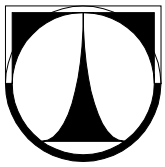
TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií



DIPLOMOVÁ PRÁCE

Liberec 2010 **Martin Podlipný**

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N 2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Využití JSR-184 v 3D grafických aplikacích mobilního telefonu

3D graphic cell-phone applications by JSR-184 API

Diplomová práce

Autor: **Bc. Martin Podlipný**
Vedoucí práce: Ing. Roman Špánek, Ph.D.
Konzultant: Ing. Pavel Tyl

V Liberci 21. 5. 2010

Fakulta mechatroniky, informatiky a mezioborových studií

Zadání diplomové práce

Jméno a příjmení studenta (os. číslo)	**Bc. Martin Podlipný (M08000224)**
Zkratka pracoviště	MTI
Datum zadání DP	16. 10. 2009
Plánované datum odevzdání	21. 5. 2010
Rozsah grafických prací	Dle potřeby dokumentace
Rozsah průvodní zprávy	cca 40–50 stran
Název DP (česky)	**Využití JSR-184 v 3D grafických aplikacích mobilního telefonu**
Název DP (anglicky)	**3D graphic cell-phone applications by JSR-184 API**

Zásady pro vypracování diplomové práce:

1. Seznamte se s problematikou modelování, zpracování a zobrazování 3D scén.
2. Nastudujte specifikace JME a JSR-184.
3. Obeznamte se s načítáním obecných 3D scén a jejich dynamických transformací na základě impulsů od uživatele.
4. Na základě předchozích bodů naprogramujte 3D aplikaci, která bude využita pro grant ESF na TUL, který si klade za cíl rozvoj zájmu o technické obory žáků základních a středních škol.

Seznam doporučené literatury:

[1] Internetové stránky Sun: http://java.sun.com/

[2] Specifikace Sun: http://developers.sun.com/techtopics/mobility/apis/articles/3dgraphics/

[3] Qusay H. Mahmoud: Naučte se Java 2 Micro Edition, O'Reilly, 2003

Vedoucí DP	Ing. Roman Špánek, Ph.D.
Konzultant DP	Ing. Pavel Tyl

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že Technická univerzita v Liberci (TUL) má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum:

Podpis:

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Romanu Špánkovi, Ph.D. za podněty, připomínky, cenné rady a vedení v průběhu tvorby této diplomové práce.

Zvláštní poděkování patří Ing. Miloši Hernychovi, koordinátorovi projektu *Starttech*, a Ing. Jaroslavu Semerádovi, zástupci koordinátora projektu za SPŠSE a VOŠ.

Abstrakt

Tato diplomová práce se zabývá problematikou 3D grafiky v aplikacích pro mobilní zařízení. Pro vytváření 3D aplikací je dnes dostupné rozhraní Mobile 3D Graphics API (M3G, JSR-184), které je rozšířením platformy Java Micro Edition. V teoretické části práce je proveden rozbor tohoto rozhraní s praktickými příklady a s popisem obecných principů zobrazování 3D grafiky.

V praktické části práce se využívá popsaných poznatků k řešení návrhu konkrétní aplikace *3D elektrické obvody*. Při použití 3D grafiky v mobilních telefonech je řešeno modelování prostorových objektů, jejich načtení, zobrazení a transformace. V aplikaci jsou zohledněny cíle projektu *Starttech* řešeného na TU v Liberci, který má za cíl podporu a rozvoj technických oborů v rámci základních a středních škol.

Klíčová slova: Mobile 3D Graphics API (M3G, JSR-184), Java Micro Edition (JME), prostorová grafika, mobilní telefon

Abstract

This diploma thesis deals with the three dimensional graphics in applications for mobile devices. For designing 3D applications there is now Mobile 3D Graphics API (M3G, JSR-184) available which is an extension of Java Micro Edition. The theoretical part consists of a survey of M3G with practical examples and description of general 3D graphics projection principles.

In the practical part author applies the gathered information and design specific application *3D Electrical Circuits*. Modeling, loading, projection and transformations of 3D objects is solved there. The application is designed regarding the goals of project *Starttech* which under the direction of TU in Liberec supports progress of young technical talents.

Keywords: Mobile 3D Graphics API (M3G, JSR-184), Java Micro Edition (JME), three dimensional graphics, cell-phone

Obsah

Seznam obrázků

Seznam příkladů

Seznam použitých zkratek

2D Two-dimensional – Plocha

3D Three-dimensional – Prostor

API Application Programming Interface – Aplikační programové rozhraní

CAD Computer aided design – Počítačem podporované projektování

CLDC Connected Limited Device Configuration - Konfigurace pro JME

ESF Evropský sociální fond

HP Hewlett-Packard – Výrobce mobilních telefonů

ID Identification number – Uživatelské číslo objektu

IDE Integrated Development Environment – Integrované vývojové prostředí

IPv6 Internet Protocol version 6 – Internetový protokol verze 6

JAD Java Application Descriptor – Popis Jav archivu I

JAR Java Archive – Java archiv

JCP Java Community Process – Společenství firem pro vývoj různých součástí platformy Java.

JME Java Micro Edition (JME)

JSR Java Specification Request – Požadavek na změnu nebo doplnění platformy Java

JSR-184 Java Specification Request 184 – Požadavek na doplnění platformy Java s číslem 184

LG Life's good – Výrobce mobilních telefonů

M3G Mobile 3D Graphics API (JSR-184) – Jiné označení pro JSR-184

MIDP Mobile Information Device Profile – Profil pro mobilní zařízení

OpenGL Open Graphics Library – Standard pro tvorbu aplikací počítačové grafiky (otevřená grafická knihovna)

OpenGL ES OpenGL for Embedded Systems – OpenGL pro vestavěná zařízení

OPVK Operační program Vzdělávání pro konkurenceschopnost

PNG Portable Network Graphics – Grafický formát pro bezeztrátovou kompresi rastrové grafiky

RIM	Research In Motion – Výrobce Blackberry smartphonu
RMS	Record Management System – Systém správy záznamů
SE	Sony Ericsson – Výrobce mobilních telefonů
SIM	Subscriber identity module – Účastnický identifikační modul
SPŠSE a VOŠ	Střední průmyslová škola strojní a elektrotechnická a vyšší odborná škola v Liberci.
TUL **TU v Liberci**	Technická univerzita v Liberci
WAP	Wireless Application Protocol – Protokol pro aplikace využívající bezdrátovou komunikaci

Úvod

V dnešní době se grafika vytvořená pomocí počítače používá v mnoha oblastech odborného i všedního života. S počítačovou grafikou se setkáváme nejen ve vědeckých publikacích, při výuce ve školách, na internetu, v televizi, v novinách, ale i v mobilních telefonech, to znamená prakticky na každém kroku. Pomocí dobře vytvořeného grafu nebo obrázku porozumíme obsahu sdělení mnohdy lépe a rychleji než psanému textu. Téměř všechny mobilní telefony používají různé grafické způsoby k tomu, aby uživatelům usnadnily jejich ovládání. Kliknutí na znak nebo symbol, který má podobu obrázku a vyjadřuje určitý pojem, jednoduchým způsobem nahrazuje zadávání složitého textového příkazu.

S použitím platformy Java Micro Edition, která je nyní standardně podporována většinou mobilních telefonů, bylo vytvořeno programové rozhraní pro 3D (prostorovou) grafiku, JSR-184. Toto rozhraní (API) je navrženo tak, aby splňovalo potřeby pro zařízení s méně výkonným procesorem, malou pamětí a bez hardwarové podpory prostorové grafiky. Lze tak vytvářet aplikace zahrnující například plastickou vizualizaci map, různá uživatelská rozhraní s ikonami, animované zprávy, graficky působivé spořiče obrazovek nebo prostorově zobrazované hry. Použití prostorové grafiky je pro uživatele velmi působivé, neboť se svým způsobem zobrazení mnohem více blíží našemu trojrozměrnému vnímání skutečnosti.

Tato diplomová práce si klade za cíl vysvětlit způsoby a možnosti načítání obecných prostorových scén a jejich dynamických transformací na základě impulsů od uživatele mobilního telefonu, pomocí rozhraní JSR-184, a také názorně vyložit funkce a možnosti při aplikaci do projektu *Starttech* (*viz dále*). V rámci rozšiřování spektra projektu *Starttech* se tak naskytla možnost vytvořit aplikaci sloužící pro mobilní telefony ve 3D prostředí simulující propojování elektrických obvodů.

Diplomová práce je rozdělena do pěti kapitol. Po úvodním představení motivace je v druhé kapitole z důvodu přiblížení dané problematiky popsána prostorová grafika se zřetelem na mobilní telefony. Druhá kapitola dále popisuje základní standardy pro 3D grafiku a možnosti způsobu vytvoření obecného prostorového modelu. Ve třetí kapitole je proveden rozbor a popis metod rozhraní JSR-184, včetně uvedení

ilustrujících příkladů. Důraz je kladen na objasnění vlastností a možností tohoto daného programového rozhraní. Čtvrtá kapitola se zabývá popisem návrhu praktické aplikace v JSR-184, která umožňuje uživateli mimo jiné rozvíjet i jeho logické myšlení a zájem o technické obory. Poslední, pátá kapitola je věnována zhodnocení významu a závěrečnému souhrnu práce.

1 Motivace práce

Diplomová páce vznikla ve spolupráci s projektem *Starttech*. Než přistoupíme k teoretické části a způsobu řešení návrhu aplikace, přestavme projekt *Starttech*, jeho cíle a vzájemné propojení se zadáním.

1.1 Projekt Starttech

Projekt *Starttech* vznikl během roku 2009 na Technické Univerzitě v Liberci. Snaží se probudit zájem dětí o technické obory, proto je i jeho podtitulek „Začni s technikou“.

Obrázek 1: Logo projektu Starttech [1]

Starttech je spolufinancován Evropským sociálním fondem (ESF), který se zabývá neinvestičními projekty pro různé skupiny lidí s ohledem na jejich uplatnění. Mezi tyto skupiny patří i mládež a její vzdělávání. *Starttech* je součástí programu „Vzdělávání pro konkurenceschopnost“, který se zabývá zkvalitněním a modernizací vzděláváním a zlepšením podmínek pro všeobecný výzkum a vývoj. Řídícím orgánem pro tento program je Ministerstvo školství, mládeže a tělovýchovy České republiky. Projekt má i své partnery, kterými jsou: Krajská vědecká knihovna v Liberci, Střední průmyslová škola strojní a elektrotechnická a Vyšší odborná škola, Liberec 1. [2]

1.1.1 Cílová skupina, cíle projektu

Hlavním zaměřením *Starttechu* je povzbuzení zájmu o výzkum a vývoj v technických oborech z řad žáků základních škol a studentů středních škol. Cílovou skupinou je tedy především mládež. Snahou je však podpořit i pedagogy a další pracovníky, kteří žáky a studenty vyučují, a mají tak rozhodující vliv na jejich budoucí zaměření.

Následují dva hlavní vybrané cíle projektu:

- Vytvořit soubor vzájemně se doplňujících polytechnických pomůcek, představujících různé technické problémy a demonstrujících interaktivní a přitažlivou formou základní fyzikální a chemické děje [1].
- Vytvořit internetový portál, propagující výsledky výzkumu a vývoje na TUL, dovolující koordinovat a řídit klíčové aktivity projektu směrem k cílové skupině a umožnit osobám, které jsou zapojeny do projektu, výměnu názorů, rad a zkušeností [1].

1.1.2 Forma podpory

Zmíněné cíle jsou realizovány pomocí mimoškolních technických aktivit pořádaných jednotlivými partnery projektu. Jedná se o různé kurzy, semináře, hry a soutěže. Mezi uskutečněné akce patří například: „Vítejte na mechatronice“, „Dny s technikou na SPŠSE Liberec“, „Photoshop a vy“, „Škola robotiky pro mladší“, „Metroškolka“. Zajímavé je, že akce jsou většinou plně obsazeny, i když jsou pořádány ve volném čase cílové skupiny mládeže.

1.2 Osobní zkušenost s JSR-184

Během svého studia jsem se již několikrát zamýšlel a řešil, jak implementovat do mobilního telefonu 3D scénu. Zdánlivě jednoduchý úkol však v sobě ale může občas skrývat problém ve vymodelování vhodné 3D scény a její kompatibility při načtení pro použití v dalším zařízení. Řešení různých 3D modelů a scén tak, aby bezchybně fungovaly (jejich transformace a zobrazení) v emulátoru i na koncovém zařízení a k tomu splňovaly požadavky zadavatele a jejich řešení přitom zůstalo přehledné, je pro mne stále velkou výzvou.

1.3 Požadavky na aplikaci

Dnes je mobilní telefon součástí výbavy téměř každého školáka. V průběhu již výše popsaného projektu *Starttech* vznikla myšlenka na vytvoření aplikace pro mobilní telefony, která by byla k dispozici pro účastníky projektu nejen v době pořádaných akcí, ale i libovolně mimo tento čas. Společně s vedoucím práce a koordinátorem projektu jsme se rozhodli, že vytvořím vhodnou aplikaci, konkrétně simulaci elektrických

obvodů ve 3D prostředí, která bude mimo jiné rozvíjet i logické myšlení a povzbuzovat zájem žáků o technické obory. Vzájemným porovnáním cílů projektu *Starttech* a zásad pro vypracování této diplomové práce byly stanoveny následující základní požadavky pro řešení aplikace:

- Ohled na cílovou skupinu a rozvoj jejího logického myšlení
- 3D prostředí pomocí JSR-184
- Technické zaměření na elektrické obvody
- Vytvoření a použití 3D scén
- Snadné ovládání na základě impulsů od uživatele
- Kompatibilita pro různé mobilní telefony

Pro širší dostupnost, testování a aktualizace aplikace je také cílem vytvořit webové rozhraní (nebo rozšířit webový portál *Starttechu*), kde by zájemci mohli získat více informací a stažení aktuální verze. Požadavky na aplikaci byly postupně upřesňovány po vzájemné konzultaci s vedoucím práce a koordinátorem projektu.

2 Zobrazování a modelování 3D scén

2.1 Počítačová grafika

Asi jen těžko by se dnes hledal dokument, který by neprošel počítačovým zpracováním. Všeobecně téměř všechny výstupy počítače, mimo text nebo zvuk, jsou označovány pojmem počítačová grafika [3]. Ta zahrnuje prostředky pro zobrazení, modelování a úpravu obrazu. Obrazem obecně rozumíme grafické vyjádření určité informace (například fotografie, mapy, technické dokumentace, animace). Protože i grafika v mobilním telefonu používá podobné principy, následuje vysvětlení několika pojmů z počítačové grafiky.

2.1.1 2D grafika

Dvourozměrná (2D) grafika se zabývá popisem objektů definovaných v rovině, jejich rastrováním (převodem do systému posloupnosti obrazových bodů) a úpravami vzniklého obrazu. S dvourozměrnou grafikou se setkáváme například při návrhu novin, webových stránek, reklamních ploch, plošných map, technického kreslení, nebo zpracování digitálních fotografií. [4]

2.1.2 3D grafika

Prostorová (3D) grafika používá mnoho stejných algoritmů jako 2D grafika. Pracuje však s objekty definovanými v prostoru (trojrozměrný souřadnicový systém). Z těchto trojrozměrných dat reprezentujících objekty je potom vytvořen 2D obraz. Výsledný obraz tak vypadá velmi realisticky a působivě. Uživatel díky tomu může mít větší představivost a prožitek. Prostorová grafika se často používá ve filmech, různých hrách, v CAD systémech (projektování, vytváření 3D modelů), vizualizacích dat (navigace), nebo ve výukových simulátorech.

2.1.3 Proces vytvoření 3D obrazu

Při návrhu 3D scény pro zobrazení se předpokládá, že jsou definovány určité grafické objekty. Objekty jsou popsány posloupnostmi bodů a dalšími atributy (například barva nebo průhlednost), které blíže určují způsob zobrazení. Grafické

objekty při zobrazování scény vstupují do zobrazovacího řetězce, kde se na ně postupně aplikují následující operace:

• Transformace – ve scéně jsou použity různé modelovací souřadnicové systémy, vlastní systém má i pozorovatel (kamera), který scénu vnímá. Z tohoto důvodu je potřeba scénu před zobrazením podrobit promítání a teprve výslednou geometrii transformovat do okna displeje.

• Ořezání – při definici scény, kterou obvykle není nutné zobrazovat celou, se definuje prostor, který se bude promítat. Stejně tak i prostor displeje má svá omezení. Výsledný obraz na displeji je průnikem obou těchto prostorů.

• Stínování – barva zobrazovaného bodu displeje (pixelu) závisí na osvětlení scény, proto je nutný výpočet odstínů barvy pixelu.

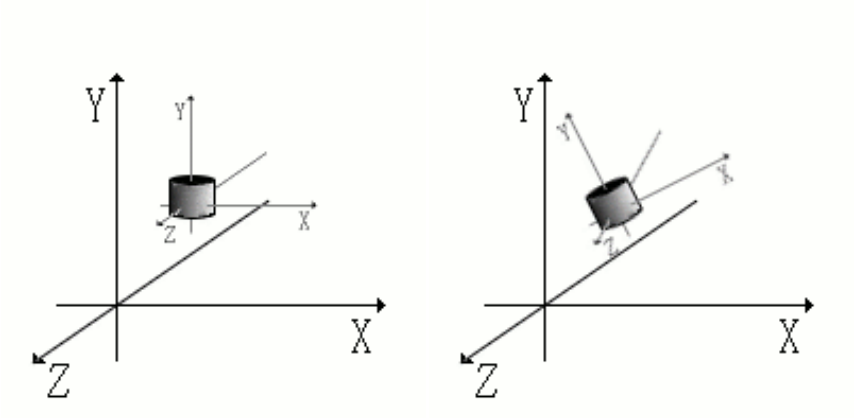
• Rastrování – během rastrování se provádí rozklad grafického objektu na jednotlivé pixely a určí se, které pixely v paměti budou patřit danému grafickému objektu. Výsledná viditelnost objektů se řeší pomocí *hloubkového bufferu*, který má konečnou velikost. Není tedy možné ukládat informace přesahující jeho velikost. Definují se tedy dvě plochy, blízká a vzdálená rovina kamery. Objekty mezi těmito dvěma rovinami jsou vykresleny, ostatní jsou ignorovány. Vzdálenosti objektů jsou porovnány pro každý pixel displeje, přičemž pro každý pixel je zachován pouze nejbližší objekt. [5]

2.1.4 Lineární algebra v 3D grafice

Pro definici a určení polohy objektů ve scéně je zapotřebí soustavy souřadnic. K prostorovému zobrazení se používá kartézské trojrozměrné, ortogonální (každá osa je kolmá ke zbylým dvěma) pravotočivé soustavy. Osy, obvykle označovány *x, y* a *z*, jsou navíc délkově normalizovány. Ve scéně je definována globální soustava souřadnic pro všechny objekty a je neměnná. Všechny objekty mohou být zobrazovány a transformovány vzhledem k této soustavě. Každý objekt může mít navíc svoji vlastní (lokální) soustavu souřadnic a v ní také může být přemísťován. Orientace vlastní soustavy se může měnit, viz obrázek 2.

Například soustava souřadnic místnosti může mít počátek v jejím rohu (globální). Předmět umístěný v místnosti může mít svou vlastní (lokální) soustavu například s počátkem v těžišti předmětu. Pokud je předmět přesunut, nebo otočen,

vlastní soustava se otáčí společně s předmětem, zatímco globální soustava místnosti zůstává neměnná.

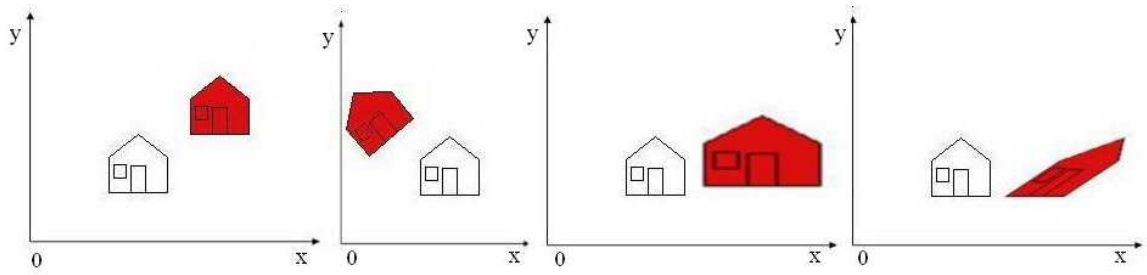


Obrázek 2: Změna orientace lokální soustavy souřadnic předmětu [6]

Libovolného bodu **P** ve 3D scéně o souřadnicích [P_x, P_y, P_z] se dosáhne posunutím z počátku souřadnic ve směru *osy x* o vzdálenost P_x, následným posunutím ve směru *osy y* o vzdálenost P_y a ve směru *osy z* o vzdálenost P_z. Dva body určují úsečku (přímku), tři body trojúhelník (plochu). Pomocí sítě vzájemně propojených trojúhelníků lze s určitou přesností definovat povrch objektu. Více objektů umístěných souhrnně v globální soustavě souřadnic pak definují virtuální svět.

Pro popis souřadnic objektů v prostoru se také často používají homogenní souřadnice. Oproti kartézským souřadnicím [X, Y, Z] se zde používá čtveřice [x, y, z, w], kde w značí váhu. V normalizovaném tvaru (w = 1) jsou složky *x*, *y* a *z* shodné s kartézskými. Pokud w = 0, jedná se o vektor, v ostatních případech o vlastní bod prostoru. Pro jiné libovolné *w* lze normalizovaný tvar získat pomocí převodu [x/w, y/w, z/w, 1]. Homogenní souřadnice se používají z toho důvodu, že pomocí nich společně s transformačními maticemi (4×4) je možné vyjádřit všechny prostorové transformace, jako je posunutí, rotace, změna měřítka, zkosení, dokonce i perspektivní a paralelní projekce. Nové souřadnice bodu (*v'*) se získají vynásobením sloupcového vektoru bodu (*v*) transformační maticí (*M*) zleva. Transformaci lze zapsat vztahem $v' = M \times v$ (OpenGL). Několik po sobě jdoucích transformací lze vyjádřit pomocí jedné matice, která vznikne postupným násobením základních matic. Součin matic (M_1, M_2) však není komutativní, proto $M_1 \times M_2$ není stejné jako $M_2 \times M_1$. [4]

Transformace objektu ve 3D scéně se docílí aplikací transformace na všechny body objektu. Nejpoužívanější jsou afinní transformace, kde spodní řádek transformační matice je [0 0 0 1]. Běžně se používají k umístění a orientaci objektu ve scéně nebo ke změně jejich velikosti nebo tvaru. Bílý objekt na obrázku reprezentuje originál, červený provedenou transformaci.



(a) Posunutí (b) Rotace (c) Změna měřítka (d) Zkosení

Obrázek 3: Základní transformace

Posunutí

Posunutí o vektor $[t_x, t_y, t_z]$ změní polohu bodu **P** $[P_x, P_y, P_z]$ na **P'** $[P_x + t_x, P_y + t_y, P_z + t_z]$. Matice transformace posunutí **T** a inverzní matrice $\mathbf{T}^{-1}$ mají tvar:

$$\mathbf{T}(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T^{-1}}(t_x, t_y, t_z) = \mathbf{T}(-t_x, -t_y, -t_z) = \begin{bmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Jak je vidět na obrázku 3 (a), tvar objektu se nemění, je jen přesunut. Inverzní transformace (získání původních souřadnic z matice transformace) je pouze posun opačným směrem.

Rotace

Rotační matice otáčí bod kolem osy vzhledem k počátku. Následuje rotační matice $\mathbf{R_z}$ kolem *osy z* o úhel α a její inverzní matice $\mathbf{R_z}^{-1}$:

$$\mathbf{R_z}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R_z^{-1}}(\alpha) = \mathbf{R_z}(-\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Podobné matice mohou být vytvořeny pro rotace kolem *osy x* a *y*. Směr otáčení je určen pravidlem pravé ruky. Při pohledu podél osy směrem k počátku, kladný směr otáčení ostatních os je proti hodinovým ručičkám. Kombinací těchto tří základních rotací lze vytvořit rotaci kolem libovolné osy.

Změna měřítka

Další transformací je změna měřítka, která mění velikost objektu, eventuelně i tvar. Matice **S** a inverzní matice $\mathbf{S^{-1}}$ jsou:

$$\mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{S^{-1}}(s_x, s_y, s_z) = \boldsymbol{S}\left(\frac{1}{s_x}, \frac{1}{s_y}, \frac{1}{s_z}\right).$$

Koeficienty s_x, s_y, s_z mění měřítko příslušné souřadnicové osy. Pokud je koeficient větší než jedna, jedná se o zvětšení transformovaného objektu. Je-li hodnota koeficientu v intervalu (0,1) dochází naopak ke zmenšení. Záporné znaménko u koeficientu má za následek zvětšení, nebo zmenšení v opačném směru. Inverzní maticí je ekvivalentní změna měřítka s převrácenou hodnotou. Pokud $s_x = s_y = s_z$, jedná se o uniformní změnu měřítka, která změní velikost objektu, ale ne tvar. Na obrázku 3 (c) nejsou koeficienty stejné, proto je i tvar objektu pozměněn (více roztáhnut v horizontálním směru). Vhodným nastavením koeficientů lze realizovat středovou a osovou souměrnost, či souměrnost podle roviny.

Zkosení

Zkosení je v počítačové grafice méně používaná transformace. Je získána změnou některé nuly horní levé 3×3 části matice na nenulovou hodnotu. Zkosení vždy deformuje tvar. Inverze ke zkosení je získána zkosením s opačnými hodnotami koeficientů. Následuje příklad transformace zkosení ve směru *roviny xy*, kde koeficienty sh_x a sh_y odpovídají míře zkosení v odpovídajícím směru.

$$\mathbf{Sh_{xy}}(sh_x, sh_y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sh_x & sh_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{Sh_{xy}^{-1}}(sh_x, sh_y) = \mathbf{Sh_{xy}}(-sh_x, -sh_y).$$

Odvozené transformace

Na obrázku 3 jsou všechny transformace provedeny vzhledem k počátku souřadnic. Zde například při zvětšení objektu se každý bod vzdaluje od počátku. Nicméně někdy je potřeba transformace vzhledem k určitému jinému bodu (*řídící bod*), například střed nebo jeden z rohů objektu. Protože počátek je jediný bod, který se při transformacích nepohybuje, je řídící bod přemístěn do počátku, pak provedena transformace a řídící bod je poté přemístěn zpět na původní místo. Například otočení

bodu **P** [P_x,P_y,P_z] kolem bodu **S** [S_x, S_y, S_z] kolem *osy z* o úhel α na nový bod **P'** [P'_x,P'_y,P'_z] by tedy mělo tuto sekvenci transformací:

$$\mathbf{P}'[P'_x, P'_y, P'_z, 1]^T = \mathbf{T}(S_x, S_y, S_z) \times \mathbf{R_z}(\alpha) \times \mathbf{T}(-S_x, -S_y, -S_z) \times \mathbf{P}[P_x, P_y, P_z, 1]^T$$

Jak již je zmíněno v předchozí podkapitole 2.1.3, aby scéna mohla být vykreslena, musí být objekty transformovány do soustavy souřadnic kamery. Avšak místo modelování přímo v soustavě kamery je jednodušší modelovat objekty v globálních osách a poté provést potřebné transformace.

2.2 3D Grafika v mobilním telefonu

Mobilní telefony se v poslední době staly univerzálním prostředkem pro komunikaci. V České republice a i v některých ostatních zemích Evropy je více zákazníků (počet aktivních SIM karet) než počet obyvatel. Současně se také zlepšují schopnosti jednotlivých zařízení. Pokud by rozlišení displeje mobilního telefonu dosáhlo hodnot srovnatelných s běžnou televizí, mohli bychom mluvit o *mobilním počítači*. Dnešní běžně dostupné mobilní telefony jsou výkonné asi jako byly stolní počítače na konci 90. let. Výkon počítačů však bude pravděpodobně vždy o určitý krok napřed. Navíc mobilní telefony nemají velkou klávesnici, jsou napájené malou baterií, která se může rychle vybít, a jsou zde omezení týkající se velikosti datového prostoru. S příchodem barevných displejů a výkonnějších procesorů jsou ovšem i mobilní telefony schopny vykreslit 3D grafiku pomocí interaktivní sekvence snímků. 3D grafika v mobilních telefonech se tak stává pro uživatele přístupná a je čím dál více žádána. [7] [8]

Kromě her a vizualizací je vhodné využít 3D technologie pro aplikace v mobilních telefonech například k inzerci a pro multimediální zprávy, neboť vizuální prvky jsou vhodné k upoutání pozornosti zákazníků. Prostorová grafika se používá nejen v přídatných aplikacích, ale často je zařazena jako součást uživatelského menu, které se tak stává více intuitivním a snadněji použitelným. Některé aplikace (CAD systémy) však z důvodu vhodnější ergonomie počítačů, na mobilních telefonech prakticky nenajdeme.

Obrázek 4: Ukázka 2D a 3D grafiky v mobilním telefonu

Playman Winter Games – Mr. Goodliving [9]

2.2.1 Rozdělení mobilních telefonů

Mobilní telefony s ohledem na podporu 3D grafiky lze dle [10] rozdělit do tří kategorií:

- Základní telefony (Basic phones)
- Pokročilé telefony (Feature phones)
- Chytré telefony (Smart phones)

Základní telefony mají obvykle vlastní specifický *firmware* (softwarová část řídící všechny funkce) připravený od výrobce. Nové aplikace pro tyto přístroje lze proto vyvíjet pouze v úzké spolupráci s daným výrobcem. Tyto telefony jsou sice levné, ale velmi omezené z hlediska velikosti a rozlišení displeje, nebo výpočetního výkonu. Omezený výkon procesoru bez hardwarové podpory 3D grafiky umožňuje provozovat pouze nejjednodušší 3D aplikace.

Pokročilé telefony představují podstatnou část trhu a většina z nich podporuje Javu Micro Edition (JME). Vlastnosti a významné prvky programovacího jazyka JME jsou popsány v podkapitole 2.2.2). Každý rok jsou vyrobeny stovky různých nových modelů telefonů s JME. JME umožňuje vyvíjet aplikace pro velký počet zařízení za pomoci jednotné programovací platformy. Nabízí dostatek programovacích rozhraní pro většinu multimediálních potřeb, včetně 3D grafiky. Vlastnostem JSR-184, 3D grafického API (application programming interface) pro JME a jeho třídám je

věnována kapitola 0 této práce a navržená aplikace je určená především pro tuto kategorii telefonů. [10]

Chytré telefony, známé stále více pod označením *smartphony* se svým charakterem přibližují počítačům. Mají větší obrazovky, výkonnější procesory, relativně dostatek paměti, podporu multimédií včetně připojení k internetu. Některé z přístrojů zahrnují i 3D grafický hardware. Smartphony používají operační systémy jako například Symbian, RIM Blackberry, Apple iPhone a Windows Mobile, což umožňuje instalaci jakéhokoli softwaru (aplikací) pro daný systém. Na smartphonech lze spouštět i JME aplikace za pomoci emulátoru (podpora JSR-184 však není vždy zaručena). V tomto je hlavní rozdíl oproti pokročilým telefonům, které umožňují provozovat výhradně JME aplikace. Kromě telefonování lze smartphony díky mnoha různým aplikacím využívat i ke správě e-mailů, pohodlnějšímu prohlížení webových stránek, práci s mobilními kancelářskými aplikacemi Office, jednoduché synchronizaci s počítačem a dalším. [11]

Na obrázku je ukázka mobilních telefonů ze zmíněných kategorií, popisováno zleva: telefon se základními funkcemi Nokia 1661, mobilní telefon s pokročilejšími funkcemi Sony Ericsson G502 a Smartphone HP imag 914.

Obrázek 5: Nokia 1661, SE G502, HP iPaq 914 [12]

2.2.2 Java Micro Edition

Java Micro Edition (JME), dříve označována J2ME, je jedna ze tří platforem programovacího jazyka Java, který si získal svoji popularitu především díky své

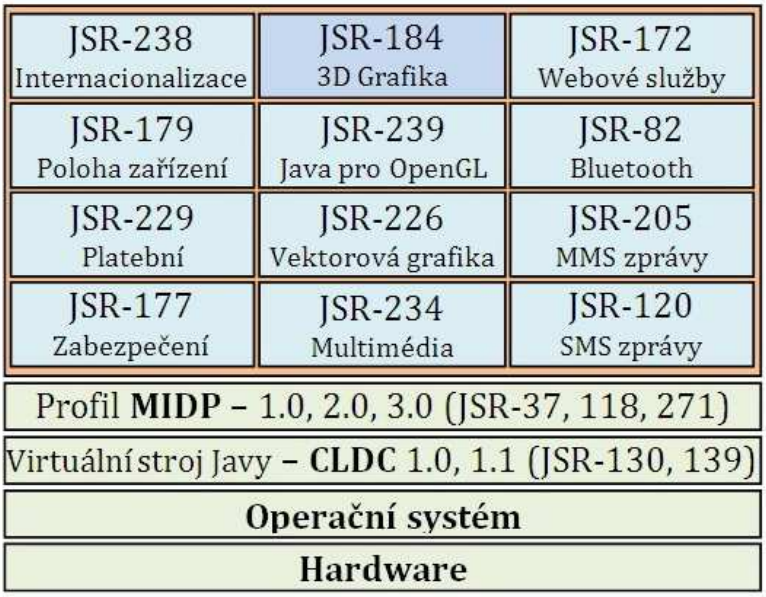
přenositelnosti a bezpečnosti. Jedná se o objektově orientovaný, interpretovaný jazyk, kde samotná aplikace je tvořena mezikódem (*byte kód*), který je poté interpretován až na koncovém zařízení pomocí virtuálního stroje. Jestliže dané zařízení má v sobě nainstalovaný virtuální stroj, pak lze Java aplikaci bez problémů spustit. Vytvořený program je nutné před spuštěním zkompilovat, což má za následek pomalejší spouštění aplikací, obzvláště v použití na mobilních telefonech [13].

JME je určena výhradně pro malá bezdrátová zařízení s omezeným paměťovým prostorem. Díky podpoře mezi téměř všemi mobilními telefony se stala jednou z nejrozšířenějších platforem na světě. Dále díky přenositelnosti obsáhne JME pokrytí značně různé hardwarové a softwarové konfigurace, což je nezbytné v prostředí mobilních zařízení, kde žádný výrobce nemá dominantní postavení. Pro vývojáře aplikací tak není podstatné, jaký firmware nebo operační systém mobilní zařízení používá. Platforma JME je velmi vhodným doplňkem k jinak uzavřenému operačnímu systému telefonu. V počátcích produkce se sice vyskytovala nekompatibilita rozhraní telefonů s konkrétními aplikacemi, ale výrobci se však postupně dohodli na společných testech a certifikacích aplikací, což umožnilo vzájemnou slučitelnost rozdílných produktů. Uživatel tak dnes již může mít větší jistotu, že přenesená aplikace funguje i na jeho telefonu. [14]

Prostředí JME obsahuje množinu standardních Java API, virtuální stroj Javy, řadu komunikačních protokolů a zabezpečení. Vydání a revize dalších API, označovány jako JSR (Java Specification Request) průběžně vznikají pod vedením Java Community Process (společenství firem mající za cíl vyvíjet rozšíření pro platformu Java). Mezi členy konsorcia patří například Sony Ericsson, Motorola a Nokia.

Základní vlastnosti pro jednotlivá zařízení jsou určeny *konfiguracemi* JME, které definují programové vybavení pro určité skupiny zařízení. Podmnožiny programového vybavení pro danou úroveň zařízení jsou dány *profily*. Ty jsou rozlišeny podle typu a velikosti paměti, typu a rychlosti procesoru a síťového připojení. Profil doplňuje konfiguraci tak, že výsledná aplikace je lépe přizpůsobena vlastnostem dané skupiny zařízení. Profil MIDP (Mobile Information Device Profile) je používán pro bezdrátová zařízení (mobilní telefony a pagery) s konfigurací CLDC (Connected Limited Device Configuration). Společně tak poskytují bohaté runtime prostředí pro tato zařízení. Kromě profilu a konfigurace jsou k dispozici i *volitelné balíčky*, které poskytují podporu

dodatečných funkcí (například 3D grafika, bluetooth, fotoaparát). Záleží na rozhodnutí výrobce, se kterými volitelnými balíčky bude virtuální stroj na vybraném zařízení pracovat. Je tedy možné, že i když JME má k dispozici API s 3D grafikou, práce s ní není podporována, protože chybí knihovna, která nebyla do zvoleného zařízení implementována. Volitelná API a architektura JME je znázorněna na následujícím obrázku. [15]



Obrázek 6: Architektura JME s MIDP, CLDC a volitelných API (JSR) [14]

CLDC verze 1.0 (JSR-30) byla první zveřejněná specifikace CLDC, která poskytovala kompaktní virtuální stroj a základní knihovny pro malá zařízení s omezenými možnostmi. V roce 2003 byla rozšířena na verzi CLDC1.1 (JSR-139), která pokračuje v zaměření na univerzálnost místo na konkrétní kategorii zařízení. Verze 1.1 je doplněná o další dodatky, jako je například podpora pro matematické operace s plovoucí řádovou čárkou (datové typy float a double).

Stejně tak byla nejdříve uvolněna verze 1.0 profilu MIDP (JSR-37), implementující základní rozhraní pro práci s uživatelem a podporou sítí. Aby se zamezilo nadbytečnému růstu volitelných API, v roce 2002 byla rozšířena na verzi MIDP 2.0 (JSR-118). Mezi významné doplnění patřila třída pro podporu zvuku a videa, zvláštní API pro usnadnění her, nebo rozšíření síťových standardů. Dnes jsou zařízení s MIDP mnohem výkonnější a mají více paměti, v prosinci roku 2009 byla schválena verze MIDP 3.0 (JSR-271). S použitím této verze bude možné očekávat spuštění více

aplikací najednou, vnitřní komunikace mezi aplikacemi, možnost automatického spuštění aplikace, vylepšení uživatelského rozhraní, zavedení podpory pro více displejů, nebo IPv6. [16]

Verze CLDC a MIDP jsou na sobě nezávislé, existují tedy zařízení s kombinací CLDC 1.1 a MIDP 2.0, ale i CLDC 1.0 a MIDP 2.0.

Java aplikace pro MIDP se nazývají Midlety. Midlet obsahuje třídy a metody pro řízení průběhu aplikace. Alespoň jedna ze zdrojových tříd musí být odvozena od abstraktní třídy *javax.microedition.midlet.Midlet*. Midlet se může nacházet ve třech stavech: přerušený, aktivní nebo zrušený. Zařízení s MIDP jsou schopna zobrazit pouze jedno uživatelské okno. Před instalací do daného zařízení se ze všech Midletů aplikace vytvoří soubor JAR (Java Archive). Do něho jsou zabaleny všechny potřebné třídy aplikace, informace o Midletu, obrázky a jiné zdrojové soubory. Informace o Midletu (velikost, název, požadovaná konfigurace a profil, výrobce, verze) lze také najít při kompilaci vytvořeném textovém souboru typu JAD (Java Application Descriptor).

2.2.3 JSR-184 a jiné standardy pro 3D rozhraní

Dnes lze najít několik standardů pro 3D grafiku v mobilních zařízeních. Nejčastěji se vyskytují v podobě API, které mohou používat různé programovací jazyky jako Java, C nebo C++.

První telefony podporující 3D grafiku uvedl na trh v roce 2001 japonský operátor JPhone, společnost tak představila API HI Corporation's Mascot Capsule. Hlavním záměrem v té době bylo zobrazení jednoduchého animovaného textu. Všichni hlavní operátoři v Japonsku toto API využívají v různých podobách. V roce 2004 byla Nokia 3410 (stále s jednobarevným displejem) prvním telefonem na evropském trhu s podporou 3D. Na jeho rozhraní se podílel Autodesk 3ds Max. Nebylo však k dispozici žádné veřejné API. Vytvoření veřejného API, které by podporovalo velké množství různých zařízení, bylo obtížným úkolem. Alternativa, že pro každý typ zařízení bude jiné API, byla téměř pro všechny výrobce a návrháře aplikací nepřijatelná. [17]

Některé operační systémy, jako například Symbian, Windows Mobile povolují instalaci aplikací napsaných v jazyku C++ nebo v assembleru. Vývojář tak může nezávisle vytvořit účinné 3D jádro aplikace (*engine*). Ostatní zařízení s vlastními uzavřenými systémy povolují instalaci Java Midletů. Jak už bylo zmíněno dříve, totožný

Midlet lze spustit na odlišných zařízeních a v různých operačních systémech. V roce 2002 konsorcium Khronos Group definovalo OpenGL ES, podmnožinu jazyka OpenGL, vhodnou pro vestavěné systémy. Současně s tím pod vedením konsorcia Java Community Process (JCP) byla zahájena normalizace 3D grafického rozhraní pro JME. Tak vzniklo API Java specification request s číslem 184, odtud výraz „JSR-184“. Standard je však známější spíše pod označením Mobile 3D Graphics (M3G). Koncem roku 2003 byly obě API, OpenGL ES 1.0 a M3G 1.0 schváleny a první implementace byly dodávány na trh přibližně o rok později [10]. Dnes jsou většinou k dostání na trhu mobilní telefony od všech hlavních výrobců podporující obě uvedená rozhraní.

OpenGL ES

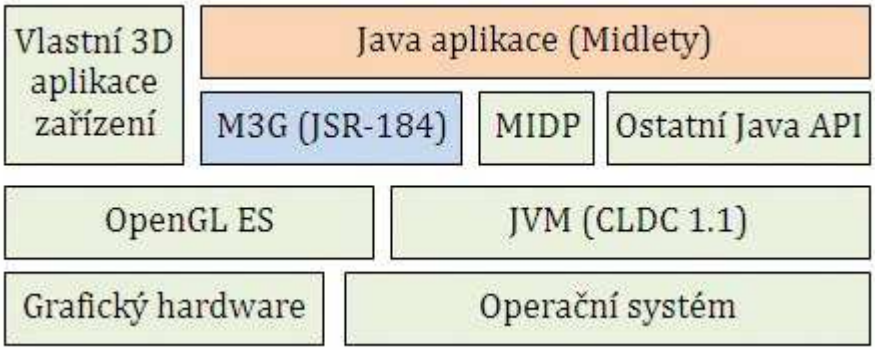
OpenGL je široce známá knihovna pro návrh otevřeného nízko-úrovňového grafického API pro různé platformy. OpenGL bylo nicméně samo o sobě pro mobilní zařízení příliš velké. Bylo proto nutné upravit zřídka používané funkce a lépe toto rozhraní přizpůsobit technickým omezením mobilních zařízení. Bylo tak vytvořeno OpenGL ES (ES – embedded systems – vestavěné systémy). Toto rozhraní poskytuje nízko-úrovňové rozhraní mezi aplikacemi, hardwarem a softwarovými grafickými enginy. Umožňuje snadný přístup k řadě pokročilých 3D grafických funkcí. Stejné grafické efekty známé z osobních počítačů jsou tak k dispozici i na mobilních zařízeních. Vzhledem k tomu, že se toto rozhraní velice úzce podobá původnímu OpenGL, vyplynuly z toho určité výhody. Standardizační proces mohl postupovat rychleji, nebylo nutno vytvářet nové technologie a je k dispozici obsáhlá literatura o OpenGL, která ulehčuje vývojářům práci s tímto API. [22]

Mobile 3D Graphics API (JSR-184)

JSR-184 (Java Specification Request 184) je rozšiřujícím Java API pro 3D grafiku na mobilních zařízeních. Toto rozšířené API je ve formě volitelného balíčku *javax.microedition.m3g* a očekává použití s profilem MIDP a konfigurací CLDC 1.1. Definuje nízko-úrovňové a především vysoko-úrovňové programové rozhraní, které přináší výkonnou interaktivní 3D grafiku do zařízení obecně s malou pamětí, malým výkonem procesoru a žádnou hardwarovou podporou 3D grafiky. S příchodem nových telefonů s rozmanitějšími funkcemi je API schopno se přizpůsobit i zařízením

s barevnými displeji, 3D grafickým hardwarem a podporu plovoucí řádové čárky. Charakteristika JSR-184 je uvedena v následujících bodech:

- Zcela nové rozhraní, které využívá myšlenek z předchozích API, jako je například Java 3D, může být použito k tvorbě jakéhokoli 3D obsahu.
- Skládá se z jednotlivých uzlů, které zastupují prostorové grafické objekty. Uzly mohou být včleněny do grafu, který představuje jednotlivé objekty a jejich vzájemné vztahy.
- Rozhraní bylo navrženo tak, aby mohlo být efektivně implementováno nad OpenGL ES a umožňuje 3D grafiku pro Midlety Javy, zatímco OpenGL ES slouží nejen pro implementaci JSR-184 , ale i pro vlastní aplikace navrhované výrobcem zařízení (tato architektura je vidět na obrázku 7). [18]



Obrázek 7: Architektura mobilního zařízení pro 3D grafiku [17]

Rozdíl mezi vysoko a nízko-úrovňovým rozhraním spočívá v míře použití abstrakce. Vysoko-úrovňové API, operuje nad komponentami grafických a uživatelských rozhraní (viz obrázek 7 a postavení M3G), je tedy více abstraktní. Aplikace je pak více závislá na implementaci rozhraní daného zařízení, které automaticky obstarává určité funkce. Oproti tomu v nízko-úrovňovém API je důraz kladen na přímý přístup ke komponentám zařízení s velice nízkou mírou abstrakce mezi API a hardwarem. Proto se říká, že nízko-úrovňové rozhraní je „blízko k hardwaru“. Tím, že vysoko-úrovňové M3G rozhraní je vzájemně kompatibilní s nízko-úrovňovým OpenGL ES, tyto dvě rozhraní si nekonkurují, ale spíše se vzájemně doplňují. Návrh pomocí M3G zvyšuje produktivitu programátorů a vývojářů, minimalizuje velikost kódu grafických aplikací a zvyšuje výkon aplikací tím, že využívá 3D akceleraci prostřednictvím OpenGL ES.

Nyní je k dispozici verze M3G 1.1 (JSR-248), která oproti původní verzi 1.0 zlepšuje výkon a schopnost sdílení dat existujících funkcí. Nepřinesla žádné nové významné rozšiřující změny, proto dodavatelé velice rychle a bez problémů aktualizovali svá zařízení pro tuto verzi. Verze M3G 2.0 (JSR-297) je stále ve vývoji. Návrh ke konečné verzi (Proposed Final Draft) byl uvolněn v dubnu 2009. Tato verze má přinést nové funkce v podobě nových zobrazovacích metod, které povedou k daleko větší přesvědčivosti 3D grafiky. Dále se pak očekává zejména snížení požadavků na velikost paměti a rychlejší vykreslování scén. [16]

JSR-184 má svůj vlastní 3D formát souboru s názvem M3G. Tento velice univerzální 3D typ souboru je schopen obsáhnout mnoho dat týkajících se prostorových modelů, způsobu jejich osvětlení, textury povrchů, umístění kamer a možnost provádění animace. Soubor může být načten pouze jako celek. Pokud se vyskytne během načítání chyba, není proveden žádný pokus o vykompenzování a soubor se nenačte. Formát M3G je velice vhodný svými vlastnostmi a také ho lze poměrně snadno načíst do navržené aplikace. [19]

Java Bindings pro OpenGL ES API (JSR-239)

Kromě JSR-184 existuje i rozhraní Java Bindings pro OpenGL ES (JSR-239), které poskytuje 3D API. JSR-184 je objektově orientované, a proto je rozhraní pro základní funkce odlišné od OpenGL. JSR-239 se podobá OpenGL ES a slouží především pro vývojáře, kteří jsou již dobře obeznámeni s OpenGL a nevyžadují žádné vyšší funkce poskytované rozhraním JSR-184. [16]

2.3 Tvorba M3G

Scéna pro zobrazení na displeji může být vytvořena vývojářem buď v kódu pomocí jednoduchých grafických elementů, nebo ji je možno předpřipravit pomocí 3D modelovacího nástroje a pak ji vyexportovat do M3G formátu. Navrhování scény pro JSR-184 je podobné návrhům pro počítačovou 3D grafiku.

Modely pro mobilní telefon by měly být co nejvíce jednoduché a kompaktní. To se zajistí minimalizací počtu polygonů na zobrazovaných objektech. Zobrazení objektů pomocí 2D reprezentace je rychlejší a výpočetně méně náročné než v prostorovém (3D) provedení. Proto je vhodné nahradit některé 3D objekty 2D obrázky

(například mraky, sněhové vločky, informační tabule). Také se doporučuje rozdělit větší objekty na menší celky a nahradit opakující se matematické funkce asociativními poli.

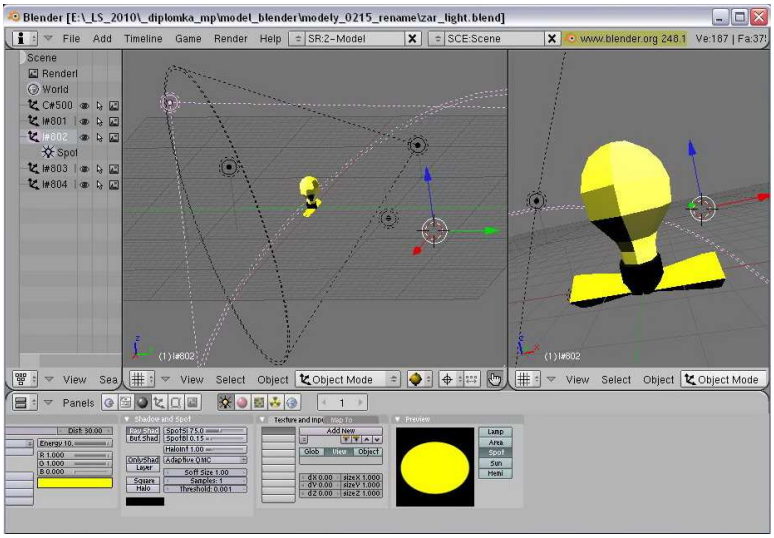
Existuje více programů pro vytvoření 3D scény ve formátu M3G pomocí dodatečně instalovaných doplňků k programům 3D Studio Max, Maya, Softimage, Lightwave, Blender a dalším. Osobně jsem měl možnost se seznámit s programy 3D Studio Max a Blender.

2.3.1 3D Studio Max

Autodesk 3ds Max je široce známý profesionální komerční program pro 3D grafiku, vizualizace a animace. K tomuto softwaru lze doinstalovat opět komerční *Export 184 plug-in* přímo pro vytvoření M3G souborů. Ovládání tohoto programu je velice intuitivní a uživatel si rychle zvykne na dané prostředí. V průběhu modelování scény pro tuto diplomovou práci se autorovi však nepodařilo v omezené zkušební verzi nastavit parametry pro export tak, aby se výsledná scéna zobrazovala zcela bez chyb. Proto pro navržení scény byl použit program Blender.

2.3.2 Blender

Blender je software určený pro 3D modelování, který je dostupný pro velké množství operačních systémů, jako je Windows, Linux, Mac OS X.



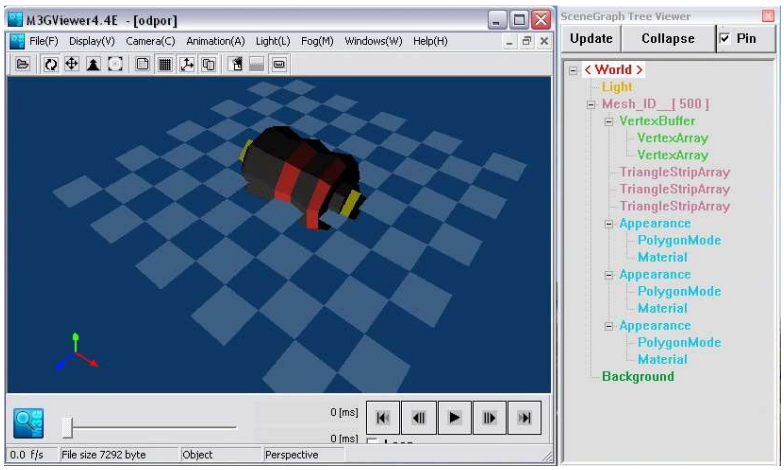
Obrázek 8: Vytváření 3D modelu v programu Blender

Další výhodou Blenderu je, že je šířen zcela zdarma, a to včetně zdrojových kódů (jedna licence komerčního programu pro 3D modelování obvykle stojí desetitisíce

až statisíce korun). Stejně tak pomocí plug-inu *m3g_export* lze exportovat scény do formátu M3G nebo java.class, které se následně využijí v aplikacích s JSR-184. [20]

2.3.3 M3G Viewer

Pro prohlížení obsahu souboru M3G, který doplňuje Mobile 3D Graphics API je k dispozici samostatná aplikace M3G Viewer [21]. S jeho pomocí je možné si prohlédnout vlastní vyexportované 3D scény, nebo prohlédnout scény již vytvořené jiným autorem, aniž by k tomu bylo potřeba zdrojový soubor, ze kterého byly scény vytvořeny. Obvykle používanými prohlížeči obrázků a multimédií nelze tento typ souboru načíst.



Obrázek 9: Náhled M3G souboru v programu M3G Viewer

V programu M3G Viewer je možno zobrazit i hierarchii objektů ve scéně (viz obrázek 9), přehrát animaci, měnit pohledy, nebo měnit způsob osvětlení scény.

3 Mobile 3D Graphics API (JSR-184)

Objektově orientované rozhraní JSR-184 se skládá z 30 tříd, pomocí kterých je schopno zobrazit komplexní 3D scény. Třída *Graphics3D* slouží k vykreslování, *World* zastupuje celou 3D scénu a *Camera* určuje způsob pohledu. 3D objekty, které jsou definovány body, polygony a síťovými modely, jsou obvykle uspořádány do stromové struktury. Architektura tříd tohoto balíčku *javax.microedition.m3g* je znázorněna v Příloze A. Po popisu zobrazovacích režimů jsou v následujících podkapitolách popsány vlastnosti základních tříd s uvedením názorných příkladů. Většina z těchto tříd byla také využita k návrhu aplikace *3D obvodů*.

3.1 Immediate a retained režim

JSR-184 umožňuje vykreslování scény ve dvou režimech. Oba jsou důležité, i když se každý používá v odlišných situacích, mohou se vzájemně kombinovat.

Immediate (okamžitý) režim umožňuje nízko-úrovňovou kontrolu nad zobrazovanou scénou. Protože tento režim dovoluje přístup k nízko-úrovňovým základním jednoduchým prvkům, je možné definovat objekty pomocí pole bodů. Objekty jsou ihned vykresleny s použitím aktuálních světel a kamery, příkazy jsou okamžitě vyhodnoceny a zpracovány renderovací jednotkou.

V *retained (zadržovaném) režimu* jsou všechny funkce řízené na vyšší úrovni. K vykreslení scény se používá stromové struktury (*scene graph*), obsahující všechny objekty společně se světly a kamerou. Každý objekt ve scéně představuje určitý list tohoto stromu, který v sobě zahrnuje informaci o svých vlastnostech, pozici v 3D prostoru a definuje vzájemné vztahy k ostatním objektům scény. Scéna není vykreslena ihned. Nejprve vykreslovací algoritmus projde *scene graph*, získá informace o viditelnosti a poloze objektů a o úhlu pohledu kamery, aby mohl redukovat výsledný počet zobrazovaných ploch před konečným vykreslením obrazu. Renderovací jednotce je předán jen kořen stromu a následně je vykreslena „zjednodušená“ scéna. V daném čase může být aktivní pouze jedna zobrazovací kamera. [23]

Oba režimy jsou navrženy tak, že je lze kombinovat i ve stejné aplikaci. *Immediate režim* je použitelný v případě s jednoduchou scénou s jedním polem vrcholů,

nebo když je potřeba zobrazení pouze určité části scény samostatně. *Retained režim* je častěji používán a je více abstraktní. Pro složitější model scény složené z několika objektů, nebo pro načtení stromové struktury scény uložené v M3G souboru je jednodušší a méně paměťově náročné použít *Retained režim*.

3.2 Loader

Tato třída slouží k načítání souborů. Pomocí zavolání metody *Loader.load* lze získat všechny reference z M3G souboru. Jako argument metody se používá adresa uloženého souboru, která je typu *String* (viz příklad 1).

```
public void loadWorld(String path) {
    try {
        Object3D[] buffer = Loader.load(path);
        for (int i = 0; i < buffer.length; i++) {
            if (buffer[i] instanceof World) {
                world = (World) buffer[i];
                break;
            }
        }
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

Příklad 1: Načtení scény z M3G souboru

Metoda vrací načtená data v podobě pole 3D objektů (*Object3D[]*). Třída *Loader* umožňuje načítat nejen M3G soubory, ale i všechny třídy odvozené od *Object3D*. Načítaná data musí být v platném M3G souboru, alternativou může být obrázek typu PNG.

3.3 Object3D

Jedná se o abstraktně založenou třídu pro všechny objekty, které mohou být součástí 3D scény. *Object3D* může reprezentovat konkrétní scénu (*World*), animace, textury, nebo ostatní uzly scény. Vše v tomto API je odvozené od *Object3D*, kromě tříd *Loader*, *Transform*, *RayIntersection*, a *Graphics3D* (viz Příloha A).

Každému prvku *Object3D* může být přiřazeno uživatelské číslo ID, a to již při návrhu M3G souboru, nebo později za běhu programu pomocí metody *setUserID*. Uživatelské číslo ID se obvykle používá k nalezení známého objektu v načtené scéně.

Výhodou je, že po načtení scény lze takto pracovat (například používat transformace) s každým objektem i odděleně.

```
public void loadObjects(){
        rezistor = (Mesh) world.find(rezistorID);
        zdroj = (Mesh) world.find(zdrojID);
        zarovka = (Mesh) world.find(zarovkaID);
        vodic1 = (Mesh) world.find(vodic1ID);

        svetlo1 = (Light) world.find(svetlo1ID);
        svetlo2 = (Light) world.find(svetlo2ID);
}
```

Příklad 2: Načtení hledaných 3D objektů

Instance objektů Object3D jsou vždy přiřazeny pomocí referencí a ne hodnotou. Změny provedené v kterémkoli objektu Object3D mají tudíž okamžitý efekt. Například změny v 2D obrázku (Image2D) připojenému k pozadí (Background) budou mít bezprostřední účinek v každém odkazujícím objektu Object3D. [17]

3.4 Node

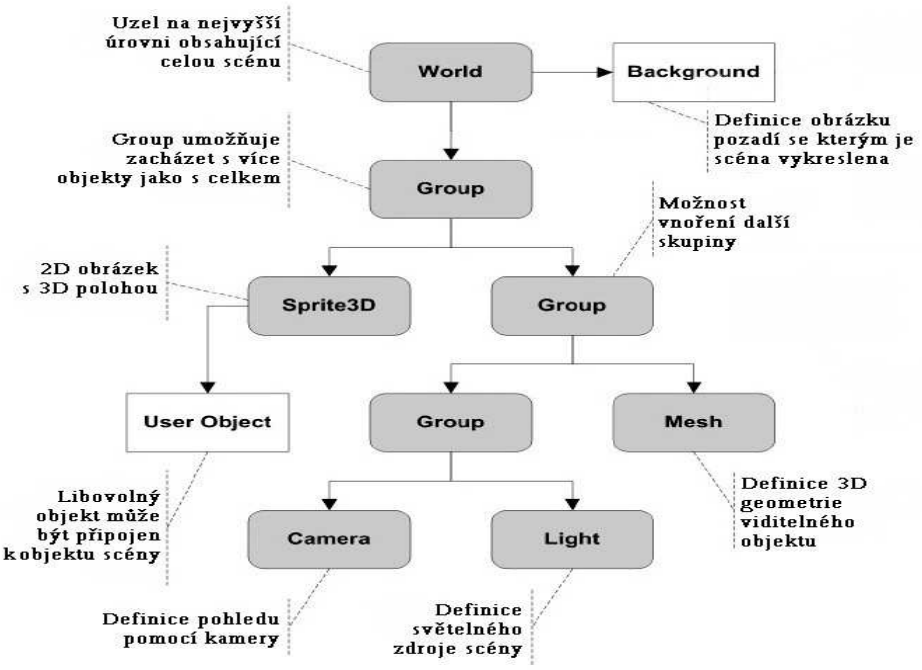
Node je abstraktní třída pro všechny uzly grafu scény. Typy těchto uzlů jsou:

- *Camera* – definuje projekci z 3D do 2D a pozici pozorovatele ve scéně
- *Mesh* – definuje 3D objekt skládající se z trojúhelníků s přiřazenými vlastnostmi materiálu.
- *Sprite3D* – definuje 2D obrázek umístěný ve 3D prostoru.
- *Light* – definuje pozici, směr, barvu a další vlastnosti světelného zdroje.
- *Group* – slouží jako kořen pro jednotlivé větve grafu scény.

Node definuje místní systém souřadnic, který se transformuje vzhledem k rodičovskému systému.

3.4.1 World

World je specifická třída odvozená od *Group* a reprezentuje uzel, který je na nejvyšší úrovni grafu scény. Graf scény je sestrojen ze struktury uzlů. V kompletním grafu scény jsou všechny uzly určitým způsobem mezi sebou propojeny pomocí společného uzlu, který je právě uzel *World.* Příklad grafu scény je ukázán na následujícím obrázku 10.



Obrázek 10: Příklad struktury uzlů grafu vykreslované scény [18]

Přestože je struktura označována jako graf, jedná se ve skutečnosti o stromovou strukturu. Z toho vyplývá, že uzel v jeden okamžik patří maximálně do jedné skupiny a cykly v grafu jsou zakázány. Pomocí oddělených metod v *Graphics3D* lze ale vykreslit části, nebo větve stromu jednotlivě. To znamená, že scéna celého grafu nemusí být vykreslena najednou. V příkladu 1 je v poli 3D objektů nalezen objekt typu *Word* a uložen do proměnné pro pozdější použití.

3.4.2 Group

Třída *Group* je uzel grafu scény, který umožňuje zacházet s objekty jako s celkem a ukládá množinu uzlů jako své potomky. Vztah rodič-potomek je obousměrný. Pokud uzel X je potomkem uzlu Y, potom Y je (jediný) rodič X. Uzel může mít v jednom okamžiku pouze jednoho rodiče, cykly jsou zakázány a kořen stromové struktury (*World*) nemůže být potomkem žádného uzlu. [18]

3.4.3 Camera

Pomocí třídy *Camera* je získána projekce ze 3D prostoru na 2D prostor displeje. Kamera používá OpenGL ořezávání, může být umístěna a nasměrována stejně jako ostatní uzly. Ve scéně lze definovat více kamer. Nastavení základních vlastností a přidání kamery do aktuálního uzlu *World* je podle následujícího příkladu.

```
public void kamera() {
    camera = new Camera();
    camera.setPerspective(60.0f,//pole pohledu
            (float) HEIGHT / (float) WIDTH,//poměr stran
            1f,//blízká rovina kamery
            100.0f);//vzdálená rovina kamery
    camera.setTranslation(-10f, 0f, 0f);//kde umístit
    camera.setOrientation(90f, 0f, 1f, 0f);//jak natočit
    world.addChild(camera);//přidání do scény
    world.setActiveCamera(camera);//kamera zvolena jako aktivní
}
```

Příklad 3: Vytvoření a nastavení nové kamery

3.4.4 Light

Jedná se o uzel grafu, který reprezentuje různé druhy světelného zdroje. Světelný zdroj lze použít k určení barvy každého objektu podle jeho materiálových vlastností. Jsou k dispozici čtyři typy světel, vyjmenované vzestupně podle složitosti výpočetního algoritmu: *ambient*, *directional*, *omnidirectional* a *spot*. Typ světla je možné kdykoli změnit, což je výhodné například při přepnutí na jednodušší světelný model, pokud vzdálenost světla od objektu přesáhne určitou mez.

3.4.5 Sprite3D

2D obrázek umístěný ve 3D prostoru v podobě třídy *Sprite3D* napomáhá zjednodušit 3D model. Tím urychluje vykreslování scény tam, kde není potřeba složitějšího 3D modelu. Proto jsou pomocné objekty často reprezentovány 2D obrázky (*sprity*). *Sprite* je zobrazen v rovině kolmé k ose vycházející z kamery. Zobrazený *sprite* může být i průhledný a otočen nebo zrcadlově převrácen. Pokud je pozměněn obrázek odkazující na daný *Sprite*, změna nastává okamžitě, čímž je možno vytvořit jednoduché animace pomocí 2D obrázků.

3.4.6 Mesh

Třída *Mesh* je definována jako struktura obsahující základní data 3D modelu a jeho vzhledu ve scéně. V podstatě se jedná o síťový model (*mesh*), neboli povrch složený z mnohoúhelníků. *Mesh* všeobecně reprezentuje neměnnou síť, kdežto odvozené třídy *MorphingMesh* a *SkinnedMesh* rozšiřují schopnost třídy *Mesh* o transformaci bodů sítě nezávazně na sobě. Síť může být složena z více pod-sítí (představující pole trojúhelníků) a jejich přiřazených vzhledů.

3.5 Transform, Transformable

Rozhraní JSR-184 používá podobné matematické vyjádření prostoru, které je popsáno v podkapitole 2.1.4 Lineární algebra v 3D grafice. Transformace jako posun, rotace a změna měřítka objektů jsou realizovány 4×4 maticemi a jejich násobením. Díky vysoko-úrovňovému API jsou však již metody pro základní transformace s pomocí násobení matic předdefinovány. Vývojáři aplikací tak mohou využít těchto připravených transformačních metod, nebo si samostatně definovat vlastní matice.

Transformace lze aplikovat na všechny objekty odvozené z třídy *Transformable* (*Mesh, Light, Camera, Group*). Výsledné transformace lze také docílit pomocí použití třídy *Transform*. Pro třídu *Transformable* jsou základní funkce *postRotate* a *translate*, pro třídu *Transform* se používá metod *postRotate* a *postTranslate* [6]. Pokud je funkce *postRotate* volána přímo na *Transformable* objekt, rotace bude provedena vzhledem k souřadnicím předka objektu. Pokud se použije třídy *Transform*, rotace objektu se provede kolem jeho vlastního středu. S funkcí *translate* je objekt posunut vzhledem ke globálním souřadicím, s použitím třídy *transform* je posun objektu proveden v souřadnicích lokálních. Je také vhodné zmínit, že u rozhraní JSR-184 je 3D prostor orientován tak, že *osa z* má kladný přírůstek souřadnic směrem ke kameře.

3.6 RayIntersection

Rozhraní JSR-184 neposkytuje detekci vzájemné kolize mezi objekty. K dispozici je třída *RayIntersection* pro nalezení objektu ve scéně pomocí paprsku. Paprsek je vyslán do 3D scény z určitého bodu **P** = (x, y) libovolným směrem. Detekován je nejbližší objekt, se kterým se paprsek střetne, společně s informacemi o jeho vzdálenosti od bodu **P**. Paprsek začíná na blízké zobrazovací rovině kamery a končí v odpovídajícím bodu vzdálené roviny kamery (viz podkapitola 2.1.3 Proces vytvoření 3D obrazu). Vzdálenost nalezeného objektu tak není určována od kamery, ale od počátečního bodu **P**. Souřadnice (x, y) bodu **P** jsou relativně vztaženy k zobrazující ploše tak, že bod (0, 0) se nachází v levém horním rohu displeje a bod (1, 1) v pravém dolním rohu. Nicméně souřadnice (x, y) nejsou omezeny jen na tento interval rozsahu. [24]

```
private void pick() {
    RayIntersection paprsek = new RayIntersection();// informace o kolizi
    // Normalizace - zobrazovaná plocha v rozmezí (0,0) - (1,1)
    float px = (float) KURZOR_X / (float) WIDTH; // WIDTH/HEIGHT
    float py = (float) KURZOR_Y / (float) HEIGHT;// šířka/výška displeje
    // Testování kolize ve všech objektech skupiny soucastky.
    boolean kolize = soucastky.pick(-1, px, py, camera, paprsek);
    if (kolize) {
        if (oznacenyMesh != paprsek.getIntersected()) {
            zobrazSipky(paprsek.getIntersected());
        }// Označeni zaznamenaného objektu šipkami
    }
}
```

Příklad 4: Detekce kolize kurzoru obrazovky a objektu 3D scény

3.7 Background, Animation

Pro nastavení pozadí se využívá třídy *Background*. Obrázek pozadí je uložen jako reference na *Image2D*. Třída *AnimationController*, *AnimationTrack* a *KeyfameSequence* slouží k vytvoření animace objektů. Protože by animovaná data z vymodelovaného M3G souboru obsadila velké množství paměti, je občas výhodnější provádět samotnou animaci pomocí transformací až ve zdrojovém kódu (*v navržené aplikaci je animace z tohoto důvodu řešena pomocí transformací*).

3.8 Graphics 3D

Zpracovávání scény pro vykreslení (rendering) je zajištěno třídou *Graphics3D*. Po vytvoření instance se musí tato třída svázat s objektem, který vykresluje zbytek zobrazovacího pole. V případě mobilního telefonu se jedná o propojení vykreslované scény s grafikou displeje. Poté už je scéna připravena na vykreslení pomocí funkce *render*.

```
public void paint(Graphics g) {
    Graphics3D g3d = Graphics3D.getInstance();//vytvoření instance
    try {g3d.bindTarget(g); //svázání
         g3d.render(world); //vykreslení
    } catch (Exception e) {e.printStackTrace();
      } finally {g3d.releaseTarget();
        }
}
```

Příklad 5: Vykreslení scény

4 Aplikace - 3D elektrické obvody

Aplikace „3D elektrické obvody“ (dále jen *3D obvody*), je určena především pro žáky základních a středních škol, kteří mají zájem o technické obory a rozvoj logického myšlení. Při jejím používání se seznamují zajímavou formou s řešením elektrických obvodů a základními výpočty elektrotechniky. Aplikace také pomáhá rozvíjet orientaci a představivost v 3D prostoru.

4.1 Pravidla 3D obvodů

Při používání *3D obvodů* je úkolem sestavit pomocí transformací předem nadefinovaných a rozmístěných součástek (vodičů, žárovek, rezistorů) elektrický obvod tak, aby všechny součástky byly správně propojeny a obvod byl uzavřený. Aplikace sleduje celkový počet otočení jednotlivých součástek a čas potřebný k dosažení správného propojení. Při správném řešení se žárovky „rozsvítí.“ Čím menší počet otočení a kratší čas, tím více bodů. Po grafické úloze ve 3D prostředí následuje teoretická otázka z elektrických obvodů, kde je možné vybírat z několika různých odpovědí. Za správnou odpověď jsou připočteny bonusové body. *3D obvody* nabízejí více možností zadání (levelů), body za jednotlivé levely se sčítají. Cílem je absolvovat všechny levely a přitom dosáhnout co možná nejvyššího součtu bodů.

4.2 Vývojové prostředí aplikace

Pro vývoj aplikace bylo zvoleno integrované prostředí NetBeans IDE 6.1 s volitelným modulem Mobility. Tento modul je optimalizovaný pro vývoj, testování a ladění aplikací pro mobilní zařízení s platformou JME a zahrnuje v sobě vlastní emulátor (virtuální mobilní telefon). Prostředí NetBeans navíc umožňuje dodatečně nainstalovat emulátory ostatních výrobců mobilních zařízení.

4.3 Vytvoření 3D modelu

3D modely použitých součástek elektrických obvodů byly navrženy v programu Blender. Modelům navržených součástek a osvětlení scény, aby mohly být v aplikaci identifikovány při načítání, byly v době návrhu přiřazeny hodnoty ID. Modely součástek mají jednoduchou strukturu, minimální počet vrcholů a ploch, aby vykreslení

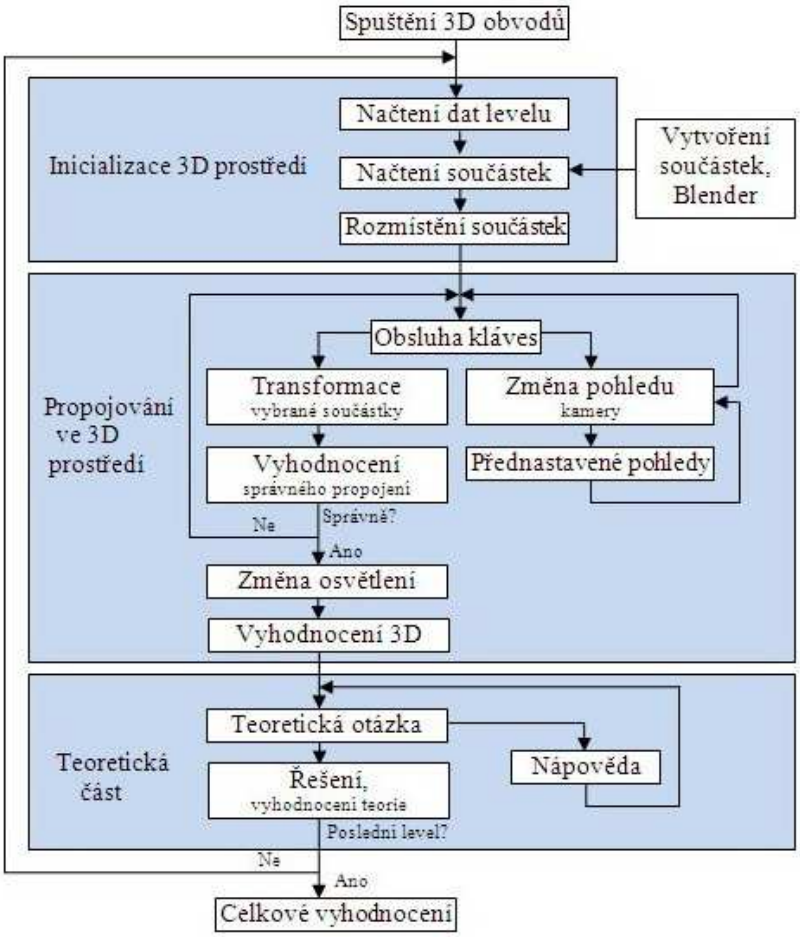
scény bylo co nejrychlejší a s co nejmenším zatěžováním procesoru mobilního telefonu. Při návrhu modelu byla také definována velikost objektů. Vodiče jsou vytvořeny tak, aby byly dobře zřetelné a snadno označitelné kurzorem. Takto nadefinované součástky byly vyexportovány do souborů formátu M3G a jsou připraveny pro další použití.

4.4 Logika aplikace

Logika 3D obvodů je sestavena ze tří hlavních funkčních bloků, které na sebe navazují:

- Inicializace 3D prostředí
- Propojování ve 3D prostředí
- Teoretická část

Tyto základní funkční bloky jsou dále rozčleněny na dílčí funkce, tak jak je znázorněno na následujícím schématu:

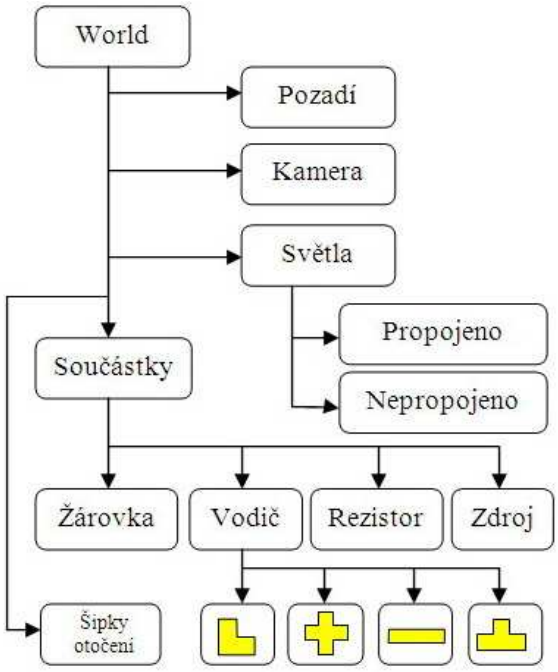


Obrázek 11: Schéma logiky aplikace 3D obvodů

4.4.1 Inicializace

Zde se provádí počáteční nastavení pro vykreslení 3D scény. Po spuštění *3D obvodů* se nejdříve ověří, zda zařízení podporuje JSR-184. Dostupnost tohoto rozhraní se určí metodou *System.getProperty("microedition.m3g.version")*, která vrací číslo verze M3G. Pokud metoda vrátí hodnotu *null*, API není k dispozici. Při pokusu o vykreslení by se scéna v tomto případě na daném zařízení nevykreslila a aplikace by nahlásila chybu. Proto bylo vytvořeno ošetření této chyby pomocí výjimky, při které je uživatel v případě nedostupnosti 3D rozhraní JSR-184 upozorněn s následným návratem do hlavní nabídky.

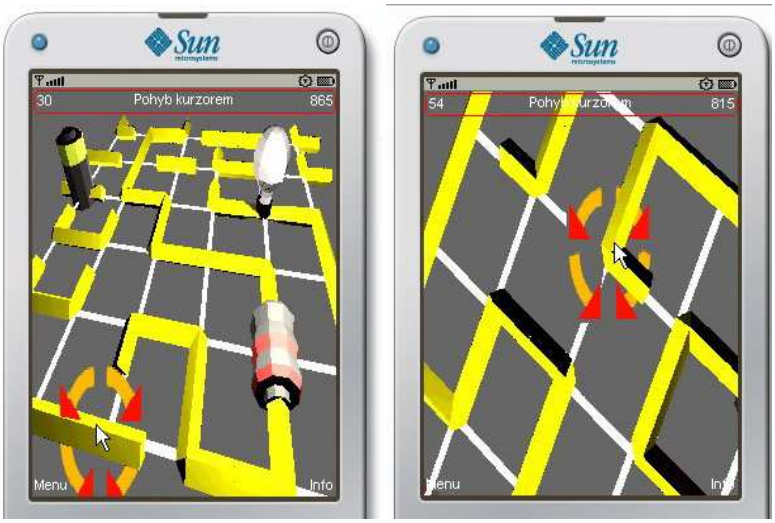
Z předem vytvořeného souboru typu M3G jsou načteny objekty v podobě součástek a světel scény. Dle daného levelu jsou potřebné součástky opakovaně použity a je tak vytvořen konkrétní elektrický obvod (například Obrázek 13). Součástky jsou na začátku náhodně pootočeny a je jim přiřazeno nové ID číslo pro pozdější identifikaci při řešení úlohy. Scéna elektrického obvodu je uložena ve stromové struktuře (obrázek 12), která se vykreslí celá najednou pomocí *Retained režimu.* Součástky jsou uspořádány do samostatné skupiny *soucastky*, která zajišťuje lepší obsluhu při jejich transformacích.



Obrázek 12: Stromová struktura scény 3D obvodů

4.4.2 Propojování obvodů ve 3D prostředí

Aktuální přechodná a výsledná požadovaná poloha součástek jsou uchovávány v maticových strukturách, které jsou v průběhu transformací vzájemně porovnávány. Pokud jsou obě matice totožné, znamená to, že elektrický obvod je správně propojen. Volba dané součástky k otočení se provádí pomocí pozice kurzoru displeje a třídy *RayIntersection*. Transformovaná součástka je označena šipkami v podobě 2D obrázku s pomocí třídy *Sprite3D* (obrázek 13). Označená součástka je k dispozici pro transformace tak dlouho, dokud není vybrána jiná.



Obrázek 13: Pomocné šipky s využitím třídy *Sprite3D*

Přiřazení prvku z maticové struktury k vybrané součástce je realizováno pomocí pozice ve 3D scéně související s jejím příslušným identifikačním číslem ID. Při transformacích pomocí třídy *Transformable* bylo nezbytné zajistit, aby všechny použité součástky simulovaly správné propojení (vyřešení otázky, kdy je součástka správně otočena).

	Úhel otočení součástky vlevo				
	0°	90°	180°	270°	360°
Vodič tvaru písmene L					
Vodič tvaru písmene I					

Obrázek 14: Vlastnosti součástek při otáčení

Po otočení o určitý úhel, je součástka opět ve funkčně stejné poloze. Tento úhel se pro tvarově různé součástky může lišit. (obrázek 14). Například pokud se provádí otáčení vodiče kolem jeho vlastní *osy z* opakovaně o úhel 90°, vodič tvaru písmene „L“ se dostane do základní (funkčně shodné) pozice až po čtyřech otočeních (úhel 360°), kdežto vodič tvaru písmene „I“ již po dvou otočeních (úhel 180°). Z tohoto důvodu je nutné rozlišovat, se kterou součástkou se provádí otáčení a není možné obecně definovat úhel, po kterém se součástka dostane do základní pozice. Analogicky je navržen i způsob řešení při otáčení (převrácení) součástek kolem *osy x* a *osy y*.

Animace pohybu součástek je realizována pomocí postupného otáčení součástky s nepatrnou časovou prodlevou během kroku pohybu. Při dosažení správného propojení obvodu následuje změna osvětlení (zapnutí a vypnutí určitých světel), což vytváří dojem, že žárovky v elektrickém obvodu se „rozsvítí“.

Obrázek 15: Rozsvícení při správném propojení obvodu

Kromě propojování součástek umožňuje aplikace volbu dvou základních pohledů kamery. Principy řešení pohybu kamery jsou obdobné s řešením transformací součástek. Kamera může být transformována i vzhledem ke globálním souřadnicím, součástky se ale mohou otáčet pouze vzhledem k vlastním (lokálním) souřadnicím.

4.4.3 Teoretická část

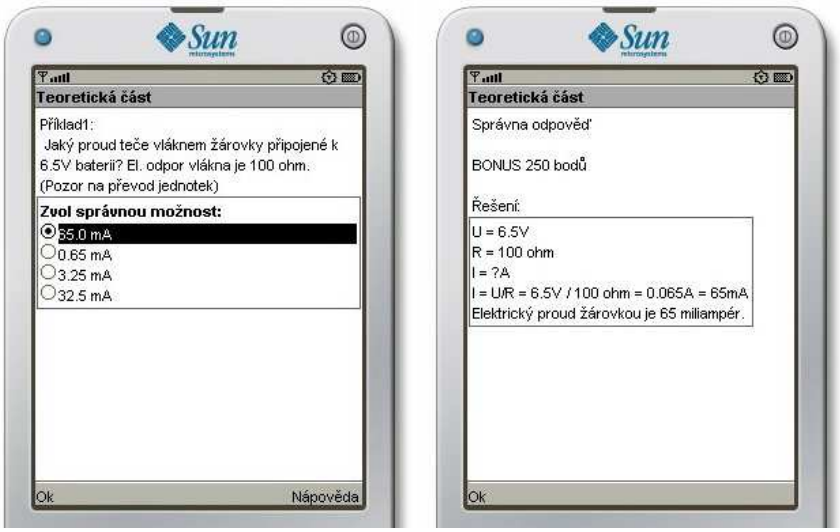
Základní vztahy pro výpočty

S ohledem na cílovou skupinu byly pro teoretické příklady vybrány tyto základní vztahy z tématiky elektrických obvodů: a) Ohmův zákon, který vyjadřuje, že elektrický

proud protékající vodičem je přímo úměrný napětí na vodiči, b) elektrický odpor je roven podílu napětí a proudu, c) výsledný odpor rezistorů zapojených za sebou je roven součtu jednotlivých odporů rezistorů, d) výsledný odpor (R) dvou rezistorů (R_1, R_2) zapojených vedle sebe je dán vztahem: $1/R = 1/R_1 + 1/R_2$. Kromě těchto vztahů jsou do příkladů zahrnuty i převody jednotek. Podrobnější popis vztahů je obsažen v uživatelské dokumentaci *3D obvodů* na webových stránkách, a také uvnitř aplikace v instrukcích *3D obvodů*. Obtížnost příkladů odpovídá znalostem žáků osmé třídy základní školy. [25]

Generování příkladů

Na výše uvedených základních vztazích z oboru elektrotechniky si žáci také procvičují násobení, sčítání a dělení. Pro každý level je jiný typ příkladu. Aby příklady nebyly identické při opakovaném spuštění aplikace, jsou pro každý příklad zvlášť generovány náhodné hodnoty. Ty jsou voleny z určitého intervalu, tak aby byla zachována reálnost příkladu. Řešení úlohy uživateli usnadňuje volba ze čtyř nabídnutých odpovědí. Správná odpověď je pouze jedna. Ostatní nesprávné odpovědi jsou generovány také ze zadaných hodnot, ale nesprávnou operací (například převrácení hodnoty, vynecháním převodu jednotek, úpravou správného výsledku koeficientem). Dále pak větší různorodost příkladů zajišťuje náhodná pozice správné odpovědi v nabídce možností. Po výběru odpovědi je zobrazena informace o správnosti volby a uveden vzorový postup řešení příkladu (viz obrázek 16).



Obrázek 16: Vygenerovaný příklad s jeho řešením

Při správné odpovědi jsou přičteny bonusové body do celkového skóre, v průběhu výběru odpovědi je k dispozici možnost náhledu do nápovědy.

4.4.4 Ostatní logika

Celkové hodnocení (skóre) závisí na:

- Počtu provedených otočení daných součástek
- Době potřebné k řešení úlohy
- Počtu správně zodpovězených teoretických otázek

Obrázek 17: Dílčí a celkové vyhodnocení

Na začátku každého levelu je počáteční vklad 2 000 bodů. Za každé otočení (převrácení) součástky je odečteno deset bodů a za každou uplynulou sekundu práce se odečítá jeden bod. Toto je navrženo z důvodu, aby se uživateli vyplatilo přemýšlet, kterým směrem součástky otáčet, namísto jen rychlého a ukvapeného otáčení. Odečítání neprobíhá do záporných hodnot. Zodpovězení teoretické otázky má nejvyšší váhu, za správnou odpověď je připočten bonus 250 bodů. Průběžná hodnota počtu otočení je zobrazována v horním levém rohu displeje, aktuální čas v sekundách je uveden v horním pravém rohu. Pokud se zvolí během propojování ve 3D zobrazení *Menu* nebo *Nápovědy*, průběžný čas se zastaví a pokračuje až po zpětném přepnutí do řešení úlohy volbou *Pokračovat*. Celkové skóre je získáno součtem bodů z jednotlivých dokončených levelů. Nejvyšší dosažené skóre se uchovává ukládáním pole bajtů pomoví RMS (Record Management System), aby bylo zachováno i při dalším spuštění

aplikace. Celkové skóre lze pomocí nabídky v menu vymazat, což může být potřebné například při pořádání soutěže. Účastníci si tímto způsobem na počátku soutěže vynulují svá skóre dosažená dříve.

Aktivace dalších levelů je zajištěna pomocí změny proměnné určující stav rozšíření (ano/ne). Změna se provede v nabídce *Rozšíření* zadáním vhodného kódu. Výhoda spočívá v tom, že pokud se uživateli již podařilo 3D obvody stáhnout a nainstalovat na jeho mobilní telefon, není již pro spuštění dalších levelů (které v průběhu testování nebyly k dispozici a jsou určeny například jen pro soutěž) potřebná nová instalace, ale pouze znalost vhodného kódu.

Zobrazení *Menu* a dalších nabídek je možné i během řečení propojovaných obvodů.

Pokud je libovolná klávesa určená pro pohyb kurzoru stisknuta déle, jeho pohyb se zrychlí.

4.5 Architektura 3D obvodů

3D obvody jsou vytvořeny pomocí 20 tříd. Většina tříd reprezentuje jednu obrazovku aplikace. Třídy *Okno.** jsou určeny pro zobrazení textu, třídy *Canvas.** pro zobrazení grafiky. Všechny třídy dohromady pak realizují výše vysvětlenou logiku *3D obvodů*. Celá aplikace je zabalena do souboru 3Dobvody.jar a její popis je obsažen v souboru 3Dobvody.jad. Následuje stručný přehled vybraných nejdůležitějších tříd:

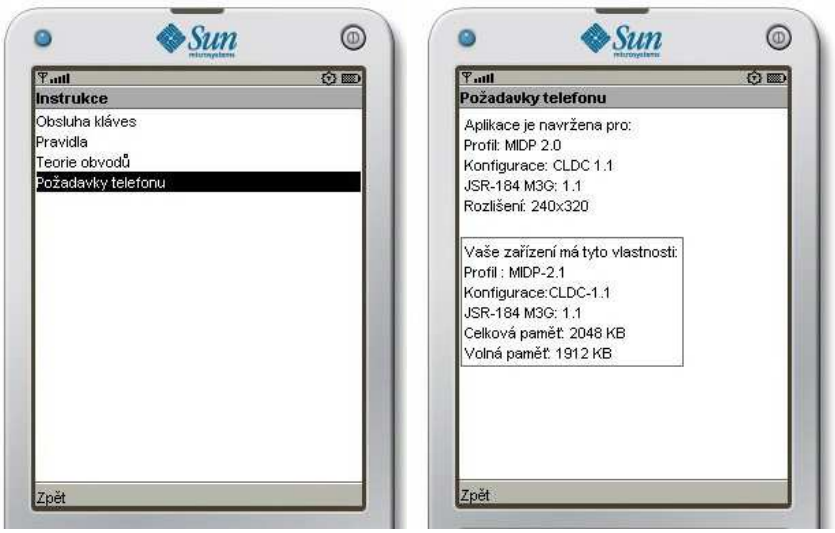
- *ObvodyMIDlet* – Hlavní třída *3D obvodů* odvozená od třídy *Midlet*. Zajišťuje běh celé aplikace, s její pomocí jsou tvořeny instance ostatních tříd týkajících se zobrazení menu, informací, nastavení, vykresleni 3D scény a vyhodnocení výsledků.
- *ObvodyCanvas* – Zde je realizována logika 3D scény, inicializace a vykreslení. Pro každý level je vytvořena nová instance této třídy. Toto zabezpečuje, aby v paměti zařízení byly pouze objekty aktuálního levelu. Pokud by se s načtením každého nového levelu pouze vytvořily další objekty součástek, mělo by to za následek zpomalení běhu celé aplikace, což bylo zjištěno v průběhu navrhování *3D obvodů*.
- *Levely* – Data pro generování 3D scén a teoretických příkladů. V podobě maticových struktur je uloženo jak úvodní náhodné rozmístění elektrických

součástek, tak i výsledné správné propojení. Dále tato třída obsahuje metody pro vygenerování náhodných hodnot teoretické části.

- *CanvasUvod* – Zobrazení úvodní obrazovky s logem projektu *Starttech* při spuštění *3D obvodů* a definice implicitní doby pro zobrazení loga. Pokud je libovolná klávesa stisknuta dříve, následuje bezprostřední přesun do hlavní nabídky. Na podobném principu je založena třída *CanvasUvodLevel* pro informační obrazovku, která je vykreslena vždy před každým novým levelem.

Obrázek 18: Využití třídy *CanvasUvod* při spuštění 3D obvodů

- *MenuList, MenuInstrukce* – Zobrazení a obsluha hlavní nabídky a nabídky instrukcí.
- *OknoTeorie, OknoTeorieKonec* – Realizace logiky pro teoretickou část 3D obvodů.
- *OknoInstrukcePozadavky* – Pomocí této třídy jsou na displej zařízení vypsány jeho základní vlastnosti (verze konfigurace CLDC, profilu MIDP a M3G a dostupnost paměti).



Obrázek 19: Submenu instrukce, požadavky na telefon

4.6 Uživatelské rozhraní

Uživatelské rozhraní vychází z výše popsané logiky 3D obvodů s tím rozdílem, že uživatel přesně neví, jak jsou funkce realizovány. Po úvodní obrazovce se objeví hlavní nabídka, kde má uživatel na výběr z několika možností (obrázek 20). Po výběru *Spustit 3D obvody* začne s řešením od počátečního levelu. Pokud uživatel v průběhu sestavování elektrického obvodu zvolí návrat do menu, v hlavní nabídce přibude možnost *Pokračovat*, která poté zajistí možnost návratu k pokračování v řešení 3D scény.

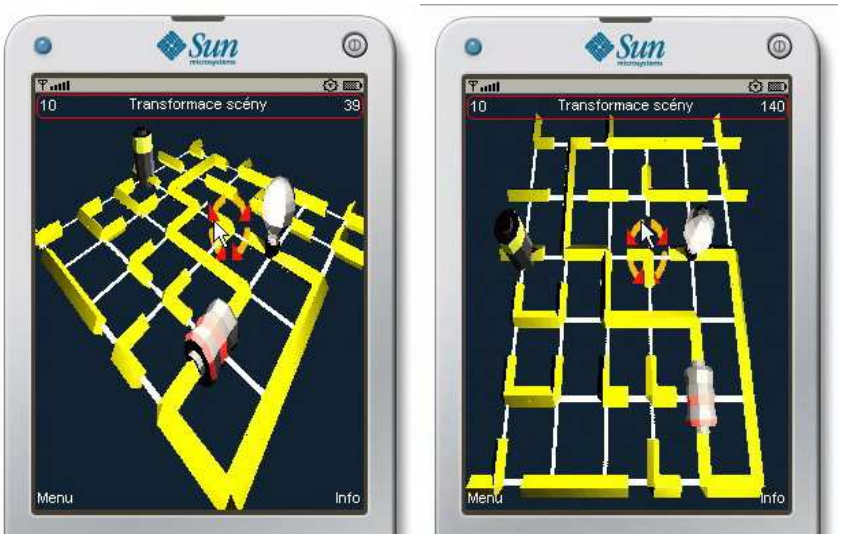
Obrázek 20: Hlavní nabídka po spuštění a v průběhu propojování 3D obvodů

Aplikace se ovládá pomocí numerických kláves nebo joysticku. Joystick nahrazuje stisknutí kláves 2, 8, 4, 6 a 7. Navržený způsob ovládání je jednoduchý a intuitivní a je prováděn následovně:

Tabulka 1: Navržený způsob ovládání *3D obvodů*

Pohyb kurzorem		Otočení součástky		Změna pohledu	
Klávesa	Funkce	Klávesa	Funkce	Klávesa	Funkce
2	Nahoru	7	Doprava (osa z)	2, 8	Rotace (osa x)
8	Dolů	9	Doleva (osa z)	4, 6	Rotace (osa y)
4	Doleva	1	Převrácení (osa x)	7, 9	Rotace (osa z)
6	Doprava	3	Převrácení (osa y)	1, 3	Posun vpřed, vzad
				5, 0	Posun dolů, nahoru

Pohyb kurzoru a otáčení součástek lze provádět současně. Kurzorem je vybrána součástka, která se zvýrazní směrovými šipkami a touto součástkou pak lze otáčet. Přepnutí mezi pohybem kurzoru a změnou pohledu se provede pomocí stisku hvězdičky (*). Základní režim „Otočení součástky a Pohyb kurzorem“ běží na šedém pozadí a nápisem v horní části displeje „Pohyb kurzorem“, pro režim „změny pohledu“ je použita černá barva pozadí a nápis „Transformace pohledu“. S pomocí stisku mřížky (#) je možné nastavení dvou základních pohledů, viz obrázek 21.



Obrázek 21: Základní Pohledy na 3D scénu elektrického obvodu

Uživatel může pomocí dostupných transformací nastavit svůj vlastní pohled na scénu (viz obrázek 13: Pomocné šipky s využitím třídy Sprite3D). V teoretické části si pak uživatel vyzkouší znalosti ze základních vztahů elektroniky (viz obrázek 16: Vygenerovaný příklad s jeho řešením)

4.7 Webové rozhraní 3D obvodů

Jako doplněk diplomové práce k aplikaci *3D obvody* byly vytvořeny webové stránky s adresou http://obvody3d.webnode.cz. Zprovoznění těchto stránek se osvědčilo z hlediska jednoduché dostupnosti a možnosti stažení uvedené aplikace. Na portálu projektu *Starttech*, http://www.starttech.cz, je odkaz na stránky 3D obvodů pomocí článku „Hrajeme si se Starttechem“ [1]. Navíc po zadání hesla „3D obvody“ ve vyhledávači Google se odkaz na vytvořené stránky v současné době (květen 2010) nachází na prvním místě. V sekci uživatelská dokumentace je uveden podrobný popis ovládání a možností hlavního menu. Ve fotogalerii jsou pro lepší pochopení zobrazeny a popsány obrázky z aplikace. Prostor pro připomínky a komentáře umožňuje zpětnou vazbu. Dále jsou zde k dispozici informace o pravidlech, teorii elektrických obvodů (základní vztahy pro výpočty), testovaných zařízeních, uskutečněných akcích, instalaci a aktualizacích *3D obvodů*. Nechybí kontakt v podobě e-mailu.

Obrázek 22: Náhled hlavní webové stránky 3D obvodů

4.8 Testování

4.8.1 Emulátory

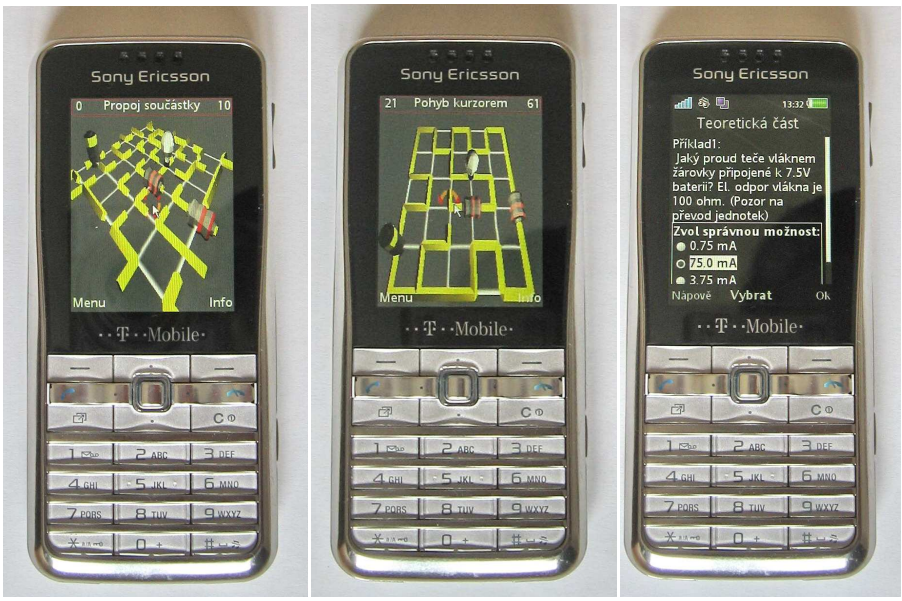
V první řadě probíhalo odstranění funkčních chyb při psaní zdrojového kódu pomocí debuggingu. *3D obvody* byly testovány na základním emulátoru (virtuální mobilní telefon v osobním počítači) modulu *Mobility*. Z tohoto emulátoru je pořízena většina obrázků této diplomové práce. Následně byly do prostředí NetBeans doinstalovány emulátory výrobců zařízení, které jsou dostupné na jejich webových stránkách, konkrétně od firem: Sony Ericsson, Nokia, LG a Samsung. Každý emulátor má k dispozici navíc různé konkrétní modely zařízení. Na počátku byl problém se spuštěním návrhu aplikace na emulátoru Nokia, problém byl však vyřešen nastavením vlastností vytvářeného projektu na profil MIDP 2.0 místo MIDP 2.1. Úspěšné testování 3D obvodů v různých emulátorech vypadalo následovně:

Obrázek 23: Simulace 3D obvodů v emulátorech LG, SE a Nokia

4.8.2 Reálné mobilní telefony

Dále byla aplikace testována na různých reálných zařízeních, nejvíce na mobilním telefonu Sony Ericsson G502. Určité problémy v reálných telefonech se všeobecně občas projevily při načítání scény elektrického obvodu při změně 3D modelu součástky (přičemž v emulátorech aplikace funguje bez problémů). Proto bylo nutné při

návrhu součástek pravidelně zálohovat aktuální verzi modelů a postupovat při jejich vývoji s malými intervaly a opakovaně návrh testovat i na reálném zařízení. Proces „upravení 3D modelu součástky, export do formátu M3G, import do projektu v prostředí NetBeans, vytvoření JAR souboru, přenos, instalace a následné spuštění v reálném telefonu“ se tak stal v jistých etapách návrhu 3D obvodů nedílnou součástí testování. V případě zjištění nefunkčnosti bylo nutné se vrátit o krok zpět. Celkově tak byl v průběhu návrhu 3D obvodů autorův mobilní telefon obohacen o více než 20 vývojových verzí zpracovávané aplikace.



Obrázek 24: 3D obvody v reálném telefonu SE G502

Následně pak byla funkční aplikace otestována i na jiných mobilních telefonech. Seznam konkrétních zařízení je uveden na webových stránkách. Spuštění aplikace bylo provedeno i na základním telefonu Nokia 3110 bez podpory 3D grafiky, kde uživatel byl upozorněn na nedostupnost rozhraní JSR-184. Stejné omezení se vyskytlo i u některých smartphonů, jelikož podporují pouze určitý virtuální stroj Javy (viz kapitola 2.2.1).

Další rozdíl v použití aplikace na reálných zařízeních oproti emulátoru je například v rychlosti zobrazování. Co proběhne v emulátoru počítače rychle a plynule, může být někdy na konkrétním mobilním telefonu pomalé a přerušované. Pro označování součástek bylo využito kolize vyhledávacího paprsku se součástkou scény (viz kapitola 3.6: RayIntersection). Byla zkoušena i verze s prohledáváním okolí

paprsku, což by mělo za následek jednodušší a rychlejší výběr součástek kurzorem. Tento způsob vyhledávání však probíhal na běžném telefonu velmi pomalu a pohyb kurzoru nebyl dostatečně plynulý, což je pro uživatele nepříjemné. Během testování na různých zařízeních se také projevila jejich konstrukční odlišnost. Mobilní telefony s výkonnějším procesorem dokážou zpracovávat a vykreslovat 3D scénu rychleji.

Při vlastním testování aplikace byla využívána příležitostná spolupráce s kolegy z autorova okolí. Takto bylo možné otestovat *3D obvo*dy na široké paletě konkrétních mobilních telefonů od různých výrobců bez nutnosti jejich zakoupení. Vzhledem k tomu, že většina kolegů také studuje technicky zaměřený obor, mobilní telefon jim není cizí a spolupráce s nimi byla poměrně přínosná.

4.8.3 Instalace 3D obvodů

Všeobecně pro každý typ mobilního telefonu se způsob instalace libovolné aplikace liší. Asi nejjednodušší způsob instalace *3D obvodů* je pomocí stažení aktuální verze z webových stránek do osobního počítače, ze kterého se aplikace nahraje prostřednictvím bluetooth nebo datového kabelu do určitého telefonu. Další možností je přímé stažení aplikace do mobilního telefonu za pomoci připojení k internetu. Například operátor T-Mobile nabízí cenu přenesených dat 6 haléřů za kB [26]. Při zkušebním stažení aplikace *3D obvodů* o velikosti přibližně 60 kB z WAP stránek, kam byla aplikace dočasně umístěna, však byl odečten kredit v hodnotě 120 Kč, místo předpokládaných 5 Kč. Tuto možnost instalace přes přímé připojení k internetu tedy nelze obecně doporučit, neboť by to mohlo být finančně náročné pro uživatele, který není plně obeznámen s tímto prostředím a nemá paušální připojení.

4.8.4 5. den s technikou na SPŠSE a VOŠ

Aplikace *3D obvody* byla oficiálně představena veřejnosti na akci „5. den s technikou na SPŠSE a VOŠ“, konkrétně skupině zájemců o programování jednočipových mikroprocesorů. Zde byl zaznamenán pozitivní ohlas. V úvodu byli všichni obeznámeni se základními funkcemi a ovládáním aplikace. Někteří účastníci této akce měli dokonce aplikaci již předem nainstalovanou ve svých telefonech z domova. Z důvodu zajištění stejných podmínek (účastníci měli různé telefony) a po dohodě se zástupcem koordinátora projektu za SPŠ, byly *3D obvody* při této akci simulovány na počítačích.

Obrázek 25: Testování 3D obvodů při „5. víkendu s technikou“

Všeobecně se potvrdilo, že většina mladých lidí používá na svém mobilním telefonu i aplikace, které nijak nesouvisejí s telefonováním, nebo psaním SMS (zábava, kalkulačka, kalendář). Teoretické otázky obsažené v aplikaci se podobají příkladům, se kterými se již zájemci (věk 14–15 let) setkali při výuce ve škole. Přesto ale bylo pro některé účastníky obtížné spočítat příklady zpaměti a určit tak správnou odpověď. Navzdory očekávání byla hojně využívána funkce nápovědy. Díky této akci bylo také zjištěno, že ovládání *3D obvodů* může být obtížné na zařízeních s dotykovým displejem.

Významným zjištěním v dané zájmové skupině také bylo, že všichni projevili svůj zájem o možnost budoucího vytváření podobných aplikací osobně.

4.9 Možnosti rozšíření

Díky třídě *Levely* je možné jednoduché rozšíření o další nové levely, nebo pozměnění stávajících konkrétních elektrických obvodů. Je také možné vymodelovat další nové elektrotechnické součástky, jako například diodu, cívku, kondenzátor, vypínač, polarizovaný zdroj a podobně. Řešení takovýchto složitějších úloh by však již u uživatele předpokládalo znalosti studenta střední školy.

Dalším zajímavým rozšířením aplikace by byla možnost vlastního sestavování elektrického obvodu uživatelem z daných součástek při dodržování určitých

technických kritérií v návaznosti na školní výuku fyziky. Rozšiřování aplikace však může mít určité hranice s ohledem na omezené vlastnosti mobilního telefonu (velikost displeje, výkon).

4.10 Výroční konference OP VK

Dne 24. března 2010 pod záštitou Evropské Unie proběhla konference Operační program Vzdělávání pro konkurenceschopnost aneb Vzdělání pro každého. Část programu nabídla bloky workshopů, v nichž byly prezentovány podporované projekty. Součásti prezentace projektu *Starttech* bylo i představení aplikace *3D obvodů*. Smyslem konference bylo informovat širokou i odbornou veřejnost o pokroku v implementaci tohoto operačního programu. Prezentace jsou volně dostupné na webových stránkách MŠMT. [27]

Závěr

Cílem diplomové práce bylo nastudovat principy zobrazování a modelování 3D scén, seznámit se s možnostmi rozhraní JSR-184 a programovacího jazyka Java Micro Edition a takto získané poznatky použít při řešení konkrétní aplikace v projektu *Starttech*.

Nejprve bylo nutné ověřit vlastnosti současných mobilních telefonů a stanovit, pro která zařízení bude navrhovaná aplikace určena. Prostorová grafika v mobilních telefonech se v mnohém svými vlastnostmi podobá grafice počítačové. Z řady rozdílných standardů pro 3D grafiku v mobilních telefonech má rozhraní JSR-184 podporu většiny předních výrobců telefonů, a proto se dá předpokládat, že právě tento standard bude široce používán v mobilních telefonech i nadále.

Celková specifikace JSR-184 je poměrně rozsáhlá, tato práce se zabývá především základními funkcemi potřebnými pro vytváření 3D grafiky. Proces vývoje aplikace zahrnoval vytvoření 3D modelů v podobě M3G souborů. Modely pro řešenou aplikaci byly vytvořeny v open-source nástroji Blender.

Výuková aplikace *3D elektrické obvody* byla začleněna do projektu *Starttech*, který je řešen v rámci grantu ESF na TU v Liberci. V rámci tohoto projektu také proběhlo její úspěšné představení a otestování uživateli. Zpracovaná aplikace je díky svému určení pro mobilní telefony a intuitivnímu ovládání obecně použitelná. Na základě provedeného testování koncovými uživateli lze konstatovat, že *3D obvody* jsou užitečnou pomůckou nejen pro cvičení logického myšlení a zopakování látky o elektrických obvodech, ale že mohou být i zajímavým podnětem pro rozšíření obzoru u zájemců o technické obory.

Seznam použité literatury

[1] *Starttech – Začni s technikou* [online]. 2009 [cit. 2010-03-02]. Dostupné z WWW: <http://www.starttech.cz/>.

[2] *Strukturální fondy EU* [online]. 2010 [cit. 2010-04-03]. Operační program Vzdělávání pro konkurenceschopnost. Dostupné z WWW: <http://www.strukturalni-fondy.cz/opvpk>.

[3] *Computer Cornell University Program of Computer Graphics* [online]. 1998 [cit. 2010-03-02]. What is Computer Graphics?. Dostupné z WWW: <http://www.graphics.cornell.edu/online/tutorial/>.

[4] ŽÁRA, Jiří, et al. *Moderní počítačová grafika*. Brno : Computer Press, 2004. 609 s.

[5] HUDEC, Bohuslav. *Základy počítačové grafiky*. Praha : ČVUT, 2007. 246 s.

[6] *Sony Ericsson* [online]. 2007 [cit. 2010-03-29]. Mobile Java 3D Tips. Dostupné z WWW: <http://developer.sonyericsson.com/>.

[7] PETERKA, Jiří. *Český mobilní sektor v roce 2009 - LUPA* [online]. 1. 3. 2010 [cit. 2010-04-04]. Český mobilní sektor v roce 2009. Dostupné z WWW: <http://www.lupa.cz/clanky/cesky-mobilni-sektor-vnbsproce-2009/>.

[8] MALIZIA, Alessio. *Mobile 3D graphics*. [s.l.] : Springer, 2006. 161 s.

[9] *Developing Mobile 3D Applications With OpenGL ES and M3G* [online]. 2006 [cit. 2010-03-29]. Course material. Dostupné z WWW: <http://people.csail.mit.edu/kapu/mobile_3D_course>.

[10] PULLI, Kari, et al. *Mobile 3D graphics: with OpenGL ES and M3G*. [s.l.] : [s.n.], 2008. 464 s.

[11] BUCHTA, Michal. *Spider.net* [online]. 15.03.2005 [cit. 2010-04-04]. Proč si koupit smartphone místo klasického telefonu?. Dostupné z WWW: <http://www.spiderer.net/15-03-2005-proc-si-koupit-smartphone.php>.

[12] *Mobilmania* [online]. 2010 [cit. 2010-03-16]. Dostupné z WWW: <http://www.mobilmania.cz/katalog-mobilu/>.

[13] *Dioné: Programovací jazyk Java* [online]. 2010 [cit. 2010-04-04]. Dostupné z WWW: <http://v1.dione.zcu.cz/java/>.

[14] *Sun Developer Network* [online]. 2010 [cit. 2010-04-15]. Java ME Sun Products. Dostupné z WWW: <http://java.sun.com/javame/overview/products.jsp>.

[15] MAHMOUD, Qusay H. *Java 2 Micro Edition*. Praha : Grada, 2002. 246 s.

[16] *The Java Community Process(SM) Program : JSRs: Java Specification Requests - platform* [online]. 2010 [cit. 2010-04-19]. Dostupné z WWW: <http://jcp.org/en/jsr/platform?listBy=1&listByType=platform>.

[17] PULLI, Kari, et al. Designing Graphics Programming Interfaces for Mobile Devices. *IEEE Computer Graphics and Applications.* 2005, 2005, November/December 2005, s. 66-75.

[18] JSR-184. *Mobile 3D Graphics API Technical Specification*. [s.l.] : Java Community Process, 2005. 280 s.

[19] BAROS, Mikael. *Sony Ericsson : 3D programming for mobile devices using M3G (JSR-184)* [online]. 2010 [cit. 2010-03-18]. Dostupné z WWW: <http://developer.sonyericsson.com/>.

[20] POKORNÝ, Pavel . *Blender - naučte se 3D grafiku*. 2., aktualiz. a rozš. vyd.. [s.l.] : BEN-Technická literatura, 2009 . 286 s.

[21] *MascotCapsule Developer Network* [online]. 2010 [cit. 2010-04-16]. MascotCapsule Developer Network - M3G. Dostupné z WWW: <http://www.mascotcapsule.com/en/m3g/index.php>.

[22] *Sun Developer Netvork (SDN)* [online]. 2004 [cit. 2010-03-20]. Getting Started With the Mobile 3D Graphics API for J2ME. Dostupné z WWW: <http://developers.sun.com/mobility/apis/articles/3dgraphics/>.

[23] HÖFELE, Claus. *IBM* [online]. 2005 [cit. 2010-03-29]. 3D graphics for Java mobile devices. Dostupné z WWW: <http://www.ibm.com/developerworks/>.

[24] *Sony Ericsson* [online]. 2005 [cit. 2010-04-19]. Picking objects in 3D space using JSR 184. Dostupné z WWW: <http://developer.sonyericsson.com/>.

[25] *Fyzikální web - ZŠ Dobrovského Lanškroun* [online]. 2006 [cit. 2010-04-12]. Fyzikální web. Dostupné z WWW: <http://www.zslado.cz/vyuka_fyzika/vyuka.html>.

[26] *T-Mobile* [online]. 1. 3. 2010 [cit. 2010-04-13]. Ceník tarifů a služeb pro tarifní a Twist zákazníky T-Mobile platný k 1. 3. 2010. Dostupné z WWW: <http://www.t-mobile.cz/FileStorage/Cenik032010.pdf>.

[27] *Ministerstvo školství, mládeže a tělovýchovy České republiky* [online]. 2010 [cit. 2010-05-07]. Výroční konference OP Vzdělávání pro konkurenceschopnost. Dostupné z WWW: <http://www.msmt.cz/strukturalni-fondy/vyrocni-konference-op-vzdelavani-pro-konkurenceschopnost>.

Přílohy

Příloha A: Hierarchie tříd rozhraní M3G [18]

- class java.lang.Object
  - class javax.microedition.m3g.**Graphics3D**
  - class javax.microedition.m3g.**Loader**
  - class javax.microedition.m3g.**Object3D**
    - class javax.microedition.m3g.**AnimationController**
    - class javax.microedition.m3g.**AnimationTrack**
    - class javax.microedition.m3g.**Appearance**
    - class javax.microedition.m3g.**Background**
    - class javax.microedition.m3g.**CompositingMode**
    - class javax.microedition.m3g.**Fog**
    - class javax.microedition.m3g.**Image2D**
    - class javax.microedition.m3g.**IndexBuffer**
      - class javax.microedition.m3g.**TriangleStripArray**
    - class javax.microedition.m3g.**KeyframeSequence**
    - class javax.microedition.m3g.**Material**
    - class javax.microedition.m3g.**PolygonMode**
    - class javax.microedition.m3g.**Transformable**
      - class javax.microedition.m3g.**Node**
        - class javax.microedition.m3g.**Camera**
        - class javax.microedition.m3g.**Group**
          - class javax.microedition.m3g.**World**
        - class javax.microedition.m3g.**Light**
        - class javax.microedition.m3g.**Mesh**
          - class javax.microedition.m3g.**MorphingMesh**
          - class javax.microedition.m3g.**SkinnedMesh**
        - class javax.microedition.m3g.**Sprite3D**
      - class javax.microedition.m3g.**Texture2D**
    - class javax.microedition.m3g.**VertexArray**
    - class javax.microedition.m3g.**VertexBuffer**
  - class javax.microedition.m3g.**RayIntersection**
  - class javax.microedition.m3g.**Transform**

Příloha B: Obsah CD

K práci je přiloženo CD, které obsahuje zdrojové kódy, zkompilovanou výslednou aplikaci *3D obvody* a Java dokumentaci aplikace. Dále se na CD nachází text této diplomové práce v elektronické podobě.