Analýza datového formátu Cabri 3D

BAKALÁŘSKÁ PRÁCE

Vedoucí bakalářské práce: Ing. Petr Vaněček, Ph.D.

Michal Moravec

VTI

2007

PODĚKOVÁNÍ

Děkuji mému vedoucímu bakalářské práce Ing. Petru Vaněčkovi, Ph.D. za hodnotné připomínky a odborné vedení v průběhu mé práce.

PROHLÁŠENÍ

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a že jsem uvedl veškeré použité zdroje.

.....

Michal Moravec

ABSTRAKT

Cílem této práce je analýza a popis datového výstupního formátu geometrického softwaru Cabri 3D – CabriML.

V druhé kapitole je obecně představen software Cabri. První podkapitolou je historie Cabri, další pak funkčnost a využití.

Třetí kapitola má za účel obeznámit čtenáře nejprve s obecným principem značkovacích jazyků, dále pak v samostatných podkapitolách stručně představit jazyky SGML a HTML. V závěru této kapitoly je věnována bližší pozornost jazyku XML.

Čtvrtá kapitola se konkrétně zabývá praktickou realizací analýzy datového formátu Cabri 3D, která je hlavním úkolem této práce. V této kapitole jsou abecedně seřazeny a popsány veškeré zjištěné tagy obsažené v CabriML.

V příloze je k nahlédnutí jednoduchá scéna z Cabri 3D a k ní odpovídající výstup formátu CabriML.

Výsledkem této práce je první dostupný popis výstupního formátu Cabri 3D a to je také jejím největším přínosem. Využití je velmi široké a to nejen v rámci projektu GREG ale i v další libovolné implementaci CabriML.

ABSTRACT

The main target of this work is to analyse and give a description.of the Cabri 3D (geometric software) output format called CabriML (Cabri Markup Language).

There is a Cabri geometric software introduction in the second chapter. The first part of the chapter is about the history of Cabri-family products and the second one about its usage.

In the third chapter, at first there is a general markup languages introduction, then parts about SGML and HTML languages and finally the part where is the detailed XML language description.

Forth chapter contains the main task of this work - complete CabriML analysis and description. There are all tags described and sorted alphabetically in this chapter.

In the appendice there is a preview of simple Cabri3D scene and its corresponding output format, which is commented.

The result of this work is the first description of CabriML ever and it's the major acquisition of this bachelor thesis. The utilization of this analysis is wide and not only for the GREG project needs, but also in other arbitrary implementation of the CabriML format.

OBSAH

1. OBECNÝ ÚVOD

Tato bakalářská práce se zabývá analýzou datového formátu Cabri 3D zvaným CabriML (Cabri Markup Language). První teoretická kapitola se bude zabývat historii a vývojem produktů Cabri, dále pak funkčností a využitím, protože se nepředpokládá obecná známost tohoto produktu. Následovat bude kapitola věnovaná značkovacím jazykům, kde bude vysvětlen obecný princip a historie těchto jazyků, krátký popis nejznámějších z nich a rozsáhlejší popis metajazyka XML, jehož je CabriML profilem. Další kapitola je již praktická a jejím obsahem je konkrétní popis, rozbor a možnosti využití datového výstupu, především pak podrobný popis jednotlivých prvků a veškerých elementů CabriML.

Hlavním přínosem této práce by měl být dosud unikátní popis výstupního formátu Cabri 3D, který by měl posloužit všem zájemcům o další využití tohoto formátu. Primárním důvodem pro výběr tohoto tématu bakalářské práce byla má spoluúčast na projektu GREG, kde bylo mým úkolem zjistit, zda vůbec nebo jak by bylo možno docílit použití scén vytvořených v Cabri 3D. Právě pro tento účel se ideálně hodí popis výstupního formátu CabriML a nyní by již neměl být problém pro mé následovníky tento výstup načíst a zpracovat.

2. CABRI

a) HISTORIE

První verze programu Cabri vznikla v roce 1986 na univerzitě Josepha Fouriera v Grenoblu ve Francii, kdy výzkumná skupina okolo matematika, počítačového vědce a průkopníka diskrétní matematiky pana profesora Jean-Marie Laborde dokončila vývoj interaktivního výukového programu zaměřeného na matematiku. Nápad vytvořit takovýto projekt vznikl v roce 1985 při přípravě Labordeho studentů na doktorát. Vývojáři historicky první

verze Cabri se tedy kromě pana Labordeho stali Philippe Cayet, Yves Baulac a Franck Bellemain. Jejich podíl na vývoji Cabri se stal součástí jejich dizertační práce a tato první verze měla označení Cabri I (Interactive Notebook). Jejich práce přinesla ovoce v podobě významné pocty v roce 1988, kdy Cabri I vyhrálo ocenění Apple Trophy, udělované každoročně největšímu úspěchu na poli výukového softwaru. V roce 1989 bylo Cabri I dostupné na francouzském trhu se vzděláním za podpory Ministerstva školství ve dvou verzích - jedna pro DOS a druhá pro MacOS.

Následovalo období klidu, které trvalo přibližně do roku 1992. Do té doby bylo Cabri I málo rozšířené a až na pár výjimek bylo Cabri známo pouze ve velmi úzkém okruhu odborníků, navíc výhradně ve Francii. Do jisté míry to bylo způsobeno také malým rozšířením dostatečně výkonných počítačů ve školách a také absencí jiných jazykových mutací. Na počátku devadesátých let minulého století se v *Institutu počítačových věd a aplikované matematiky v Grenoblu* (IMAG) kolem profesora Laborde sjednotil tým složený z řad specialistů v oborech jako počítačová věda, matematika, psychologie, umělá inteligence, didaktika matematiky a v neposlední řadě z pedagogů. Tou dobou se začalo mluvit o nástupci první verze Cabri a vývojářský tým začal s vývojem Cabri II.

Největší podíl na vzniku Cabri II měla kromě již zmiňované skupiny profesora Laborde spolupráce s firmou Texas Instruments. V roce 1994 byla druhá verze programu dokončena a nastal první rozsáhlý boom Cabri. Cabri II bylo distribuováno celosvětově a rychle si našlo dostatečně širokou skupinu uživatelů především v řadách pedagogů. Za úspěchem Cabri stál také fakt, že v roce 1995 bylo Cabri II implementováno do světově populárních grafických kalkulaček firmy Texas Instruments s označením TI-92.

Jean-Marie Laborde založil v roce 2000 v Grenoblu firmu Cabrilog a začal vývoj nových verzi Cabri – Cabri II Plus a Cabri 3D. Prvně jmenovaná

verze vyšla celosvětově v roce 2002 a Cabri II Plus poprvé proniklo do dalších odvětví matematiky, zejména do algebry a matematické analýzy. Po vydání Cabri II Plus vzniklo motto produktů firmy Cabrilog - *Cabri : Software serving pedagogy* (volně přeloženo Cabri : Software sloužící pedagogice). Tou dobou také začalo pravidelné pořádání konferencí zaměřených na Cabri s názvem Cabriworld.

Počínaje rokem 2003 byla uvedena verze Cabri Junior, která jako první přinesla na pole grafických kalkulátorů dynamickou geometrii, zejména se tato verze stala součástí modelů již zmiňované firmy Texas Instruments TI-83 a TI-84.

V září roku 2004 na konferenci Cabriworld v Římě představila společnost Cabrilog verzi Cabri II plus pro MacOS X a zároveň také uvedla úplně novou a inovativní verzi s názvem Cabri 3D. Cabri 3D veleúspěšně navázalo na své předchůdce a rozšířilo se po celém světě. Jak již napovídá označení 3D, tato verze se stala průkopníkem v oblastech prostorového zobrazování matematiky, zejména překvapující je způsob s jakou lehkostí a dynamičností se Cabri 3D vypořádává se zobrazováním matematiky a geometrie v prostoru. Nutno poznamenat, že Cabri 3D není v žádném případě pokračovatelem Cabri II, nýbrž se jedné o dva odlišné, navzájem si nekonkurující produkty.

Důkazem dobré práce týmu kolem pana Laborde je i nejnovější úspěch, Cabri 3D se stalo 10. ledna tohoto roku vítězem BETT Trophy 2007. Tato prestižní cena je udílena každoročně v 11 kategoriích v rámci vzdělávacího odvětví a výukových postupů. Tentokrát se z 340 přihlášených produktů dostalo do hlavního výběru 210, ze kterých pak porota poslala 49 do finálového výběru. Z těchto pak rozhodla devětašedesátičlenná porota na základě kritérií (např. design, poměr cena/kvalita, náročnost na uživatele,

výukový přínos, preciznost a přímočarost řešení problematiky) o vítězství Cabri 3D,

V současnosti jsou produkty firmy Cabrilog rozšířeny ve více než 35 zemích po celém světě a Cabri stále pokračuje v expanzi zlepšování výuky ve třídách a školách po celém světě. Více informací o produktech Cabri viz [1] a [2].

b) FUNKČNOST A VYUŽITÍ

V současné době jsou od firmy Cabrilog k dispozici víceméně tři verze. Jedná se o verze Cabri Junior, Cabri II Plus a Cabri 3D. O verzi Cabri Junior se příliš zmiňovat ale nebudu, protože ta existuje jen jako součást kalkulátorů firmy Texas Instruments.

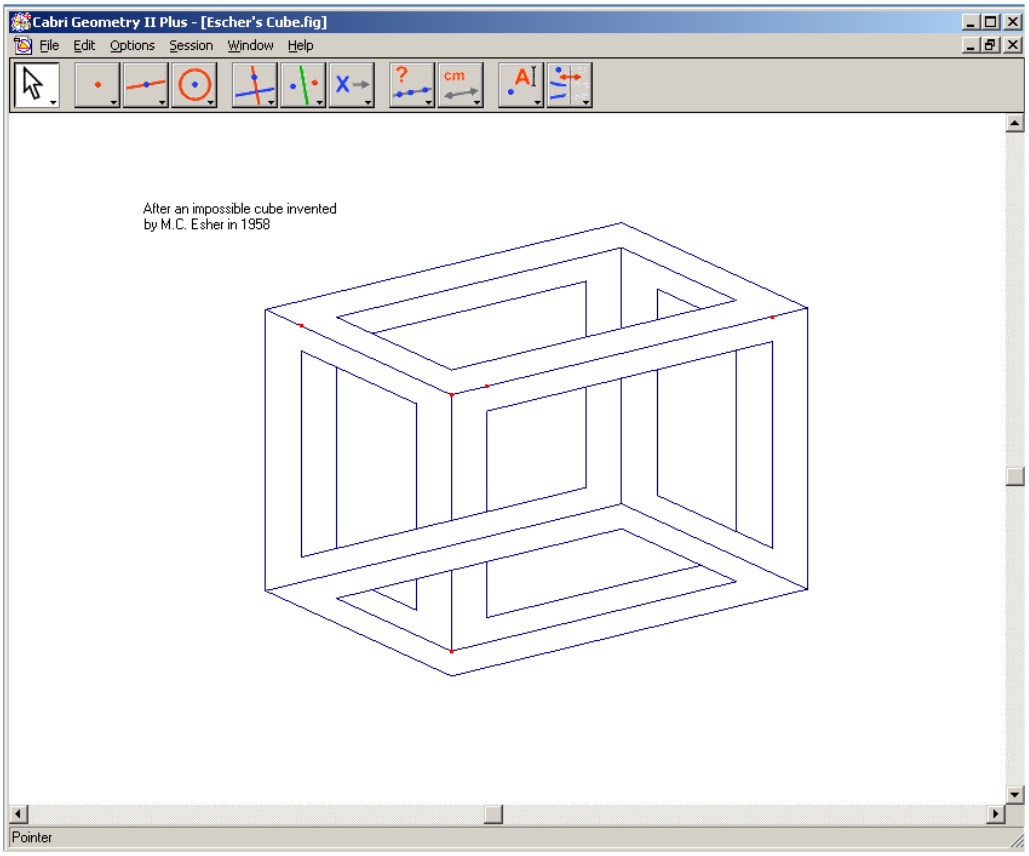
CABRI II PLUS

Další verzí je geometrický výukový program Cabri II Plus, který je jedním z programů typu DGE (*Dynamical Geometry* Environment) sloužící jako dynamické prostředí k vytváření geometrických konstrukcí na obrazovce počítače. I přes existenci dalších velice kvalitních programů dynamické geometrie má Cabri II spoustu nesporných výhod, které Cabri staví jako nejvhodnější pro výuku geometrie nejen na základních a středních školách. Svědčí o tom také fakt, že se Cabri dostalo na Seznam výukového a vzdělávacího softwaru schváleného Ministerstvem školství, mládeže a tělovýchovy pro výuku na základních a středních školách.

Mezi hlavní výhody Cabri II Plus patří:

- česká lokalizace
- kontextová nápověda při práci s geometrickými tvary
- rychlost a přesnost rýsování
- možnost počítat a graficky znázorňovat výrazy funkcí
- rychlost provádění animací
- možnost přiřazení obrázku ve formátu BMP, .JPEG nebo GIF k jednotlivým geometrickým pozicím
- možnost pojmenování všech objektů bez omezení

- výstupy do dalších aplikací (vektorové obrázky do textových editorů, tabulky do tabulkových procesorů atd.)
- subtilní rýsování, silné konstrukční nástroje (pracuje se shodnými i neshodnými zobrazeními, stopu pohybujících se objektů může otiskovat do nákresny a vytvářet tak krásné křivky, pracuje s množinami bodů, měří délky, úhly a obsahy, tyto hodnoty umí dosadit do vzorců a výsledek zanést zpět do konstrukce, lze sestrojit i pohyblivé obrázky. Pokročilý uživatel může v Cabri řídit činnost i velmi složitých virtuálních mechanizmů, a to bez znalosti programování – stačí rozumět geometrii
- malá velikost programu i souborů s vytvořenými konstrukcemi
- dostupnost velkého množství metodických a dalších materiálů pro učitele, pokračující výzkum na českých vysokých školách (např. Jihočeská univerzita viz [3]) i široké mezinárodní nasazení Cabri ve výuce v řadě zemí Evropy i v USA.
- již řadu let se pomocí tohoto programu připravují budoucí učitelé matematiky na Pedagogických fakultách Univerzity Karlovy, Jihočeské univerzity, Západočeské univerzity a dalších
- Společnost Cabrilog neustále pracuje na zdokonalení možnosti výuky geometrie jednotlivými upgrady svých produktů včetně vývoje nových produktů



Obr. č. 1 Ukázka proslulé Escherovské krychle vytvořená v Cabri II Plus. Obrázek je součástí examples Cabri II Plus

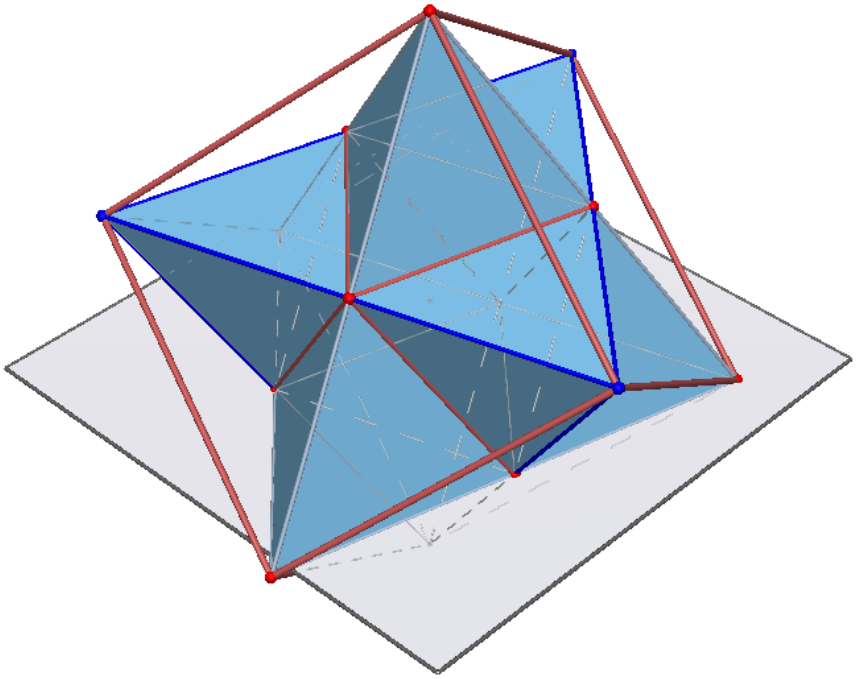
CABRI 3D

„ „Otec“ Cabri, profesor Laborde, kdysi pronesl větu: „Pohyb (nebo též eventualita, možnost pohybu) přidává do dynamické geometrie další dimenzi.“ Cabri II (Cabri II Plus) přinesla tuto dimenzi svými nástroji dynamiky.

Cabri 3D přináší nyní další rozměr, třetí prostorový rozměr, protože nákresna je prostorová a konstrukční prvky bez výjimky také. Na druhou stranu v Cabri 3D z pohledu předchozího citátu ubyl jeden rozměr, rozměr pohybu.

Lze tedy říci, že Cabri 3D nepřináší další dimenzi, přináší jinou dimenzi než Cabri II.“

[9] VANÍČEK, Jiří, *Cabri 3D-Cesta do další dimenze?*. 2 konference Užití počítačů ve výuce matematiky, České Budějovice, listopad 2005. Dostupný z WWW <http://www.pf.jcu.cz/cabri/cabri3d/dalsi_dimenze.pdf>.



Obr. č. 2 Takovéto věci není v Cabri 3D problém udělat

Mezi hlavní výhody Cabri 3D patří:

- česká lokalizace
- kontextová nápověda při práci s geometrickými tvary
- možnost pojmenování všech objektů bez omezení
- výstupy do dalších aplikací (např. pomocí TeléCabri)
- možnost vyučovat trojrozměrnou geometrii a to zejména díky komplexnosti designu v perspektivách a modelech, které by jinak zabraly mnoho času zpracováním.
- výstupní formát CabriML

3. ZNAČKOVACÍ JAZYKY

a) ÚVOD

Každý existující jazyk se skládá z určitých symbolů. Když je smysluplně sestavíme, můžeme vyjadřovat nebo předávat různé informace. Existují určitá pravidla, která nám říkají, jak dané symboly poskládat do slov, vět či odstavců tak, aby informace byla srozumitelná i ostatním. To jistě známe například z gramatiky.

Podobně je tomu i s počítačovými jazyky. Nadefinujeme jeden speciální jazyk, metajazyk, který vymezuje pravidla a symboly jiných jazyků. Pomocí metajazyka jsou přesně určena obecná pravidla pro definici jazyků a podle nich pak jeden, či více konkrétních jazyků nadefinován. Metajazyky pomáhají ve vytváření automatizovaných agentů, které zobrazují nebo zpracovávají obsah dokumentu. Vhodnou ukázkou metajazyku jsou značkovací jazyky.

Značkovací jazyk (Markup Language) je jazyk, jehož zdrojový text obsahuje současně jak vlastní text, tak instrukce pro jeho zpracování. Ty se zpravidla vyskytují v podobě příkazů (commands) či značek (tags). Zdrojovým textem bývá obyčejný ASCII soubor, což umožňuje jeho snadnou editaci i nejjednoduššími textovými editory.

Typickým rysem značkovacích jazyků jsou znaky se speciálním významem. Ty slouží k vymezení řídicích konstrukcí - příkazů či značek. Hlavní výhodou značkovacích jazyků je to, že díky otevřenému textovému formátu nevyžadují speciální programové vybavení pro svou editaci.

Specializované programy či alespoň adaptace některých výkonnějších editorů ale jejich editaci podstatně usnadní a zpříjemní. Dokumenty v těchto jazycích lze snadno strojově generovat. Nástroje pro jejich zpracování většinou bývají k dispozici zdarma a s otevřeným zdrojovým kódem. Na druhé straně

jejich použití vyžaduje určité základní znalosti, nutno ale poznamenat, že křivka učení bývá poměrně strmá. Na rozdíl od WYSIWYG (*What you se is chat you get*) nástrojů není možné si ke značkovacímu jazyku sednout a bez znalosti jeho příkazů či značek jen tak experimentovat. Informace získány z [4] a [5].

Mezi značkovacími jazyky lze vymezit dvě základní skupiny.

JAZYKY POPISNÉ (DESKRIPTIVNÍ)

Jejich konstrukce slouží k popisu, k čemu jsou informace obsažené v dokumentu. Typickými představiteli jsou XML nebo HTML. Jejich prostřednictvím lze například vymezit na stránce nadpis, odkud kam sahají jednotlivé odstavce či popsat odkaz na jinou stránku. Je ponecháno na zpracovávajícím programu, jak s těmito informacemi naloží a jak je promítne například do zobrazení dokumentu.

JAZYKY VÝKONNÉ (PROCEDURÁLNÍ)

Obsahují i výkonné instrukce na úrovni programovacího jazyka - typicky určitou formu pamětí či proměnných a nástroje pro přiřazování a využívání jejich hodnot. Výkonné jazyky zpravidla také umožňují velmi detailně popsat vizuální charakteristiky výstupu. Uživatel může tedy velmi přesně řídit vzhled výsledného dokumentu. Mezi procedurální jazyky patří TeX nebo PostScript. K demonstraci vyjadřovací síly prvního z nich v něm byl dokonce vytvořen interpreter jazyka BASIC, přestože se jedná o typografický program.

b) JAZYK SGML

SGML (*Standard Generalized Markup Language*) je univerzální značkovací metajazyk, který umožňuje definovat značkovací jazyky jako své vlastní podmnožiny. SGML je komplexní jazyk poskytující mnoho značkovacích syntaxí, ale jeho složitost brání většímu rozšíření. Jedná se o standardní jazyk určený k formálnímu popisu struktury dokumentů.

Počátkem 70. let vyvinula firma IBM značkovací jazyk, který měl usnadnit přenositelnost dat. Základem byl jazyk GML (*Generalized Markup Language*) vytvořený v roce 1969. Teprve roku 1986 byl jazyk SGML standardizován normou ISO-8879 (organizace ANSI – *American National Standardization Institute*).

Vznikl v rámci projektu ODA (*Open Document Architecture*). Cílem ODA je poskytnout standardní architekturu pro vytváření, předávání, uchovávání a zpracování různorodých dokumentů v elektronické podobě. Zahrnuje proto různé standardy pro formáty dat, architekturu předávání zpráv, zabezpečení informací atd. Pro potřeby projektu ODA bylo zapotřebí formátu umožňujícího uložení textů v elektronické podobě a zároveň nezávislého na softwarové a hardwarové platformě. A přitom aby tento formát poskytoval dostatečnou flexibilitu. Jediným možným řešením se stal značkovací jazyk (*Markup Language*).

SGML je velmi vhodný pro popis vzájemného vztahu dat a hlavně pro následné strojové zpracování. Díky výše uvedené složitosti se používá jen

v systémech pracujících s velkými objemy dat, kde se možnosti tohoto jazyka plně uplatní. Programy, které s ním pracují, jsou značně složité a hardwarově náročné. Tento fakt by brzdil rozvoj internetu a zpomalilo vývoj aplikací a prohlížečů webových stránek. Proto se hledal jiný jazyk, který by byl jednodušší než SGML, byl přístupný pro široké spektrum vývojářů

a neodrazoval od použití. Tímto jazykem se stal následně HTML, později pak XML.

„Jazyk SGML je skutečně hodně obecný, samozřejmě umožňuje definici vlastních značkovacích jazyků (sad značek a jejich vzájemných vztahů) pomoci tzv. definic typu dokumentu (DTD). Navíc má spoustu volitelných parametrů počínaje maximální délkou názvů značek a konče určením znaků použitelných jako oddělovače značek od textu. Komplexnost standardu SGML poněkud zbrzdila jeho praktické využití. Velkou podporu pro SGML znamenalo americké ministerstvo obrany, které od svých dodavatelů vyžadovalo dokumentaci pravě ve formátu SGML. Důvod byl zřejmý, bylo třeba, aby dokumentace byla použitelná v poměrně dlouhém období. Nebylo tedy možné použít nějaký proprietární formát textového procesoru, který se každých par let mění.“

[6] KOSEK, Jiří, *XML pro každého, podrobný průvodce*. vyd. Grada Publishing 2000, 164 s., ISBN 80-7169-860-1

VYUŽITÍ SGML

SGML není omezenou definicí značek. Nejedná se o formátovací jazyk ani o konkrétní značkovací jazyk. Je to specifikace, která umožňuje vytvářet vlastní značkovací jazyky. Tento metajazyk umožňuje definovat vlastní sady elementů a jejich vzájemné vztahy pomocí tzv. definic typu dokumentu (DTD).

Původně to měl být snad jediný značkovací metajazyk, jelikož je v něm možno nadefinovat víceméně všechno, od klínového písma, přes čínské písmo až třeba po jazyk XHTML. Je však příliš obecný a má tak velmi komplexní šíři záběru, že s ním normální člověk nemůže ani pracovat. Z těchto důvodů byly postupně vytvořeny i další značkovací

jazyky, s nimiž se pracuje lépe, protože mají omezenější, zároveň však postačující možnosti.

Je zcela otevřeným standardem nezávislým na hardware nebo aplikacích. Soubory SGML jsou ukládány jako prostý text ve formátu ASCII.(*American Standard Code for Information Interchange* – Americký standardní kód pro výměnu informací, osmibitová abeceda tvořící základ pro většinu dnes používaných abeced pro kódování písmen a číslic).

Hlavní význam SGML se projeví v okamžiku, kdy se začnou používat příslušné značkovací příkazy. Jimi je definována struktura a vnitřní vztahy v dokumentu. Narozdíl od nestrukturované informace otevírá zcela nové možnosti zpracování, publikování a opakovaného používání této informace.

Je vhodný pro projekty, kde se vyžaduje velké množství podobně strukturovaných údajů (katalogy, manuály, seznamy, přepisy, statistické přehledy apod.).

Jak již bylo naznačeno, největší výhodou systémů využívajících SGML je jejich otevřenost. Používá se standardu, který je neměnný, a proto nehrozí další náklady finančních prostředků v důsledku nutnosti přechodu na nové verze. Při vytváření SGML dokumentů se zajímáme pouze o obsah a nikoliv o jejich formu. Tu určíme až podle potřeby v závislosti na požadovaném výstupu. Znamená to, že stačí změnit pouze pravidla formátování a dokument je připraven pro výstup (například na tiskárnu, libovolnou aplikaci nebo webový server).

c) JAZYK HTML

V roce 1989 spolupracovali Tim Berners-Lee a Robert Caillau na propojeném informačním systému pro CERN, výzkumné centrum fyziky poblíž Ženevy ve Švýcarsku. V té době se pro tvorbu dokumentů obvykle používal TeX, Postscript a také již zmiňovaný SGML. Berners-Lee si uvědomoval, že potřebují něco jednoduššího a v roce 1990 byl tedy navržen jazyk HTML (*HyperText Markup Language*) a protokol pro jeho přenos v síti - HTTP (*HyperText Transfer Protocol*).

V roce 1991 CERN zprovoznil svůj web. Současně NCSA (*National Center for Supercomputer Applications*) vybídlo Marca Andreessena a Erica Binu k vyvinutí prohlížeče Mosaic. Byl vyvinut v roce 1993 pro počítače PC a Macintosh a měl obrovský úspěch. Byl to první prohlížeč s grafickým uživatelským rozhraním. Došlo k velkému rozvoji webu a bylo nutné pro HTML definovat standardy.

VERZE JAZYKA

- verze 0.9 – vydána zhruba v roce 1991. Nepodporuje grafický režim
- verze 2.0 – zachycuje stav jazyka v polovině roku 1994. Standard vydala komunita IETF (*Internet Engineering Task Force*). Jedná se o první verze, která odpovídá syntaxi SGML. Přidává k původní specifikaci interaktivní formuláře, podpora grafiky.
- verze 3.2 – vydána 14. ledna 1997 a zachycuje stav jazyka v roce 1996. Připravovaná verze HTML 3.0 nebyla nikdy přijata jako standard, protože byla příliš složitá a žádná firma nebyla schopna naprogramovat její podporu ve svém prohlížeči. Standard už vydalo konsorcium W3C, stejně jako následující

verze. Přidává k jazyku tabulky, zarovnávání textu a stylové elementy pro ovlivňování vzhledu.

- verze 4.0 – vydána 18. prosince 1997. Do specifikace jazyka přidala nové prvky pro tvorbu tabulek, formulářů a nově byly standardizovány rámy. Tato verze se snaží dosáhnout původního účelu - prvky by měly vyznačovat význam (sémantiku) jednotlivých částí dokumentu, vzhled má být ovlivňován připojovanými styly. Některé prezentační elementy byly zavrženy.
- verze 4.01 – vydána 24. prosince 1999. Tato verze opravuje některé chyby verze předchozí a přidává některé nové tagy. Je to poslední verze HTML, dále se nevyvíjí, nahrazeno novějším XHTML, jehož základem je právě tato poslední verze HTML.

Jak již uvedeno, jazyk HTML je od verze 2.0 aplikací SGML. Je charakterizován množinou značek a jejich atributů pro danou verzi definovaných. Mezi značky se uzavírají části textu dokumentu a tím se určuje význam obsaženého textu. Názvy jednotlivých značek se uzavírají mezi úhlové závorky ("<" a „>“). Část dokumentu uzavřená mezi značkami tvoří tzv. element (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu. Značky (také nazývané tagy) jsou obvykle párové. Rozlišujeme počáteční a koncové značky. Koncová značka má před názvem značky znak lomítko.

Ukázka zdrojového kódu HTML verze 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<html>
<!-- toto je komentář -->
   <head>
      <title>Titulek stránky</title>
   </head>
<!-- tělo dokumentu -->
   <body>
      <h1>Nadpis stránky</h1>
      <p>Toto je tělo dokumentu</p>
   </body>
</html>
```

Tento jazyk se velmi rychle rozšířil a stal se nejpoužívanějším formátem na platformě WWW. Na jeho rozšíření se výraznou měnnou podílely i samotné prohlížeče. Pokud prohlížeč příslušnému tagu rozumí, provede ho, v opačném případě ho ignoruje. Toto platí i o atributech. Pokud někde chybí koncový tag, prohlížeč se jej snaží odhadnout. Tyto skutečnosti ulehčovaly amatérským tvůrcům webových stránek jejich práci a přispěly k masovému rozšíření jazyka HTML. Na druhou stranu rychlý nástup jazyka HTML měl za následek jeho nekonzistentnost. Mnoho programátorů vytvářelo své stránky a jazyk HTML je nijak neomezoval. Takto zapsaná data jsou proto nevhodná pro strojové zpracování a není ani žádný jednotný algoritmus, pomocí kterého by bylo možno takováto data analyzovat (v mnoha případech nejsou použitelná vůbec).

Jazyk HTML původně sloužil jen jako nástroj pro psaní velmi strohých dokumentů a praxi tudíž příliš nevyhovoval. Umožňoval použití pouze několika druhů zvýraznění textu, vkládání odkazů a obrázků. Byl využíván převážně pro vědecké dokumenty. Chybělo mu totiž jakékoli zkrášlení, které dnes známe.

Mnoho tagů tohoto jazyka bylo v rámci konkurenčního boje navrženo výrobci jednotlivých prohlížečů a pouze následně byly některé z nich zahrnuty do doporučení organizace W3C o struktuře HTML. Důsledkem je nekompatibilita některých tagů v prohlížečích různých výrobců. Ještě dnes nalezneme tagy, které jsou správně zobrazeny jen v určitých prohlížečích a v jiných jsou ignorovány. Nejvýrazněji do tohoto konkurenčního boje zasáhly firmy Netscape (Netscape browser), později Microsoft (Internet Explorer) a v současnosti i Mozilla Foundation (prohlížeče Mozilla a Firefox) a Open SSL Project (Opera). Bližší informace k dispozici viz lit. [4] a [5].

d) JAZYK XML

XML (*eXtensible Markup Language*, česky *rozšiřitelný značkovací jazyk*) je obecný značkovací metajazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat.

Formální počátek tohoto jazyka můžeme datovat rokem 1998, kdy vzniklo oficiální doporučení konsorcia W3C nesoucí název W3C XML 1.0 Recommendation.

Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí. Prezentace dokumentu (vzhled) se potom definuje připojeným stylem. Další možností je pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML.

Poznámka:

World Wide Web Consortium (W3C) je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web. Cílem konsorcia je *„Rozvíjet World Wide Web do jeho plného potenciálu vývojem protokolů a směrnic které zajistí dlouhodobý růst Webu“*. W3C se také zabývá vzděláním a přístupností, vyvíjí software a nabízí otevřenou diskuzi o Webu prostřednictvím fóra. Konsorciu předsedá Tim Berners-Lee, původní autor služby WWW (World Wide Web) a primární autor specifikací URL (Uniform Resource Locator), HTTP (HyperText Transfer Protocol) a HTML (HyperText Markup Language) - základních pilířů Webu.

Původní jazyk pro publikování HTML již přestal vyhovovat především pro svou složitost, která vznikla jeho postupným (a svévolným) rozšiřováním. Jazyk XML nemá žádné předdefinované značky (tagy, názvy jednotlivých elementů) a také jeho syntaxe je podstatně přísnější, než HTML.

Důvodů je hned několik. Jedním z nich je paradoxně i samotný jazyk HTML. Jazyk HTML se neustále rozšiřoval o nové a nové prvky, skripty a rozšíření, které vývojářům umožňovaly komplexnější manipulaci s daty ovšem za cenu „nestandardního” kódu. Takto rozšířený kód se z důvodu korektního zobrazení ve všech prohlížečích rozrostl a stal složitým natolik, že spousta atributů je víceméně zbytečná a podporována pouze některými prohlížeči.

Jiným důvodem je využívání dat z internetu. Protože data nemají definovanou strukturu, je jejich převod z formátu HTML obtížný a nelze použít univerzální převodní programy do jiných formátů (např. databází). Pokud by data byla ve formátu XML, tento převod by byl bezproblémový. XML lze tedy považovat za použitelný kompromis mezi složitým jazykem SGML (vhodným pro strojové zpracování) a uživatelsky přívětivým jazykem HTML.

Snahou jazyka XML je využít kladné vlastnosti jazyka SGML a vytvořit nástroj, který nebude komplikovaný a přesto bude výkonný. XML je typem formátu, který nám říká, jak můžeme data zapsat i spolu s jejich významem.

Jazyk XML je od počátku definován a vyvíjen jako závazné doporučení upravující detailně syntaxi i možná další rozšíření jazyka. Jeho hlavní filosofií je co nejširší přenositelnost, z čehož vyplývá úplná nezávislost na použitém programovacím jazyce a na platformě. XML je jazyk, který se nebude používat a ani nepoužívá jako náhrada HTML (od toho tu je XHTML).

Podstatnou novinkou jazyka XML je logické uspořádání dat. Využívá se hlavně tam, kde je třeba elektronicky zpracovávat data či hledat nějaké informace. Důvodů, proč logicky uspořádávat data na internetu je mnoho. Jedním z nich je například možnost snadnějšího a přesnějšího vyhledávání dat, neboť u textových údajů bude definována i jejich logická struktura. Nebudeme se tudíž muset spoléhat na fulltextové vyhledávače, jejichž pomocí nalezneme všechny stránky, které obsahují hledaný text, ať už má s námi hledaným významem něco společného nebo ne.

Díky své filosofii co největší nezávislosti, kráčí např. ve stopách jazyka Java, má velké ambice stát se univerzálně použitelným formátem nejen pro přenos WWW dat mezi počítači, ale také např. mezi mobilními telefony a jinou účelověji zaměřenou technikou. Díky možnosti tvorby vlastních značek si může najít spoustu uplatnění téměř kdekoli. Stačí totiž, aby se vyrábějící firmy dohodly na svém vlastním typu dokumentu (tzv. DTD), kterému XML poskytne své mantinely, co se syntaxe týče a nový formát může vyrazit do světa. Výhodou takového formátu pak je, že se v podstatě řídí stejnými pravidly jako ostatní dokumenty, ačkoli používá vlastní značky.

Nespornou výhodou XML je také jeho snadná strojová zpracovatelnost. Ta je dána tím, že jazyk XML je už více zaměřen na strojové zpracování, než jazyk HTML a proto také obsahuje daleko důkladnější omezení, co se syntaxe kódu týče. Zatímco totiž analyzátor XML kódu na PC může být poměrně masivní program o velikosti řádově megabytů, analyzátor v telefonu či rádiu takové možnosti asi jen tak brzo mít nebude.

Z toho důvodu musí být XML, aby mohl být použitelný i zde, mnohem přísnější a strojově „příjemnější“ než HTML. Díky této snadnosti může být XML zajímavý i pro výrobce elektroniky, která by pomocí poměrně jednoduché logiky mohla s tímto formátem pracovat a být jím například i řízena.

V důsledku by XML mohl být oním jazykem, který by v domácnostech zprostředkovával úsporný provoz spotřebičů (např. řízení vytápění, ventilace apod.). V oblasti počítačů se už dnes pomocí dalších specializovaných verzí přenáší data mezi počítači, zprostředkovává elektronický obchod nebo třeba komunikuje mezi uživateli (např. komunikační klient Jabber).

V podstatě by mohl existovat jediný jazyk, který by měl spoustu speciálních odnoží, založených však na stejném jádře, čímž by byly tyto jazyky do jisté míry srozumitelné i ostatním zařízením pracujícím s XML.

XML dokumenty lze rozdělit na dvě skupiny - datově orientované dokumenty a dokumentově orientované dokumenty.

Datově orientované XML dokumenty se nazývají XML dokumenty, které plní funkci obálky pro přenos dat nebo manipulaci s nimi, která sebe sama popisuje. Datově orientované XML dokumenty se vyznačují tím, že se nejmenší jednotky dat nacházejí na úrovni hodnot typu PCDATA a atributů.

Obvykle nezáleží na pořadí sourozeneckých elementů. Datově orientované XML dokumenty jsou většinou ukládány do (objektově) relačních databází nebo jsou z těchto databází generovány. Jedním z typických použití je přenos tabulek z relačních databází bez ztráty informace o vzájemných relacích. Jako další příklady použití datově orientovaných XML dokumentů lze uvést např. objednávky, faktury, různé seznamy, vědecká a technická data atd.

Ukázka struktury datově orientovaného XML dokumentu, např. možnost správy kontaktů

```
<kontakt>
    <id>1</id>
    <jmeno>Michal</jmeno>
    <prijmeni>Moravec</prijmeni>
    <telefon>381231468</telefon>
    <mobil>604123456</telefon>
    <email>michal@moravec.cz</email>
</kontakt>
```

Dokumentově orientované XML dokumenty se nazývají XML dokumenty, které mají méně pravidelnou nebo nepravidelnou strukturu a data v těchto dokumentech nejsou rozdělena na tak malé části jako v případě datově orientovaných XML dokumentů. Dokumentově orientované XML dokumenty mají velmi často smíšený obsah, ve kterém obvykle záleží na pořadí, v jakém jsou jednotlivé elementy uvedeny. Tyto dokumenty bývají často psány ručně přímo ve formátu XML nebo jsou do něj exportovány z různých textových

formátů. Jako příklady dokumentově orientovaných XML dokumentů lze uvést např. knihy, články nebo emaily.

Ukázka struktury dokumentové orientovaného XML dokumentu

```
<kniha id="123456">

    <nazev>Kniha o knize</nazev>

    <nadpis>Uvod /nadpis>

    <strana>1</strana>

    <text> Uvodem bych rad řekl ze tato kniha
je o knize o které je tato kniha. O cem je tato
kniha to nevi ani sam autor</text>

  </kniha>
```

BLIŽŠÍ POPIS

Pro pochopení principu XML je nutno rozlišovat navzájem XML File – soubor obsahující posloupnost znaků, které definují XML soubor a XML Document – což je abstraktní model dat z tohoto souboru, který je umístěn v paměti. XML dokument je výsledkem analýzy XML souboru. Je možné, aby několik rozdílných XML souborů definovalo tentýž XML dokument (stejnou logickou strukturu). XML dokument je vlastně strom, který má jeden kořenový element – ROOT ELEMENT, a několik dalších uzlů, které jsou poduzly kořenového uzlu. Z nejobecnějšího pohledu musí každý z uzlů být jedním z následujících typů:

- znak
- zpracovávající instrukce
- komentář

- element

Znak uzel obsahuje jeden znak a nemá žádnou jinou strukturu. V XML jsou povoleny všechny znaky ze sady UNICODE/ISO-10646. Uzel zpracovávající instrukce má dvě pole: jméno (také nazývané cíl – PI target) a obsah - skládající se ze sekvence znaků. Uzel komentář obsahuje jedinou položku komentář, která obsahuje posloupnost znaků.

Uzly ZNAK, PI, a COMMENT nemají potomky, to znamená, že jsou to vždy listy ve stromě XML dokumentu. Jejich rodičovským uzlem je vždy uzel typu ELEMENT. Ve skutečnosti, XML dokumenty musí obsahovat alespoň jeden element, kterým je kořenový element dokumentu. Uzly ELEMENT jsou více komplexní. Nejenom, že mohou mít potomky, ale také mají jména (něco co je nazýváno jako „generický identifikátor“) a sadu atributů. Atributy jsou páry dat typů: klíčové_slovo - hodnota. Každé klíčové slovo (tj. vlastně jméno atributu) se může v jednom elementu vyskytnout maximálně jednou.

Elementy mohou mít nula nebo více dětí, ty jsou uspořádány. Lze tedy např. lze najít pátého potomka daného uzlu (tohoto faktu se využívá jak v DOM tak i při práci s odkazy v XML). Toto je v kontrastu s atributy elementu, které jsou pojmenovány, ale ne seřazeny, takže nelze určit, v jakém pořadí atributy po sobě následují. Element, který nemá další potomky (má 0 dětí) je nazýván prázdný element.

Element také může obsahovat jisté schéma. Toto schéma uspořádání vlastních prvků má ve specifikaci XML 1.0 možnost mít jen kořenový element (DOCUMENT). Schéma, které je dáno tomuto elementu je pak aplikováno na všechny jeho podprvky (podstromy) tj. v XML na celý dokument. Více informací viz [7].

Rozdělení na skupiny, které bylo uvedeno výše je však pouze abstraktní a prakticky, z hlediska programování v XML lze XML dokument dělit na

následující součásti: elementy, entity, reference (odkazy), komentáře, zpracovávající instrukce, značkované sekce a definice typů dokumentu (DTD).

Ukázka fotogalerie vytvořené pomocí XML

```
<galerie>
    <jmeno>Fotogalerie hub</jmeno>
    <popis>Muj les</popis>
    <album>
    <nazev>Hriby</nazev>
    <autor>Michal</autor>
    <foto>
    <nazev>Hrib_smrkovy</nazev>
    <url>01.jpg</url>
    <datum>2006-07-16</datum>
    <autor>Michal</autor>
    <technika>
    <fotoaparát              typ="digital">Kodak
75</fotoaparat>
    <optika>Kodak CR18-60G</optika>
```

```
        </technika>

        </foto>

        </album>

    </galerie>
```

Tato jednoduchá ukázka XML souboru, ukazuje možnost zpracování například fotogalerie. To ovšem není až tak důležité jako demonstrace vlastní struktury tohoto dokumentu. Jazyk XML, jak už je zmíněno výše, si klade za cíl uspořádat data souboru do nějakého logického celku. Jeho smyslem je logicky uspořádat data tak, aby byla přenositelná mezi jednotlivými počítači ale i platformami či dokonce zařízeními.

Všimněme si, že ve smyslu XML, jsou všechna důležitá data označkována. Tak např. kolem slova „Hrib_smrkovy“ je značka NAZEV, udávající, že se jedná o název fotografie. Pokud bychom tedy měli nějaký program (parser), který by z tohoto kódu vždy vybral označkovaná data, mohli bychom jednoduše dostat např. tabulku s pořízenými fotografiemi.

Název	Datum	Autor	Fotoaparát
Hrib_smrkovy	16. 07. 2006	Michal Moravec	Kodak 75

Na ukázce je jasně vidět, proč je XML tak dobrým nástrojem. Lze totiž data jednoduše označit a pomocí jednoduchých programů pak lehce strojově zpracovat. Nyní se pokusím vysvětlit jednotlivé části XML dokumentu:

ZPRACOVÁVAJÍCÍ INSTRUKCE

Pomocí těchto instrukcí se dají zpracovávající aplikaci poskytnout potřebná data. Vykonávající instrukce nejsou (zrovna tak jako komentáře) textovou součástí dokumentu, přesto jsou ale XML procesorem odesílána zpracovávající aplikaci. Zpracovávající instrukce musí mít následující formát:

```
<?jmeno data?>
```

Jméno, nazývané taky PI target (PI cíl), identifikuje vykonávající instrukci aplikaci. Aplikace by měla zpracovat pouze ty instrukce, kterým rozumí a ignorovat všechny ostatní. Jména instrukcí začínající znaky XML jsou rezervována pro další verze XML, a proto by se v uživatelských dokumentech neměla vůbec vyskytovat.

V našem příkladě je použita konkrétní zpracovávající instrukce v tomto tvaru:

```
<?xml version="1.0" encoding="windows-1250"?>
```

Protože je to standardní velmi často používaná a důležitá instrukce, povězme si o ní něco blíže. Jako první příkaz v programu se objevuje řádek

```
<?xml version="...
```

Tento příkaz oznamuje zpracovávajícímu klientovi, že se jedná o soubor typu XML, proto se také někdy označuje jako prolog. Tato instrukce má následující syntaxi:

```
<?xml version="xx.xx" encoding="..." standalone="yes|no"?>
```

Parametr version je vyžadován, přičemž současná verze XML se označuje jako 1.1 - pod tímto označením je verze oficiálně schválena konsorciem W3C.

Parametr *encoding* je sice volitelný, avšak pokud používáte znaky národních abeced, je důležitý. Bez specifikace kódování jsou totiž diakritické znaky považovány za neznámé a jako takové budou působit potíže v sekcích označených jako CDATA (viz dále). Proto se doporučuje specifikaci kódování používat. Více o kódování lze zjistit na webové stránce [http://cs.wikipedia.org/wiki/Znakov%C3%A1_sada]. Současná běžně používaná česká kódování:

- ISO 8859-2 Latin-2, Východoevropský
- Windows-1250 Latin-2, Středoevropský (podobný na ISO8859-2)
- UTF-8 kódování unicode s nejmenší kódovou jednotkou délky osm bitů. Nezávisí na tom, zda je systém „little-endian“ nebo „big endian“, je kompatibilní s ASCII.

Jazyk XML je postaven a plně podporuje 16. bitové kódování typu UNICODE. To je také důvod, proč mají některé programovací jazyky jako např. Perl s podporou XML potíže. Je totiž třeba nejdříve upravit jádra těchto jazyků, aby podporovala 16. bitová kódování. Z tohoto pohledu je nejlepší oporou XML jazyk Java, který je už od počátku svého vývoje koncipován pro kódování UNICODE, tj. nemá s podporou těchto nových znakových sad používaných v XML žádné problémy. Podrobnější popis dostupný v [8].

Parametr *standalone* je volitelný a udává, zda se daný soubor odkazuje na nějaké další (externí) zdroje, jako např. další soubory XML, formátovací styly XSL či CSS apod. Pokud soubor stojí "sám o sobě" (je tzv. standalone, tj.

má všechny informace v jediném souboru) pak má parametr standalone hodnotu 'yes'. Jinak má hodnotu 'no'. Ukázka:

```
<?xml version="1.0" encoding="windows-1250" standalone="yes"?>
```

ELEMENTY

Každá značka je vlastně částí dokumentu, je jeho elementem. Proto se pro značky používá tento název, ELEMENT je klíčové slovo jazyka XML. Značky či elementy, se zapisují stejným způsobem jako v HTML. Je to samozřejmě dáno tím, že oba tyto jazyky mají stejného rodiče (SGML) a proto mají i téměř shodnou syntaxi.

Název značky je uzavřen mezi znaky **< >**. Každá uživatelsky vytvořená značka se v XML bere jako párová, každé dvojici **< >** musí odpovídat dvojice **< />** znaků. Syntaxe značek je tudíž následující:

```
<název_značky> ...data... </název_značky>
```

Zatímco v HTML existovaly i nepárové značky, jako např. BR nebo HR, v XML se i nepárové značky ukončují. Vzniká tak tzv. prázdná značka (empty tag). Např. IMG značka by v XML podání vypadala takto:

```
<IMG SRC="obrazek.gif"/>
```

Vše je opět podřízeno snadné zpracovatelnosti, tj. i nepárová značka musí být ukončena znaky **/>** aby analyzátor snadno poznal, že se jedná o ukončení značky. Značky si v XML vytváří sám programátor. Jediné omezení, které je mu kladeno je pravidlo, že značky mohou být složeny pouze ze znaků písmen, číslic a podtržítka, přičemž musí začínat podtržítkem nebo písmenem.

Elementy tvoří interface, neboli rozhraní, jak přistupovat k jednotlivým částem dokumentu. Tím, že se určitá část dokumentu obklopí značkou, stává se z této části dokumentu obsah elementu, kterým je ohraničen.

- Pro jakoukoli část dat v souboru lze vytvořit element, kterým je část označena.
- Značky do sebe lze libovolně vnořovat.
- Musí se zachovat správné vhnízdění značek, aby byla zachována logika vnoření, tj. značky se nesmí překrývat.

Zatímco HTML, potažmo analyzátory tohoto jazyka (prohlížeče) zpracují zápis typu:

```
<u><b>text</u></b>
```

pravidla XML takové konstrukce neumožňují. Konkrétně značka <*b*> je vhnízděna do značky <*u*>, ovšem při ukončení </*u*> je značka <*b*> stále otevřená. To je v rozporu s pravidly XML a analyzátory nahlásí chybu. Je nutno opravit zápis do následující správné formy:

```
<u><b>text</b></u>
```

Další odlišností od HTML je nutnost veškeré argumenty uzavírat do uvozovek. Jak je vidět, jazyk XML je mnohem přísnější než HTML. V důsledku nutí XML programátora k mnohem ucelenějšímu programovacímu stylu a většímu pořádku kódu jak je uvedeno v [6].

KOMENTÁŘE

Stejně jako v jazyce HTML a spoustě ostatních jazyků, má i jazyk XML možnost vkládání komentářů. Komentáře se v XML značí stejně jako v HTML tj. komentář je vložen mezi sekvence znaků **<! -- a -->.**

Uvnitř komentáře se mohou nacházet jakékoli znaky, význam speciálních znaků **<**, **>**, **?**, **&** je v komentářích potlačen. (To proto, že analyzátory XML úseky komentářů nezpracovávají). Jediné, co by se v komentáři nemělo objevit je dvojice znaků '**--**'. Takovou dvojici by totiž analyzátor mohl považovat za ukončení komentáře, tak jak je definován v SGML. Komentář může být ve zdrojovém kódu umístěn kdekoli vyjma vnitřků značek. Komentář také nebude fungovat ve značkované sekci CDATA (, kde se bude považovat za součást kódu.

OVĚŘENÍ PLATNOSTI (VALIDITA)

Nejčastějším problémem dokumentů je ověření platnosti. Jazyk HTML to řeší ověřením kódu na straně prohlížeče (klienta či zpracovávající aplikace). Ovšem jak už bylo zmíněno, existují různá rozšíření, které ten či onen prohlížeč nepodporuje, takže problém platnosti dokumentu přetrvává dál a to co je pro jeden prohlížeč platné, to druhý prohlížeč vůbec nemusí umět načíst.

V XML si podobné věci nemůže programátor dovolit. Musíme totiž předpokládat, že náš kód bude zpracovávat i jednoduchá logika, která bude dokument načítat jako platný, jinak dojde k nepředvídatelné chybě. V XML se díky použití DTD neboli šablon dokumentu situace poněkud komplikuje. Díky DTD lze totiž rozlišovat dva stavy správnosti XML:

- správná utvořenost (well-formed document) – dokument je správně utvořen, pokud jeho syntaxe odpovídá výše popsaným pravidlům, tj. správné vnoření, správný zápis značek apod.
- platnost (valid document) – toto je vyšší stupeň správnosti XML souboru. Platný může být pouze dokument, který obsahuje šablonu neboli DTD, podle které se ověří, jestli uspořádání dokumentu odpovídá této šabloně, na kterou se dokument odvolává. Pokud tomu tak je, je dokument platný (valid), jinak může být maximálně dobře

utvořen (well-formed) a to za předpokladu, že je syntakticky správně zapsán.

- Alternativou k použití DTD je tzv. XML schéma – XSD (*XML Schema Definition).*

SHRNUTÍ

Vzhledem k potřebám dnešní doby, kdy je potřeba používat nějaký jednoduchý otevřený formát, který není vázán s platformou nebo technologií se jeví XML jako nejvhodnější kandidát.

„Práci s XML usnadňuje i to, že celý formát je založen na obyčejném textu. I když pro většinu lidi zůstane kód XML skryt a budou ho používat pouze aplikace pro vzájemnou komunikaci, není problém kdykoliv otevřít XML dokument v libovolném textovém editoru a pár potřebných úprav provést ručně. Použití textového formátu může někomu připadat jako zbytečné plýtvaní místem. Dnes se však mnohem větší důraz klade na srozumitelnost a snadnou práci s daty, jestli ušetříme par kilobajtů paměti, již nikoho příliš netrápí. Navíc většina protokolů pro síťovou komunikaci (včetně protokolu HTTP používaného na Webu) umožňuje zcela transparentně pro potřeby přenosu data komprimovat a u příjemce zase dekomprimovat do původní podoby.“

[6] KOSEK, Jiří, *XML pro každého, podrobný průvodce*. vyd. Grada Publishing 2000, 164 s., ISBN 80-7169-860-1

4. ANALÝZA DATOVÉHO FORMÁTU – PRAKTICKÁ ČÁST

Dříve než se začnu zabývat konkrétní analýzou CabriML bych rád uvedl, že jednotlivé značky, které jsou uzavřeny mezi úhlové závorky ("<" a „>“) budou pro účely popisu nazývány tagy. Informace uvnitř tagů umístěné budou nazývány hodnoty. Popis jednotlivých tagů bude vysvětlován pomocí ukázek a možných hodnot. Jelikož CabriML je profilem jazyka XML, všechny

tagy jsou párové, až na jedinou výjimku a tou je specifikace verze XML. Popis tagů je řazen abecedně.

<ACTIVE_VIEW>

Tag určující aktivní pohled na scénu. Hodnota odkazuje na ID tagu <GVIEW3>, která se dále pomoci vnořeného projekčního tagu <PROJECTION> odkazuje na tag pro transformaci <TRANSFO3>.

Ukázka:

```
<active_view>%8</active_view>
```

<ACTIVE_PAGE>

Jeden z nejjednodušších tagů odkazuje na stranu, která má být znázorněna, resp. nastavena jako aktuální. Implicitně je hodnota nastavena na první stránku.

Ukázka:

```
<active_page>%1</active_page>
```

<BASE_POINT_SIZE>

Tag pro uživatelsky defaultně nastavenou velikost grafické reprezentace bodu. Konkrétní hodnota se nastavuje v uživatelském prostředí Cabri 3D a neukládá se do CabriML. Tento tag má pouze výchozí hodnoty s názvem příslušné entity (např. vector3_base_point_size, subline3_base_point_size atd.). Každý uživatel může mít jinak nastavené výchozí velikosti bodů u přímek než u vektorů atd. Pro potřeby dalšího zpracování CabriML je tento tag víceméně zbytečný.

Ukázka:

```
<base_point_size>vector3_base_point_size</base_point_
size>
```

<BASE_CURVE_RADIUS>

Tag analogický k <BASE_POINT_SIZE>, s rozdílem, že se jedná o nastavení tloušťky křivky.

<BASE_SURFACE_THICKNESS>

Tag analogický k <BASE_POINT_SIZE>, hodnoty pro tloušťku povrchu.

<BITMAP>

Tag pro zobrazení náhledu. Má v sobě jediný vnořený tag <PNG_DATA>, který obsahuje pomocí BASE64 kódovaný obrázek ve formátu PNG.

Ukázka:

```
<bitmap>
    <png_data>
    iVBORw0KGgoAAAANSUhEUgAAAFoAAAB/CAYAAABvydWHAAAG
    +0lEQVR4nO3dS09j5xnA8f+xnA8f+xDQZz
    </png_data>
</bitmap>
```

<CIRCLE_BY_THREE_POINTS>

Tag definující kružnici zadanou třemi body. Povinné tagy jsou <ID>, <IN_POINTS> a <OUT_CONIC>.

Ukázka:

```
<circle_by_three_points3>
    <id>%22</id>
```

```
    <out_conic>%23</out_conic>
    <in_points>%18 %20 %24</in_points>
</circle_by_three_points3>
```

<CLIPPING>

Tento tag vyjadřuje typ clippingu (ořezávání). Využití a možnosti nebyly nalezeny. Jediná známá je výchozí hodnota DEFAULT_CLIPPING_TRANSFO3.

Ukázka:

```
<clipping>default_clipping_transfo3</clipping>
```

<CONIC3>

Tag určující grafickou reprezentaci kuželosečky. Povinné tagy jsou <ID>, <VALUE>, <GSTATE>, <POINT_STYLE>, <POINT_SIZE>, <POINT_COLOR>, <GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_POINT_SIZE> a <BASE_CURVE_RADIUS>.

Ukázka:

```
<conic3>
    <id>%23</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>orange</curve_color1>
```

```
    <base_curve_radius>conic3_base_curve_radius</bas
e_curve_radius>
</conic3>
```

<CONIC_BY_FIVE_COPLANAR_POINTS3>

Tag definující kuželosečku zadanou pěti body (ohnisková definice). Podle posloupnosti a rozložení bodů se jedná buď o kružnici, elipsu, parabolu nebo hyperbolu. Povinné vnořené tagy jsou <ID>, <OUT_CONIC> a <IN_POINTS>.

Ukázka:

```
<conic_by_five_coplanar_points3>
    <id>%34</id>
    <out_conic>%35</out_conic>
    <in_points>%26 %28 %30 %32 %36</in_points>
</conic_by_five_coplanar_points3>
```

<CURVE_COLOR>

Tag pro reprezentaci barvy křivky po najetí na ni myší. Na výběr je 135 barev, které se dají vybrat z palety přímo v uživatelském rozhraní Cabri 3D. Všech 135 barev má své slovní vyjádření, takže kromě běžných RED, BLUE atd. se jedná např. o ROSY_RED, LIGHT_CYAN, DARK_OLIVE_GREEN apod.

Ukázka:

```
<curve_color>steel_blue</curve_color>
```

<CURVE_RADIUS>

Jelikož je zde díky 3D prostředí jakákoliv křivka reprezentována jako jakási hůlka, tento tag definuje tloušťku křivky. Na výběr jsou opět jen předdefinované VERY_SMALL_POINT_SIZE (velmi malá velikost), SMALL_POINT_SIZE (malá velikost), NORMAL_POINT_SIZE (normální velikost), LARGE_POINT_SIZE (velká velikost) a VERY_LARGE_POINT_SIZE (velmi velká velikost).

Ukázka:

```
<curve_radius>small_curve_radius</curve_radius>
```

<CURVE_STYLE>

Tímto tagem se určuje styl, jak má vypadat křivka. Možné styly jsou předdefinovány. Tag může mít tyto hodnoty NO_CURVE_STYLE (žádná čára), SOLID_CURVE_STYLE (plná čára), DASH_CURVE_STYLE (čárkovaná čára), DOT_CURVE_STYLE (tečkovaná čára) nebo DASH_DOT_CURVE_STYLE (čerchovaná čára).

Ukázka:

```
<curve_style>dash_dot_curve_style</curve_style>
```

<CYLINDER_BY_AXIS_AND_POINT3>

Tag definující válec podle osy a bodu. Povinné tagy jsou <ID>, <IN_LINE>(odkazuje na osu), <IN_POINT> a <OUT_QUADRIC>.

Ukázka:

```
<cylinder_by_axis_and_point3>
    <id>%18</id>
    <in_point>%20</in_point>
    <in_line>%15</in_line>
```

```
    <out_quadric>%19</out_quadric>
</cylinder_by_axis_and_point3>
```

<DISPLAY_SCALE>

Tento tag znázorňuje měřítko zobrazení přímo v uživatelském rozhraní. Hodnoty mohou nabývat pouze číselných hodnot. Hodnota 1.0 znamená zobrazení 1:1. Hodnoty pod 1.0 zmenšují, hodnoty nad 1.0 zvětšují.

Ukázka:

```
<display_scale>4.000</display_scale>
```

<DOCUMENT>

Tento tag víceméně vymezuje dokument a obsahuje všechny zde popsané tagy, které jsou potřebné pro popis konkrétní scény, vyjma tagu určujícího verzi XML.

Ukázka:

```
<document>
    <display_scale>1.0</display_scale>
</document>
```

<FREE_CRYSTAL_BALL3>

Tag definující vlastnosti kamery. Povinné vnořené tagy jsou <ID>, <AZIMUTH>, <ELEVATION>, <IN_CENTER> a <OUT_TRANSFO>. Tag <AZIMUTH> udává azimut (směr) kamery ve stupních, <ELEVATION> elevaci (sklon) kamery také ve stupních. Tag <IN_CENTER> má pouze defaultní hodnotu ORIGIN3 (počátek) a <OUT_TRANSFO> se odkazuje na danou transformaci.

Ukázka:

```
<free_crystal_ball3>
     <id>%6</id>
     <elevation>30.000</elevation>
     <azimuth>30.000</azimuth>
     <in_center>origin3</in_center>
     <out_transfo>%7</out_transfo>
</free_crystal_ball3>
```

<FREE_POINT_IN_SPACE>

Víceméně identický tag jako <POINT3> na který se i odkazuje. Jeho existence znamená osamocený bod, který v době uložení výstupu není vázán k jinému objektu. Pakliže se později stane součástí nějakého objektu jeho definice je zachována a rozdíl od <POINT3> není žádný.

Ukázka:

```
<free_point_in_space3>
     <id>%13</id><
     position>(3,0,1)</position>
<out_point>%12</out_point>
```

<FREE_POINT_ON_LINE>

Analogický tag k <FREE_POINT_IN_SPACE> s tím rozdílem, že bod leží libovolně na přímce.

<FREE_POINT_ON_PLANE>

Analogický tag k <FREE_POINT_IN_SPACE> a <FREE_POINT_ON_LINE> s tím rozdílem, že bod leží libovolně v rovině.

<GRAPHIC_LAYERS> NEBO <LAYERS>

Tag reprezentující grafické vrstvy. Cabri 3D nabízí dvě vrstvy jednu výchozí, která je vidět a druhou skrytou. Možné hodnoty jsou tedy dvě – DEFAULT_LAYER (výchozí viditelná) a HIDDEN_LAYER (skrytá).

Ukázka:

```
<graphic_layers>hidden_layer</graphic_layers>
```

<GSTATE>

Tento tag je součásti grafické reprezentace objektů. Výchozí hodnota je DEFINED u všech objektů. Existuje ještě hodnota DEFINED_NO_HIDDEN, která se používá u objektů, které jsou ve skryté vrstvě (layer), ale mají být zobrazeny.

Ukázka:

```
<gstate>defined</gstate>
```

<GVIEW3>

Tento tag je poměrně bohatý na vnořené tagy, a jeho úkolem je definovat pohled na scénu. Povinné vnořené tagy jsou <ID>, <X>, <Y>, <HEIGHT>, <WIDTH>, <SURFACE_CLIPPING>, <PROJECTION>, <OPTICS>, <LAYERS> a <CLIPPING>.

Ukázka:

```
<gview3>
     <id>%8</id>
     <y>1.500</y>
     <x>0.500</x>
     <height>18.000</height>
     <width>19.000</width>
     <surface_clipping>default_surface_clipping_trans
fo3</surface_clipping>
     <projection>%7</projection>
     <optics>central_medium_lens</optics>
     <layers>default_layer</layers>
     <clipping>default_clipping_transfo3</clipping>
</gview3>
```

<HEIGHT>

Rozměrový tag pro výšku. Výška je udávána v centimetrech. Používá se například pro definici výšky stránky nebo pohledu.

Ukázka (nastavení výšky strany formátu A4):

```
<height>29.700</height>
```

<ID>

Tento tag slouží pouze k identifikaci objektu, transformace případně pohledu kamery a případnému následnému odkazování na něj. Hodnota tohoto tagu může být víceméně jakákoliv – např. Bod1. Veškeré geometrické figury musí tento tag povinně obsahovat. Implicitně je hodnota nastavena na „%1“, přičemž s každým dalším objektem se tato hodnota zvyšuje o jedničku – nesmí existovat dva totožné tagy <ID>

Ukázka:

```
<point3>
       <id>Bod1</id>
       <value>(0,-4,0)</value>
 </point3>
```

Definice bodu, jehož ID je „Bod1“

```
<point3>
<id>%27</id>
<value>(0,-4,0)</value>
</point3>
```

Definice bodu, jehož ID je „%27“, což znamená, že tento bod byl vytvořen jako 27. v pořadí bez zásahu uživatele při identifikací objektů.

<IN_AXIS>

Tag vyjadřující náležitost objektu k ose otáčení.

<IN_CENTER>

Tag vyjadřující odkaz na střed objektu.

<IN_LINE/S>

Tag vyjadřující náležitost objektu k přímce/přímkám.

<IN_OBJECT/S>

Tag vyjadřující náležitost k určitému objektu.

<IN_ORIGIN>

Tag vyjadřující počátek objektu v jiném objektu, zpravidla bodu.

<IN_PLANE/S>

Tag vyjadřující náležitost objektu k rovině/rovinám

<IN_POINT/S>

Tag vyjadřující náležitost objektu k bodu/bodům.

<IN_POLYNET>

Tag vyjadřující náležitost objektu k mnohostěnu.

<IN_SUBLINE/S>

Tag vyjadřující náležitost objektu ke křivce/křivkám.

<IN_VECTOR/S>

Tag vyjadřující náležitost objektu k vektoru/vektorům.

<LINE_BY_TWO_PLANES3>

Tag definující přímku pomocí dvou rovin, přímka se nachází na průniku dvou rovin (průsečnice). Povinné vnořené tagy jsou <ID>, <IN_PLANES> a <OUT_LINE>.

Ukázka:

```
<line_by_two_planes3>
     <id>%26</id>
     <out_line>%25</out_line>
     <in_planes>rovina1 rovina2<in_planes>
```

```
</line_by_two_planes3>
```

<LINE3>

Tag definující přímku. Povinné vnořené tagy jsou <ID>, <VALUE>, <GSTATE>, <POINT_STYLE>, <POINT_SIZE>, <POINT_COLOR>, <GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_POINT_SIZE> a <BASE_CURVE_RADIUS>.

Ukázka:

```
<line3>
    <id>%23</id>
    <gstate>defined no_hidden</gstate>
    <value>(0,0,-1,0,0,0)</value>
    <graphic_layers>hidden_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>medium_aquamarine</curve_color1>
    <base_curve_radius>line3_base_curve_radius</base
_curve_radius>
</line3>
```

<LINE_OF_VECTOR3>

Tag určující definici přímky pomocí vektoru. Povinné vnořené tagy jsou <ID>, <OUT_LINE> a <IN_VECTOR>.

Ukázka:

```
<line_of_vector3>
     <id>%24</id>
     <out_line>%23</out_line>
<in_vector> vektor1</in_vector>
```

<LINE_PARALLEL_TO_LINE_BY_POINT3>

Tento tag definuje přímku procházející zadaným bodem a je rovnoběžná s jinou přímkou. Mezi povinné vnořené tagy patří <ID>, <IN_POINT>, <IN_LINE> a <OUT_LINE>.

Ukázka:

```
<line_parallel_to_line_by_point3>
     <id>%84</id>
     <out_line>%85</out_line>
     <in_point>%82</in_point>
     <in_line>%79</in_line>
</line_parallel_to_line_by_point3>
```

<LINE_PERPENDICULAR_TO_LINE_IN_PLANE_BY_POINT3>

Tag definující přímku kolmou k přímce v rovině a procházející zadaným bodem. Povinnými tagy jsou <ID>, <IN_POINT>, <IN_LINE>, <IN_PLANE> a <OUT_LINE>.

Ukázka:

```
<line_perpendicular_to_line_in_plane_by_point3>
```

```
    <id>%14</id>
    <out_line>%15</out_line>
    <in_point>%16</in_point>
    <in_line>%12</in_line>
    <in_plane>Z0_Plane</in_plane>
</line_perpendicular_to_line_in_plane_by_point3>
```

<LINE_PERPENDICULAR_TO_PLANE_BY_POINT3>

Tag definující přímku kolmou na rovinu a procházející zadaným bodem. Povinně tagy jsou <ID>, <IN_POINT>, <IN_PLANE> a <OUT_LINE>.

Ukázka:

```
<line_perpendicular_to_plane_by_point3>
    <id>%28</id>
    <out_line>%29</out_line>
    <in_point>%26</in_point>
    <in_plane>%17</in_plane>
</line_perpendicular_to_plane_by_point3>
```

<OBJECT_IMAGE_BY_AXIS_AND_POINTS_ROTATION3>

Tag definující otočení objektu kolem osy nebo bodu. Povinné vnořené tagy jsou <ID>, <IN POINTS>, <OUT_OBJECT>, <IN_AXIS> a <IN_OBJECT>.

Ukázka:

```
<object_image_by_axis_and_points_rotation3>
    <id>%31</id>
    <in_points>%22 %13</in_points>
    <out_object>%32</out_object>
    <in_axis>%23</in_axis>
    <in_object>%29</in_object>
</object_image_by_axis_and_points_rotation3>
```

<OPTICS>

Tag znázorňující použitou čočku kamery. Tento tag má zřejmě jen výchozí hodnotu CENTRAL_MEDIUM_LENS. Jiné hodnoty nebyly objeveny.

<OUT_CONIC3>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci kuželosečky.

<OUT_QUADRIC>

Tag určený k odkazování nějakého objektu na grafickou reprezentaci kvadratické plochy.

<OUT_LINE>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci přímky.

<OUT_OBJECT>

Tag určený k odkazování nějakého objektu na grafickou reprezentaci jiného libovolného objektu. Slouží k předávání, proto je obecný.

<OUT_PLANE>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci roviny.

<OUT_POINT>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci bodu.

<OUT_POLYNET>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci mnohostěnu.

<OUT_SUBLINE>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci křivky.

<OUT_SUBPLANE>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci plochy.

<OUT_VECTOR>

Tag určený k odkazování nějakého objektu na určitou grafickou reprezentaci vektoru.

<PAGE>

Definiční tag pro stránku (nákresnu). Povinné vnořené tagy jsou <ID>, <NUMBER>, <HEIGHT>, <WIDTH> a <VIEWS>. Tag <ID> je zřejmý, <NUMBER> definuje číslo strany, <HEIGHT> a <WIDTH> velikost a <VIEWS> odkazuje na použité pohledy.

Ukázka (strana A4):

```
<page>
	<id>%1</id>
	<number>1</number>
	<height>29.700</height>
	<width>21.000</width>
	<views>%8</views>
</page>
```

<PLANE3>

Tag reprezentující vzhled roviny. Tento tag i jeho vnořené tagy jsou nepovinné, ale závisí na nich případný vzhled – daná rovina sice bude existovat, ale nebude vidět. Mezi vnořené tagy patří <ID>, <GSTATE>, <SURFACE_STYLE>, <SURFACE_THICKNESS>, <SURFACE_COLOR>, <GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_SURFACE_THICKNESS> a <BASE_CURVE_RADIUS>.

Ukázka:

```
<plane3>
	<id>%17</id>
	<gstate>defined</gstate>
	<value>(0,0,-1,0)</value>
	<surface_thickness>normal_surface_thickness</sur
face_thickness>
```

```
    <surface_style>solid_surface_style</surface_styl
e>
    <surface_color1>cornflower_blue</surface_color1>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>small_curve_radius</curve_radius>
    <curve_color1>dim_gray</curve_color1>
    <base_surface_thickness>plane3_base_surface_thic
    kness</base_surface_thickness>
    <base_curve_radius>plane3_base_curve_radius</bas
e_curve_radius>
</plane3>
```

<PLANE_BY_THREE_POINTS>

Tag definující rovinu zadanou třemi body. Povinně vnořené tagy <ID>, <IN POINTS> a <OUT_PLANE>.

Ukázka:

```
<plane_by_three_points3>
    <id>%16</id>
    <in_points>%12 %14 %18</in_points>
    <out_plane>%17</out_plane>
</plane_by_three_points3>
```

<PLANE_PERPENDICULAR_TO_LINE_BY_POINT3>

Tag definující rovinu kolmou k zadané přímce a bodu. Vnořené tagy jsou <ID>, <IN_POINT>, <IN_LINE> a <OUT_PLANE>.

Ukázka:

```
<plane_perpendicular_to_line_by_point3>
    <id>%86</id>
    <in_point>%88</in_point>
    <in_line>%85</in_line>
    <out_plane>%87</out_plane>
</plane_perpendicular_to_line_by_point3>
```

<PNG_DATA>

Tento tag má jako hodnotu pomocí BASE64 zakódovaný náhled ve formátu obrázku PNG.

<POINT3>

Tímto tagem definujeme bod v prostoru pomocí jeho souřadnic. Takovýto bod Tento tag má povinné další dva vnořené tagy a těmi jsou tag <ID> a tag <VALUE>. Zajímavostí je, že vnořený tag <VALUE> má u tohoto tagu povinné tří hodnoty a to X, Y a Z souřadnice, může však mít i další, tzv. homogenní souřadnice. Mezi nepovinné vnořené tagy patří tagy definující vzhled. Mezi tyto tagy patří <GSTATE>, <POINT_STYLE>, <POINT_SIZE>, <POINT_COLOR>, <GRAPHIC_LAYERS> a <BASE_POINT_SIZE>. Bod, který není nijak vázán na jiný objekt je pak ještě odkazován tagem <FREE_POINT_IN_SPACE>.

Ukázka

```
<point3>
      <id>Bod1</id>
      <value>(0,-4,0)</value>
</point3>
```

Bod definovaný pouze souřadnicemi X, Y a Z.

```
<point3>
      <id>Bod1</id>
      <value>(0,-4,0,1)</value>
</point3>
```

Bod definovaný navíc s jednou homogenní souřadnicí

<POINT_COLOR>

Jeden z tagů pro grafickou reprezentaci bodu. Tento tag vyjadřuje barvu bodu po najetí na něj kurzorem myši. Možných hodnot je 135, které se dají vybrat z palety přímo v uživatelském rozhraní Cabri 3D. Všech 135 barev má své slovní vyjádření, takže kromě běžných RED, BLUE atd. se jedná např. o ROSY_RED, LIGHT_CYAN, DARK_OLIVE_GREEN apod.

Ukázka:

```
<point_color>moccasin</point_color>
```

<POINT_MIDDLE_OF_TWO_POINTS>

Tag definující bod umístěný ve středu imaginární přímky mezi dvěma dalšími body. Povinné tagy <ID>, <OUT_POINT> a <IN_POINTS>.

Ukázka:

```
<point_middle_of_two_points3>
      <id>%21</id>
      <out_point>%22</out_point>
```

```
    <in_points>%19 %13</in_points>
</point_middle_of_two_points3>
```

<POINT_SIZE>

Tag pro velikost bodu při grafické reprezentaci. Možné hodnoty jsou VERY_SMALL_POINT_SIZE (velmi malá velikost), SMALL_POINT_SIZE (malá velikost), NORMAL_POINT_SIZE (normální velikost), LARGE_POINT_SIZE (velká velikost) a VERY_LARGE_POINT_SIZE (velmi velká velikost). Zajímavostí je, že tyto názvy neodpovídají názvům v menu Cabri 3D, kde jsou velikosti stupňovány smallest-small-normal-big-biggest.

Ukázka:

```
<point_size>very_small_point_size</point_size>
```

<POINT_STYLE>

Tag určený pro grafickou reprezentaci bodu. Jeho hodnota určuje styl, jak bude bod znázorněn. Na výběr jsou pouze čtyři předdefinované možnosti – NO_POINT_STYLE (žádný), SPHERE_POINT_STYLE (koule), CUBE_ POINT_STYLE (krychle) nebo DIAMOND_POINT_STYLE (diamant – 3D kosočtverec - tetragonální dipyramida)

Ukázka:

```
<point_style>sphere_point_style</point_style>
```

<POLYGON_BY_N_COPLANAR_POINTS3>

Tento tag definuje n-úhelník pomocí n bodů ležících v jedné rovině. Povinné vnořené tagy jsou <ID>, <IN_POINTS> a <OUT_SUBPLANE>.

Ukázka:

```
<polygon_by_n_coplanar_points3>
```

```
    <id>%221</id>
    <in_points>%108 %202 %187 %204</in_points>
    <out_subplane>%222</out_subplane>
</polygon_by_n_coplanar_points3>
```

<POLYNET3>

Tag definující mnohostěn. Povinné vnořené tagy jsou <ID>, <GSTATE>, <POINT_STYLE>, <POINT_SIZE>, <POINT_COLOR>, <GRAPHIC_LAYERS>, <BASE_POINT_SIZE>, <SURFACE_STYLE>, <SURFACE_THICKNESS>, <SURFACE_COLOR>, GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_SURFACE_THICKNESS> a <BASE_CURVE_RADIUS>. Jediným volitelným vnořeným tagem je <VALUE>, který definuje a určuje mnohostěn a hodnotami jsou vrcholy, pozice a další případně potřebné. Tento tag je povinný v případě vlastního mnohostěnu. V případě použití předdefinovaného mnohostěnu, který je určen tagem <POLYNET_COPY_BY_THREE_POINTS> se hodnota <VALUE> nepoužívá a odkazovaný tag reprezentuje pouze grafickou reprezentaci.

Ukázka:

```
<polynet3>
    <id>%23</id>
    <surface_thickness>normal_surface_thickness</surface_thickness>
    <surface_style>solid_surface_style</surface_style>
    <surface_color1>crimson</surface_color1>
```

```
<point_style>sphere_point_style</point_style>

<point_size>small_point_size</point_size>

<point_color>tomato</point_color>

<graphic_layers>default_layer</graphic_layers>

<curve_style>solid_curve_style</curve_style>

<curve_radius>small_curve_radius</curve_radius>

<curve_color1>fire_brick</curve_color1>

<base_surface_thickness>polynet3_base_surface_th
ickness</base_surface_thickness>

<base_point_size>polynet3_base_point_size</base_
point_size>

<base_curve_radius>polynet3_base_curve_radius</b
ase_curve_radius>

</polynet3>
```

.

<POLYNET_COPY_BY_THREE_POINTS>

Tag označující mnohostěn uživatelsky zadaný pomocí tří bodů. Tento tag odkazuje na tag <POLYNET3>, který jej konkrétně definuje a specifikuje jeho grafickou reprezentaci. Vnořené tagy jsou <ID>, <IN_POINTS>, <OUT_POLYNET> a <IN_POLYNET>. Tag <IN_POLYNET> může obsahovat název předdefinovaných pravidelných mnohostěnů. Mezi tyto patří CANONIC_TETRAHEDRON, CANONIC_OCTAHEDRON, CANONIC_CUBE, CANONIC_ICOSAHEDRON, CANONIC_DODECAHEDRON

Ukázka:

```
<polynet_copy_by_three_points3>
    <id>%24</id>
    <in_points>%20 %22 %26</in_points>
    <out_polynet>%25</out_polynet>
    <in_polynet>canonic_octahedron</in_polynet>
</polynet_copy_by_three_points3>
```

<POSITION>

Vnořený tag u volných objektů určující pozici, která je identická jako hodnota odkazovaného objektu.

<PROJECTION>

Tag projekce s jedinou hodnotou odkazující na transformační tag <TRANSFO3>.

Ukázka:

```
<projection>%7</projection>
```

<QUADRIC3>

Tag vyjadřující grafickou reprezentaci kvadratické plochy. Mezi povinné vnořené tagy patří <ID>, <GSTATE>, <SURFACE_STYLE>, <SURFACE_THICKNESS>, <SURFACE_COLOR>, <GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_SURFACE_THICKNESS> a <BASE_CURVE_RADIUS>.

Ukázka:

```
<quadric3>
```

```
    <id>%23</id>

    <gstate>defined</gstate>

    <surface_thickness>normal_surface_thickness</sur
face_thickness>

    <surface_style>solid_surface_style</surface_styl
e>

    <surface_color1>medium_turquoise</surface_color1
>

    <graphic_layers>default_layer</graphic_layers>

    <curve_style>solid_curve_style</curve_style>

    <curve_radius>small_curve_radius</curve_radius>

    <curve_color1>dim_gray</curve_color1>

    <base_surface_thickness>quadric3_base_surface_th
ickness</base_surface_thickness>

    <base_curve_radius>quadric3_base_curve_radius</b
ase_curve_radius>

</quadric3>
```

<RAY_BY_TWO_POINTS>

Tento tag definuje polopřímku. Má povinné tagy <ID>, <IN_POINT>, <OUT_SUBLINE> a <IN_ORIGIN>. Tag <IN_ORIGIN> udává v jakém bodě je počátek polopřímky.

Ukázka:

```
<ray_by_two_points3>
     <id>%14</id>
     <in_point>%16</in_point>
     <out_subline>%15</out_subline>
     <in_origin>%12</in_origin>
</ray_by_two_points3>
```

<REF>

Tag určující uživatelské prostředí Cabri 3D. Slouží pouze k identifikaci a definici panelů nástrojů. Pro účely této analýzy, a použití CabriML jinou aplikací, naprosto zbytečný tag. Vnořené tagy <ID> a <TARGET>.

Ukázka:

```
<ref>
     <id>document_toolbar</id>
     <target>3d_geometry_toolbar</target>
</ref>
```

<SECTOR_BY_THREE_POINTS3>

Tag definující výseč pomocí třech bodů (jeden z nich je vrchol a odkazuje se pomocí <IN_ORIGIN>). Povinné tagy jsou <ID>, <IN_POINTS>, <IN_ORIGIN> a <OUT_SUBPLANE>.

Ukázka:

```
<sector_by_three_points3>
     <id>%16</id>
     <in_points>%14 %18</in_points>
```

```
    <out_subplane>%17</out_subplane>
    <in_origin>%12</in_origin>
</sector_by_three_points3>
```

<SEGMENT_BY_TWO_POINTS3>

Tento tag určuje úsečku zadanou dvěma body. Povinné vnořené tagy jsou <OUT_SUBLINE> a <IN_POINTS>. Tag <OUT_SUBLINE> odkazuje na grafickou reprezentaci křivky (je jedno jestli se jedná o úsečku, přímku atd. - při definici odkazuje na sebe sama), tag <IN_POINTS> udává mezi jakými body (jejich ID) je úsečka definována.

Ukázka:

```
<segment_by_two_points3>
    <out_subline>usecka1</out_subline>
    <in_points>A B</in_points>
</segment_by_two_points3>
```

Úsečka je definována body s ID „A“ a „B“ a odkaz na ní a její grafickou reprezentaci je přes ID „usecka1“

<SPHERE_BY_CENTER_AND_POINT3>

Tag definující kouli pomocí dvou bodů - středu a bodu na povrchu. Povinné tagy <ID>, <IN_POINT>, <IN_CENTER> a <OUT_QUADRIC>.

Ukázka:

```
<sphere_by_center_and_point3>
```

```
    <id>%22</id>

    <in_point>%24</in_point>

    <in_center>%20</in_center>

    <out_quadric>%23</out_quadric>

</sphere_by_center_and_point3>
```

<SPHERE_BY_FOUR_POINTS3>

Tag definující kouli pomocí čtyř bodů ležících na povrchu. Povinné vnořené tagy jsou tagy <ID>, <IN_POINTS> a <OUT_QUADRIC>.

Ukázka:

```
<sphere_by_four_points3>

    <id>%28</id>

    <in_points>%22 %24 %26 %30</in_points>

    <out_quadric>%29</out_quadric>

</sphere_by_four_points3>
```

<SUBLINE3>

Tento tag udává grafickou reprezentaci křivky nebo její části. Tento tag i jeho vnořené tagy jsou nepovinné, ale závisí na nich případný vzhled – daná čára sice bude existovat, ale nebude vidět. Mezi vnořené tagy patří <ID>, <GSTATE>, <POINT_STYLE>, <POINT_SIZE>, <POINT_COLOR>, <GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_POINT_SIZE> a <BASE_CURVE_RADIUS>.

Ukázka:

```
<subline3>
```

```
        <id>tb</id>
        <gstate>defined</gstate>
        <point_style>sphere_point_style</point_style>
        <point_size>normal_point_size</point_size>
        <point_color>tomato</point_color>
        <graphic_layers>default_layer</graphic_layers>
        <curve_style>solid_curve_style</curve_style>
        <curve_radius>normal_curve_radius</curve_radiu
        s>
        <curve_color>brown</curve_color>
      <base_point_size>subline3_base_point_size</base_
      point_size>
      <base_curve_radius>subline3_base_curve_radius</b
      ase_curve_radius>
</subline3>
```

<SUBPLANE3>

Analogie k <SUBLINE3> s tím rozdílem že se jedná o libovolnou část plochy a její grafickou reprezentaci.

<SURFACE_CLIPPING>

Analogie k tagu <CLIPPING> s tím rozdílem, že se jedná o clipping povrchu objektů. Jediná známa hodnota je výchozí DEFAULT_SURFACE_CLIPPING_TRANSFO3.

<SURFACE_COLOR>

Tag vyjadřující barvu povrchu objektu, v němž je tag vnořen. Možných hodnot je 135, které se dají vybrat z palety přímo v uživatelském rozhraní Cabri 3D. Všech 135 barev má své slovní vyjádření, takže kromě běžných

RED, BLUE atd. se jedná např. o ROSY_RED, LIGHT_CYAN, DARK_OLIVE_GREEN apod.

Ukázka:

```
<surface_color>ivory</surface_color>
```

<SURFACE_TYPE>

Tag určující typ povrchu. Tento tag může nabývat následující hodnoty – NO_SURFACE_STYLE, NORMAL_SURFACE_STYLE, SMALL_DOT_SURFACE_STYLE (tečkované), LARGE_DOT_SURFACE_STYLE, SMALL_HOLE:SURFACE_STYLE (dírkované), LARGE_HOLE_SURFACE_STYLE, SMALL_HATCH_SURFACE_STYLE (šrafované) a LARGE_HATCH_SURFACE_STYLE.

Ukázka:

```
<surface_style>small_hole_surface_style</surface_styl
e>
```

<SURFACE_THICKNESS>

Tag definující tloušťku povrchu. Možné hodnoty jsou VERY_SMALL_SURFACE_THICKNEES, SMALL_SURFACE_THICKNESS, NORMAL_SURFACE_THICKNESS, LARGE_SURFACE_THICKNESS a VERY_LARGE_SURFACE_THICKNESS.

Ukázka

```
<surface_thickness>very_small_surface_thickness</surf
ace_thickness>:
```

<TARGET>

Existence toho tagu je jen pro potřeby tagu <REF>, ve kterém je vnořen. Hodnotou tagu jsou implicitní názvy panelů nástrojů.

Ukázka:

```
<target>3d_geometry_toolbar</target>
```

<TRANSFO3>

Transformační tag, na kterém závisí projekce pohledu. Povinně vnořené tagy jsou <ID> a <VALUE>. Hodnotami <VALUE> jsou obvykle řádky transformační matice typu 4x4. Na tento tag se odkazuje zobrazovací tag <GVIEW3> pomocí vnořeného tagu <PROJECTION>.

Ukázka:

```
<transfo3>

     <id>%3</id>

     <value>((-0.49999999999999978,-
0.43301270189221947,0.75,0),(0.86602540378443871,-
0.24999999999999994,0.43301270189221913,0),(0,0.86602
54037844386,0.50000000000000011,0),(0,0,0,1))</value>

</transfo3>
```

<TRIANGLE_BY_THREE_POINTS3>

Tag definující trojúhelník zadaný třemi body. Povinně vnořené tagy jsou <ID>, <IN_POINTS> a <OUT_SUBPLANE>.

Ukázka:

```
<triangle_by_three_points3>
```

```
    <id>%4</id>
    <in_points>%1 %2 %3</in_points>
    <out_subplane>%5</out_subplane>
</triangle_by_three_points3>
```

<VALUE>

Hodnota tohoto tagu konkrétně udává hodnoty pro tag objektu, v němž je vnořen. Hodnoty mohou být různé u různých objektů, pro geometrické figury většinou pozice nebo vrcholy, případně vektory nebo úhly.

Ukázka:

```
<point3>
<id>Bod1</id>
<value>(0,-4,0)</value>
</point3>
```

Tag <VALUE> je vnořen do tagu <POINT3> tudíž specifikuje hodnoty pro bod, konkrétně udává X, Y a Z souřadnice bodu s ID „Bod1“

<VECTOR3>

Tag definující vektor. Povinné tagy jsou <ID>, <GSTATE>, <ORIGIN> - tag určující počátek vektoru, <VALUE>, <POINT_STYLE>, <POINT_SIZE>, <POINT_COLOR>, <GRAPHIC_LAYERS>, <CURVE_STYLE>, <CURVE_RADIUS>, <CURVE_COLOR>, <BASE_POINT_SIZE> a <BASE_CURVE_RADIUS>.

Ukázka:

```
<vector3>
    <id>I_vector</id>
```

```
    <gstate>defined</gstate>
    <origin>(0,0,0,1)</origin>
    <value>(2,0,0)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>very_small_point_size</point_size>
    <point_color>tomato</point_color>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>small_curve_radius</curve_radius>
    <curve_color1>orange_red</curve_color1>
    <base_point_size>plane3_base_point_size</base_po
    int_size>
    <base_curve_radius>plane3_base_curve_radius</bas
    e_curve_radius>
</vector3>
```

<VERTEX_OF_VECTOR3>

Tag určující vrchol vektoru. Povinné tagy jsou <ID>, <POSITION>, <OUT_POINT> a <IN_VECTOR>.

Ukázka:

```
<vertex_of_vector3>
    <id>%20</id>
    <position>(1)</position>
    <out_point>%19</out_point>
    <in_vector>I_vector</in_vector>
</vertex_of_vector3>
```

<WIDTH>

Tag pro definování šířky pohledu nebo strany. Hodnoty jsou v centimetrech.

Ukázka:

```
<width>21.000</width>
```

<X>

Tento tag má využití při nastavení pohledu a určuje vzdálenost pohledu od horizontálního okraje strany v centimetrech.

Ukázka:

```
<x>0.500</x>
```

<? XML VERSION="1.0" ENCODING="ISO-8859-1?">

Tento tag je neměnný a je na začátku každého CabriML dokumentu. Tento tag říká, že je použito XML verze 1.0 a kódování ISO 8859-1 Latin-1, Západoevropský.

<Y>

Tento tag má využití při nastavení pohledu a určuje jeho vzdálenost od vertikálního okraje strany v centimetrech.

Ukázka:

```
<y>1.500</y>
```

SHRNUTÍ

Díky této analýze je k dispozici první existující popis formátu CabriML, který lze využít několika způsoby jako např.:

- možnost vytvořit grafickou vizualizaci jiným programem než Cabri
- formát lze relativně jednoduše parsovat, když jsou veškeré tagy popsány
- příprava textové scény
- využití v projektu GREG v rámci stereoskopické projekce - možnost vytvořit modul, který bude načítat a posléze zobrazovat scény vytvořené v Cabri

5. ZÁVĚR

V této bakalářské práci jsem měl za úkol zanalyzovat a popsat do té doby neznámý výstupní formát CabriML. Z toho důvodu jsem nejprve lehce uvedl program Cabri 3D, následné nastínil problematiku značkovacích jazyků a poté se věnoval komplexní analýze a popisu zmiňovaného formátu.

Analýza datového formátu Cabri3D – CabriML splnila víceméně očekávání do ní vkládané. Formát se ukázal složitější, než se na první pohled jevil a to zejména kvůli struktuře a hierarchii jednotlivých tagů. Některé konstrukce se zdají dosti nelogické, těžko říct zda to byl záměr tvůrců nebo je formát komplexněji navržen, než jak ho Cabri 3D využívá. Každopádně se na něm ukázala síla metajazyka XML a je vidět, že i komerční produkty mohou fungovat na bázi výstupu s otevřeným formátem.

Celá analýza vznikala na bázi testování dostupných možností uživatelského prostředí Cabri 3D v součinnosti se změnami v textovém výstupu a zároveň i s konfigurací výstupu a zkoušení zobrazení scény vytvořené mimo Cabri v textovém editoru. Největším problémem bylo odhalit funkčnost některých tagů a pochopit systém vzájemného odkazování – zejména definice geometrických figur a jejich závislost na grafické reprezentaci a naopak. I přes veškeré potíže se ale nakonec podařilo odhalit, jakým způsobem vše funguje, a že vzniklá analýza resp. popis je velkým přínosem pro širokou skupinu uživatelů Cabri a dalších podobných produktů.

Zároveň je potřeba podotknout, že tato analýza nemusí být nutně konečná - může existovat tag, který nebyl při výzkumu objeven. Nicméně pro odpovídající zobrazení scény by takový tag neměl být důležitý, protože zásadní tagy definující celou scénu jsou téměř jistě popsány.

Výsledkem této práce je první existující popis CabriML, a to nejen v rámci České Republiky. Podobný popis nelze v současné době nikde jinde nalézt, a právě z tohoto důvodu považuji tento výsledek za největší přínos mé bakalářské práce.

Využití tato práce najde nejen v projektu GREG, kde umožní vytvořit načítání Cabri scén v rámci stereoskopické projekce, ale i v řadách zájemců o libovolnou další implementaci datového výstupu Cabri 3D.

6. POUŽITÉ ODKAZY A LITERATURA

[1] *Domovská stránka Cabri.* http://www.cabri.com, (cit. 8. 4. 2007)

[2] *Akademická stránka Cabri.* http://www-cabri.imag.fr, (cit. 10. 4. 2007)

[3] *Cabri Geometri – český portál.* http://www.pf.jcu.cz/cabri, (cit. 11. 4. 2007)

[4] *Česká verze encyklopedie Wikipedia*. http://cs.wikipedia.org, získáno 8. – 13. 4. 2007

[5] *Anglická verze encyklopedie Wikipedia*. http://en.wikipedia.org, získáno 10. – 16. 4. 2007

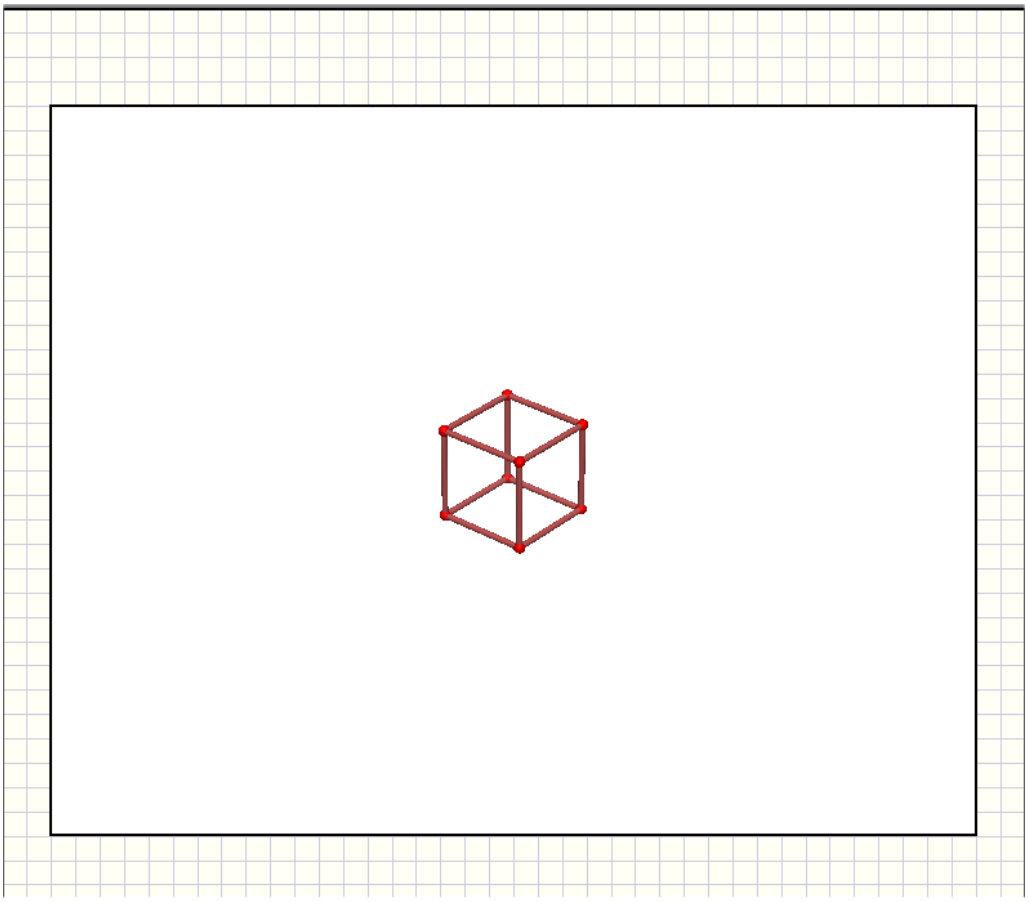
[6] KOSEK, Jiří. *XML pro každého, podrobný průvodce*. Grada Publishing 2000, 164 s., ISBN 80-7169-860-1

[7] VLIST van der, Eric. *XML Schema.* O'Reilly 2002, 400 s., ISBN 0-596-00252-1

[8] BINSTOCK, C. – PETERSON, D. – SMITH, M., *The XML Schema Complete Reference,* Addison Wesley 2002, 1006 s., ISBN 0-672-32374-5

[9] VANÍČEK, Jiří, *Cabri 3D-Cesta do další dimenze?*. 2 konference Užití počítačů ve výuce matematiky, České Budějovice, listopad 2005. Dostupný z WWW <http://www.pf.jcu.cz/cabri/cabri3d/dalsi_dimenze.pdf>.

PŘÍLOHA Č. 1 – SCÉNA V CABRI 3D A JEJÍ REPREZENTACE V CABRIML



```
<?xml version="1.0" encoding="UTF-8"?>
<document>      //nastaveni dokumentu
  <display_scale>1.000</display_scale>
  <active_view>%1</active_view>    //nastaveni pohledu
  <active_page>%9</active_page>
  <subline3>      //grafická reprezentace úseček
    <id>S_Cmmp_Cpmp</id>
```

```
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cmmp_Cmpp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cpmp_Cppp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
```

```
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cppm_Cppp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cmmm_Cmmp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
```

```
  </subline3>
  <subline3>
    <id>S_Cmmm_Cmpm</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cmmm_Cpmm</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cmpm_Cppm</id>
    <gstate>defined</gstate>
```

```
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cmpm_Cmpp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cpmm_Cpmp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>
```

```
<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cpmm_Cppm</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
  <subline3>
    <id>S_Cmpp_Cppp</id>
    <gstate>defined</gstate>
    <graphic_layers>default_layer</graphic_layers>
    <curve_style>solid_curve_style</curve_style>
    <curve_radius>normal_curve_radius</curve_radius>
    <curve_color1>indian_red</curve_color1>

<base_curve_radius>subline3_base_curve_radius</base_c
urve_radius>
  </subline3>
```

```
  <point3>     // definice bodu
    <id>Cppp</id>
    <gstate>defined</gstate>
    <value>(1,1,1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
    <id>Cmmp</id>
    <gstate>defined</gstate>
    <value>(-1,-1,1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
```

```
    <id>Cpmp</id>
    <gstate>defined</gstate>
    <value>(1,-1,1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
    <id>Cppm</id>
    <gstate>defined</gstate>
    <value>(1,1,-1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
    <id>Cmmm</id>
    <gstate>defined</gstate>
```

```
    <value>(-1,-1,-1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
    <id>Cmpm</id>
    <gstate>defined</gstate>
    <value>(-1,1,-1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
    <id>Cpmm</id>
    <gstate>defined</gstate>
    <value>(1,-1,-1,1)</value>
    <point_style>sphere_point_style</point_style>
```

```
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <point3>
    <id>Cmpp</id>
    <gstate>defined</gstate>
    <value>(-1,1,1,1)</value>
    <point_style>sphere_point_style</point_style>
    <point_size>normal_point_size</point_size>
    <point_color>red</point_color>
    <graphic_layers>default_layer</graphic_layers>

<base_point_size>point3_base_point_size</base_point_s
ize>
  </point3>
  <ref>
    <id>document_toolbar</id>
    <target>3d_geometry_toolbar</target>
  </ref>
  <gview3>        //nastavení zobrazení pohledu
    <id>%1</id>
```

```
    <y>2.000</y>
    <x>1.000</x>
    <height>15.000</height>
    <width>19.000</width>
    <surface_clipping>%4</surface_clipping>
    <projection>%2</projection>
    <optics>central_medium_lens</optics>
    <layers>default_layer</layers>
    <clipping>%3</clipping>
  </gview3>
  <transfo3>     //nastavení transformace
    <id>%2</id>
    <value>((-0.64278760968653936,-
0.38302222155948912,0.66341394816893839,0),(0.7660444
4311897801,-
0.32139380484326974,0.55667039922641937,0),(0,0.86602
540378443849,0.5,0),(0,0,0,1))</value>
  </transfo3>
  <transfo3>
    <id>%3</id>

<value>((1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,20))</va
lue>
  </transfo3>
  <transfo3>
    <id>%4</id>
```

```
<value>((1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,6))</val
ue>
  </transfo3>
  <segment_by_two_points3>     //definice úsečky
    <id>%5</id>
    <in_points>Cpmm Cppm</in_points>
    <out_subline>S_Cpmm_Cppm</out_subline>
  </segment_by_two_points3>
  <segment_by_two_points3>
    <id>%6</id>
    <in_points>Cmpp Cppp</in_points>
    <out_subline>S_Cmpp_Cppp</out_subline>
  </segment_by_two_points3>
  <free_crystal_ball3>      //nastavení kamery
    <id>%7</id>
    <elevation>30.000</elevation>
    <azimuth>40.000</azimuth>
    <in_center>origin3</in_center>
    <out_transfo>%2</out_transfo>
  </free_crystal_ball3>
  <segment_by_two_points3>
    <id>%8</id>
    <in_points>Cpmm Cpmp</in_points>
    <out_subline>S_Cpmm_Cpmp</out_subline>
```

```
</segment_by_two_points3>
<page>     // nastavení stránky
  <id>%9</id>
  <number>1</number>
  <height>29.700</height>
  <width>21.000</width>
  <views>%1</views>
</page>
<segment_by_two_points3>
  <id>%11</id>
  <in_points>Cmpm Cmpp</in_points>
  <out_subline>S_Cmpm_Cmpp</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
  <id>%12</id>
  <in_points>Cmpm Cppm</in_points>
  <out_subline>S_Cmpm_Cppm</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
  <id>%13</id>
  <in_points>Cmmm Cpmm</in_points>
  <out_subline>S_Cmmm_Cpmm</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
```

```
  <id>%14</id>
  <in_points>Cmmm Cmpm</in_points>
  <out_subline>S_Cmmm_Cmpm</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
  <id>%15</id>
  <in_points>Cmmm Cmmp</in_points>
  <out_subline>S_Cmmm_Cmmp</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
  <id>%16</id>
  <in_points>Cppm Cppp</in_points>
  <out_subline>S_Cppm_Cppp</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
  <id>%17</id>
  <in_points>Cpmp Cppp</in_points>
  <out_subline>S_Cpmp_Cppp</out_subline>
</segment_by_two_points3>
<segment_by_two_points3>
  <id>%18</id>
  <in_points>Cmmp Cmpp</in_points>
  <out_subline>S_Cmmp_Cmpp</out_subline>
</segment_by_two_points3>
```

```
  <segment_by_two_points3>
    <id>%19</id>
    <in_points>Cmmp Cpmp</in_points>
    <out_subline>S_Cmmp_Cpmp</out_subline>
  </segment_by_two_points3>
  <bitmap>      //náhled
    <png_data>
iVBORw0KGgoAAAANSUhEUgAAAFoAAAB/CAYAAABvydWHAAADgklEQ
VR4nO3dz0vTcRzH8dc0yOjS
qSRCSVCGyx8TV3YIdupimnhxIhP8GzYQb/MiCE4Qj56EKZuKUGrgb
UKeFBm6hjqIJAYiFiER+Wvr
oqabxDJ87TN7PW77fj+HN08+fMbgu82SSqVSkGtXkOsB/hcKTaLQJ
ApNotAkCk2i0CS30i/4fL4c
jHGzOJ1OOJ3OC9cyQgOK/a98Pl9GaB0dJApNotAkCk2i0CQ6TaLQJ
ApNotAkCk2i0CR5F/pgYSHXI1zJpQ/QmOhLN
IrttTX8WFxEYmYGzq4u3LPZ
cj1W1ozf0Z8jEbz1eHCcSsHW3g40NKBlYAB7BweY9XqxGQzmesSsG
Lujt8Nh7CQSOEwm0ez3Z9wv
sdtRYrcDACKBALZ6e/F8ZAT30x7FMoWxOzoxPIxPKyt4XFd34brFY
rnwejcaRVFhIfYbG42NDBgc
+rCpCc1+P34eHeGNx4N4KJSxZsbrxebSEqzt7Sg92d2mMvboON25D
2tq8Nrvx/rSEjaDQeyurmJn
bw/v43G0Dg7+Xp+rQbNkbOh0VocDcDiwl0xicWoKrdPTF+6nHymmM
fbouMyHiQncnp/Ho5aWjHum
fys1L0J/jcUQDQZRVF6OqtFRbEUieNfTg/jk5Nkas/ezwUfH6VEw2
92NZ243nrhcZ/dKa2vh6OwE
```

AMRCIWwsL+NOgdl7xtjQ3+bmsLCxgVf9/X9cV9nWhgc2G2JDQ6TJr
sbY0C9DIWyvrmJtbAwAUNXR

kbEmEYng+/o67lqteDEywh7xrxgbGgCKq6tRXF2NjysrmPV6UeFwo
KKtDQAQGx/H/vEx7G53jqfM

jtGhT5XV1aHs5BPiejCIeF8fngYCqKyvz/Fk2cuL0OdZXS5Yz70x5
guz36pvEIUmUWgShSZRaBKF

JlFoEoUmUWgShSZRaBKFJlFoEoUmUWgSh
SZRaJKM5zrC4bB+1vgaWPSH

Nxw6OkgUmkShSRSaRKFJFJpEoUkUmkShSRSaRKFJFJpEoUkUmkShS
RSaRKFJFJpEoUkUmkShSRSa

RKFJFJpEoUkUmkShSRSaRKFJFJpEoUkUmkShSRSaRKFJFJpEoUkUm
kShSRSaRKFJFJpEoUkUmkSh

SRSaRKFJFJpEoUkUmkShSRSaRKFJfgEqr7s6cPEmXAAAAABJRU5Er
kJggg==

</png_data>

  </bitmap>

</document>