Nyní následují vlastní implementace kontejnerů, které definují vkládání a uspořádání komponent.

Fixní kontejner (Třída ContainerFixed)

Do tohoto kontejneru se komponenty vkládají na přesně zadané souřadnice, kde bod [0,0] leží v levém dolním rohu. Při vkládání se vždy uvádějí informace o vzdálenosti od levého a spodního okraje. Pokud je vzdálenost od dolního okraje záporná, tak se bere za počáteční bod levý horní roh. Při změně velikosti kontejneru se pozice nemění, pouze při změně pozice se všechny komponenty též posunou.

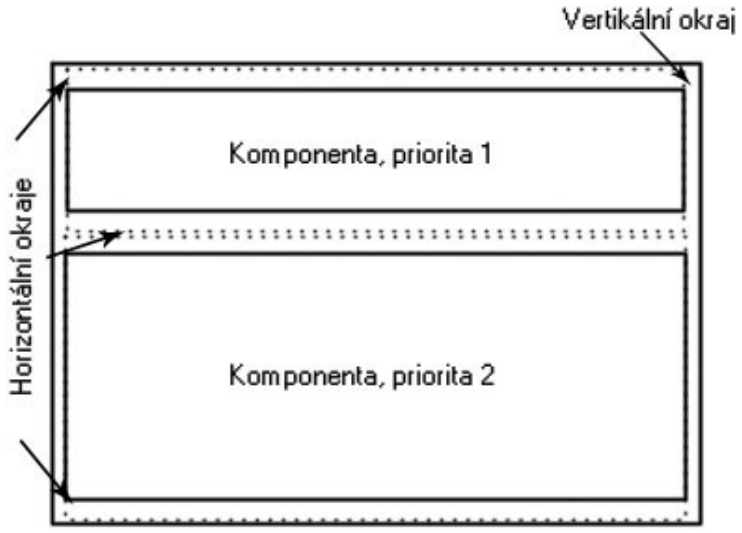
Relativní kontejner shora dolů (Třída ContainerRelativeUpToDown)

Při přidávání komponent do tohoto kontejneru se udává jako parametr priorita, jinak se pouze vkládají komponenty do prvního seznamu komponentu a vytváří se druhý. Priorita udává právo na vymezený prostor pro komponenty. Navíc lze nastavit, aby se komponenty přidávali zdola nahoru jednoduchým parametrem konstruktoru (tato vlastnost je společná pro všechny potomky). Jelikož jde o kontejner shora dolů, jedná se v tomto případě o výšku, která se bude dělit. Tato priorita se postupně sčítá. U tohoto kontejneru lze navíc nastavit prostor mezi komponentami hodnotou (0-1) a výsledná velikost okraje se zjistí vynásobením šířkou (vertikální okraj) nebo výškou (horizontální okraj). Tento okraj se pak dělí mezi komponenty.

Při každé změně velikosti kontejneru nebo přidaní nové komponenty se volá funkce „recalculate“, která uspořádá komponenty. To bude jediným rozdílem všech potomků této třídy a nyní ji popíši pro tento kontejner:

Komponenty zde mají všechny stejnou šířku, která je rovna šířce ořezávacího boxu bez vertikálního okraje. Výška se pak nastavuje dle poměru priority komponenty ku součtu priorit. Situaci ukazuje obrázek 16. Komponenta s prioritou 2 si bere 2/3 (3 je součet priorit) výšky ořezávacího boxu kontejneru mínus horizontální okraj. Horizontální okraj se pak dělí mezi dvě komponenty a dále ještě dvěma, protože se umísťuje nahoru i dolu. Dohromady tedy na čtyři části, které jsou na obrázku 16 vidět. Výpočet výsledných souřadnic komponent je pak pouhým sčítáním vzdáleností.

Pozn.: Priorita ovlivňuje pouze velikosti komponent, ale nikoliv okraje.



Ilustrace 16: Uspořádání v kontejneru

Relativní kontejner zleva doprava (Třída ContainerRelativeLeftToRight)

Tato třída je až na pár drobností stejná jako ta předchozí. Jediným rozdílem je to, že zde se komponenty uspořádávají zleva doprava, takže pro ilustraci si stačí představit obrázek 16 otočený o 90 stupňů s tím, že je třeba zaměnit horizontální a vertikální okraje. Komponenty tak budou mít vždy stejnou výšku a budou si mezi sebou dle priorit rozdělovat šířku ořezávacího boxu kontejneru.

Semi-relativní kontejner shora dolů (Třída ContainerSemiRelativeUpToDown)

V přidávání komponent zde není žádná změna oproti dvěma předchozím kontejnerům. Rozdíl je v uspořádání, protože zde se prostor nedělí, ale každá komponenta dostane takový prostor jaký chce, jen šířka zůstává stejná jako šířka ořezávacího boxu kontejneru bez vertikálního okraje. Tím však může nastat situace, že komponenta nemusí být vidět, jelikož se už nevejde do ořezávacího boxu kontejneru. K zobrazení je pak potřeba kontejner roztáhnout například zvětšením okna, ve kterém je umístěn. Principy horizontálních a vertikálních okrajů zůstávají stejné jako u relativních kontejnerů. Obrázek XXX ilustruje tento typ kontejneru.

Semi-relativní kon. zleva doprava (Tř. ContainerSemiRelativeLeftToRight)

Od svého předka se liší pouze směrem přidávání komponent, jinak principy zůstávají naprosto stejné. Výška komponent zůstává stejná jako výška ořezávacího boxu kontejneru bez horizontálního okraje.

Skupina ostatní

Sem patří komponenty, jejichž společným předkem je pouze kořenová komponenta. Často ke své funkci využívají ostatní komponenty, aby se jejich implementace zjednodušila a systematizovala. Skupina se skládá z těchto prvků:

- Okno
- Pole tlačítek volby
- Dělící čára
- Šoupátko (slider)
- Textové pole
- Indikátor stavu operace (progress bar)
- Záložky
- Rolovací panel
- Seznam
- Dialog (Message box)

Okno (Třída SubWindow)

Tato třída implementuje základ každého uživatelského rozhraní, protože se do ní obvykle vkládají ostatní komponenty. V tomto případě sem lze vložit pouze jednu, ale pokud to bude například kontejner, tak dosáhnu stejného efektu jako kdybych umožnil vkládat více komponent. Okno vždy přizpůsobuje její velikost svému ořezávacímu boxu, takže neumožňuje nějaké možnosti uložení.

V horní části této komponenty je ovládací panel, který obsahuje dvě tlačítka, která mají tyto funkce: zabalení / rozbalení okna, zavření okna. Zabalení zde znamená, že se okno jednoduchou animací zmenší tak, že zůstane pouze ovládací panel. Po opětovném kliknutí na toto tlačítko se uvede do

původní podoby. Při zavření okna se mění jeho průhlednost, až není vůbec vidět. Při otevření okna je animace opačná.

Tím, že okno obsahuje horní ovládací panel, se mění i jeho nastavení okraje (čili ořezávacího boxu). Ten je v horní části větší o výšku panelu než ostatní tři. Pokud uživatel najede myší na některý z okrajů, stiskne tlačítko myši a táhne, tak se bude měnit velikost okna. Horní okraj je u této funkce opět speciální, protože jeho velikost je vždy 1/3 výšky panelu. Samozřejmostí je možnost táhnout dva okraje, pokud stisknu tlačítko myši a kurzor je v některém z rohů. Při každé změně velikosti se mění ořezávací box a tím pádem i velikost ovládané komponenty. Pokud uživatel stiskne tlačítko myši v oblasti horního panelu, kde není tlačítko a ani okraj pro měnění velikosti, tak pomocí táhnutí lze okno posouvat. Tím by se však mohl ovládací panel octnout mimo hlavní okno, takže zde existuje kontrolní funkce, která ho v tomto případě vrátí zpět do viditelné oblasti.

Tím, že okno obsahuje dvě tlačítka a může mít i jednu komponentu, tak jim musí předávat přijaté i vygenerované zprávy. Zejména pak události vstupního kontroléru, bez nichž by nebylo možné komponenty ovládat a samozřejmě vykreslování. Dále předává zprávu „běh“, alfa multiplikátor, který samo generuje, nastavení povolení, dočasné zakázání.

Při vykreslování se používají dvě textury. Jedna pro ovládací panel, který se pak vykresluje modrou barvou a pokud má okno fokus, tak se vykreslí světlejším odstínem. Druhá je pro zbylou oblast a je nekontrastní, protože zde budou ležet ostatní komponenty. Po vykreslení panelu o této oblasti teprve dojde řada na zobrazení obou tlačítek a ovládané komponenty, pokud ji okno má, a popisek okna, který leží uprostřed horního panelu.

Pole tlačítek volby (Třída Option)

Tato třída pracuje s komponentami RadioButton. Vytváří seznam těchto prvků a umožňuje uživateli vybrat právě jednu možnost. Používá se například při výběru rozlišení nebo pohlaví. Ke své funkci používá jeden kontejner, který kontroluje a nastavuje mu vždy pozici, šířku a výšku dle ořezávacího boxu. kontejner si nastaví programátor podle svého uvážení. Pokud ale bude chtít přiřadit neprázdný, tak ho nejprve vyprázdní.

Poté už lze přidávat pouze RadioButton. Komponenta si udržuje odkaz na vybranou volbu a při inicializaci je to vždy první přidané tlačítko.

Všechny přijaté události předává okamžitě kontejneru, protože sama komponenta nemá z tohoto hlediska téměř žádný význam. Pouze reaguje na vstupní kontrolér (přesněji uvolnění tlačítka myši), protože musí obsluhovat výběr ze seznamu tlačítek. Takže prochází seznam komponent v kontejneru a jakmile nějaké tlačítko obslouží tuto událost a není již vybrané, tak ho nastaví jako označené, zároveň získá fokus a RadioButton, který byl předtím označený, se samozřejmě musí odznačit. V opačném případě se nic nezmění.

Dělící čára (Třída ComponentLine)

Je to horizontální čára, která se vykresluje přes celou šířku komponenty a vertikální pozice se nastaví jako střed zadané výšky, která se poté vynuluje.

Šoupátko (Třída Slider)

Tato komponenta umožňuje měnit nějakou hodnotu v zadaného intervalu pomocí jezdce, který je implementován pomocí tlačítka. Dále ještě využívá dělící čáru jako dráhu. Tlačítku definuje nový vzhled, jeho výška je vždy poloviční než celá komponenta a pohybuje se ve ¾ výšky komponenty, kde je vykresluje i dělící čára. Tyto pozice a velikosti se musí při každé změně velikosti nebo pozice

aktualizovat.

V dolní polovině komponenty se zobrazuje text, který znázorňuje aktuální hodnotu. Při nastavení intervalu se zadává minimální hodnota a rozsah. Pozice jezdce pak udává aktuální hodnotu, u které lze nastavit počet zaokrouhlovacích míst.

Události se musí samozřejmě předávat i dělící čáře a jezdci (tlačítku). Nejdůležitější je zde vstupní kontrolér, kde je potřeba ovládat pohyb jezdce. To lze provést dvěma způsoby:

- Stisknutí tlačítka myši nad jezdcem a pohybem myši měnit hodnotu
- Kliknout myší do prostoru dělící čáry a tím nastavit okamžitě novou hodnotu

Jelikož je jezdec tlačítko, tak je jednoduché zjistit, zda nad ním uživatel stiskl tlačítko myši. Pokud ano, tak se přepne komponenta do stavu pohybu jezdce a testuje pouze nové souřadnice myši, podle níž změní pozici jezdce, která pak přesně udává výslednou hodnotu (samozřejmě jezdec se smí pohybovat jen ve vymezeném prostoru komponenty). Při druhé možnosti ovládání nastává problém s tím, že čára nepřijímá zprávy vstupního kontroléru, takže je třeba provést detekci manuálně. Čára má výšku vždy nula, takže jsem si vytvořil určitý interval, který výšku simuluje. Jednoduše pak testuji, zda kurzor myši při kliknutí leží v obdélníku daném šířkou dělící čáry a výškou dle mého intervalu. Problém ilustruje obrázek 17.

Ilustrace 17: Testovací oblast okolo dělící čáry

Pokud uživatel do tohoto intervalu klikne, tak dle souřadnice X se vypočítá nová pozice jezdce a samozřejmě nová hodnota.

Textové pole (Třída TextBox)

Tato komponenta se používá pro textovou komunikaci s uživatelem. Implementuje kurzor, který představuje místo v textu a může se pomocí kurzorových šípek nebo myši přesunovat na jiné místo, kde lze vkládat další písmena a nebo je mazat. Třída přímo spolupracuje s objektem textu, protože využívá vždy jen jednu řádku a text musí po každé změně (přidání, odebrání) překompilovat. Popis textové knihovny není předmětem textu tohoto rozhraní.

Pro účely manipulace s textem definuje dvě proměnné První určuje pozici kurzoru v textu dle počtu písmen a druhá dle vzdálenosti od levého okraje komponenty. Pomocí nich se pak text upravuje a kurzor vykresluje. Pro přehlednost je zde implementován jednoduchá animace posunu textu, aby byl kurzor vždy vidět. Algoritmus se udržuje kurzor v horizontálním středu komponenty, pokud není příliš blízko u kraje. Animace pak celek plynule posunuje do tohoto cíle a vytváří tak příjemný pohled. Při vykreslování se nejprve zobrazuje rámeček textového pole, poté se provede posun dle animace a namaluje se text s kurzorem.

U této komponenty je nejdůležitějším prvkem vstupní kontrolér. Reaguje se na vstup z klávesnice i z myši a nyní je zde popíši:

- Klávesa BACKSPACE – pokud textové pole obsahuje text a kurzor není na začátku, tak se smaže první písmeno před ním. Na to se musí zmenšit celková šířka textu, který se musí překompilovat, a pozice kurzoru. Následně se provede aktualizace cílového posunu textu.

- Klávesa DELETE – provádí téměř stejnou operaci jako klávesa BACKSPACE, ale maže první znak za kurzorem, pokud takový existuje.
- Kurzorové šipky (vlevo, vpravo) – posunují kurzor vlevo resp. Vpravo. Pokaždé je potřeba aktualizovat jeho pozici a cílový posun textu.
- Stisk tlačítka myši – nejdříve se testuje, zda je kurzor myši nad textovým polem. Pokud ano, tak je třeba projít postupně všechna písmena a zjistit, na které je ukazováno. Jestli takové existuje, tak se podle směru od středu tohoto písmena rozhodne, zda se kurzor umístí před nebo za písmeno. Vlevo od středu znamená umístění před a vpravo od středu znamená umístění za písmeno. Poté už se jen aktualizuje cílový posun textu.
- Pohyb myší – pokud je kurzor myši nad komponentou, tak ze změní grafická podoba kurzoru myši na standardní „I“.

Indikátor stavu operace (Třída ProgressBar)

Tato komponenta se používá u operací, které trvají delší dobu, pro zobrazení průběhu a indikaci činnosti. Ve své podstatě pracuje pouze s jedinou hodnotou v intervalu 0-1, která určuje podíl splnění úkolu. Hodnotu pak násobí 100 a vykresluje jako procentuální zápis. Pro grafické znázornění se používá obdélník, který mění svou šířku v závislosti na referenční hodnotě. K tomu jsou k dispozici čtyři textury, které celou komponentu zobrazují. Jedna se používá pro vykreslení obdélníku (indikátoru) a ostatní pro ohraničení komponenty (pravý okraj, střed, levý okraj). Použití těchto třech textur místo jediné bylo potřeba kvůli eliminaci nežádaných deformací při změně velikosti. Využívá se toho, že se roztahuje pouze střed, jehož textura je imunní vůči těmto jevům. (eliminuje se deformace pouze horizontálně, při změně šířky, což je ale u indikátoru, který je také horizontální, důležité).

Tato komponenta není fokusovatelná a nezpracovává žádné události vstupního kontroléru. Při každé změně velikosti se musí přepočítat pozice a velikost indikujícího obdélníku.

Záložky (Třída Tabs)

Tato komponenta umožňuje ukládat komponenty do panelů (ukládají se do spojového seznamu), mezi kterými lze přepínat pomocí tlačítek (přepínačů), které dohromady vytvářejí horní panel záložek. Panel, který je právě ovládán a je vykreslován, je označován jako aktivní, dostane i fokus, a nemůže jich být současně více než jeden. Díky záložkám lze rozdělit okno na více součástí, které se nevejdou do jediného okna, a mezi nimi přepínat.

Pro vytváření nových záložek je nejprve potřeba vytvořit panely, kde každý dostane unikátní identifikační číslo. Poté jim lze pomocí této hodnoty přiřadit právě jednu komponentu, kterou budou ovládat a vykreslovat. Po inicializaci je aktivní ten panel, který byl přidán jako první. Jejich tlačítka (přepínače, které tvoří panel záložek) se přidávají postupně zleva a jejich šířka je úměrná šířce textu, který jim byl nastaven.

Pro funkčnost je důležitá právě informace, že aktivní může být jen jeden panel. Jemu pak musí tato komponenta posílat všechny události (povolení, dočasné zakázání, alfa multiplikátor atd.). Panelům přiřazená komponenta má vždy stejnou velikost a pozici jako ořezávací box této třídy a je nutné ji aktualizovat při každé změně velikosti nebo pozice. Při vykreslovaní se vždy zobrazí nejprve vlastní objekt a teprve poté aktivní panel.

Už jsem uvedl, že se zde vytváří panel záložek. Je umístěn v levém horním rohu a při přidávání nových panelů se rozšiřuje směrem doprava. Skládá se z přepínacích tlačítek, které je nutné též ovládat. Svoji velikost sice nemění, ale je třeba aktualizovat jejich pozici při každé změně pozice

vlastní komponenty. Těmto tlačítkům byla též přiřazena nestandardní textura přímo pro třídu záložky.

Pro ovládání tlačítek a přepínání mezi panely je potřeba zpracovávat události vstupního kontroléru. Při události pohybu myši ji musím předávat panelu záložek a poté aktivnímu panelu. Při uvolnění myši je třeba zkontrolovat, zda nebylo na nějaké tlačítko kliknuto (pokud ano, tak se jeho panel nastaví jako aktivní) a předat zprávu také aktivnímu panelu.

Rolovací panel (Třída ScrollBar)

Rolovací panel je využíván ostatními komponentami pro možnost rolování pohledu, a tak možnosti zobrazeni více množství dat, než by se vešlo do základní vykreslovací oblasti. Panel je pouze vertikální. Horizontální není problém vytvořit, ale pro mé účely nebyl potřeba. Aby panel správně fungoval, tak potřebuje dvě tlačítka (jedno nahoře pro pohyb vzhůru a jedno dole pro pohyb dolů) a nějakého jezdce, kterým půjde hýbat pomocí myši, a tak měnit výslednou hodnotu. Ta může být z intervalu (0-1) a udává velikost posunutí vzhledem k poměru velikost zobrazovací oblasti a velikosti zobrazovaných dat.

V implementaci se používají tři obyčejná tlačítka. Dvě z nich jsou určené pro krokové posouvání jezdce a výsledné hodnoty. Jeden je umístěn v nahoře a druhý dole. Mezi nimi se může pohybovat třetí tlačítko, které představuje jezdce, kterým lze pomocí myši pohybovat a také měnit výsledek. Prostor mezi horním a dolním tlačítkem je dráha po které se prostřední může pohybovat. Jeho velikost se mění podle aktuální výsledné hodnoty tak, že jí vynásobí výškou dráhy.

Vzhledem k tomu, že velikost jezdce je závislá na velikosti celé komponenty, tak se musí aktualizovat při každé změně velikosti, stejně tak jako ostatní tlačítka. Při posunu nevznikají žádné komplikace, jen je třeba všechny posunout. Vykreslování je také jednoduché. Zobrazují se pouze tlačítka, z nichž to spodní se nejdříve otočí o 180 stupňů, aby textura šipky směřovala správným směrem. Tím není třeba mít dvě textury, které by se lišily pouze v otočení.

Nejdůležitější a nejsložitější je zde ovládání. Je velice podobné tomu u „Šoupátka“ (Slider), ale navíc obsahuje dvě ovládací tlačítka. Události pohybu myši se jen předávají všem třem. Při stisknutí tlačítka myši se testuje, zda je horní resp. dolní stisknuté, protože pokud ano, tak se provede příslušný posun jezdce a výsledné hodnoty. Zde navíc zafunguje zpráva „Běh“, kde pokud je jedno z tlačítek stisknuté, tak pokračuje v posunu jezdce a hodnoty. Při stisknutí jezdce tlačítkem myši se komponenta přepne do stavu posouvání jezdce a dále testuje zprávu pohybu myši se stisknutým tlačítkem. Změna souřadnice kurzoru za jednu kontrolu pohne jezdcem, který má ale dané určité hranice, přes které se nesmí dostat. Po změně pozice se musí vypočítat nová výsledná hodnota.

Při používání této komponenty je potřeba pouze správně nastavovat dva vstupní parametry což jsou:

- Výška oblasti, ve které se data zobrazují
- Výška dat, která se mají vykreslit

Poté už stačí pouze sledovat výslednou hodnotu a pomocí ní měnit zobrazovací oblast.

Seznam (Třída ListBox)

Třída umožňuje skládat různé komponenty pod sebe a tím vytvářet seznam. Jednotlivé položky jsou panely, z nichž vždy pouze jeden může být aktivní. Seznam se může rozrůstat až bude jeho výška větší než výška vlastní komponenty. Proto se zde využívá rolovacího panelu, takže jde zobrazovací oblast rolovat. Programátor, který chce seznam využívat musí pouze přidávat komponenty, protože o ostatní věci se postará sám seznam.

Každá položka seznamu má vždy stejnou šířku jako jeho ořezávací box. Výška je individuální,

takže komponenta není homogenní, ale i proto, že panely v seznamu mohou obsahovat každý různou komponentu. Panel, který je aktivní, získává od seznamu události a to i od vstupního kontroléru. Lze tak vytvářet složité uživatelské rozhraní.

Na této komponentě je důležitá synchronizace s rolovacím panelem. Kdykoliv se změní jeho výsledná hodnota, tak je třeba na to reagovat aktualizací posunutí všech panelů. Jenže mezi aktivními panely lze přecházet pomocí tlačítka myši a kurzorových šipek nahoru a dolu a pak se mění posunutí, takže při každé této události se musí aktualizovat výsledná hodnota rolovacího panelu. Seznam se vždy snaží dostat aktivní panel do středu jeho výšky.

Panel se může stát aktivním, pokud na něj uživatel klikne myší a nebo pokud se na něj postupně dostane pomocí kurzorových šipek z právě aktivního panelu. Položka seznamu, která se stane aktivní zesvětlí a naopak ta, která aktivitu ztratila ztmavne. Proces probíhá postupně a vytváří tak příjemný dojem.

Jak už jsem napsal, aktivní panel se nachází vždy co nejblíže středu seznamu, takže vždy když dojde k jeho změně, přidá se nový nebo se změní velikost celého seznamu, se musí přepočítat cílový posun. Cílový je proto, že zde existuje ještě animovaný posun, který postupně zvyšuje/snižuje svoji hodnotu, aby dosáhl cílového posunu. Čím větší je jejich vzdálenost, tím větší posun se v jednom kroku provede. Celkově to vytváří další animaci, protože kdyby se posun prováděl skokově, tak by to bylo nepříjemné pro oči.

Při vykreslování se začíná ohraničením seznamu, poté rolovacím seznamem a nakonec jednotlivými položkami. Pouze aktivní se ale zobrazí světleji a navíc označena modrým obdélníkem, který je však dostatečně průhledný, aby nerozptyloval.

Ostatní události se po vlastní obsluze předají dále. Mám na mysli nastavení povolení, dočasného zakázání, běhu.

Dialog (Třída MessageBox)

Tato třída není složitým objektem, ale pouze ovládá jedinou komponentu. Je vlastně podobná oknu, ale nemá žádné ovládací prvky. Při vykreslování zobrazuje poloprůhledné pozadí a pak už jen ovládanou komponentu. Samo nezpracovává žádné události vstupního kontroléru, tyto zprávy předává komponentě, stejně jako ostatní zprávy.

Uspořádání komponent v dialogu si musí programátor udělat sám a nebo využít některou z předdefinovaných funkcí, které vytvářejí standardní dialogy. To už jsem zmínil v kapitole o komponentách. Funkce je zatím pouze jedna. Vytváří uspořádání s jedním obrázkem, jedním popiskem a dvěma tlačítky. Všechny texty lze nastavit a obrázek lze vybrat dle konstanty. Předdefinovány jsou čtyři možné obrázky dle významu dialogu:

- informativní
- otázka
- upozornění
- chyba

U této komponenty je ale důležitá komunikace s hlavním oknem, což popíši v následujícím odstavci.

Hlavní okno (Třída CMainWindow)

Hlavní okno nepatří přímo mezi komponenty, ale operuje s nimi. Nejvíce se podobá fixnímu kontejneru. Neumožňuje ale měnit fokus klávesnicí. Tento objekt je tu jako prostředník mezi aplikací a uživatelským rozhraním. Umožňuje vkládat komponenty a rozesílat jim události. Lze

měnit fokus pomocí kliknutí myší. Při použití tohoto systému se provede vytvoření nové třídy odvozené od hlavního okna a definuje se v ní uspořádání komponent.

Hlavní okno je také důležité pro práci s komponentou dialog, protože obsahuje funkce, které dokáží dialog spustit a odstranit. Tím, že se zde spustí dialog, se zakáže interakce s ostatními komponentami lze pracovat pouze s dialogem, dokud ho někdo neodstraní. Samo okno se vůbec nevykresluje, zobrazují se pouze ovládané komponenty.

10. Shrnutí a závěr

V předchozích odstavcích jsem popsal všechny komponenty, které jsou v rozhraní implementovány. Uvedl jsem ale pouze základní principy a konkrétní hodnoty a výpočty jsem zde nezmiňoval. Pro účely tohoto textu je tento způsob dostačující.

V devíti kapitolách jsem popisoval uživatelské rozhraní, které je vhodné pro použití ve hrách, které k vykreslování používají OpenGL. Ovšem přepracovat vykreslovací funkce pro jinou grafickou knihovnu by nebyl příliš problém. Organizaci a uspořádání komponent jsem se snažil vysvětlit srozumitelně a dostatečně podrobně. Pro více informací se lze podívat přímo do zdrojových kódů systému. Rozhraní využívá i některé další součásti (texty, fonty, textury), které jsem v textu uváděl a jejich principy jsou vysvětleny jinde.

Díky tomuto rozhraní jsem byl po jeho dokončení schopen vytvořit hlavní menu ke hře ChronoPhobia a dále pak okno, které se zobrazuje při načítání levelu a rozhraní indikátorů stavu (HUD), které si může hráč přizpůsobit jak bude chtít.

Některé vlastnosti tohoto rozhraní se určitě ještě změní, ale základní principy, které jsem v tomto textu uvedl se pravděpodobně nezmění.

11. Zdroje a literatura

[1] GUICHAN – jednoduchá knihovna pro tvorbu uživatelského rozhraní pro hry

http://guichan.sourceforge.net/

[2] GTK+ - oblíbené platformě nezávislé GUI

http://www.gtk.org/

[3] QT – další platformě nezávislé GUI

http://trolltech.com/products

[4] Seriál o OpenGL na www.root.cz

http://www.root.cz/clanky/graficka-knihovna-opengl-1/