Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

ŠTÁTNY PEDAGOGICKÝ ÚSTAV
NATIONAL INSTITUTE FOR EDUCATION

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Web, Multimédiá

Predmet: Web, Multimédiá

Línia: Vlastný odborový kontext informatiky a informatickej výchovy

Operačný program VZDELÁVANIE

EURÓPSKA ÚNIA
Európsky sociálny fond

Európska únia
Európsky sociálny fond

Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov ES

Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Web, Multimédiá

Identifikácia modulov

Aktivita projektu:	1.3 Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ
Línia aktivity:	Vlastný odborový kontext informatiky a informatickej výchovy
Predmet:	Web, Multimédiá

Zaradenie modulov

Moduly **Webové technológie a publikovanie na webe 2 (3Web)** a **Multimédiá (3Multi)** sú súčasťou vlastného odborového kontextu informatickej výchovy a informatiky v projekte DVUi a tvoria dve alternatívy jedného predmetu. Účastníci Aktivity 1.3 vzdelávania DVUI si vyberajú a absolvujú jeden z nich. Oba obsahovo nadväzujú na druhý modul z línie Digitálna gramotnosť učiteľa (Webové technológie a publikovanie na webe 1).

Abstrakty modulov

V module **Webové technológie a publikovanie na webe 2** účastníci vzdelávania získajú prehľad o vybraných, v súčasnej dobe aktuálnych, postupoch a metódach používaných pri tvorbe webových stránok. Bližšie sa oboznámia s jazykom kaskádových štýlov CSS a naučia sa prostredníctvom neho vytvárať netriviálny dizajn webovej stránky. Druhá časť modulu bude zameraná na programovanie na strane klienta. Účastníci sa tu oboznámia s účelom a so základnými konceptmi client-side aplikácií, získajú veľmi stručný úvod do jazyka JavaScript a sami si vyskúšajú tvorbu jednoduchej client-side aplikácie pomocou knižnice jQuery.

Modul **Multimédiá** poskytne účastníkom prehľad o pokročilejších postupoch tvorby a programovania multimédií. Oboznámia sa s princípmi reprezentácie rastrových a vektorových obrázkov (statických aj animovaných) i s postupmi tvorby interaktívnej grafiky. Zameriame sa nie len na tvorbu, ale aj na programovanie multimédií a ich umiestnenie na webe. V závere sa účastníci oboznámia s princípmi tvorby a úpravy zvukov a videa.

Požadované prerekvizity

Pre oba moduly: Spoločenské aspekty digitálnych technológií / Slobodný a otvorený softvér (3DG1), Webové technológie a publikovanie na webe 1 (3DG2). Pre modul Multimédiá aj Programovanie 1 (3Prog1).

Garanti predmetu:

RNDr. Martin Homola
KAI FMFI UK, Bratislava
FBK, Trento, Taliansko
homola@fmph.uniba.sk

Mgr. Ján Guniš
PF UPJŠ v Košiciach
jan.gunis@upjs.sk.sk

Autori:

RNDr. Martin Homola
KAI FMFI UK, Bratislava
FBK, Trento, Taliansko
homola@fmph.uniba.sk

RNDr. Zuzana Kubincová, PhD.
KZVI FMFI UK, Bratislava
kubincova@fmph.uniba.sk

Mgr. Ján Guniš
ÚINF PF UPJŠ v Košiciach
jan.gunis@upjs.sk.sk

Mgr. Martin Cápay, PhD.
KI FPV UKF v Nitre
mcapay@ukf.sk

PaedDr. Martin Magdin
KI FPV UKF v Nitre
mmagdin@ukf.sk

RNDr. Ľubomír Šnajder, PhD.
ÚINF PF UPJŠ v Košiciach
lubomir.snajder@upjs.sk.sk

Modul má e-learningovú podporu

Obsah

Časť 1: Webové technológie a publikovanie na webe 2

Autori:

RNDr. Martin Homola
KAI FMFI UK, Bratislava
FBK, Trento, Taliansko
homola@fmph.uniba.sk

RNDr. Zuzana Kubincová, PhD.
KZVI FMFI UK, Bratislava
kubincova@fmph.uniba.sk

Kapitola 1: Úvod

Jazyk CSS (kaskádové štýly, z angl. cascading style sheets) patrí k základným stavebným kameňom dnešného webu a je nosnou technológiou moderného webdizajnu. Bez kvalitnej CSS šablóny sa dnes nezaobíde žiadna poriadna webová stránka. Profesionálny webový dizajn síce začína v niektorom z grafických editorov, ale jeho implementácia na webovej stránke je dnes plne v jazyku CSS. Preto je znalosť tejto technológie podstatná aj pre učiteľov, ktorí chcú svojich žiakov naučiť správne postupy pri tvorbe webových stránok s moderným dizajnom.

Na webe sa dnes stále viac presadzujú tzv. RIA aplikácie (angl. rich Internet applications), ktoré sa snažia zvýšiť interaktívnosť webových aplikácií na úroveň aplikácií desktopových. Nosnou súčasťou architektúry RIA je client-side programovanie (t.j. tvorba programov interpretovaných prehliadačom) v programovacom jazyku JavaScript. Vďaka nesmiernej popularite tejto technológie sa o ňu začínajú zaujímať už aj študenti na stredných školách. Považujeme preto za veľmi užitočné, aby mali o tejto technológii aspoň základný prehľad aj učitelia informatiky.

Tak ako mnohé iné technológie, aj CSS a JavaScript sú vo vysokej miere zbraňou dvojsečnou. Ani podrobná znalosť jednotlivých konštruktov jazyka nie je zárukou jeho správneho použitia na daný účel. Preto znalosť technológie ide ruka v ruke s metodológiou tvorby kvalitného webového dizajnu, ktorý je návštevníkom webových stránok na osoh a nie naopak na príťaž. Modul sa sústredí na oba tieto aspekty, technológiu aj metodológiu, rovným dielom.

Modul má formu tutoriálu, čiže série postupne riešených úloh, ktorých cieľom bude postupné vytvorenie uceleného dizajnu webovej stránky v podobe blogu. Budeme pracovať s predpripraveným HTML dokumentom, ktorý budeme postupne formátovať v CSS a neskôr stránku doplníme aj o niekoľko jednoduchých funkcií v JavaScripte.

Ukážku dokumentu pred naformátovaním a po ňom môžeme vidieť na nasledujúcej dvojstrane (obr. 1 a 2).

Kapitola 2: Kaskádové štýly

Ako už vieme z modulu 3DG2 [8], kým v jazyku HTML charakterizujeme štruktúru webových dokumentov, teda z akých prvkov sa dokument skladá, jazyk CSS používame na opis **prezentácie** dokumentov, teda ako dokument vyzerá. Navyše, čo je veľká výhoda, CSS nám umožňuje opísať prezentáciu dokumentov *oddelene od štruktúry* – k HTML dokumentu prilinkujeme súbor so štýlom, ktorý opisuje, ako majú byť jednotlivé HTML elementy naformátované. Výhodou je, že jeden CSS súbor môžeme použiť s viacerými HTML dokumentmi, a keď následne potrebujeme formátovanie zmeniť, môžeme to urobiť rýchlo, jednoducho a konzistentne.

V tejto kapitole si vyskúšame prácu s CSS v praxi. Aby sme si prácu uľahčili a mohli sa sústrediť len na CSS, stiahnime a preštudujme si štartovací balíček s HTML predpripraveným dokumentom a ďalším materiálom (pozri stĺpec).

Úloha 1

Stiahnite si štartovací balíček s HTML dokumentmi a grafickým materiálom. Otvorte vo webovom prehliadači súbor `post.html` a preštudujte si jeho zdrojový kód. Vytvorte (zatiaľ prázdny) CSS súbor `main.css` a prilinkujte ho k dokumentu `post.html`.

Štartovací balíček ako aj ďalší doplnkový materiál k tomuto modulu nájdete na adrese: http://ii.fmph.uniba.sk/~homola/download/dvui/

Keď vo webovom prehliadači otvoríme súbor `post.html`, dokument sa nám zobrazí tak, ako vidíme na obr. 1. Našou úlohou bude postupne dokument naformátovať tak, aby vyzeral približne tak, ako vidíme na obr. 2. Otvorme teraz súbor `post.html` aj v textovom editore a preskúmajme jeho obsah. Všimnime si, že celé telo stránky je zapuzdrené v elemente `div` s ID `container`. Tento obsahuje tri podelementy `div`

Obrázok 1 - Ukážka pôvodného dokumentu

Pripomeňme si, že v HTML môžeme elementom priradiť jednoznačný identifikátor, tzv. ID, a to pomocou atribútu `id`. Toto budeme pri formátovaní často používať.

reprezentujúce hlavičku (ID `header`), obsahovú časť dokumentu (ID `content`) a pätu stránky (ID `footer`). Podľa ukážky výsledného formátu našej stránky si vieme približne predstaviť, ako majú byť tieto tri elementy na stránke rozmiestnené. Napr. pravý stĺpec v obsahovej časti stránky je v dokumente reprezentovaný elementom `div` s ID `right-column`. Zjednodušene vyzerá štruktúru dokumentu takto:

```
<html>
  <head> <title>Šajny Blog</title> </head>
  <body>
  <div id="container">
    <div id="header"> ... </div>
    <div id="content">
      <div id="right-column"> ... </div>
      <div id="left-column"> ... </div>
    </div>
    <div id="footer"> ... </div>
  </div>
  </body>
</html>
```

Okrem súboru `post.html` sa v štartovacom balíčku nachádza ešte aj súbor `index.html`. Tieto dva súbory spolu tvoria predlohu blogu, s ktorou budeme pracovať. Jednotlivým elementom, z ktorých sa dokumenty skladajú, budeme postupne nastavovať rôzne CSS vlastnosti tak, aby sme postupne dosiahli ich požadované rozmiestnenie a vzhľad.

Medzi výhody CSS patrí možnosť oddeliť prezentáciu od štruktúry. V praxi to funguje tak, že jeden CSS súbor (tiež CSS šablóna) prilinkujeme k viacerým HTML dokumentom a tým pádom sa ten istý CSS kód použije sa na formátovanie každého z nich. Vytvorme teda v priečinku s našimi HTML dokumentmi textový súbor `main.css`. Jeho **prilinkovanie** k HTML dokumentu vykonáme vložením nasledujúceho elementu do hlavičky dokumentu.

```
<link rel=”stylesheet” type=”text/css” href=”main.css”/>
```

Obrázok 2 - Ukážka dokumentu na konci tutoriálu

Prázdny štýlový súbor nateraz nijako neovplyvní zobrazenie dokumentu v prehliadači, pokiaľ doň nevložíme nejaký CSS kód. S tým začneme v nasledujúcej časti.

2.1 Základy CSS

Ako už vieme z modulu 3DG2, CSS súbory obsahujú tzv. **pravidlá**, ktoré sa skladajú zo selektora a zo série deklarácií. Selektor určuje, na ktoré elementy dokumentu bude príslušné pravidlo aplikované a deklaráciami vhodne nastavujeme hodnoty CSS vlastností, čím efektívne meníme vzhľad týchto elementov, a teda vlastne aj vzhľad celej webovej stránky.

CSS Selektory

Aby sme si selektory preopakovali, pripravíme si malý kus CSS kódu, ktorého aplikáciu na niektorý HTML element ľahko spozorujeme. Vyskúšajme si najprv **selektor elementu**. Z modulu 3DG2 vieme, že tento selektor má tvar názvu elementu, napr. pre element `div` má tvar „`div`“, pre element `p` má tvar „`p`“, a pod. Pokúsme sa kód aplikovať na element `h2` - nadpis druhej úrovne. Do súboru `main.css` vložíme nasledovné CSS pravidlo:

```
h2 {
  background: red;
  color: white;
}
```

Po načítaní stránky v prehliadači vidíme, že sa nám na červeno nezafarbil iba jeden nadpis, ale všetky nadpisy `h2`. Toto je v poriadku, pretože každé CSS pravidlo je aplikované vždy na všetky elementy, pre ktoré je selektor splnený. Ak by sme napr. v našom pravidle vymenili selektor `h2`, za selektor `div`, zistili by sme, že takmer celý dokument sa nám sfarbí do červena. Je to preto, že náš dokument je prakticky vyskladaný z elementov typu `div`.

HTML elementy môžu mať pomocou atribútu `id` priradený identifikátor, tzv. ID. Z modulu 3DG2 vieme, že v CSS existuje tzv. **ID-selektor**, ktorý nám umožňuje

CSS šablóna je vlastne obyčajný textový súbor. Môžeme s ním teda pracovať v každom textovom editore. Lepší je programátorský editor so zvýrazňovaním syntaxe. Napr. editor PSPad je voľne dostupný:

http://www.pspad.com/

Správnosť CSS kódu si môžeme najľahšie overiť pomocou online W3C validačnej služby:

http://jigsaw.w3.org/css-validator/

Prácu nám môžu uľahčiť vývojárske rozšírenia webových prehliadačov. Populárnym rozšírením je Firebug pre Mozilla Firefox. Podobné rozšírenia sú však zadarmo dostupné aj pre MS Internet Explorer, Safari, Google Chrome, či pre prehliadač Opera.

selektovať elementy podľa ID. ID-selektor sa skladá zo znaku mriežka („#“) nasledovaného hodnotou ID niektorého z elementov tvoriacich stránku. V našom HTML dokumente už máme ID elementy rôzne prednastavené, čo teraz využijeme. Ak zmeníme selektor v našom pravidle na `#header`, presvedčíme sa, že CSS deklarácie budú aplikované iba na jediný element `div` s ID hodnotou `header`.

```
#header {
  background: red;
  color: white;
}
```

Zopakujme si z modulu 3DG2 v čom je rozdiel medzi ID a triedou elementu. Kým ID je vždy unikátne, v jednej triede môže byť viacero elementov.

Okrem ID-selektora budeme často používať aj **selektor triedy**. V module 3DG2 sme sa naučili, že elementom v HTML môžeme priraďovať triedu, a to pomocou atribútu `class`. Aby sme si to vyskúšali, zadefinujeme triedu `red`. Použijeme ten istý kód, iba vymeníme selektor za selektor triedy. Z modulu 3DG2 si pamätáme, že tento selektor sa začína znakom bodka („.“):

Triedu `red` si v CSS súbore `main.css` ponechajme. Opäť sa nám bude hodiť v kapitole 2.

```
.red {
  background: red;
  color: white;
}
```

Po načítaní stránky nevidíme žiadny červený element. Je to preto, lebo žiadny element nie je v triede `red`. Doplňte postupne triedu `red` na niekoľko elementov v súbore s HTML kódom a pozorujte, čo sa bude diať.

Okrem týchto troch základných typov selektorov poznáme z 3DG2 ešte aj **selektor pseudo-triedy** (začína znakom „:“), ktorý selektuje elementy v určitom stave. Napr. `:visited` selektuje odkazy, ktoré už boli navštívené, alebo `:hover` selektuje element, nad ktorým je kurzor myši.

Okrem toho poznáme **zložené selektory**, ktoré vzniknú kombináciou niekoľkých jednoduchších selektorov. Pokiaľ selektory pospájame bez medzier či iných oddeľovačov, vznikne ich **konjunkcia** - takto zložený selektor je aktivovaný len u takých elementov, pre ktoré sú všetky pôvodné selektory súčasne splnené. Napr. selektor `a.menulink:hover` selektuje také elementy `a`, ktoré sú v triede `menulink` a je nad nimi práve umiestnený kurzor myši. Pochopiteľne, takýto element môže byť na stránke iba jeden. Pokiaľ viacero selektorov oddelíme čiarkou, vznikne **disjunkcia selektorov** - dané pravidlo je aplikované na každý z nich.

Často sa stretneme so **selektorom potomka**. Ten vznikne tak, že niekoľko selektorov oddelíme od seba znakom „ “ (medzera). Označme dva selektory písmenami `A` a `B`. V takom prípade selektor `A B` selektuje všetky elementy, pre ktoré je splnený selektor `B` a zároveň sú potomkom nejakého elementu, pre ktorý je splnený selektor `A`. Napr. selektor `.menu a:visited` selektuje všetky navštívené odkazy, ktoré sa nachádzajú vo vnútri nejakého elementu (v ľubovoľnej hĺbke), ktorý je v triede `menu`.

Ďalej v CSS existuje ešte zložený **selektor priameho potomka** (znak „>“) ako aj **selektor nasledovníka** (znak „+“). Ako ich názvy nasvedčujú, selektor `A>B` selektuje taký element, pre ktorý je splnený selektor `B` a zároveň je priamym potomkom elementu, ktorý spĺňa selektor `A`. Rozdiel medzi potomkom a priamym potomkom je v tom, že potomok sa môže nachádzať vo vnútri elementu v ľubovoľnej hĺbke, kým priamy potomok sa vo vnútri svojho rodiča nachádza v hĺbke 1. Selektor nasledovníka v tvare `A+B` selektuje element, ktorý sám spadá pod selektor `B`, `a` zároveň je priamym nasledovníkom iného elementu v strome dokumentu, spadajúceho pod selektor `A` - t.j. `B` je bezprostredne nasledujúcim „bratom“ `A`.

Selektory > a + musíme používať opatrne, pretože niektoré staršie prehliadače (napr. IE6) ich nepodporujú.

Fonty, vlastnosti textu a farby

Preopakujme si najprv niektoré najdôležitejšie CSS **vlastnosti písma a textu**. Následne nastavíme tieto vlastnosti pre náš dokument:

- `font-family`: nastavenie fontu. Hodnota je čiarkami oddelený zoznam. Každá položka je buď konkrétny názov fontu alebo generická rodina. Základ-

né generické rodiny sú `serif` (pätkové písmo) a `sans-serif` (bezpätkové písmo). Niekedy sa hodí tiež rodina `monospace` (neproporcionálne);
- `font-size`: veľkosť fontu. Hodnota je dĺžka alebo percentuálny údaj;
- `font-weight`: šírka písma. Hodnoty `normal` (normálne) a `bold` (tučné);
- `font-style`: zošikmenie. Možné hodnoty: `normal`, `italic` (kurzíva), `oblique` (zošikmené);
- `text-decoration`: hodnoty `none`, `underline` (podčiarknuté), `overline` (nadčiarknuté), `line-through` (prečiarknuté);
- `text-align`: zarovnanie textu. Hodnoty `left` (na ľavý okraj), `right` (na pravý okraj), `center` (na stred) a `justify` (na oba okraje).

Preopakujme si vedomosti o fontoch, vlastnostiach textu a farbách, ktoré sme získali v module 3DG2.

Okrem vlastností písma budeme nastavovať **farbu textu a pozadia**. Často budeme tiež využívať obrázok na pozadí. Budeme potrebovať dve vlastnosti `color` (farba textu) a `background` (pozadie). Hodnotou oboch je údaj o farbe, ktorý nastavujeme pomocou kľúčových slov (napr. `white`, `red`, `yellow`, ...) alebo pomocou farebných kódov v tvare `#RRGGBB` (`RR`, `GG` a `BB` sú čísla v šestnástkovej sústave v rozsahu `00-ff`) alebo v tvare `rgb(R,G,B)` (`R`, `G` a `B` sú desiatkové hodnoty s rozsahom `0-255`). Prácu s obrázkom na pozadí uvidíme nižšie.

Aby bol výsledok dobre čitateľný pre každého používateľa, bez ohľadu na počítač, operačný systém, či prehliadač, ktorý používa, musíme sa pri práci s týmito vlastnosťami pridržať základných **zásad webovej typografie**:

1. **nastavenie fontu**. ako hodnotu `font-family` vždy uvedieme zoznam niekoľkých podobných fontov zakončený generickou rodinou, do ktorej patria. Prehliadač má pre každú rodinu prednastavený nejaký font daného typu. Tento font bude použitý, ak používateľ nemá žiadny z predchádzajúcich fontov nainštalovaný. Väčšinou používame bezpätkový font;
2. **veľkosť písma** nikdy nenastavujme absolútnu. Najlepšie je nastaviť ju v percentách, pričom základnú veľkosť nemeniť (použije sa veľkosť prednastavená v prehliadači používateľom) a meniť len tam, kde potrebujeme písmo zväčšiť alebo zmenšiť. Nikdy nezmenšujme na menej ako 80% pôvodnej veľkosti;
3. na **zvýrazňovanie v texte** podľa potreby používame kurzívu alebo tučné písmo, nikdy nie však podčiarknuté, pretože podčiarknuté bývajú webové odkazy – používateľa by to miatlo;
4. nemeňme **zaužívaný štýl odkazov** (modré a podčiarknuté) pokiaľ to nie je naozaj nutné. Pokiaľ sa rozhodneme ho zmeniť, musíme zabezpečiť aby používateľ dokázal odkazy ľahko odlíšiť od okolitého textu, a aby si hneď uvedomil, že ide o odkaz;
5. dodržme vždy **vysoký kontrast** medzi popredím a pozadím, aby text ľahko prečítali aj používatelia s obmedzeným zrakom.

Formátovanie webovej stránky začneme tým, že odstránime HTML elementy, ktoré sa nám do výsledného dizajnu nehodia. Konkrétne, na obr. 3 môžeme vidieť horizontálne čiary (v HTML kóde element `hr`). Tieto čiary sú v dokumente preto, aby bol ako-tak čitateľný aj bez formátovania pomocou CSS (napr. ak je súbor s CSS z nejakého dôvodu nedostupný). V CSS ich jednoducho vypneme pomocou malého triku za pomoci vlastnosti `display`, s ktorou sa podrobnejšie oboznámime neskôr:

```
hr {
  display: none;
}
```

Prejdime teraz k samotnému formátovaniu stránky. Ako prvú vec nastavíme vlastnosti fontu, a tiež pozadie celej stránky.

Úloha 2

Nastavte pre celý dokument font `Tahoma` a pre nadpisy a názov stránky font `Georgia`. Farba textu bude čierna a v pozadí dokumentu bude obrázok `background.png`.

Font a farbu písma, ale tiež obrázkové pozadie nastavíme pre celú stránku v elemente `body`, ktorý celý dokument zapuzdruje:

Je dobré nastaviť vlastnosti `background` a `color` *vždy* obe naraz a to hlavne u elementu `body`. Keďže používateľ môže mať tieto farby prednastavené rôzne, vlastným nastavením zabránime tomu, aby bola výsledná kombinácia málo kontrastná.

Tiež je vhodné nastaviť okrem obrázka na pozadí aj farbu pozadia, ak by obrázok bol z nejakého dôvodu nedostupný.

```
body {
  font-family: Tahoma, Helvetica, sans-serif;
  color: black;
  background: white url("background.png") repeat-x;
}
h1, h2, h3, h4, h5, h6, .title {
  font-family:  Georgia, "Times New Roman", Times, serif;
}
```

Vlastnosti `background` sme odovzdali až tri hodnoty: najprv sme nastavili bielu farbu pozadia (`white`); následne sme povedali, že na pozadí bude obrázok s URI adresou `background.png` (použili sme relatívnu adresu, takže ide o obrázok z toho istého priečinka, v ktorom je celá stránka). Posledná hodnota `repeat-x` určuje, akým spôsobom sa obrázok na pozadí bude opakovať. My sme nastavili opakovanie v smere osi X (ďalšie hodnoty sú `repeat-y`, `repeat` a `no-repeat`).

Druhým pravidlom nastavíme font pre všetky nadpisy a pre názov blogu. Všimneme si, že vlastnosť `font-family` je v oboch prípadoch ukončená generickou rodinou. Využili sme selektor elementu pre elementy `body` a `h1-h6` a selektor triedy pre triedu `title`, keďže názov stránky sa v oboch dokumentoch nachádza vždy v elemente s touto triedou. Všimnime si tiež zložený selektor *disjunkcie* v spoločnom pravidle pre všetky nadpisy.

Každý webový dokument by mal mať práve jeden nadpis prvej úrovne `h1`. Kým prirodzeným nadpisom hlavnej stránky nášho blogu je „Šajnyho blýskavý blog“, na stránke s článkom je to už názov článku.

Názov stránky však stále neodpovedá predlohe, a navyše vyzerá v každom z dokumentov inak. Je to preto, že v jednom nich je tvorený elementom `h1` a v druhom z nich elementom `div`.

Úloha 3

Názov blogu naformátujte tak, aby na prvý pohľad nebolo vidieť, že ide o odkaz: jeho text nebude podčiarknutý, písmo v nadpise bude čierne a dvojnásobne veľké. Dbajte na to, aby nadpis vyzeral v `index.html` aj v `post.html` rovnako.

Názov je v každom z dvoch dokumentov zakódovaný odlišne. V dokumente `post.html` jednak nie je v nadpise `h1`, a okrem toho je zároveň hypertextovým odkazom. Preto je malý, farebný a podčiarknutý. My však chceme, aby názov v oboch prípadoch vyzeral rovnako, jediným rozdielom bude, že mimo hlavnej stránky (`index.html`) sa naň bude dať kliknúť. Budeme postupovať nasledovne:

```
.title {
  font-size: 225%;
  font-weight: bold;
}
.title a {
  color: black;
  text-decoration: none;
}
```

Keďže chceme aby bol názov výrazný, zväčšili sme písmo na 225% oproti zvyšku dokumentu a nastavili sme tučný rez. V súbore `post.html` zistíme, že odkaz (element `a`) sa nachádza až vo vnútri nadpisu. Preto, ako vidíme vyššie, použijeme selektor potomka. Nastavíme mu farbu textu na čiernu a zrušíme podčiarknutie.

Obrázok 3 - Nadpis a pozadie stránky (`post.html`)

Na obr. 3 vidíme výsledok nášho snaženia. Nadpis sme už naformátovali dobre. Keď však otvoríme v prehliadači dokument `index.html`, zistíme, že nadpis je posunutý. Tento problém vyriešime v ďalšej časti, kde sa zoznámime s tzv. box modelom.

Samostatná úloha

Úloha 4

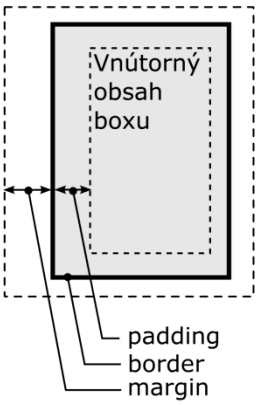
Naformátujte menu v hlavičke a päte stránky, aby neboli jednotlivé jeho odkazy podčiarknuté.

Čo sme sa naučili

Zopakovali sme si základy CSS: prilinkovanie šablóny, selektory, vlastnosti písma, farby textu a pozadia. Vieme selektovať podľa mena elementu, ID a triedy. Ovládame CSS vlastnosti `color`, `background`, `font-family`, `font-size`, `font-style`, `font-weight` a `text-decoration`.

2.2 Box model

Jednotlivé elementy sú v na webovej stránke vizuálne reprezentované obdĺžnikovým tvarom, ktorému hovoríme *box*. Súbor CSS vlastností, pomocou ktorých môžeme tento tvar meniť nazývame **box model**. Každý box má dĺžku, šírku, rámik, vonkajší a vnútorný okraj (pozri stĺpec).



Box model v CSS.

Rozmery boxu meníme pomocou vlastností `width`, `height`, `min-width`, `max-width`, `min-height` a `max-height`. Vlastnosťami `margin` (vonkajší okraj) a `padding` (vnútorný okraj) meníme odsadenie elementu voči okolitým elementom v dokumente, ako aj odsadenie obsahu elementu od jeho rámika. Na nastavenie šírky a farby rámika elementu používame vlastnosť `border`.

Ako hodnoty týmto vlastnostiam najčastejšie odovzdávame dĺžkový alebo percentuálny údaj. **Dĺžkový údaj** sa skladá z čísla a jednotky. Ak je dĺžka 0, môžeme jednotku vynechať (inak v žiadnom prípade). Najčastejšie používané jednotky sú `em`, `ex` a `px`. **Percentuálny údaj** sa skladá z čísla a znaku „`%`“ (napr. `10%`). Percentá sú odvádzané od tzv. referenčného elementu (pozri nižšie). Pokiaľ nenastavíme inak, je ním rodičovský element. Ďalej v tejto časti nastavíme vlastnosti box modelu hlavným obsahovým častiam stránky.

Dĺžkové jednotky delíme na absolútne (`cm`, `mm`, `in`, `pt`, `pc`), relatívne vzhľadom k zariadeniu (`px`) a relatívne vzhľadom k veľkosti písma (`em`, `ex`). Na webe výrazne preferujeme práve poslednú menovanú skupinu.

Úloha 5

Naformátujte obsahovú časť dokumentu podľa obr. 4. Obsah bude mať biele pozadie a obmedzenú šírku, primeraný vnútorný okraj, a bude odsadený od vrchu stránky. Na šírku má byť v okne vycentrovaný.

Obrázok 4 - Obsahová časť stránky

Začneme s elementom `body`. Aby sme docielili rovnaký výsledok vo všetkých prehliadačoch, nastavíme vlastnosti `margin` a `padding` (jednotlivé prehliadače môžu mať rôzne prednastavené hodnoty). My nastavíme hodnotu 0, keďže okraje obsahovej časti už o chvíľu vyriešime inak:

```
body {
  ...
  margin: 0;
  padding: 0;
}
```

Celú obsahovú časť v HTML kóde reprezentuje element `div` s ID `container` Vytvoríme preto nové pravidlo so selektorom `#container`:

```
#container {
  background: white;
  min-width: 22em;
  max-width: 42em;
  padding: 1ex;
  margin: 1em auto;
}
```

Vyskúšajte si, ako sa pri zmene šírky okna prehliadača správa stránka v prípade, že sme elementu s ID `container` nastavili pevnú šírku a v prípade, že sme mu nastavili pružnú šírku.

Elementu sme nastavili biele pozadie. Keďže vieme, že farba textu bude odvodená od rodičovského elementu `body`, ktorému sme nastavili čiernu farbu, tentoraz vlastnosť `color` môžeme vynechať. Nastavili sme tiež šírku elementu. Možno ju nastaviť ako pevnú (vlastnosťou `width`) alebo pružnú (`min-width` a `max-width`). My sme sa rozhodli pre pružnú šírku, vďaka čomu obsahová časť nikdy nepresiahne `42em`, ak ale používateľ okno zúži na menšiu šírku, stránka sa prispôsobí. Vyhneme sa tak horizontálnemu rolovaniu, čo je žiaduce (pozri 3DG2). Zároveň, ak je okno priširoké, text sa nerozleje do príliš dlhých riadkov, ktoré sú zle čitateľné.

Okraje môžeme nastavovať aj pre každú stranu zvlášť, s použitím vlastností `margin-top`, `margin-right`, `margin-bottom` a `margin-left` a podobne pre `padding`.

Ďalej sme nastavili **vnútorné a vonkajšie okraje**. Vnútorný okraj (vlastnosť `padding`) nastavujeme preto, aby sa obsah elementu bezprostredne nedotýkal krajov. V našom prípade sme týmto docielili výrazné odsadenie celej obsahovej časti od okolitého bieleho pozadia v šírke `1em`. Vlastnosti `padding` sme pritom odovzdali len jednu hodnotu. V takom prípade hodnota platí pre všetky štyri okraje boxu. Vonkajší okraj nastavujeme pomocou vlastnosti `margin`. Tu sme odovzdali dve hodnoty. Horizontálny okraj sme nastavili na `1em`, čím sme element efektívne odsadili od okraja okna prehliadača. Pre vertikálny okraj sme použili hodnotu `auto`, vďaka čomu bude obsah umiestnený horizontálne v strede okna.

Vlastnostiam `margin` a `padding` môžeme odovzdať jednu, dve alebo štyri hodnoty. Ak odovzdáme jednu hodnotu, platí pre všetky štyri okraje boxu; ak odovzdáme dve hodnoty, prvá platí pre horizontálne okraje a druhá pre vertikálne; ak odovzdáme hodnoty štyri, priraďujeme ich v poradí horný, pravý, dolný a ľavý okraj. Prípustné hodnoty sú dĺžkový údaj, percentuálny údaj alebo kľúčové slovo `auto`. Hodnota `auto` znamená „dorátať automaticky“. Ak nastavíme `auto` pre ľavý aj pravý okraj, bude obom dorátaná rovnaká hodnota, čím sa element dostane do stredu. Ďalej sa naučíme nastavovať rámik boxu a naformátujeme hlavičku stránky:

Pozor na nedostatočnú podporu box modelu v starších prehliadačoch. Napr. IE6 nepodporuje centrovanie blokových elementov pomocou `margin` hodnoty `auto`. Problém možno obísť tak, že v rodičovskom elemente nastavíme `text-align` na hodnotu `center` a v samotnom centrovanom elemente naspäť na `left`, alebo inú želanú hodnotu.

IE6 však nepodporuje ani vlastnosti pre maximálnu a minimálnu šírku a výšku. Tento problém už nevieme nijako obísť. Ak sa nám to hodí, môžeme pomocou jednoduchého triku nastaviť pre IE6 šírku a výšku pevne. Pozri napr.: http://www.quirksmode.org/css/condcom.html

Úloha 6

Naformátujte hlavičku stránky podľa obr. 5. Nastavte okraje a rámik, do pozadia vložte obrázok `header.png`.

Obrázok 5 - Základné formátovanie hlavičky a nadpisu

Okraje a pozadie už nastavovať vieme. Vonkajší okraj je nulový, čo je u elementu div preddefinovaná hodnota, takže ju už nastavovať nemusíme.

```
#header {
  background: white url("header.png") repeat-x;
  padding: 1.5ex 1ex 1ex 1ex;
  border: solid black 1px;
  border-top: solid black 2px;
}
```

Spomeňme si ale na nedoriešenú úlohu 3. Keďže element `h1` má okraje prednastavené, vyzerá názov stránky inak v súbore `index.html` a inak v `post.html`. Problém teraz už ľahko odstránime vynulovaním okrajov v triede `title`:

```
.title {
  margin: 0;
}
```

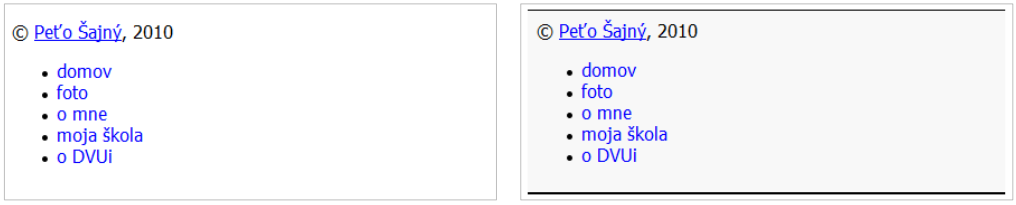
Nové je tu **orámovanie boxu**. Na tento účel slúži vlastnosť `border`, ktorá vyžaduje trojicu hodnôt: typ, farbu a šírku orámovania. Typ `solid` určuje súvislú čiaru orámovania. Okrem toho je možné napr. čiarkované (`dashed`), bodkované (`dotted`) a dvojité orámovanie (`double`). Hrúbku rámika sme zadali v pixeloch, pretože chceme jemný, čo najužší rámik. Keďže hornú hranu rámika chceme mať širšiu, vzápätí sme ju predefinovali za pomoci vlastnosti `border-top`.

Existujú tiež vlastnosti `border-left`, `border-right` a `border-bottom`. Ak sa nám to hodí, môžeme ich využiť.

Úloha 7

Naformátujte pätu stránky podľa obr. 6. Zobrazte jej len hornú a dolnú hranu rámika, dolnú širšiu. Nastavte vnútorný okraj a svetlosivé (napr. `#f8f8f8`) pozadie.

Samostatná úloha



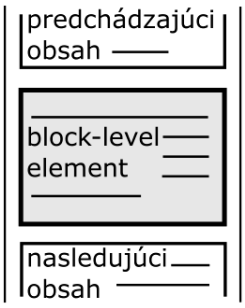
Obrázok 6 - Päta stránky pred naformátovaním a po ňom.

V hlavičke stránky ako aj v jej päte sa nachádza navigačné menu, ktoré je v dokumente reprezentované elementom `ul` (nečíslovaný zoznam) a zároveň je v triede `menu`. Menu chceme podľa predlohy naformátovať do formy **horizontálneho panelu**, v ktorom sú jednotlivé položky umiestnené vedľa seba.

Úloha 8

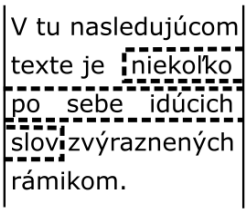
Naformátujte navigačné menu dokumentu tak, aby jeho položky boli umiestnené vedľa seba v riadku, bez odrážok zoznamu, a aby sa pri pohybe kurzorom myši nad nimi zobrazil okolo príslušnej položky rámik.

Obrázok 7 - Hlavička stránky pred naformátovaním menu a po ňom



Blokový element.

Aby sme sa s úlohou dokázali popasovať, zoznámime sa najprv podrobne s vlastnosťou `display`. V module 3DG2 sme sa naučili, že vizuálne elementy v HTML sú buď blokové alebo riadkové. Správanie sa týchto dvoch typov elementov je odlišné a úzko súvisí s box modelom. **Blokový element** má priradený jeden box, je teda vždy obdĺžnikového tvaru. Blokové elementy sa vždy radia pod seba a pokiaľ im nenastavíme šírku „rozlejú“ sa na celú šírku rodičovského elementu. Pre **riadkové elementy** je naopak typické, že sa radia vedľa seba do riadkov. Pokiaľ je element dlhší než voľná šírka riadku, dôjde k jeho zalomeniu na nový riadok. Riadkový element má na každom riadku, v ktorom figuruje, jeden box. Pre blokové aj riadkové elementy platia všetky vlastnosti box modelu, výnimkou sú iba vertikálne vonkajšie okraje, ktoré na riadkové elementy nemajú vplyv.



Riadkový element.

Vlastnosť `display` umožňuje prepínať typ elementu podľa potreby. Základné hodnoty sú `block` (blokový element), `inline` (riadkový element) a tiež hodnota `none`, s ktorou sme sa už zoznámili, a ktorá nám umožňuje zobrazovanie elementu úplne vypnúť. Okrem týchto hodnôt sú možné ešte niektoré špeciálne hodnoty, napr. hodnota `list-item` (položka zoznamu) - špeciálny typ blokového elementu s odrážkou, vyskytujúci sa v zoznamoch.

Jednotlivé HTML elementy majú túto vlastnosť prednastavenú, napr. elementy `div`, `h1`, ..., `h6` a `p` sú blokové, elementy `span`, `a`, `strong` a `em` sú riadkové. Element `li` má hodnotu `display` prednastavenú na `list-item`. Ak chceme, aby sa menu zobrazovalo v riadku, využijeme práve vlastnosť `display`, a pre položky menu nastavíme hodnotu `inline`. Keďže odrážky zoznamu sú naviazané na hodnotu `list-item`, nebudú sa už viac zobrazovať a jednotlivé položky menu sa poukladajú do riadku vedľa seba. Použijeme nasledovný kód:

```
.menu {
  margin: 1ex 0;
  padding: 0;
}
.menu li {
  display: inline;
  padding: 0.25ex 1ex;
  margin: 0;
}
.menu li:hover {
  border: solid black 1px;
}
```

Okrem nastavenia `display` musíme venovať zvýšenú pozornosť vnútorným aj vonkajším okrajom menu a položkám v ňom. V HTML kóde sú totiž reprezentované zoznamom (elementy `ul`, `li`) a pre zoznamy je typické odsadenie, ktoré je predformátované práve pomocou vlastností `margin` a `padding` (navyše v každom prehliadači trochu inak). Musíme preto tieto vlastnosti dôsledne predefinovať.

Preformátovať zoznam tvoriaci menu do formy vodorovného panelu nie je úplne triviálne. Prečo sme si teda pre menu nevybrali nejaké iné HTML elementy, napr. `div` alebo `span`, ktorých predefinované formátovanie je jednoduchšie?

Je to preto, lebo menu v zozname je prístupnejšie pre zrakovo postihnutých a prehľadnejšie v prípade, ak je CSS nedostupné.

Posledné pravidlo rámčekom zvýrazní položku menu pri pohybe myšou. Je to vďaka pridanému selektoru `:hover`, ktorý je aktivovaný len ak sa nad elementom práve nachádza kurzor myši. Keď však dokument otvoríme v prehliadači, zistíme, že položky pri pohybe myšou ponad ne akosi poskakujú. Je to preto, lebo položka menu, nad ktorou je kurzor, je širšia - o šírku rámiku na oboch stranách. Riešením je pravidlo, ktoré vidíme nižšie. Ním sa nastaví priesvitný rámik pre položky, nad ktorými kurzor nie je.

```
.menu li {
  ...
  border: solid transparent 1px;
}
```

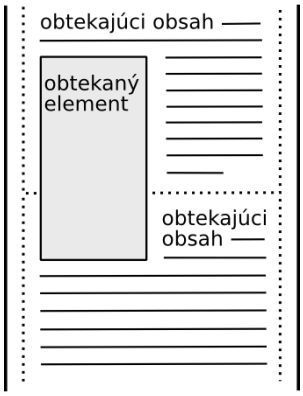
Čo sme sa naučili

Podrobne sme sa zoznámili s box modelom v CSS, ktorý nám umožňuje meniť vzhľad jednotlivých elementov v HTML dokumente. V praxi sme si vyskúšali prácu s vlastnosťami `margin`, `padding`, `border`, `width`, `min-width`, `max-width`, `height`, `min-height`, `max-height`, a `display`.

2.3 Obtekanie

Kým box model nám umožňuje nastaviť rozmery jednotlivých elementov, prípadne okraje, tzv. v*izuálny formátovací model* nám umožňuje ovplyvniť rozmiestnenie elementov na stránke. Doteraz sme sa vždy stretli s tzv. *statickým polohovaním*, pri ktorom sa jednotlivé elementy ukladajú za sebou, prípadne vedľa seba, podľa toho či ide o blokový alebo riadkový element. Okrem týchto dvoch základných možností CSS umožňuje viaceré ďalšie špeciálne formátovacie módy, pomocou ktorých môžeme elementy organizovať na webovej stránke flexibilnejšie, podľa potreby.

Vlastnosť float



Obtekanie, vlastnosť `float`

Jednou z týchto možností je tzv. **obtekanie** (angl. **floating**). Obtekanie nám umožňuje vytvoriť blokové elementy špeciálneho typu, tzv. obtekané elementy. Obtekaný element je špeciálny v tom, že ďalší obsah, ktorý by normálne nasledoval až pod ním, miesto toho vyplní priestor vedľa neho. Hovoríme, že tento obsah element obteká. Obtekanie je možné sprava alebo zľava. Nastavíme ho pomocou vlastnosti `float`. Preddefinovaná hodnota je `none` (obtekanie vypnuté). Obtekanie sprava zapína hodnota `left`, obtekanie zľava hodnota `right`.

Presnejšie, pri zapnutom obtekaní sprava je obtekaný element posunutý čo najviac vľavo a nasledujúci obsah ho obteká sprava. Podobne pri obtekaní zľava je obtekaný element posunutý vpravo a nasledujúci obsah ho obteká zľava. Posunutie vľavo či vpravo je vždy k okraju rodičovského elementu, prípadne po okraj iného obtekaného elementu, ktorý do aktuálneho riadku zasahuje. Túto techniku si teraz prakticky vyskúšame pri formátovaní obrázkov v obsahovej časti stránky `post.html`.

Úloha 9

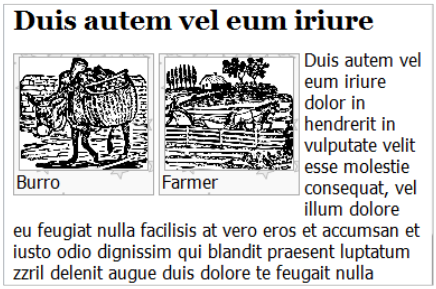
Navrhnite formátovanie rámčekov s obrázkami v obsahovej časti stránky `post.html` podľa obr. 8 tak, aby obrázky boli v rámčeku spolu s popisom a okolitý text ich obtekal sprava. Obrázky by sa mali nachádzať vždy pod sebou.

Obrázok 8 - Formátovanie obrázkov obtekaním

Ak si prezrieme HTML kód nášho dokumentu, zistíme, že obrázkové rámce sú reprezentované elementom `div` s triedou `floatbox`. Vo vnútri tohto elementu sa nachádza obrázok ako aj popis. Nastavenia preto urobíme pre triedu `floatbox`, čím zabezpečíme, že budú aplikované na každý takýto element `div`.

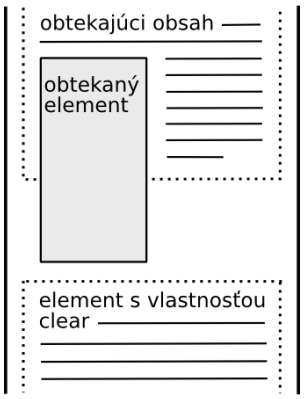
Obtekanie rámca sprava zapneme nastavením vlastnosti `float` na hodnotu `left`. Ďalšie formátovanie rámcov - centrovanie obrázka v rámci, nastavenie obrázka na pozadí, orámovania a okrajov - už urobiť vieme, naučili sme a to v predchádzajúcej časti. Môžeme použiť napr. nasledovné formátovanie:

```
.floatbox {
  float: left;
  margin: 0.5ex 0.5ex 0.5ex 0;
  padding: 0.25ex;
  width: 7.5em;
  border: solid #aaa thin;
  background: white url('header.png') repeat-x;
}
.floatbox img {
  display: block;
  margin: auto;
  width: 95%;
  border: solid #aaa thin;
  background: white;
}
```



Obrázok 9 - Jeden obrázok obteká druhý

Vonkajší okraj rámcov (`margin`) sme nastavili nenulový zhora, sprava aj zdola, aby sa rámec textu nedotýkal. Zľava sme však nastavili nulu, aby rámec s obrázkom lícoval s ľavým okrajom textu; tu by bolo ďalšie odsadenie naopak nevhodné. Centrovanie obrázka v rámčeku sme dosiahli podobnou technikou ako v prípade celej stránky, t.j. nastavením okrajov na `auto`. Aby to fungovalo, musíme však obrázku nastaviť hodnotu vlastnosti `display` na `block`, pretože preddefinovaná hodnota je iná.



Zamedzenie obtekaniu pomocou vlastnosti `clear`

Vlastnosť clear

Keď stránku načítame v prehliadači, zistíme, že formátovanie (obr. 9) ešte celkom neodpovedá našej pôvodnej predstave (obr. 8 vpravo). Druhý rámec s obrázkom sa nachádza vedľa prvého, pretože ho tak, ako aj ostatný obsah, obteká sprava. Rámec je teda obtekaný a súčasne obtekajúci. Potrebujeme preto nájsť spôsob, ako druhému rámcu povedať, že on už obtekajúci byť nemá. Presne toto nám umožňuje vlastnosť `clear`. Vlastnosť `clear` môže mať hodnoty `left`, `right`, `both` a `none`, ktoré umožňujú **zamedziť obtekaniu** zľava, sprava, prípadne z oboch strán. Prednastavenou hodnotou je `none`, t.j. obtekanie povolené. Nastavme teda v triede `floatbox` vlastnosť `clear` nasledovne:

```
.floatbox {
  ...
  clear: left;
}
```

Výsledkom je požadované umiestnenie obrázkov na stránke. Sú obtekané zvyšným obsahom, ale ony samotné nie sú obtekajúcimi elementmi. Precvičme si teraz polohovanie obtekaním na nasledujúcej úlohe.

Samostatná úloha.

Úloha 10

Naformátujte rámce s obrázkami tak, aby ich text obtekal zľava a aby boli umiestnené vedľa seba; potom formátovanie zmeňte tak, aby boli pod sebou. Ako sa správa zvyšný text pri rôznych polohách textového rámca?

Čo sme sa naučili

Zoznámili sme sa s technikou obtekania, pomocou ktorej môžeme jednoducho do textu webových dokumentov vkladať rôzne obtekané objekty. Obtekanie sme si prakticky vyskúšali pri formátovaní obrázkových rámcov. Naučili sme sa pracovať s vlastnosťami `float` a `clear`. Preopakovali sme si vlastnosti `display`, `margin`, `padding` a `background`.

2.4 Absolútne polohovanie

V predchádzajúcej časti sme sa zoznámili s technikou obtekania, ktorá nám poskytla nové možnosti ako polohovať elementy v texte, v ktorom sa nachádzajú. Možnosti polohovania elementov v CSS sú však oveľa bohatšie. Ďalšie polohovacie módy nám sprístupňuje vlastnosť `position`. Nastavením jej hodnoty môžeme prepínať medzi špeciálnymi polohovacími módmi, ako sú napr. absolútne polohovanie (`absolute`) a relatívne polohovanie (`relative`). Preddefinovaná hodnota `static` znamená základné statické polohovanie, s ktorým sme sa už podrobne oboznámili.

Absolútne polohovanie umožňuje umiestniť vybraný element na ľubovoľné miesto stránky prostredníctvom súradnicových osí. Využijeme pri tom vlastnosť `position` (hodnota `absolute`), a dvojicu vlastností súradnicových osí. Tieto vlastnosti sú dohromady štyri: `left` a `right` (horizontálne), `top` a `bottom` (vertikálne). Zvyčajne používame vždy jednu horizontálnu a jednu vertikálnu os. Absolútne polohovanie si vyskúšame na umiestnení loga DVUi v hlavičke stránky podľa predlohy, teda vpravo hore (obr. 10). Predtým však logo ešte naformátujme.

Úloha 11

Naformátujte logo DVUi a text „Som DVUík, či?“ v hlavičke stránky tak, aby sa tento text nachádzal pod obrázkom s logom a aby boli navzájom vycentrované.

Obrázok 10 - Naformátovaná hlavička stránky

Všimnime si, že v HTML kóde dokumentu máme časť s logom reprezentovanú elementom `div`, ktorý je v triede `logo`. Formátovanie preto urobíme pre túto triedu:

```
.logo {
  width: 8em;
  text-align: center;
}
.logo img {
  display: block;
  margin: auto;
}
```

Prejdime teraz k samotnej polohe loga v hlavičke stránky.

Úloha 12

Umiestnite logo DVUi v hlavičke stránky vpravo hore, podľa obr. 10. Použite pritom absolútne polohovanie.

Pre aktiváciu absolútneho polohovania v prvom rade nastavíme vlastnosť `position` na hodnotu `absolute`. Keďže logo chceme umiestniť do pravého horného rohu hlavičky, nastavíme súradnice za pomoci vlastností `top` a `right`, napr. takto:

```
.logo {
  position: absolute;
  top: 1ex;
  right: 0;
}
```

Výsledok je však trochu iný, než sme očakávali. Na obr. 11 vidíme, že logo „ušlo“ z hlavičky von a posunulo sa do pravého horného rohu stránky:

Obrázok 11 - Absolútne polohované logo „ušlo“ z hlavičky stránky

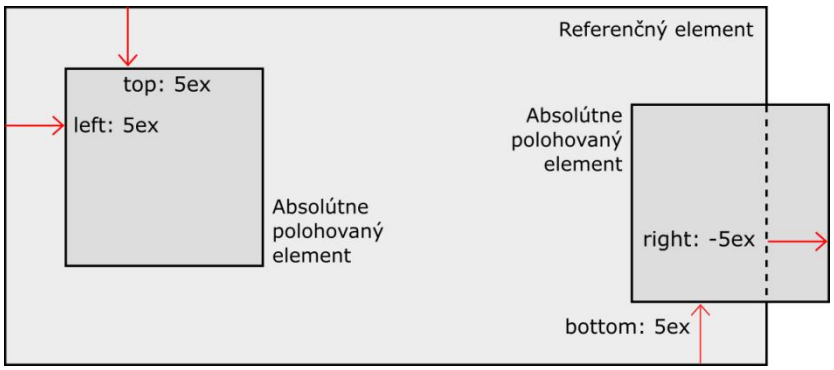
Pri absolútnom polohovaní totiž vlastnosti `top`, `right`, `bottom` a `left` určujú polohu elementu vzhľadom k jeho **referenčnému elementu**. Tým je pre daný absolútne polohovaný element vždy najbližší z jeho predchodcov v strome dokumentu, ktorý má hodnotu `position` nastavenú na inú, než je preddefinovaná hodnota `static`.

Keďže sme vlastnosť `position` nikde inde nenastavovali, žiadny referenčný element sme neurčili. V takom prípade je referenčným elementom vždy element `body`.

To presne vysvetľuje umiestnenie loga – posunulo sa do pravého horného rohu stránky. My však chceme logo umiestniť vzhľadom k hlavičke, musíme teda z hlavičky spraviť jeho referenčný element. Pritom ale nechceme samotnú hlavičku polohovať absolútne. Pomôžeme si malým trikom – nastavíme v hlavičke vlastnosť `position` na hodnotu `relative`. Ako uvidíme, na polohu hlavičky samotnej to nebude mať vplyv (neskôr si vysvetlíme prečo), ale hlavička sa stane referenčným elementom pre absolútne polohované elementy v jej vnútri:

```
#header {
  ...
  position: relative;
}
```

Princíp absolútneho polohovania si môžeme lepšie ozrejmiť podľa obr. 12, kde vidíme názornú ilustráciu dvoch elementov, ktoré sú rôznym spôsobom polohované.



Obrázok 12 - Absolútne polohovanie

Sústreďme sa na vlastnosti reprezentujúce **súradnicové osi**: `top`, `bottom`, `left` a `right`. Vlastnosť `top` nastavuje vzdialenosť horného okraja elementu od horného okraja jeho referenčného elementu, smerom dolu. Podobne, vlastnosť `left` nastavuje vzdialenosť ľavého okraja elementu od ľavého okraja referenčného elementu, smerom doprava. Analogicky fungujú aj vlastnosti `bottom` a `right`. Vhodnou voľbou dvojice spomedzi týchto štyroch vlastností môžeme teda polohovať vzhľadom na jednu zo štyroch súradnicových sústav, ktorú si vyberieme. Začiatok každej sústavy (bod 0,0) sa nachádza v príslušnom rohu referenčného elementu.

Precvičme si absolútne polohovanie na rôznych umiestneniach loga hlavičky.

Samostatná úloha.

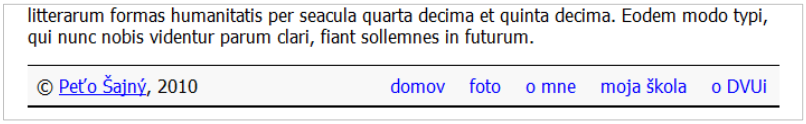
Úloha 13

S použitím absolútneho polohovania si vyskúšajte rôzne pozície loga voči hlavičke. Pokúste sa logo umiestniť postupne do všetkých štyroch rohov hlavičky. Je možné logo umiestniť aj mimo hlavičky? Pozorujte, ako sa v jednotlivých prípadoch správa logo a ako ostaný obsah, a to aj pri zužovaní stránky.

Naformátujme teraz pomocou absolútneho polohovania aj menu v päte stránky.

Úloha 14

Umiestnite menu v päte stránky doprava, podľa obr. 13. Použite absolútne polohovanie.



Obrázok 13 - Absolútne polohované menu v päte stránky

Naším cieľom je umiestniť menu v päte stránky doprava, zarovno s textom, ktorý je na ľavej strane. Keďže polohovanie má platiť len pre menu nachádzajúce sa v päte stránky, ale nie pre menu v hlavičke, musíme opäť použiť zložený selektor nasledovníka. Nesmieme zabudnúť ani na nastavenie referenčného elementu, ktorým bude

v tomto prípade päta stránky:

```
#footer .menu {
  position: absolute;
  top: 0;
  right: 0;
}
#footer {
  ...
  position: relative;
}
```

V prípade potreby nastavte päte aj primeranú výšku, či vnútorné okraje, aby výsledok pôsobil pekne a súmerne (pozri obr. 13).

Čo sme sa naučili

Zoznámili sme sa s technikou absolútneho polohovania. Táto technika nám umožňuje úplne oddeliť vybrané elementy od zvyšku dokumentu a ľubovoľne ich umiestniť na želané miesto podľa súradnicových osí. Dôležitým pojmom je referenčný element, t.j. element, vzhľadom na ktorý polohovanie prebieha. Ukázali sme si, ako môžeme tento element zvoliť sami. Naučili sme sa používať nové vlastnosti: `position`, `left`, `right`, `top`, `bottom`. Zopakovali sme si použitie: `display`, `margin`, `padding`, `width`, `height`, `background`.

2.5 Relatívne polohovanie

V predchádzajúcej časti sme sa zoznámili s absolútnym polohovaním ako s nástrojom vhodným na voľné rozmiestňovanie elementov na stránke prostredníctvom súradnicových osí. Táto technika sa hodí predovšetkým vtedy, ak chceme element umiestniť niekde úplne inde, ako bol pôvodne vykreslený. Teraz sa zoznámime s technikou relatívneho polohovania, ktorá funguje veľmi podobne, ale jej účel a význam je úplne iný: umožní nám robiť *jemné korektúry* v polohe elementov, ak to potrebujeme. Prezrime si teraz lepšie menu v hlavičke dokumentu. Jeho aktuálnu polohu zobrazuje obr. 14 (vľavo).

Obrázok 14 - Aktuálna a želaná poloha menu.

Problémom je, že menu nelícuje pekne s názvom stránky nachádzajúcim sa nad ním. Pre lepšie zvýraznenie tohto problému sme červenými prerušovanými čiarami vyznačili pomyselný vnútorný okraj hlavičky vyplývajúci opticky z polohy názvu stránky. Našim cieľom bude čo najjednoduchšie posunúť menu k červeným prerušovaným čiaram, ako to vidíme na obr. 14 (vpravo).

Na tento účel je vhodné použiť **relatívne polohovanie**, ktoré umožňuje posunúť element ľubovoľným smerom *relatívne k jeho pôvodnej polohe*. Pre aktiváciu relatívneho polohovania nastavíme vlastnosť `position` na hodnotu `relative`. Posun elementu špecifikujeme pomocou vlastností `top`, `right`, `bottom a left`, podobne, ako keď sme polohovali absolútne. Vyskúšajme si to teraz v praxi.

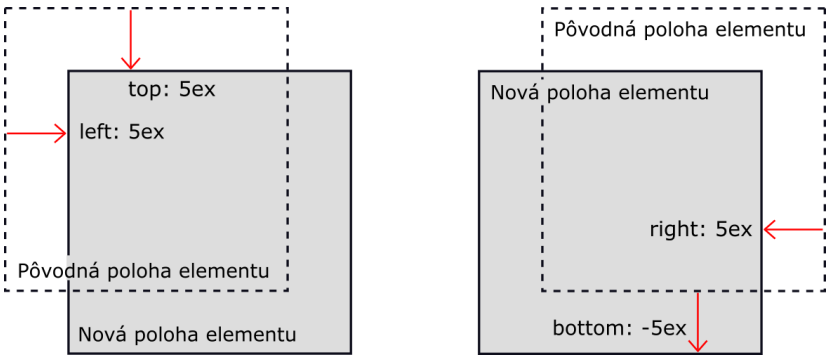
Úloha 15

Posuňte menu v hlavičke stránky dolu a doľava, aby jeho ľavý okraj zodpovedal ľavému okraju hlavného nadpisu.

Môžeme použiť napr. nasledovný kód:

```
#header .menu {
  position: relative;
  top: 1ex;
  right: 1.25ex;
}
```

Pri relatívnom polohovaní sa nemusíme starať o referenčný element, pretože vždy polohujeme vzhľadom na pôvodnú polohu elementu, kde bol umiestnený pred posunom (pozri obr. 15). T.j. ako keby bol element sám sebe zároveň referenčným elementom, resp. je ním práve pôvodná poloha elementu.



Obrázok 15 - Relatívne polohovanie

Všimnime si tiež, že pri použití vlastností `right` a `top` sa menu posunie smerom doľava a dolu. Mohlo by sa nám to zdať neintuitívne, uvedomme si však opäť, čo presne tieto vlastnosti robia. Rovnako ako pri absolútnom polohovaní, vlastnosť `right` udáva posun medzi pravým okrajom elementu a pravým okrajom jeho referenčného elementu, teda od jeho pôvodnej polohy, a to smerom doľava. Vlastnosť `top` podobne udáva posun od horného okraja referenčného elementu smerom dolu. Precvičme si techniku relatívneho polohovania ešte na nasledujúcej úlohe.

Samostatná úloha.

Úloha 16

S použitím relatívneho polohovania odsuňte celú hlavičku dokumentu o `7ex` najprv doľava, potom doprava, hore a dolu. Pozorujte, čo sa deje s hlavičkou a čo s obsahom naokolo.

Všimnime si, že keď hlavičku posúvame, zostane na jej pôvodnom mieste prázdne miesto. Všimnime si tiež, že v prípade absolútneho polohovania to tak nie je. Môžeme teda konštatovať, že medzi absolútnym a relatívnym polohovaním sú dva podstatné rozdiely:

1. absolútne polohované elementy sú z dokumentu vyčlenené, a ich miesto zaberú ďalšie elementy nasledujúce v dokumente za nimi. Pri relatívnom polohovaní ostane miesto pre element v dokumente stále vyhradené;
2. pri absolútnom polohovaní špecifikujeme nové umiestnenie elementu vzhľadom na referenčný element. Pri relatívnom polohovaní polohujeme vzhľadom na pôvodné umiestnenie elementu.

Teraz by nám už malo byť jasné, prečo sme referenčný element stanovovali pomocou deklarácie `position: relative`. Potrebovali sme to urobiť deklarovaním inej hodnoty `position` ako `static`. Práve zapnutie relatívneho polohovania (bez špecifikovania nejakého posunu) nám to umožní urobiť bez toho, aby v dokumente došlo k neželaným zmenám.

Čo sme sa naučili

Zoznámili sme sa s technikou relatívneho polohovania, ktorá nám umožňuje robiť jemné úpravy v polohe niektorého z elementov bez toho, aby to ovplyvnilo elementy naokolo. Relatívne polohovanie zapneme nastavením vlastnosti `position` na hodnotu `relative`. Zopakovali sme si vlastnosti `position`, `left`, `right`, `top`, `bottom`.

2.6 Viacstĺpcové stránky

V predchádzajúcich častiach sme sa už naučili o CSS dosť na to, aby sme dokázali vytvoriť moderný dvojstĺpcový formát stránky. Všimnime si teraz, že v zdrojovom kóde sa vo vnútri elementu div s ID `content` nachádzajú dva elementy `div` s ID `left-column` a `right-column`. V týchto dvoch elementoch sa nachádza skoro

celý obsah dokumentu, okrem hlavičky a päty. My tieto dva elementy použijeme na vytvorenie dvoch stĺpcov. Element `div` s ID `content` bude tvoriť vonkajší obal dvojstĺpcovej časti. Rozmiestnenie elementov, ktoré chceme dosiahnuť, znázorňuje nákres v stĺpci. Výsledný dizajn by mal vyzerať približne tak, ako vidíme na obr. 16.

Obrázok 16 - Ukážka dvojstĺpcového dizajnu stránky

Nákres dvojstĺpcového dizajnu

Úloha 17

Pokúste sa samostatne navrhnúť formátovanie dvojstĺpcového dizajnu blogu podľa obr. 16. Dizajn by mal vyzerať rovnako dobre v prípade oboch dokumentov, `index.html` aj `post.html`.

Na základe vedomostí získaných v prechádzajúcich častiach, by každý mal dokázať túto úlohu vyriešiť aj samostatne. V nasledujúcom texte ponúkame jedno vzorové riešenie, ktoré si predstavíme a rozoberieme. Ako prvý naformátujeme pravý stĺpec. Neskôr ho umiestnime na správne miesto. Za pomoci základných vlastností box modelu nastavíme šírku, rámik, okraje a pozadie:

```
#right-column {
  background: white url('header.png') repeat-x;
  width: 10em;
  padding: 0.5ex;
  border-top: solid black 1px;
  border-bottom: solid black 1px;
  margin: 0 0 1.5em 0.5ex;
}
```

Nastavili sme vnútorný okraj (`padding`), aby text v stĺpci nesplýval s jeho okrajom, pretože by to pri čítaní pôsobilo rušivo. Aby bol stĺpec patrične oddelený od okolitého obsahu, nastavili sme aj vonkajší okraj stĺpca (`margin`). Keďže stĺpec umiestňujeme na pravú stranu, nastavili sme mu len ľavý a dolný vonkajší okraj. Obrázok so Šajnyho „fotografiou“ v stĺpci obvyklým spôsobom vycentrujeme:

```
#right-column img {
  display: block;
  width: 9em;
  margin: 1ex auto;
}
```

Pravý stĺpec by v tejto chvíli mal vyzerať presne tak, ako vidíme na obr. 16, ale nachádza sa nad zvyšným obsahom a nie vedľa neho. Umiestnime teraz stĺpec doprava a zvyšok obsahu doľava od neho. Ak sa o to pokúsime pomocou absolútneho polohovania, narazíme na problém, že sa vám následne nepodarí naformátovať dostatočne dobre pätu stránky a umiestniť ju až pod oba stĺpce (pozri obr. 17).

Obrázok 17 - Pokazená päta stránky

Problémom pri absolútnom polohovaní je to, že nevieme, ktorý zo stĺpcov je dlhší. Keďže chceme aby sa päta stránky nachádzala až pod (oboma) stĺpcami, potrebujeme absolútne umiestniť ten kratší z nich. Ak by sme absolútne polohovali dlhší stĺpec, nastane problém, ktorý vidíme na obr. 17. Spomeňme si, že akýkoľvek absolútne polohovaný obsah je vyňatý zo statického polohovania a na nasledujúci statický obsah nijak nevplýva.

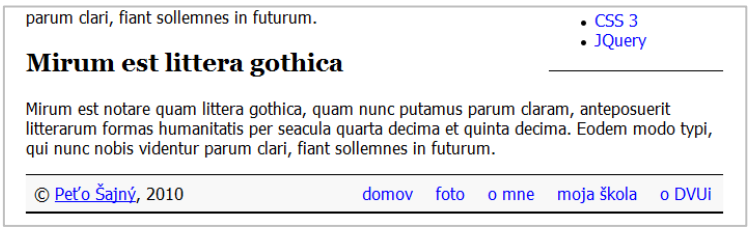
My ale chceme, aby riešenie bolo univerzálne, a aby sa päta nachádzala vždy až naspodku stránky, bez ohľadu na to, ktorý zo stĺpcov je dlhší. Preto formátovanie urobíme pomocou obtekania a pätu „upraceme“, kam patrí pomocou vlastnosti `clear`. Pri absolútnom polohovaní nemáme žiadnu obdobnú možnosť. Formátovanie bude fungovať tak, že pravý stĺpec bude obtekaný a zvyšný obsah ho bude obtekať zľava. Tým vznikne efekt pravého stĺpca. Možné sú aj iné metódy, táto má však niektoré výhody, ku ktorým sa vrátime neskôr. Doplníme preto do elementu `#right-column` vlastnosť `float`:

```
#right-column {
  ...
  float: right;
}
```

Tým sme skutočne docielili efekt dvoch súbežných stĺpcov a horná časť stránky už vyzerá takmer presne podľa predlohy. V dokumente `index.html` si ale všimneme problém v spodnej časti, ktorý je podobný tomu, na ktorý sme narazili pri pokuse sa absolútnym polohovaním (pozri obr. 17). Päta stránky nie je úplne naspodku, pretože obteká pravý stĺpec. Päta a stĺpec sa navyše prekrývajú, pretože vo vnútri päty sa nachádza absolútne polohovaný obsah. Tu práve využijeme vlastnosť `clear`. V päte stránky nastavíme `clear` na hodnotu `right` (prípadne `both`). Tým sa päta dostane na správne miesto pod oba stĺpce:

```
#footer {
  ...
  clear: right;
}
```

Keď sa však teraz pozrieme do `post.html`, zistíme, že efekt dvoch stĺpcov stále nie je úplne dokonalý. Na tejto stránke je ľavý stĺpec dlhší ako pravý. Keďže pravý stĺpec je obtekaný obsahom ľavého stĺpca, keď skončí, zvyšok ľavého stĺpca sa rozšíri na celú uvoľnenú šírku.



Obrázok 18 - Dlhší stĺpec preteká pod kratší

Ak sa nám takýto výsledok nepáči, možno tomu ľahko pomôcť. Stačí, ak ľavému stĺpcu nastavíme pravý okraj:

```
#left-column {
  margin-right: 10.5em;
}
```

Týmto je naša dvojstĺpcová stránka hotová. Výhodou tohto riešenia je, že je veľmi robustné, funguje v každom prehliadači. Je odolné a nerozpadne sa, ak v prehliadači zväčšujeme a zmenšujeme písmo. Navyše nám umožní zúžiť okno a prispôsobí sa aj užšiemu oknu či obrazovke s menším rozlíšením, napr. ak stránku otvoríme v prehliadači na mobilnom telefóne. Pri zúžení okna sa prispôsobí širší ľavý stĺpec a šírka užšieho pravého stĺpca sa nemení. Toto je žiaduce, pretože pravý stĺpec je akýmsi bočným, doplnkovým stĺpcom, ktorého šírka je už zmenšená na nutné minimum, a ďalšie zužovanie je už neprípustné. Môžeme teda skonštatovať, že sme sa naučili veľmi kvalitné a moderné riešenie pre dvojstĺpcový dizajn webových dokumentov.

Čo sme sa naučili

V tejto časti materiálu sme si prakticky vyskúšali pokročilé formátovanie webových stránok. Využili sme pri tom kombináciu techník, ktoré sme sa naučili už predtým. Oboznámili sme sa podrobne s jednou z možností pre tvorbu dvojstĺpcového dizajnu webových stránok.

Kapitola 3: Client-side programovanie

Pod client-side programovaním, či programovaním na strane klienta, ako by sme mohli tento termín lepšie preložiť do slovenčiny, rozumieme možnosť obohatenia webových stránok o programový kód, ktorý je interpretovaný prehliadačom. Keďže ide zväčša o kratšie programy, ktoré sú interpretované, nazývame ich niekedy aj client-side skripty.

3.1 JavaScript

Jediným štandardizovaným jazykom, ktorý môžeme použiť, je jazyk JavaScript (JS, tiež ECMAScript). JavaScript je pomerne jednoduchý, interpretovaný, objektovo orientovaný programovací jazyk, ktorého syntax sa podobá na C, C++, Java, PHP, či mnohé iné podobné jazyky. V tejto kapitole sa oboznámime so základnými konštruktmi JS, a vyskúšame si programovanie v JS pomocou knižnice JQuery.

Vloženie skriptu do HTML dokumentu

JavaScriptový program môžeme do webovej stránky vložiť pomocou HTML elementu `script`. Napr.:

```
<script type="text/javascript">
   /* ... programový kód ... */
</script>
```

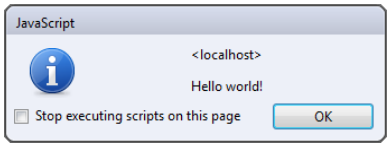
Text medzi `/*` a `*/` je v JS komentár. Jednoriadkové komentáre môžeme vkladať aj pomocou dvojice znakov `//`. V takom prípade je komentárom text od `//` až po koniec riadku.

Prípadne môžeme program uložiť do samostatného súboru. V našom prípade vytvoríme súbor s názvom `main.js`, a tento potom prilinkujeme k webovej stránke, ako vidíme v príklade:

```
<script type="text/javascript" src="main.js"/>
```

Element `script` môžeme vložiť ako do hlavičky, tak aj do tela HTML dokumentu. Prehliadač program vykoná v momente, keď ho načíta. Vyskúšajme si prvý príklad. Do hlavičky dokumentu, s ktorým sme pracovali v predchádzajúcej kapitole, vložíme nasledovný kód:

```
<script type="text/javascript">
   alert("Hello world!");
</script>
```



Obrázok 19 - Hello world, dialógové okno `alert()`

Po načítaní stránky prehliadač zobrazí dialógové okno (obr. 19), ktoré sme vyvolali funkciou `alert()`, a s oznamom, ktorý sme poslali ako parameter tejto funkcie. Všimnime si, že okno sa zobrazí skôr, ako samotná stránka. To preto, lebo sme element `script` umiestnili do hlavičky dokumentu, a teda bol vykonaný skôr, než prehliadač celý zdrojový kód spracoval a stránku zobrazil.

Takéto dialógy sú dobré, pokiaľ potrebujeme používateľovi oznámiť niečo veľmi urgentné a dôležité. To však nebýva príliš často. Vyskúšajme preto zapísať výstup zo skriptu priamo do dokumentu. Tentoraz program vložíme do súboru `main.js`, ktorý sme k dokumentu prilinkovali podľa ukážky vyššie. Do tohto súboru vložíme nasledovný kód:

```
document.write("Hello world!");
```

Výsledok po opätovnom načítaní dokumentu v prehliadači môžeme vidieť na obr. 20.

Obrázok 20 - Hello world, priamo v dokumente

Tentoraz sme využili vstavaný objekt `document`, ktorý nám umožňuje pracovať pomocou JS s dokumentom aktuálne načítaným v prehliadači. Ako môžeme vidieť, text, ktorý sme odoslali ako parameter metóde `document.write()` bol doplnený na úplný začiatok dokumentu. Na začiatku sa ocitol preto, lebo tento kód sme opäť prilinkovali do hlavičky dokumentu. Skript je možné vložiť aj na iné miesto - do tela dokumentu. V takom prípade metóda `document.write()` zapíše odovzdaný text na rovnaké miesto, kde sa nachádza element `script`, z ktorého bola volaná.

Základné konštrukty jazyka JavaScript

V JS môžeme s prehliadačom a s aktuálne načítaným dokumentom pracovať pomocou vstavaných objektov, ktoré nám prehliadač dáva k dispozícii. Ide napr. o objekt `document`, ktorý sme vyššie využili. Tento objekt umožňuje pristupovať k načítanému dokumentu, čítať, spracovávať a meniť jeho obsah. Iným dôležitým objektom je objekt `window`. Tento objekt nám sprístupňuje niektoré vlastnosti okna prehliadača. Keďže aktuálne zobrazený dokument sa tiež nachádza v okne prehliadača, aj objekt `document` a funkcia (či vlastne metóda) `alert()` sú v skutočnosti súčasťou objektu `window`. Plný tvar `window.alert()`, `window.document` a `window.document.write()` nemusíme písať, pretože JS nám umožňuje úvodné „`window.`“ vynechať.

Uvedomili sme si, že JS je objektový jazyk. S objektmi sa podrobne oboznámime neskôr. Pozrime sa teraz na niektoré základnejšie konštrukty tohto jazyka. Vyskúšajme do súboru `main.js` doplniť nasledovný kód:

```
var a, b, c;
a = 1; b = 2.5;
c = (a + b) * b / (a + b) - a;
c++;
document.write(c);

a = "Hello "; b = "world!";
c = a + b;
document.write("<p>" + c + " " + c.length + "</p>");
```

V kóde môžeme vidieť deklaráciu troch premenných pomocou kľúčového slova `var`. Následne využívame príkaz priradenia, ten zapíšeme pomocou znaku „=“ ako v jazyku C, nie „:=“ ako napr. v Pascale. Použili sme základné aritmetické operátory súčtu

(+), rozdielu (-), násobenia (*) a delenia (/), a operátor inkrementácie (++).

Ako môžeme vidieť, základná syntax nápadne pripomína C/C++ a od neho odvodené jazyky. JavaScript je však jazykom beztypovým. Premenné `a` a `b` sme najprv naplnili číselnými hodnotami a vykonávali s nimi aritmetické operácie. Neskôr sme tým istým premenným jednoducho priradili reťazcové hodnoty. Operátor `+` pracuje aj s reťazcami, pričom vykoná ich zreťazenie.

Čo však, ak je jeden z operandov číslo a druhý reťazec? V takomto prípade je najprv číslo automaticky „pretypované“ na reťazec, a potom sú oba reťazce zreťazené. Presvedčili sme sa o tom v poslednom riadku ukážky, keď sme najprv do premennej `c` uložili reťazec `"Hello world!"` a následne sme získali dĺžku tohto reťazca pomocou vlastnosti `length`. Tento číselný údaj sme potom „pripočítali“ do výstupu. Ako ale môžeme vidieť, keď si ukážku prakticky vyskúšame, v skutočnosti došlo k zreťazeniu. Pozrime sa teraz na niektoré vybrané riadiace štruktúry JS:

```
for (i=1; i<=10; i++) {
  if (i % 2 == 0) {
    // nerob nič
  } else {
    document.write("<p>" + i + "</p>");
  }
}
```

Tento kód vloží na začiatok dokumentu zoznam všetkých nepárnych čísel od jedna do desať, každé v novom odseku. V ukážke môžeme vidieť syntax cyklu `for`. V tomto konkrétnom prípade riadiacu premennú `i` inicializujeme na hodnotu `1` a následne ju v každom behu cyklu inkrementujeme, až kým nedosiahne hodnotu `10`. Vo vnútri cyklu vidíme syntax podmieneného príkazu `if-else`.

Všimnime si tiež aritmetický operátor zvyšok po delení (`%`) a predovšetkým logický operátor porovnania, ktorý zapisujeme dvojicou znakov `==`, čo je rozdiel oproti Pascalu. Operátor nerovnosti je `!=` a JavaScript tiež obsahuje príkazy `while`, `do-while` a `switch`, ktorých syntax je rovnaká ako v jazyku C. V našom krátkom tutoriáli ich nepotrebujeme, preto ich nebudeme uvádzať.

Pokiaľ ste zvyknutí na syntax jazyka Pascal, dobre si uvedomte, že v JS je priradenie `=` a porovnanie `==`. Pokiaľ omylom napíšete namiesto porovnania priradenie, prehliadač vás na to neupozorní, ale kód nebude fungovať správne (!)

Funkcie

Pozrime sa teraz na prácu s funkciami v JS. Do súboru `main.js` vložme nasledujúcu ukážku a vyskúšajme, čo sa bude diať, keď dokument načítame v prehliadači.

```
function greet(greeting) {
  alert(greeting);
}

greet("hello");

function count(a,b) {
  return a + b;
}

document.write(count(1,count(2,3)));
greet(count("hel","lo"));
```

Zadefinovali sme dve funkcie `greet()` a `count()`. Prvá z nich má jeden parameter, ktorý bez zmeny odovzdá vstavanej funkcii `alert()`. Už vieme, že funkcia `alert()` obsah parametra vypíše v dialógovom okne prehliadača. Okrem toho sme zadefinovali funkciu `count()`. Tá má dva parametre, ktoré sčíta pomocou operátora `+` a výsledok vráti ako návratovú hodnotu pomocou kľúčového slova `return`.

V dolnej časti príkladu môžeme vidieť, že vďaka univerzálnosti operátora `+` môžeme funkciu `count()` použiť tiež na zreťazenie textových hodnôt. V ďalšej ukážke uvidíme, že s funkciami môžeme v JS pracovať podobne ako s premennými. Môžeme ich napr. priradiť do niektorej z premenných a následne volať pomocou mena premennej

(t.j. ako keby sme funkciu premenovali). K predchádzajúcemu príkladu pridáme nasledovný kód:

```
var concat; concat = count;
alert(concat("aaa","bbb"));
```

Takéto manipulovanie s funkciami nám príde vhod v ďalších príkladoch, napr. hneď v nasledujúcej časti pri práci s objektmi.

Objekty

JavaScript je objektový programovací jazyk. Objekty zapuzdrujú rôzne hodnoty, ku ktorým pristupujeme pomocou znaku „.“, a ktorým sa hovorí aj **vlastnosti objektu**. S objektmi sme sa už stretli. Napr. už vieme, že každý reťazec je reprezentovaný objektom, a že dĺžku reťazca môžeme získať pomocou vlastnosti `length`. Vyskúšali sme si tiež prácu so vstavanými objektmi, ako sú `window`, či `document`. Naučili sme sa, že `window.document` je to isté ako `document`, t.j. že objekt `window` obsahuje objekt `document` ako jednu zo svojich vlastností.

Pozrime sa teraz bližšie na to, ako môžeme sami objekty vytvárať a používať:

```
var john = new Object();
john.name = "John";
john.age = 42;
document.write("<p>Hello, I'm "
  + john.name + " and I'm " + john.age + " years old.</p>");
```

Nový objekt sme vytvorili pomocou kľúčového slova `new` a funkcie `Object()`. Objekt sme zároveň uložili do premennej `john`, aby sme s ním mohli ďalej pracovať. Po tom, ako sme objekt vytvorili, sme inicializovali jeho vlastnosti `name` a `age`. Následne ku nim môžeme pristupovať pomocou `john.name` a `john.age`.

Vidíme, že objektový model v JS je veľmi pružný, nikde sme nemuseli deklarovať, aké vlastnosti bude presne daný objekt mať. Vlastnosti môžeme do objektu pridávať podľa potreby, kedykoľvek to potrebujeme. Okrem toho môže mať objekt aj metódy. **Metódy objektu** sú vlastne vlastnosti, v ktorých nie sú uložené konkrétne hodnoty, ale funkcie. Metódy primárne pracujú s dátami, ktoré sú uložené vo vlastnostiach daného objektu. V nasledujúcej ukážke vidíme priradenie metódy do objektu, ako aj volanie metódy:

```
john.greet = function() {
  document.write("<p>Hello, I'm " + this.name + " and I'm "
    + this.age + " years old.</p>");
}
john.greet();
```

Náš objekt sme rozšírili o metódu `greet()`. Použitím **anonymnej funkcie**, pomocou konštrukcie `function(){ ... }`, sme vytvorili novú funkciu, ktorú sme následne priradili do vlastnosti `john.greet`, čím vznikla nová metóda objektu `john`. Anonymné funkcie môžu mať aj parametre, v tomto prípade to ale nepotrebujeme. Vo vnútri metódy k jednotlivým vlastnostiam objektu pristupujeme pomocou kľúčového slova `this`, ktoré tu reprezentuje aktuálny objekt.

V JS sa často stretávame s objektmi, ktorých vlastnosti sú opäť objekty (spomeňme si opäť na objekt `window.document`). Takéto objekty nazývame kolekcie a môžu mať hneď niekoľko úrovní vnárania. Vyskúšajme si opäť veľmi jednoduchý príklad:

```
john.daughter = new Object();
john.daughter.name = "Lucy";
john.daughter.age = "16";
john.daughter.greet = john.greet;
john.daughter.greet();
```

Vytvorili sme objekt `john.daughter` s podobnými vlastnosťami, ako má objekt `john`

(name a age). Všimnime si, že metódu greet() sme jednoducho „skopírovali“ príkazom priradenia z objektu john. Táto metóda funguje správne aj v objekte john.daughter. Je to práve vďaka tomu, že sme v nej použili kľúčové slovo this, keď sme pristupovali k dátam. Pri volaní john.greet() teda pracuje s vlastnosťami objektu john. Následne pri volaní john.daughter.greet() však už pracuje s vlastnosťami objektu john.daughter.

Udalosti v prehliadači

Doteraz sme vždy napísali skript tak, aby sa vykonal hneď po načítaní (hlavičky) dokumentu. Toto nie je veľmi užitočné pokiaľ potrebujme spraviť niečo s už načítaným dokumentom. Alebo, čo budeme potrebovať oveľa častejšie, spustiť skript po tom, ako používateľ vykonal nejakú akciu (napr. niekde klikol). Na tento účel sú určené udalosti v prehliadači. **Udalosti** sú vyvolané nejakým podnetom, najčastejšie akciou používateľa. Takéto udalosti sú napr. onclick, onmouseover, onmouseout či onkeypress a ďalšie. Významnou výnimkou je udalosť onload, ktorá nastane po tom, ako je nejaký element načítaný (t.j. nečaká sa na akciu používateľa).

Atribúty udalostí majú rovnaké názvy ako daná udalosť, teda napr. onclick, onmouseover, onload, atď. Môže ich mať každý vizuálny HTML element. Hodnotou atribútu je ovládač – skript, ktorý sa spustí, keď udalosť v danom elemente nastane.

Pokiaľ napíšeme skript, ktorý sa vykoná po spustení niektorej udalosti, hovoríme, že sme nastavili tzv. **ovládač udalosti**. Najjednoduchším spôsobom, ako to urobiť, je využiť dedikované atribúty udalostí v HTML. Vyskúšajme najprv udalosť onload. Doplňme do nášho HTML kódu do elementu body atribút onload nasledovne:

```
<body onload="greet('ahoy');">
```

Nastavili sme teda ovládač pre udalosť onload pre element body. Ovládačom je volanie funkcie greet(), ktorú sme zadefinovali vyššie. V tomto prípade je to len jeden príkaz, ale mohol by to byť aj dlhší program. V typickom prípade si na obsluhu udalosti napíšeme funkciu, ktorú potom z atribútu udalosti voláme, aby sme množstvo JS kódu, ktorý sa nachádza priamo v HTML dokumente minimalizovali. Nižšie v časti 3.2 sa dokonca naučíme ako ho zminimalizovať na nulu. Vyskúšajme si ešte udalosť onclick. Doplňme do prvého elementu a, ktorý sa v dokumente nachádza, ovládač:

```
<a href="index.html" onclick="greet('ciao');">
  Šajnyho blýskavý blog
</a>
```

Keď stránku načítame v prehliadači, objaví sa dialógové okno s pozdravom „ahoy“. Toto okno vyvolala funkcia greet() po tom, ako nastala udalosť onload v elemente body. Všimnime si, že dialógové okno bolo naozaj vyvolané až v momente, keď už je celá stránka načítaná a zobrazená. Po tom, ako sme dialóg potvrdili kliknutím na „OK“, môžeme ďalej pracovať s načítanou stránkou. Ak klikneme na nadpis „Šajnyho blýskavý blog“, vykoná sa skript pre udalosť onclick a naša funkcia greet() nás pozdraví dialógom „ciao“. Pokiaľ dialóg potvrdíme, prehliadač zobrazí stránku odkazu, na ktorý sme klikli. Toto je ale niekedy neželané. Môžeme tomu zabrániť tak, že udalosti vrátime hodnotu false. Vyskúšajme nasledovné:

```
<a href="index.html" onclick="greet('ciao'); return false">
  Šajnyho blýskavý blog
</a>
```

V novej verzii už po kliknutí na odkaz a nasledovnom spustení ovládača pre udalosť onclick nedôjde k zobrazeniu stránky, na ktorú odkaz ukazuje. Správanie sa prehliadača po kliknutí na tento odkaz sme teda úplne predefinovali.

Doteraz sme ako výstup dokázali použiť napr. funkciu alert() a oznámiť tak niečo používateľovi za pomoci dialógového okna, alebo sme pomocou metódy document.write() zapísali výstup do dokumentu na miesto, kde sa nachádzal element script, z ktorého bola táto metóda volaná. Ani jedna z týchto možností však nie je dostatočne pružná. Neumožňuje nám skutočne meniť dokument podľa potreby. Na tento účel je určená kolekcia vstavaných objektov nazývaná objektový model dokumentu.

DOM – objektový model dokumentu

So vstavanými objektmi DOM sa oboznámime len veľmi stručne aby sme videli, akým spôsobom je práca s DOM v štandardnom JS implementovaná. V nasledujúcej časti sa naučíme pracovať s DOM oveľa efektívnejšie pomocou knižnice JQuery.

Na čítanie dát z dokumentu, ako aj na jeho ľubovoľnú úpravu, sa používa tzv. DOM (z angl. Document Object Model, t.j. objektový model dokumentu). DOM nám ponúka kolekciu objektov, ktoré reprezentujú prehliadačom aktuálne načítaný dokument. S týmito objektmi môžeme pracovať podľa ľubovôle, a teda ich aj meniť. Kolekciu DOM objektov zastrešuje objekt `document`, s ktorým sme už pracovali. Vyskúšajme si malú ukážku manipulácie s dokumentom pomocou DOM:

```
function change() {
  document.images[1].src="dvui-logo.png";
  document.getElementById("header").innerHTML
    = "<p><em>Hello</em> world!</p>";
}
```

Funkcia `change()` nastavuje vlastnosti hneď dvom objektom. Prvý z nich je objekt `document.images[1]`, ktorý odpovedá druhému obrázku v dokumente v poradí od začiatku dokumentu (číslovanie začína od nuly). Pomocou vlastnosti `src` nastavíme novú zdrojovú URI adresu obrázka. Ďalší objekt získame volaním metódy `document.getElementById()`, ktorá vráti objekt reprezentujúci element so zadaným ID. Keďže táto metóda vracia objekt, môžeme rovno pristúpiť k niektorej z jeho vlastností. Vlastnosť `innerHTML` nám umožňuje načítať alebo prepísať HTML kód vo vnútri elementu.

Pripomeňme si, že v HTML nastavujeme ID pomocou atribútu `id`. Dva elementy s rovnakým ID nie sú dovolené, takže pomocou metódy `getElementById()` vždy získame práve jeden element.

Kód, ktorý sme napísali, zatiaľ nerobí nič, keďže sme zatiaľ funkciu iba zadefinovali. Nastavme teraz volanie funkcie po udalosti `onclick`. Využijeme ten istý element `a` ako vyššie:

```
<a href="index.html" onclick="change(); return false">
  Šajnyho blýskavý blog
</a>
```

Po kliknutí na odkaz „Šajnyho blýskavý blog“ (obr. 21, dolu) v nadpise môžeme vidieť dve zmeny, odpovedajúce kódu, ktorý sme napísali. Obrázok v stĺpci sa zmení na logo DVUi, a obsah hlavičky je nahradený textom „*Hello* world!“. Vidíme aj to, že HTML, ktoré sme vložili do elementu `a` je interpretované - prvé slovo sa zobrazilo kurzívou - tak, ako sme to zadefinovali vo funkcii `change()`.

Obrázok 21 - Manipulácia s DOM

Čo sme sa naučili

Zoznámili sme sa so základnými princípmi programovania na strane klienta a s jazykom JavaScript. Naučili sme sa vložiť skript do HTML dokumentu. Naučili sme sa tiež používať premenné, riadiace príkazy `for` a `if`, funkcie a objekty. Dozvedeli sme sa, že skripty sú v prehliadači volané zväčša prostredníctvom udalostí reagujúcich na akcie používateľa. Vieme tiež, že na prácu s dokumentom slúži DOM a videli sme príklad jeho využitia.

3.2 JavaScriptové frameworky a knižnica JQuery

V predchádzajúcej časti sme sa zoznámili so základnými konštruktmi programovacieho jazyka JavaScript a vyskúšali sme si niekoľko krátkych ukážok práce s ním. V praktickom využití JavaScriptu nám ale zatiaľ bráni fakt, že jeho implementácie v rôznych prehliadačoch, dokonca v ich rôznych verziách, nie sú úplne kompatibilné. Ak používate niektorý iný prehliadač ako Microsoft Internet Explorer, ktorý má majoritný podiel na trhu, určite sa vám už stalo, že niektorá webová stránka vo vašom prehliadači nefungovala. Je to spôsobené tým, že vývojári stránku otestovali len v najpoužívanejšom prehliadači a na váš prehliadač jednoducho zabudli.

Musíme teda akceptovať fakt, že podpora JavaScriptu, DOM, ale aj CSS v prehliadačoch nie je jednotná. Hoc sa aj tento problém postupne zlepšuje, stav stopercentnej kompatibility zrejme nikdy nedosiahneme. Preto je pri tvorbe webových stránok nutnosťou testovať vo viacerých prehliadačoch.

V tejto časti sa zoznámime s novou metodikou, ktorá nám výrazne uľahčí prácu s JavaScriptom a umožní nám kód písať tak, aby sme jeho kompatibilitu mali dopredu z veľkej časti zaručenú (otestovať výsledok vo viacerých prehliadačoch by sme, pre istotu, tak či tak mali). Takúto výhodu nám prinášajú tzv. **JavaScriptové frameworky**. Ide o JavaScriptové knižnice, ktoré nám sprostredkúvajú prácu s DOM, (a teda v konečnom dôsledku s jednotlivými HTML elementmi, ktoré webovú stránku tvoria) nepriamo, pomocou objektov, funkcií a metód, ktoré sú súčasťou knižnice. Nebudeme teda pracovať s objektmi DOM, ktoré nám poskytuje priamo prehliadač, ako je napr. kolekcia `document.images[]`, či objekty, ktoré sme získali pomocou volania metódy `document.getElementById()`, tak ako sme to videli v príkladoch vyššie. Miesto toho použijeme nové objekty, ktoré nám poskytne nami zvolená knižnica a budeme pracovať s ich vlastnosťami, či metódami.

Výhodou tohto nového prístupu je, že vývojári JS frameworku už za nás odviedli veľkú časť práce. Konkrétne, naprogramovali framework tak, že po volaní niektorej z jeho metód, či nastavení hodnoty vlastnosti, knižnica najprv automaticky zistí typ a verziu prehliadača, a následne kód vykoná spôsobom kompatibilným a optimálnym pre daný prehliadač. Uvidíme tiež, že vhodná voľba kvalitného JS frameworku nám zároveň umožní vykonať nami zamýšľané akcie jednoducho a elegantne.

Knižnica JQuery

V našom krátkom tutoriáli sa zoznámime s knižnicou JQuery. Ide o kvalitný, voľne dostupný JS framework, zaručujúci vysokú kompatibilitu so všetkými aktuálne relevantnými prehliadačmi. Navyše, JQuery kladie dôraz na efektivitu kódu, a jednoduchosť práce so samotnou knižnicou, o čom sa v krátkosti sami presvedčíme.

Po tom, ako sme si knižnicu JQuery stiahli (viď stĺpec) a uložili sme ju do pracovného priečinka webovej stránky pod názvom `jquery.js`, prilinkujeme ju k HTML dokumentu, ako každý iný skript:

```
<script type="text/javascript" src="jquery.js"/>
```

Dáme si pritom pozor, aby sme tento element `script` vložili vyššie, než akýkoľvek iný skript, v ktorom chceme JQuery použiť. V nasledujúcej ukážke uvidíme ako možno **JQuery inicializovať**. Do dokumentu vložíme nasledovný kód:

```
<script type="text/javascript">
  $(document).ready(function() {
    alert("Hello world!");
  });
</script>
```

Ako sa môžeme presvedčiť, po načítaní dokumentu sa zjaví nám dobre známe dialógové okno s pozdravom „Hello world!“. Všimnime si, že pod týmto dialógovým oknom vidíme celý načítaný a zobrazený dokument. Funkcia `alert()` bola teda volaná až udalosťou `onload`, alebo nejakou neskoršou. Poďme sa teraz pozrieť bližšie na to, čo sa presne stalo.

Knižnicu si možno stiahnuť z http://jquery.com/

Váš štartovací balíček už knižnicu JQuery obsahuje, takže ju sťahovať nemusíte.

Funkcia `$()` sa nazýva **JQuery konštruktor.** Táto funkcia vracia **JQuery objekt** prislúchajúci k elementu na stránke, ktorý identifikujeme jej parametrom. Tentoraz sme jej odovzdali DOM objekt `document`, preto volanie `$(document)` vrátilo JQuery objekt zapuzdrujúci celý dokument, s ktorým pracujeme. Celá funkcionalita JQuery je následne dostupná cez vlastnosti a metódy, ktoré má každý JQuery objekt.

My sme volali metódu `ready()` tohto JQuery objektu, ktorá očakáva ako parameter funkciu a túto funkciu nastaví ako ovládač udalosti `ready` objektu `document`. Táto udalosť je veľmi podobná udalosti `onload`. JQuery spustí udalosť `ready` bezprostredne potom, ako je dokument načítaný prehliadačom a samotná knižnica je plne zinicializovaná.

Keďže metóda `ready()` vyžaduje ako parameter funkciu, túto funkciu sme vytvorili pomocou konštruktora anonymnej funkcie (pozri časť 3.1). V ďalšom pokračovaní tutoriálu uvidíme, že prácu s JQuery vlastne vždy začíname presne takto. Vytvoríme vždy anonymnú funkciu, do ktorej umiestnime všetok kód, ktorý potrebujeme, a túto funkciu následne nastavíme ako ovládač pre udalosť `$(document).ready()`. Môžeme sem umiestniť kód ktorý sa vykoná hneď, alebo môžeme priamo pomocou JQuery nastaviť ovládače pre rôzne ďalšie udalosti podľa potreby.

Manipulácia dokumentu

Pri práci s JQuery sa nám bude hodiť referenčná príručka [7] ako aj ďalšia dokumentácia k tejto knižnici [6,5].

Vyskúšajme si teraz niekoľko príkladov, ktoré demonštrujú, ako pomocou JQuery môžeme pracovať s DOM.

Úloha 18

Dovnútra `$(document).ready(function(){ ... })` vložte postupne po jednom nasledujúce riadky. Dokument zakaždým otvorte v prehliadači a pozorujte, čo sa bude diať:

```
$("#header").html("Hello world!")
$("img").attr("src", "dvui-logo.png");
```

V prvom prípade dôjde k zmene obsahu hlavičky. Namiesto našej obvyklej hlavičky, sa objaví len text „Hello world!“, ako vidíme na obrázku.

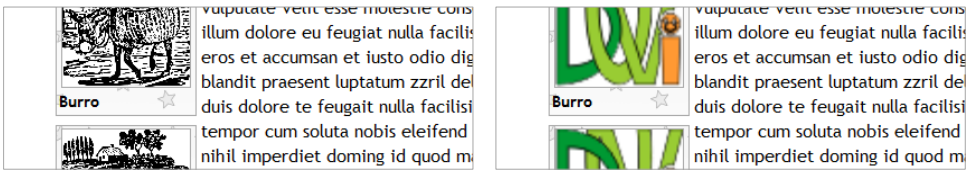
Obrázok 22 - Zmena obsahu cez JQuery

Opäť sme využili JQuery konštruktor `$()`, tentoraz sme mu ale neodovzdali ako parameter DOM objekt, ale reťazec `"#header"`. Toto je druhý spôsob inicializácie JQuery objektu, ktorý budeme využívať oveľa častejšie. Reťazec obsahuje takzvaný **JQuery výraz** (angl. JQuery expression). Keď sa lepšie prizrieme výrazu, ktorý sme použili, zistíme, že vyzerá rovnako ako ID selektor v CSS. Keďže sme si kód už vyskúšali, vieme, že aj pracuje rovnako ako tento CSS selektor. Volanie `$("#header")` nám skutočne vrátilo JQuery objekt odpovedajúci HTML elementu s ID `header`, čiže v našom prípade element predstavujúci hlavičku stránky.

JQuery výrazy sú obdobou CSS selektorov. V skutočnosti ide o rozšírenie CSS selektorov o ďalšie užitočné možnosti. Pre nás ale bude podstatné, že všetky CSS selektory, tak ako sme sa ich naučili v module 3DG2 [8] a ako sme ich používali v kapitole 2, fungujú aj ako JQuery výrazy.

V ďalšej časti tutoriálu budeme často pracovať tak, že pomocou JQuery výrazu a konštruktora `$()` skonštruujeme JQuery objekt zodpovedajúci niektorému elementu na stránke a následne s týmto elementom niečo urobíme pomocou volania metód, ktoré nám JQuery objekt dáva k dispozícii.

Druhý riadok z nášho príkladu spôsobí, že všetky obrázky, ktoré sa nachádzajú na stránke sa ihneď po načítaní zmenia na obrázok `dvui-logo.png`. Na obr. 23 vidíme výrez zo stránky pred a po zmene.



Obrázok 23 - Zmena atribútu cez JQuery

V tomto prípade sme použili JQuery výraz `"img"`, ktorý podobne ako v CSS vráti JQuery objekt zapuzdrujúci všetky HTML elementy `img` nachádzajúce sa v dokumente. Keď sme potom volali nejakú metódu tohto objektu, manipulovali sme so všetkými týmito objektmi naraz, podobne ako keby sme v CSS napísali pravidlo so selektorom `img`.

Pozrime sa teraz na metódy, ktoré sme použili na **manipuláciu s dokumentom**. Zatiaľ sme pracovali s metódami `html()` a `attr()`. Tieto metódy má každý JQuery objekt. Môžu mať nasledovný tvar:

- `html()` - vracia HTML obsah elementu;
- `html(text)` - nastaví HTML obsah elementu na novú hodnotu určenú parametrom `text`;
- `attr(name)` - vracia hodnotu atribútu s menom `name`;
- `attr(name,value)` - nastaví atribút s menom `name` na novú hodnotu `value`.

Pomocou týchto metód môžeme teda jednoducho prečítať ľubovoľnú časť dokumentu, a to buď vnútro nejakého elementu alebo hodnotu niektorého z atribútov elementu. Rovnako dobre však môžeme aj meniť obsah elementov a atribútov podľa potreby. Vyskúšajme teraz o niečo zložitejší príklad.

Úloha 19

Dovnútra `$(document).ready(function(){ ... })` vložme a jeden po druhom v prehliadači vyskúšajme tieto riadky:

```
$("h2:first").css("background", "red");
$("h2 + p a:last").addClass("red");
```

Tentoraz sme použili trochu komplikovanejšie JQuery výrazy. Prvý výraz `"h2:first"` selektuje prvý zo všetkých elementov `h2` v dokumente. Druhý výraz je ešte zložitejší. Výraz `"h2 + p a"` selektuje všetky elementy `a`, ktoré sa nachádzajú vo vnútri nejakého elementu `p`, ktorý nasleduje za elementom `h2`. O niečo zložitejší výraz `"h2 + p a:last"` následne zo všetkých týchto elementov `a` vyberie len ten posledný.

Všimnime si, že tento JQuery výraz opäť využíva CSS selektory. Tentoraz ide o selektory potomka a nasledovníka. Pozri časť 2.1.

Pozrime sa teraz na metódy, ktoré sme použili. Ide o **metódy pre prácu s CSS**, ktoré je prepojené s elementom odpovedajúcim JQuery objektu. Môžeme si vybrať z nasledovných metód:

- `css(property)` - vráti reťazec zodpovedajúci aktuálne nastavenej hodnote CSS vlastnosti `property`;
- `css(property,value)` - nastaví CSS vlastnosti `property` novú hodnotu `value`;
- `addClass(class)` - pridá elementu CSS triedu `class`;
- `removeClass(class)` - odoberie elementu CSS triedu `class`;
- `toggleClass(class)` - prepne elementu CSS triedu `class` (ak element už má túto triedu, tak mu ju odoberie, ak ju nemá, triedu mu pridá).

V našom príklade sme teda v prvom riadku nastavili prvému nadpisu úrovne `h2` červené pozadie priamo nastavením CSS vlastnosti `background` na hodnotu `red`. V druhom riadku sme si na stránke vyhľadali jeden z odkazov a zaradili sme ho do CSS triedy `red`, ktorú sme vytvorili v predchádzajúcej kapitole, t.j. následne mal tiež červené pozadie a k tomu ešte aj rámik.

CSS triedu `red` sme zadefinovali v časti 2.1. Pozri str. 6.

Udalosti a efekty

V časti 3.1 sme sa zoznámili s udalosťami v prehliadači. Vieme, že prehliadač reaguje na niektoré akcie používateľa spustením udalosti. K udalosti môžeme zadefinovať ovládač udalosti, teda skript, ktorý bude vykonaný, keď udalosť nastane. Pôvodne sme ovládače udalostí nastavovali pomocou HTML atribútov, napr. `onload`, `onclick`, a pod. Ovládače môžeme nastavovať aj oveľa praktickejšie a efektívnejšie pomocou JQuery **metód pre prácu s udalosťami**.

Už sme sa zoznámili s metódou `ready()`, nastavujúcou ovládač pre špeciálnu JQuery udalosť `ready`, ktorá nám umožnila inicializovať JQuery kód až potom, ako je dokument načítaný prehliadačom a knižnica JQuery inicializovaná. Ukážeme si ešte dve ďalšie metódy pracujúce s udalosťami:

- `click(fn)` - očakáva ako parameter funkciu, ktorú nastaví ako ovládač pre udalosť `onclick` elementu reprezentovaného JQuery objektom. Funkcia `fn` bude teda vykonaná po každom kliknutí na daný element;
- `hover(fn1,fn2)` - očakáva dve funkcie, ktoré nastaví ako ovládače pre udalosti `onmouseover` (kurzorom prejdeme nad element) a `onmouseout` (kurzorom opustíme element).

Vyskúšajme si teraz sami obohatiť náš príklad o ovládače udalostí.

Úloha 20	Napíšte JQuery kód, ktorý po kliknutí na ľubovoľný obrázok vymení všetky obrázky v dokumente za logo DVUi.

Úlohu môžeme vyriešiť napr. nasledovne. Nezabudnime na to, že JQuery kód je potrebné vložiť dovnútra `$(document).ready(function(){ ... })`, teda dovnútra ovládača pre udalosť `ready`, pre celý dokument:

```
$("img").click(function() {
  $("img").attr("src","dvui-logo.png");
});
```

Najprv sme pomocou JQuery konštruktora získali JQuery objekt odpovedajúci množine všetkých obrázkov v dokumente. Volaním metódy `click()` tohto objektu sme mu nastavili ako ovládač anonymnú funkciu, ktorá sa postará o zmenu obrázka. Toto nastavenie je platné pre všetky elementy `img`, ako si môžeme sami vyskúšať klikaním na jednotlivé obrázky. T.j. všetky obrázky majú odteraz takto nastavený ovládač udalosti `onlick`, a po prvom kliknutí bude zmena vykonaná na všetkých obrázkoch súčasne.

Určite by sa nám hodilo vykonať niečo len s elementom, ktorý udalosť vyvolal. Tu sa nám opäť hodí kľúčové slovo `this`. Vo vnútri metódy JQuery objektu `this` referuje na element (DOM objekt), ktorý k objektu prislúcha. My ale potrebujeme JQuery objekt prislúchajúci k elementu. Ten môžeme opäť poľahky skonštruovať pomocou JQuery konštruktora `$()`. Želaný efekt dosiahneme nasledovným kódom:

```
$("img").click(function() {
  $(this).attr("src","dvui-logo.png");
});
```

Po tejto úprave majú opäť všetky obrázky nastavený ovládač pre udalosť `onclick`, ale zmena sa vždy vykoná len na obrázku, na ktorý sme klikli. Poďme si teraz vyskúšať metódu `hover()`, ktorá reaguje na pohyb kurzora myši.

Úloha 21	Upravte JQuery kód z prechádzajúcej úlohy tak, aby sa každý obrázok zmenil na logo DVUi nie po kliknutí, ale po tom, ako nad ním umiestnime kurzor myši. Potom ako obrázok kurzorom opustíme, zmení sa späť na pôvodný.

Vyriešme najprv prvú časť úlohy, t.j. zmenu obrázka po prejdení kurzorom myši. Použijeme metódu `hover()`, ktorá, ako vieme, očakáva ako parametre dve funkcie:

```
$("img").hover(function() {
  $(this).attr("src","dvui-logo.png");
}, function() {

});
```

Zatiaľ sme vyriešili iba prvú časť úlohy, a tak sme druhú funkciu ponechali prázdnu. Vyskúšajte stránku načítať a prejdite ponad obrázkami kurzorom myši. Všetky sa zmenia na logo DVUi a už zmenené zostanú. Keď teraz chceme vyriešiť aj druhú časť, máme problém.

Keď totiž atribút `src` zmeníme na novú hodnotu, stará sa stratí – nemáme ju nikde odloženú a zapamätanú. Pomôžeme si faktom, že jednotlivé elementy sú reprezentované DOM objektmi, a že prislúchajúci DOM objekt máme vždy poruke vďaka kľúčovému slovu `this`. V JS môžeme objekty upravovať podľa ľubovôle vrátane pridávania vlastností a metód. A tak teda po prejdení obrázka kurzorom myši zapíšeme pôvodnú hodnotu atribútu `src` do novej vlastnosti `srcOrig`, ktorú vytvoríme. Už vieme, že DOM objekt priradený k obrázku získame pomocou kľúčového slova `this`. Výsledný kód vyzerá nasledovne:

```
$("img").hover(function() {
  this.srcOrig = $(this).attr("src");
  $(this).attr("src","dvui-logo.png");
}, function() {
  $(this).attr("src",this.srcOrig);
});
```

Uvedomme si teraz dobre, aký je rozdiel medzi objektmi `this` a `$(this)`. Prvý z týchto objektov je DOM objektom, ktorý reprezentuje element, a je *v prehliadači stále prítomný*. Preto doňho môžeme doplniť novú vlastnosť `srcOrig`, a keď po chvíli pohneme kurzorom myši von z obrázka, ten istý objekt, s tou istou vlastnosťou a s tou istou hodnotou tam stále sú. Naproti tomu objekt `$(this)` je pri každom výskyte udalosti *vždy nanovo skonštruovaný* JQuery objekt, ktorý je akousi „nadstavbou“ nad DOM objektom `this`. Preto, ak by sme sa pokúsili vlastnosť srcOrig priradiť tomuto objektu, neuspejeme. Pri pokuse o načítanie tejto hodnoty pri odchode kurzorom myši z elementu je objekt `$(this)` nanovo skonštruovaný a táto hodnota v ňom pochopiteľne nie je.

Existujú ešte mnohé ďalšie metódy, pracujúce s udalosťami, ktorým sa však pre krátkosť nášho tutoriálu nemôžeme venovať. Tieto metódy nám umožňujú priamo v JQuery kóde nastavovať ovládače pre akékoľvek udalosti prehliadača a doplniť dokument o programovú funkcionalitu. Navyše, takýmto spôsobom ovládače dopĺňame bez toho, aby sme akýmkoľvek spôsobom zasahovali do HTML kódu a dopĺňali doň atribúty, či viac elementov `script`. Vďaka JQuery je všetok potrebný kód zapuzdrený v samostatnom súbore, ktorý je k dokumentu prilinkovaný. Podobne ako pri CSS, kde sme sa snažili docieliť oddelenie štruktúry a prezentácie, docieľujeme tu *oddelenie štruktúry od funkcionality*. Výhodou je univerzálnosť a ľahká údržba kódu.

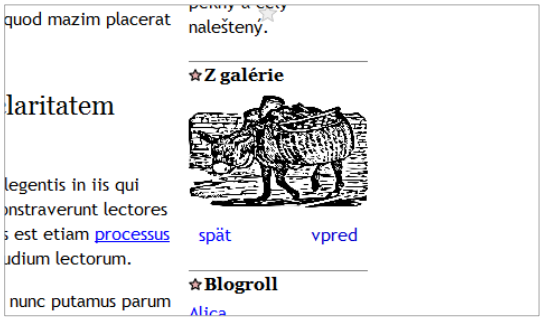
Praktický príklad: Náhľad do galérie

Doteraz sme si mohli vyskúšať prácu s JS a s JQuery na niekoľkých, nie príliš zmysluplných príkladoch, ktoré nám poslúžili na to, aby sme sa s týmito technológiami zoznámili, ale ich praktický význam je malý. Vyskúšajme si teraz prácu s JQuery na o niečo zložitejšom, zato zmysluplnom a praktickom príklade.

Do pravého stĺpca nášho blogu implementujme malý náhľad do galérie, aký vidíme na obr. 24. Bude pozostávať z jedného obrázka a dvoch tlačidiel, pomocou ktorých môžeme prepínať medzi viacerými obrázkami. Tieto sa zobrazia vždy na tom istom mieste. Našim cieľom bude, aby po kliknutí na linky „späť“ a „vpred“ bol obrázok vymenený za ďalší, resp. predchádzajúci obrázok z galérie. Obrázky máme predpripravené v priečinku `gallery`. Všimnime si, že majú vhodný názov: `img1.png` až `img5.png`. Toto názvoslovie v príklade šikovne využijeme a poľahky vypočítame poradové číslo a názov ďalšieho obrázka, ktorý sa má načítať.

Úloha 22

Pomocou JQuery dogenerujte do elementu s ID `gallery`, ktorý sa nachádza v dokumente, potrebný kód, aby sa zobrazil náhľad ako to vidíme na obr. 24.



Obrázok 24 - Náhľad do galérie v stĺpci

Potrebný kód dogenerujeme do stránky pomocou JQuery, akonáhle je dokument inicializovaný. Už vieme, že je potrebné nastaviť ovládač udalosti `ready` pre JQuery objekt reprezentujúci celý dokument. Následne využijeme fakt, že v dokumente už máme predpripravený element `div` s ID `gallery`, do ktorého dogenerujeme potrebný obsah. Okrem toho inicializujeme dve pomocné premenné, s ktorými budeme pracovať. Premenná `imgID` obsahuje číslo aktuálne zobrazeného obrázku a premenná `imgCount` obsahuje celkový počet obrázkov v galérii.

```
$(document).ready(function() {
  $("#gallery").html(
    '<h2>Z galérie</h2>'
    +'<span>'
    +'<img src="gallery/img'+ imgID +'.png" width="100"'
         +'height="100"/>'
    +'<a href="#" id="gLeft">späť</a>'
    +'<a href="#" id="gRight">vpred</a>'
    +'</span>'
  );
  var imgID = 1;
  var imgCount = 5;
});
```

Okrem HTML kódu potrebujeme ešte CSS kód. Môžete sa pokúsiť naformátovať náhľad v stĺpci sami. Ak si s niečím neviete rady, inšpirujte sa nasledovným kódom:

```
#gallery span {
  display: block;
  position: relative;
  height: 17ex;
}
#gallery img {
  display: block;
  width: 100%;
}
#gallery a#gLeft {
  position: absolute;
  bottom: 0;
  left: 0;
}
#gallery a#gRight {
  position: absolute;
  bottom: 0;
  right: 0;
}
```

Potrebný HTML kód sme vygenerovali a doplnili sme CSS pravidlá, ktoré ho formátujú. Môžeme sa teda pustiť do jeho oživovania.

Úloha 23

Napíšte v JS kód, ktorý pomocou JQuery nastaví taký ovládač pre udalosť `onclick` na elemente `a#gRight`, ktorý po kliknutí vymení obrázok v náhľade za nasledujúci v poradí.

Najprv pomocou JQuery výrazu nájdeme v dokumente správny odkaz. Tentoraz je to jednoduché, lebo poznáme ID elementu, ktorý hľadáme. Nájdenému elementu nastavíme ovládač pre udalosť `onclick`:

```
$("#gRight").click(function(){
  imgID = (imgID % imgCount) + 1;
  $("#gallery img").attr("src","gallery/img" +imgID+ ".png");
  return false;
});
```

V ovládači najprv inkrementujeme hodnotu premennej `imgID`. Nechceme ju však zvýšiť vyššie, než je celkový počet obrázkov. Preto použijeme operátor `%`, aby sme v takom prípade preskočili späť na obrázok číslo 1. Následne pomocou JQuery výrazu nájdeme správny element `img` v dokumente a zmeníme mu atribút `src`. Nájsť tento element je opäť jednoduché, keďže v elemente `div` s ID `gallery` je len jeden element `img`. Na záver vrátime hodnotu `false`. Ako už vieme, zabránime tým tomu, aby bolo kliknutie interpretované prehliadačom štandardne, t.j. aby prehliadač prešiel na stránku kam smeruje odkaz.

Ďalej oživíme kliknutie na „späť“.

Úloha 24

Napíšte v JS kód, ktorý pomocou JQuery nastaví taký ovládač pre udalosť `onclick` na elemente `a#gLeft`, ktorý po kliknutí vymení obrázok v náhľade za predchádzajúci v poradí.

Úloha je v podstate analogická k predchádzajúcej úlohe. Jediným rozdielom je, že teraz musíme premennú `imgID` dekrementovať, a to už nevieme urobiť tak elegantne (keďže opäť nechceme prejsť pod minimálnu hodnotu 1):

```
$("#gLeft").click(function(){
  if (imgID == 1) {
    imgID = imgCount;
  } else {
    imgID--;
  }
  $("#gallery img").attr("src","gallery/img" +imgID+ ".png");
  return false;
});
```

Problém môžeme ale poľahky vyriešiť podmieneným príkazom, ako vidíme vyššie. Ak je aktuálna hodnota rovná `1`, nastavíme maximálnu hodnotu (nezabudnite že pre porovnanie musíme použiť dva znaky `==`). V opačnom prípade dekrementujeme.

JQuery podporuje aj niekoľko jednoduchých efektov, ktoré nám umožnia oživiť našu aplikáciu. Medzi **JQuery efekty** patria napr. dve metódy, ktorými môžeme vybraný element na stránke schovať a opäť zobraziť:

- `hide()` - umožňuje skryť element zodpovedajúci JQuery objektu. Element ostane súčasťou stránky, ale jeho zobrazovanie bude vypnuté nastavením CSS vlastnosti `display: none`.
- `show()` - zobrazí element, ktorého zobrazenie je momentálne vypnuté.

Okrem toho existuje aj metóda `toggle()`, ktorá skryje element, ak je zobrazený, a zobrazí element ak je skrytý. Metódami `hide()` a `show()` je možné docieliť aj jednoduchý animačný efekt, pokiaľ metóde nepovinným parametrom povieme, ako rýchlo má element skryť/zobraziť. Tento parameter môže mať hodnoty `"slow"`

(pomaly) a "fast" (rýchlo). Vyskúšajme si teraz efekt pomalej animácie pri prepínaní obrázkov v našom náhľade. Najprv element skryjeme, potom zmeníme hodnotu atribútu src na novú, a až následne element pomaly opäť zobrazíme za pomoci animácie. Pozrime sa na ukážku akcie „vpred“:

```
$("#gRight").click(function(){
  imgID = (imgID % imgCount) + 1;
  $("#gallery img").hide();
  $("#gallery img").attr("src","gallery/img" +imgID+ ".png");
  $("#gallery img").show("slow");
  return false;
});
```

Animáciu pre akciu „späť“ možno implementovať veľmi podobne.

Samostatná úloha.

Úloha 25 Naprogramujte samostatne analogicky aj animáciu pre akciu „späť“.

JQuery podporuje ešte niektoré **ďalšie efekty**. Nasledovné metódy nám rovnako umožňujú niektorý element skryť a opätovne zobraziť, ale animácie sú odlišné:

- slideUp() - element je ukrytý vyrolovaním smerom dohora;
- slideDown() - ukrytý element je zobrazený vyrolovaním zhora nadol;
- fadeOut() - element je postupne zneviditeľnený;
- fadeIn() - neviditeľný element je opäť zviditeľnený.

Všetky štyri metódy podporujú reguláciu rýchlosti pomocou voliteľného parametra s hodnotami "slow" a "fast", ako sme to videli u metód show() a hide(). Práca s týmito animáciami je naozaj jednoduchá. Môžeme ich vyskúšať priamo na našej jednoduchej aplikácii.

Samostatná úloha.

Úloha 26 Postupne vyskúšajte pre prepínanie medzi obrázkami v galérii všetky druhy efektov. Nastavte pomalú aj rýchlu animáciu.

Postupne sme pomocou JQuery naprogramovali jednoduchú client-side aplikáciu. Vyskúšali sme si generovanie kódu, zmenu atribútov, ako aj prácu s udalosťami a jednoduché JQuery efekty. S efektmi by sme mali pracovať opatrne, príliš veľa animácie môže byť viac na škodu, než na úžitok. Animované efekty sa hodia predovšetkým na to, aby sme upozornili používateľa, že niekde na stránke bol dodatočne vygenerovaný nejaký dôležitý obsah. O tom, kedy a ako správne používať JS sa viac dočítame v nasledujúcej časti.

Čo sme sa naučili

Zoznámili sme sa JavaScriptovou knižnicou JQuery, ktorá umožňuje podstatne efektívnejšiu prácu s JavaScriptom a zároveň zaručuje kompatibilitu s rôznymi prehliadačmi. Naučili sme sa manipulovať s jednotlivými HTML elementmi pomocou JQuery objektov, ktoré konštruujeme pomocou JQuery konštruktora $() a JQuery výrazov. Výhodou JQuery výrazov je, že pripomínajú CSS selektory, s ktorými sme sa podrobne oboznámili v kapitole 2 ako aj v module 3DG2. Zároveň sme sa zoznámili s efektívnym spôsobom ako nastavovať ovládače udalostí, a tak teda pomocou JQuery dokážeme tvoriť veľmi jednoduché JavaScriptové programy bežiace na strane klienta.

3.3 Ako správne používať JavaScript

V module 3DG2 sme sa oboznámili so zásadami prístupnosti a použiteľnosti webových stránok a s potrebou ich pri tvorbe webu dodržiavať. Client-side programovanie a JavaScript sú technológie, ktoré obohacujú webové stránky o interaktivitu. Pri takýchto webových stránkach musíme venovať kontrole kvality zvýšenú pozornosť, pretože môžu klásť zvýšené nároky na možnosti a schopnosti používateľa, ako aj na jeho softvérové a hardvérové vybavenie. Ľahko sa môže stať, že používateľ našu JS fun-

kcionalitu nemôže vôbec použiť tak, ako si to my predstavujeme, a to najmä ak sa vyskytne jeden z nasledovných troch problémov:

- **JavaScript nie je dostupný:** webové prehliadače umožňujú JS deaktivovať, a niektorí používatelia si ho z bezpečnostných dôvodov deaktivujú. Niektoré počítačové vírusy a červy totiž využívajú JS a kombináciu známych bezpečnostných dier v prehliadačoch. Ešte častejšie zablokuje JS na všetkých počítačoch na pracovisku zamestnávateľ;
- **nekompatibilita prehliadačov:** implementácia JS a DOM nie je vo všetkých prehliadačoch navzájom kompatibilná. Ak sme skript testovali len v jednom prehliadači, nemusí byť zaručené, že funguje rovnako dobre aj v ostatných prehliadačoch a v ich rôznych verziách;
- **prílišná interaktivita:** pomocou JS dokážeme tvoriť inovatívne používateľské rozhrania so značnou mierou interaktivity. Pokiaľ sa pri tom ale odkloníme od zaužívaných používateľských rozhraní alebo ak to s interaktivitou preženeme, môže to byť viac na škodu – používateľ naše rozhranie nepochopí a nebude vedieť stránku použiť.

Preto by sme mali pri tvorbe webových stránok, ktoré obsahujú JS (či akékoľvek iné interaktívne prvky) vyššie uvedené problémy zohľadniť. **Testovať** by sme vždy mali vo všetkých hlavných prehliadačoch na trhu. A rovnako by sme stránku mali otestovať aj pri vypnutom JS. Stránka by za žiadnych okolností nemala byť rozbitá, či nekonzistentná, a nemala by obsahovať „pokazené“ interaktívne prvky, ako napr. rôzne tlačidlá, ktoré bez JS nefungujú. V našom krátkom tutoriáli sme sa naučili, že ak takéto prvky potrebujeme na ovládanie JS funkcionality, môžeme ich jednoducho dogenerovať tiež pomocou JS, a to len v prípade, že je JS naozaj dostupný.

NETMARKETSHARE℠

Voľne dostupné štatistiky trhových podielov prehliadačov nájdeme napr. na: http://marketshare.hitslink.com/

Ešte dôležitejšie je, aby sme výhradne pomocou JS neimplementovali **kritickú funkcionalitu**, ktorú používatelia bez JS vôbec neuvidia, alebo ju nebudú môcť použiť. Predovšetkým ide o rôzne navigačné prvky, ktoré používateľom umožňujú dostať sa ku všetkým súčastiam našej stránky, ale aj o akúkoľvek inú funkcionalitu, ktorú považujeme za podstatnú. Takúto funkcionalitu je vždy nutné implementovať pomocou základných technológií ako sú HTML a CSS a pomocou server-side programového kódu, ktorý je vždy dostupný pre každého používateľa, keďže beží na serveri, ktorý je pre všetkých používateľov ten istý. Samozrejme, takéto webové stránky je možné pomocou JS oživiť, či dokonca ich používanie uľahčiť. Základné pravidlo však je, že všetko podstatné musí fungovať aj bez JS, či iných interaktívnych technológií.

Programovanie na strane servera je mimo sylabus nášho modulu. Pozri [10] a [11].

Posledným problémom je prílišná, či **samoúčelná interaktivita**. Akokoľvek sa nám zdá naše inovatívne riešenie praktické a intuitívne na ovládanie, bežný používateľ, ktorý chce na našej stránke predovšetkým rýchlo nájsť hľadané informácie a s podobným riešením sa nikdy predtým nestretol, to môže vidieť presne naopak. Zohľadnime aj neskúsených používateľov, ktorí s internetom len začínajú, a tiež znevýhodnených používateľov, ktorí nemajú plnohodnotné ukazovacie zariadenie (myš), alebo majú nejakú motorickú poruchu.

Riešenia ako napr. interaktívne menu, v ktorých sú položky animované, alebo menu, ktoré zobrazia názov položky až po prejdení myšou, a pod. sú veľmi nevhodné. Mimoriadne opatrní musíme byť tiež so stránkami obsahujúcimi skrytý obsah, ktorý sa zobrazí až keď používateľ niekam klikne. V takomto prípade musí byť na prvý pohľad jasne, kam je potrebné kliknúť (napr. tak, že aktívne prvky zobrazíme ako záložky známe z desktopových aplikácií). Pri akomkoľvek interaktívnom riešení sa vždy zamyslime, či je interaktivita naozaj opodstatnená, alebo či by niekomu mohla byť viac na škodu ako na osoh.

Čo sme sa naučili

Dozvedeli sme sa, že JavaScript môže byť pri nesprávnom použití dvojsečnou zbraňou a pre niektorých používateľov môže byť viac na škodu než na osoh. Preto musíme JS používať s rozvahou a zabezpečiť testovanie vo všetkých aktuálne používaných prehliadačoch. Kritická funkcionalita by nikdy nemala byť implementovaná iba pomocou JS, a mali by sme sa tiež vystríhať samoúčelnej interaktivite, ktorá môže byť pre niektorých používateľov prekážkou.

Kapitola 4: Čo sme sa naučili v tomto module

Naučili sme sa pracovať s kaskádovými štýlmi, predovšetkým sme sa sústredili na vlastnosti box modelu a vizuálny formátovací model. Vyskúšali sme si pokročilé techniky ako je obtekanie, ale i absolútne a relatívne polohovanie. Tieto techniky dokážeme použiť pri tvorbe viacerých pokročilých prvkov webového dizajnu, ako sú napr. plávajúce textové rámce, rôzne horizontálne panely a menu, až po zložité viacstĺpcové stránky.

V druhej časti materiálu sme sa formou tutoriálu oboznámili so základmi programovania na strane klienta. Zoznámili sme so syntaxou a s vybranými konštruktmi jazyka JavaScript. Stručne sme sa oboznámili tiež so vstavaným objektovým modelom DOM, ktorý nám umožňuje spracovávať ako aj meniť aktuálne načítaný HTML dokument. Následne sme sa oboznámili s JS knižnicou JQuery, pomocou ktorej sme sa naučili pracovať s dokumentom jednoduchšie a efektívnejšie. Výhodou tejto knižnice je, že zaručuje vysokú mieru kompatibility s najrôznejšími prehliadačmi, čo nie je pri programovaní v JS úplne samozrejmé.

Literatúra a použité zdroje

[1] Bos, B., Çelik, T., Hickson, I. Wium Lee, H. (2009) Cascading style sheets level 2 revision 1 (CSS 2.1) specification. W3C Candidate Recommandation. Dostupné na internete: http://www.w3.org/TR/CSS2/

[2] Zeldman, J. (2005) Tvorba webů podle standardů, XHTML, CSS, DOM, ECMASkript a dalších, Computer Press. ISBN: 80-251-0347-1

[3] CSS3.info blog. Dostupné na internete: http://www.css3.info/

[4] Flanagan, D.(2002) JavaScript Kompletní průvodce, Computer Press ISBN: 80-7226-626-8

[5] Chaffer, J., Swedberg, K. (2009) Learning jQuery 1.3, Packt Publishing Ltd. Birmingham UK. ISBN: 978-1-847196-70-5

[6] JQuery documentaion. Dostupné na internete: http://docs.jquery.com/

[7] JQuery API reference. Dostupné na internete: http://api.jquery.com/

[8] Homola, M., Guniš, J. (2009) Webové technológie a publikovanie na webe 1, ŠPÚ Bratislava, ISBN: 978-80-8118-016-3

[9] Sivčo, M. (2009) 3-stlpcový layout platinum edition, http://saiffer.blog.matfyz.sk/p12489-3-stlpcovy-layout-platinum-edition

[10] Guniš, J. Základy progamovania v jazyku PHP. In: Palmárová, V., Siláči, J., Moravčík, M., Kaukič, M., Guniš, J. (2010) Programovanie 5, ŠPÚ Bratislava, ISBN: 978-80-8118-040-8, s. 41-61

[11] Oliver, L., Schmidt, J. (2010) PHP v praxi: pro začátečníky a mírně pokročilé. Grada. ISBN: 978-80-247-3060-8

Časť 2:Multimédiá

Autori modulu:

Mgr. Ján Guniš
ÚINF PF UPJŠ v Košiciach
jan.gunis@upjs.sk.sk

Mgr. Martin Cápay, PhD.
KI FPV UKF v Nitre
mcapay@ukf.sk

PaedDr. Martin Magdin
KI FPV UKF v Nitre
mmagdin@ukf.sk

RNDr. Ľubomír Šnajder, PhD.
ÚINF PF UPJŠ v Košiciach
lubomir.snajder@upjs.sk.sk

Kapitola 1: Úvod

Od začiatku 90-tych rokov dvadsiateho storočia sa hovorí o využívaní nových informačných technológií – multimédií. Multimédia predstavujú kombináciu mediálnych elementov: textu, grafiky, animácie, videa, zvuku. Sú určené na informovanie, prácu vzdelávanie a zábavu. Niekedy sa označujú aj ako elektronické médiá slúžiace na zálohovanie a predstavenie multimediálneho obsahu. Pod médiom sa chápe prostriedok, ktorý sa používa alebo môže použiť, na zobrazenie informácie rôzneho druhu, napr. textu, grafiky, obrazu, animácie alebo zvuku. Najjednoduchšie to vyjadruje obrázok:

Obrázok 1: Multimédiá

K jednotlivým mediálnym elementom sa často pridáva ešte jeden element – interaktivita. Niektorí autori však uvádzajú, že interaktivita je skôr vlastnosť multimédií.

Pod pojmom mediálny element rozumieme základný stavebný prvok multimédií, ktorý obsahuje danú mediálnu formu, teda podobu, v ktorej je reprezentovaná konkrétna informácia.

Úloha 1	Kde všade sa najčastejšie môžeme stretnúť aspoň s jedným z mediálnych elementov? Vymenujte aspoň päť možností, ako by ste vedeli využiť multimédia pri výučbe! Do prehliadača zadajte túto URL adresu a vymenujte mediálne elementy, ktoré www stránka obsahuje: http://www.topky.sk/.
Úloha 2	Pokúste sa spolu s lektorom stanoviť základné požiadavky na hardvérové komponenty, ktoré by mali byť súčasťou multimediálneho systému (multimediálny PC + príslušenstvo v podobe V/V zariadení).
Úloha 3	Definujte vlastnými slovami, čo si predstavujete pod pojmom multimediálna prezentácia a multimediálna aplikácia. Aký je medzi nimi rozdiel? Uveďte aspoň tri oblasti, v ktorých je možné využiť multimediálne prezentácie a aspoň tri oblasti, v ktorých je možné využiť multimediálne aplikácie.

Multimédia je veľmi často vyhľadávaný a frekventovaný pojem – v Google sa nachádza preň 191 000 000 odkazov.

Pod pojmom grafika, grafická informácia rozumieme akúkoľvek formu vizualizovanej informácie v podobe obrázku, grafu, fotografie, schémy a pod., ktorou sa snažíme vyjadriť podstatu preberaného javu, alebo zvýrazniť jeho význam. Vhodne spracovaná obrazová informácia je mnohonásobne efektívnejšia, pre pochopenie podstaty skúmaného javu, ako písaný text.

Základné kritéria pre graficky spracovanú informáciu sú:

a) názornosť,
b) jednoznačnosť,
c) zrozumiteľnosť,
d) vecnosť.

Príklad graficky spracovanej informácie, ktorá spĺňa všetky základné kritéria

Čo sme sa naučili

- Definovať pojem mediálny element a určiť oblasť jeho využitia.
- Vysvetliť vlastnými slovami pojem multimédia v nadväznosti na projektovanie multimediálnych aplikácií.
- Určiť oblasť využitia nielen multimédií, ale aj vytvorených multimediálnych aplikácií.
- Poznať základné psychologické a pedagogické požiadavky na tvorbu jednotlivých mediálnych elementov a multimediálnych aplikácií.

Kapitola 2: Text

Komunikácia prostredníctvom písaného textu je popri hovorenom slove jedným z najstarších spôsobov prenosu informácie. Použitie textu v multimediálnych aplikáciách je veľmi dôležité, keďže má vysokú informačnú hodnotu a uľahčuje orientáciu, pričom vo veľkej miere vplýva na zrozumiteľnosť. Množstvo textu v multimediálnych aplikáciách závisí od ich využitia. Napríklad aplikácia určená na vysvetlenie práce jednotlivých zariadení v rámci počítača, bude obsahovať určite viac textu ako obrázkový slovník.

Text plní dve funkcie:

- komunikačnú (navigačnú) – upozornenia, výber z ponuky, poznámky, atď.
- informačnú (informovanie o predmetnej udalosti), s vplyvom na prehľadnosť.

Text, ktorý obsahuje aplikácia musí mať svoju logickú stavbu a nadväzovať na seba. Nežiaduce sú dlhé pasáže textu, ktoré pôsobia na používateľa nudne. Správne pripravený a použitý text v multimediálnych aplikáciách určených napr. pre výučbu, môže poslúžiť aj pre samoštúdium študenta, pričom umožňuje študentovi aj bez priameho kontaktu s vyučujúcim pochopiť maximum preberanej látky bez toho, aby bolo potrebné dopĺňať tieto materiály dodatočným výkladom.

Úlohou textu použitého v takýchto multimediálnych aplikáciách však nie je nahrádzať klasickú formu vyučovania, ale efektívne pomôcť novou formou pri rýchlejšom získavaní vedomostí. Jednotlivé časti textu, alebo kapitoly by mali byť spracované tak, aby nekopírovali učebnice, ale sa najjednoduchšou formou za pomoci interaktívnych animácií alebo obrázkov snažili objasniť danú problematiku.

Na tvorbu textu do multimediálnych aplikácií používame textové procesory, resp. priamo text vkladáme napr. do autorského systému (napríklad MS PowerPoint). Dôležité je dodržiavať štandardný font písma, ktorý je základnou zložkou každého textového procesora. Celkovú podobu fontu charakterizuje:

a) tvar písma,
b) názov písma,
c) veľkosť písma,
d) rez písma,
e) farba písma.

Porovnanie rastrového a vektorového písma pri veľkom zväčšení

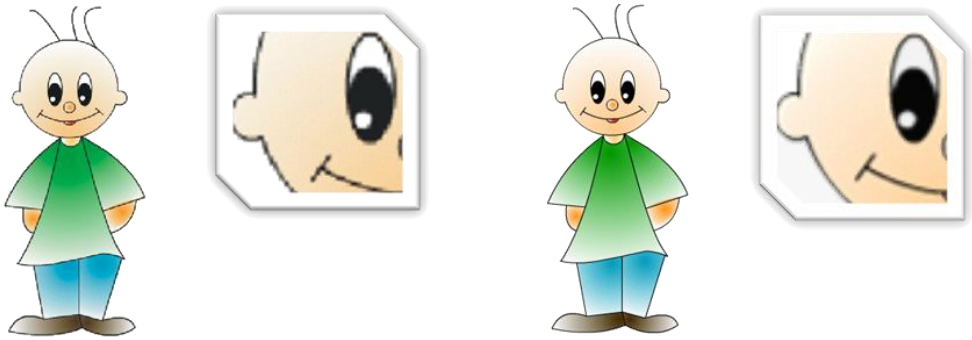
Z hľadiska tvorby písma rozdeľujeme písmo na rastrové a vektorové. Rastrové písmo, ktoré je bežnou súčasťou textových editorov, podobne ako obrázok v počítačovej grafike, je tvorené množinou bodov a jeho kvalita závisí od DPI (Dots Per Inch) v danom rozlíšení. Pri vektorovom type je tvar písma definovaný pomocou úsečiek a kriviek, ktoré na seba nadväzujú v určitej postupnosti. Vektorové písmo používame najmä v grafických programoch, kde požadujeme zachovanie jeho pôvodných vlastností aj pri veľkom zväčšení, keďže rastrové písmo stráca zaoblenie pri zmene veľkosti a často i tvar.

Čo sme sa naučili

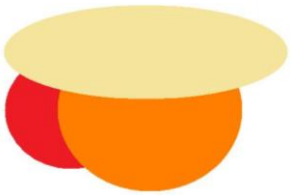
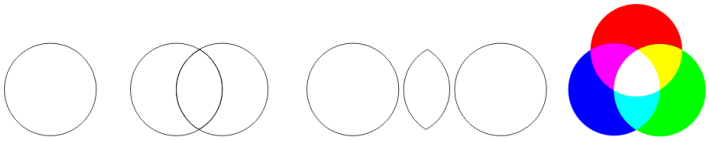
- Definovať funkcie textu v multimediálnych aplikáciách.
- Definovať vlastnosti správne napísaného a použitého textu v multimediálnej aplikácii.

Kapitola 3: Statické a animované rastrové obrázky

Rastrová grafika, nazývame ju niekedy aj bitmapová grafika, je reprezentovaná tzv. bitovou mapou. Každý bod je zakódovaný pomocou bitov. Obrazová informácia sa ukladá pomocou matice rôznofarebných izolovaných bodov, ktoré tvoria výsledný obraz a od jej veľkosti závisí aj kvalita výsledného obrázka. Problémom rastrovej grafiky je zmena kvality výsledného obrázka pri zmene jeho rozmerov.



Obrázok 2: Porovnanie rastrovej a vektorovej grafiky

Úloha 4	Aké formáty rastrovej grafiky poznáte? Čo myslíte, ktoré sú vhodné pre publikovanie na webe? Ktoré z nich sú proprietárne (uzatvorené, teda viazané na licenciu), a ktoré neproprietárne?
Úloha 5	Nakreslite v programe LogoMotion obrázok podľa predlohy a uložte ho v ponúkaných formátoch! Porovnajte, čo sa stalo s kvalitou obrázka pri jeho uložení do rôznych formátov, a akú veľkosť na pevnom disku zaberá každý z vytvorených súborov. Zistite, či pre veľkosť súboru vo formáte BMP platí: veľkosť = počet bodov obrázka x počet bitov palety farieb Napr. pre 256 farebnú bitovú mapu veľkosti 800 x 600 pixelov platí, že veľkosť súboru = 800 x 600 x 1 B = 480 000 B = 468,75 KB. Ako sa zmení veľkosť súboru, ak obrázok uložíme vo formáte 24 bitovej mapy? Ako sa zmení veľkosť súboru ak z pôvodného obrázka uložíme len výrez 400 x 300 bodov? 
Úloha 6	Vytvorte v programe LogoMotion obrázok podľa predlohy. Prečo je pre nás zložité nakresliť napr. dva rovnako veľké a navzájom sa prekrývajúce kruhy? Je vôbec možné nakresliť posledný farebný obrázok tak, ako je vyobrazený v predlohe? 

Pri rastrovej grafike sa stretávame s pojmom **animovaná grafika**, ktorá môže byť pohyblivá a často aj ozvučená. Je určená predovšetkým na tvorbu jednoduchých animovaných sekvencií až po zložité počítačom realizované filmové sekvencie – napr. Jurský park.

Najčastejšie animovanú grafiku môžeme vytvoriť pomocou krátkeho programu obsahujúceho cyklus. V prípade, ak požijeme formát GIF, realizujeme animovanú

grafiku ako sekvenciu za sebou nasledujúcich obrázkov. Získavame jednoduchú animáciu z generovaných snímok, pri ktorej sa využíva nedokonalosť ľudského oka rozlíšiť rýchlo sa premietajúcu sériu objektov, obrazov, textov. Výslednú animáciu teda vnímame ako spojitý pohyb.

Okrem formátu GIF sa môžeme stretnúť v rámci animovanej rastrovej grafiky aj s týmito formátmi:

ANI – je určený pre tvorbu animovaného kurzora v operačných systémoch Windows.

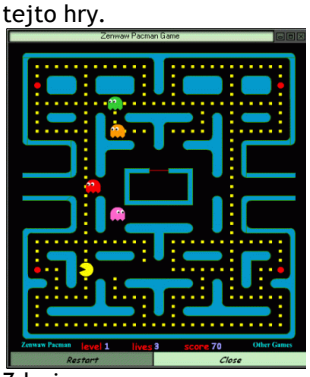
SWF – v súčasnosti dominuje na webových stránkach v oblasti animovanej grafiky, keďže (okrem rastrovej a vektorovej grafiky) umožňuje využívať aj text, audio a hlavne interakciu.

Úloha 7

Vytvorte v programe Logomotion sekvenciu obrázkov znázorňujúcich pohyb Pampúcha zo známej počítačovej hry „Pampúch a duch“ (pôvodný názov PAC-MAN), z ľavej strany na pravú. Nastavte rýchlosť prehrávania jednotlivých fáz a výsledok uložte vo formáte GIF.

Viete takto vytvorenú animovanú grafiku ovládať?

Populárna počítačová hra PAC-MAN. Vznikla v roku 1980 v Japonsku a veľmi skoro sa stala populárnou takmer na celom svete. Ešte aj dnes nájdeme na internete niekoľko mutácii tejto hry.



Zdroj: http://www.freewebs.com/agpx/evilpacmancomix.htm

Čo sme sa naučili

- Vysvetliť princíp reprezentácie rastrovej grafiky.
- Určiť vhodnú reprezentáciu rastrovej grafiky podľa situácie.
- Vysvetliť funkciu základných nástrojov rastrového grafického editora.
- Demonštrovať použitie základných nástrojov rastrového grafického editora.
- Analyzovať zložitejšiu výslednú grafiku a rozlíšiť tak použitie jednotlivých nástrojov.
- Vysvetliť princíp tvorby rastrovej animácie.
- Popísať základné typy rastrovej animácie.
- Demonštrovať použitie základných typov rastrovej animácie.

Kapitola 4: Statické a animované vektorové obrázky

4.1 Princíp reprezentácie vektorovej grafiky v počítači

Scalable Vector Graphics, http://www.w3.org/Graphics/SVG/

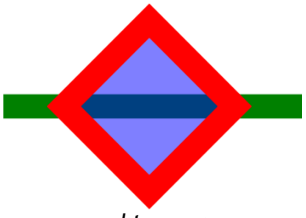
Predpokladá sa, že formát SVG sa stane základným otvoreným formátom pre vektorovú grafiku na internete.

Vektorová grafika predstavuje principiálne odlišný spôsob reprezentácie grafických súborov (obrázkov) v počítači ako pri rastrovej grafike. V rastrovej grafike je každý obrázok reprezentovaný postupnosťou farebných bodov. Každý bod má jednoznačne určenú svoju pozíciu a svoju farbu.

Vo vektorovej grafike je obrázok zostavený z objektov, geometrických tvarov. Každý objekt je vyjadrený matematickým popisom, akýmsi návodom na jeho zostrojenie. Tento návod obsahuje aj ďalšie informácie o objekte, napr. farby objektu, spôsob transformácie, úroveň priehľadnosti atď.

Nižšie uvedený kód je zápisom obrázka na okraji vo formáte SVG. Všimnime si jeho popis. Z popisu, bez toho, aby sme obrázok videli, vieme obrázok zostrojiť. Podobne postupuje aj grafický vektorový editor.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
          "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1"
     xmlns="http://www.w3.org/2000/svg">
  <line
    x1="50" y1="100" x2="300" y2="100"
    stroke-width="20" stroke="green"
  />
  <rect
    width="100" height="100"
    transform="translate(100 100) rotate(-45)"
    fill="blue" fill-opacity="0.5"
    stroke-width="20" stroke="red"
  />
</svg>
```



vektor.svg

Formát SVG je otvorený formát na popis dvojrozmerných vektorových obrázkov a grafických aplikácií. Formát SVG je len jeden z množstva formátov. Aj keď súbory vo formáte SVG sú „obyčajné“ textové dokumenty, editovať ich v textovom editore nie je najvhodnejší spôsob ako vytvárať grafické dokumenty. Výhodnejšie je použiť niektorý z grafických editorov.

Úloha 8	
	Aké vektorové formáty poznáte?
	Ktoré sú vhodné pre publikovanie na webe?
	Ktoré z nich sú proprietárne a ktoré neproprietárne?

INKSCAPE

INKSCAPE, open source vektorový grafický editor, http://www.inkscape.org/

Sketsa, multiplatformový proprietárny editor SVG, http://www.kiyut.com/products/sketsa/index.html.

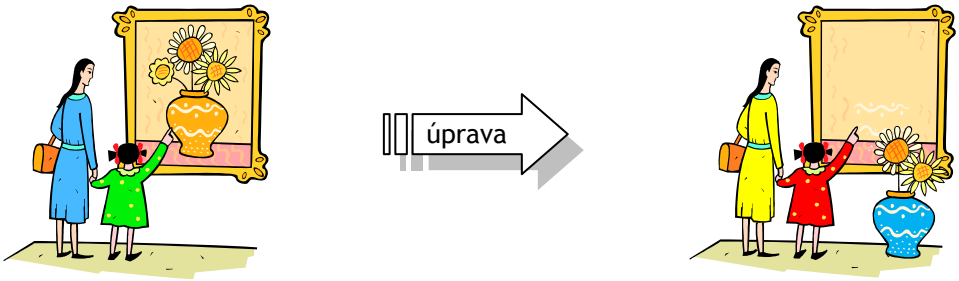
4.2 Možnosti uplatnenia vektorovej grafiky

Aj keď sa nám to na prvý pohľad nezdá, s vektorovou grafikou sa stretávame pomerne často.

Väčšina súborov písmen (fontov), ktoré používame na počítači, obsahuje vektorový popis tvaru kompletnej sady znakov. Aj tento text, ktorý práve teraz čítate, bol zobrazený na monitore (a následne vytlačený na tlačiarni) na základe popisu tvaru znakov zo súboru *trebuc.TTF*.

Časť ukážky počítačového písma Trebuchet MS:

Cliparty, jednoduché obrázky ktoré vkladáme do svojich dokumentov, sú často popísané vo vektorovom formáte. Výsledný obrázok je zložený z niekoľkých samostatných objektov. Takýto spôsob popisu nám umožňuje jednoduchšiu manipuláciu nielen s obrázkom samotným, ale aj s jeho časťami.

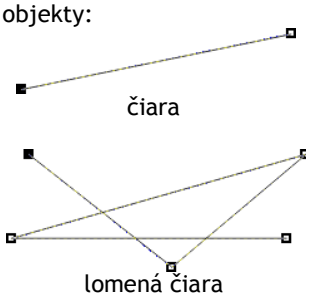


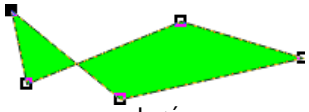
Obrázok 3: Manipulácia s vektorovým obrázkom a jeho časťami

Automatické tvary, umelecký text a rôzne ďalšie symboly môžu byť tiež reprezentované ako vektorové obrázky.

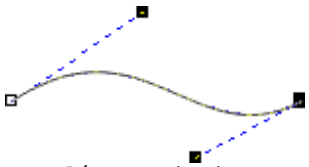
Nezriedka sa stretneme aj s prípadom, keď časť obrázka je reprezentovaná ako vektorová a časť ako rastrová grafika.

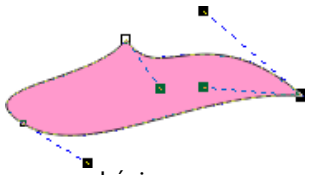
Vektorové primitívne objekty:

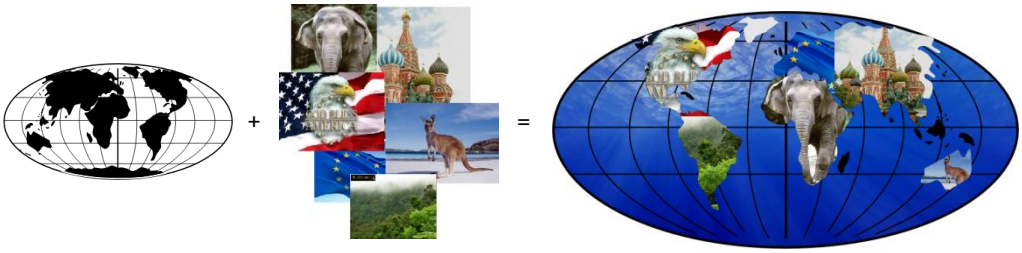



polygón


elipsa, kružnica


Bézierova krivka


béziergon



Obrázok 4: Kombinácia vektorovej a rastrovej grafiky

Úloha 9	V prostredí textového procesora vložte do dokumentu obrázok typu ClipArt. Dokument uložte. Obrázok zväčšite na dvojnásobok a dokument uložte pod iným menom. Porovnajte veľkosti súborov. Zdôvodnite!
Úloha 10	V prostredí textového procesora vložte do dokumentu obrázok typu ClipArt. Obrázok upravte (pridajte, zmažte niektoré jeho časti, presuňte objekty v ňom, prefarbite časti obrázka a pod.). Viete podobnú úpravu spraviť v každom vloženom obrázku? Zdôvodnite!

4.3 Návody a postupy na riešenie úloh

Vektorové grafické editory ponúkajú prepracovanú sadu jednoduchých, často až primitívnych nástrojov. Ich znalosť a ovládanie je nutnou, nie však postačujúcou podmienkou na to, aby sme vedeli vytvárať kvalitné a zaujímavé grafické diela (obrázky). V tejto časti si ukážeme niektoré jednoduché nástroje a ich kombinácie.

Pri vytváraní obrázkov pomocou vektorovej grafiky budeme vychádzať z primitívnych objektov: čiara, lomená čiara, polygón, kružnica, elipsa, Bézierova krivka, béziergon a text. Úpravou a vzájomnou kombináciou týchto útvarov vieme vytvárať zložitejšie objekty. Všimnime si obrázok ženy v krajnom stĺpci. Bol vytvorený vo vektorovom grafickom editore. Je zložený zo stoviek, premyslene prepracovaných jednoduchých objektov.

Výsledný obrázok je často len ilúziou budiacou dojem niečoho, čo v skutočnosti nie je. Toto tvrdenie je celkom zrejmé, ak si uvedomíme, že často zobrazujeme objekty reálneho, trojrozmerného sveta v prostredí dvojrozmerného grafického editora.

Hlavná cena v súťaži CorelDRAW® International Design Contest 2009, Aleksey Oglushevich, Rusko

Zdroj: http://www.corel.com/servlet/Satellite/us/en/Content/1231434231476

Reliéf

Pod pojmom reliéf rozumieme druh plastiky určenej pre frontálny pohľad (na rozdiel napr. od sochy, ktorá je určená pre viacero pohľadov). Reliéf teda predstavuje trojrozmerný útvar. Aj napriek tomu, že budeme používať editor dvojrozmernej grafiky, vieme tento efekt napodobniť. Všimnime si, ako je docielený efekt reliéfu na obrázku mince. Hrany, na ktoré dopadá svetlo, sú jasnejšie než tie, ktoré sú v tieni. Vhodným rozmiestnením kópií objektu (svetlejšie a tmavšie) docielime efekt jednoduchého reliéfu. Čím vzdialenejšie budú kópie objektu od jeho originálu, tým plastickejšie bude reliéf pôsobiť (od istej vzdialenosti to však už prestáva byť prirodzené).

Obrázok 5: Efekt reliéfu v grafickom editore

Úloha 11	Vo vybranom grafickom editore vytvorte efekt reliéfu pre vybraný objekt. Môžeme rovnaký efekt docieliť na bielom, resp. čiernom podklade? Zdôvodnite!

Reliéf Kriváňa na zadnej strane slovenskej päťcentovej mince.

Efekt reliéfu napodobňujú aj tlačidlá grafických aplikácií.

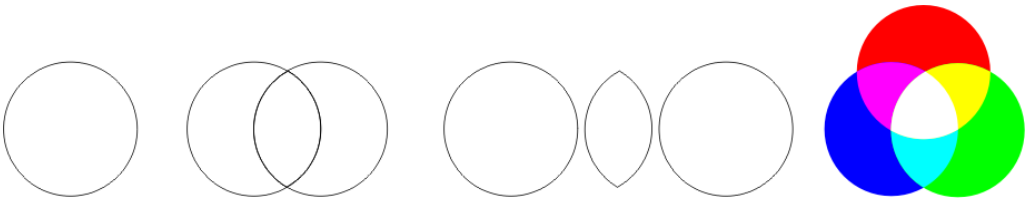
Logické operácie

Logické operácie predstavujú silný nástroj pre vytváranie nových tvarov. Zrejme každý vektorový editor ponúka aspoň základné z nich: zjednotenie, rozdiel a prienik. Najťažším krokom je rozhodnúť sa, aké zdrojové útvary a aké logické operácie použijeme. Ich použitie si ukážeme pri vytváraní obrazu RGB farebného modelu. Z tvaru jednotlivých častí tohto modelu vieme, že počiatočným útvarom bude kružnica, resp. tri kružnice umiestnené rovnomerne po obvode kružnice.

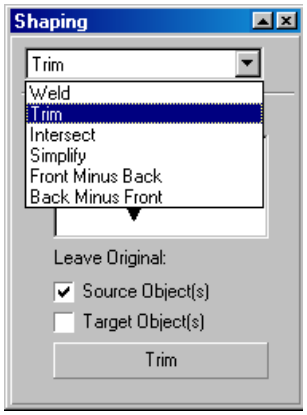
Rozmiestnenie kružníc vieme dosiahnuť niekoľkými spôsobmi. Napr.:

- Zmenou pozície kružníc na základe výpočtu ich súradníc.
- Prichytením stredov kružníc k vrcholom (pomocného) rovnostranného trojuholníka.
- Vytvorením duplikátov kružnice otočením o uhol 120° okolo určeného stredu.

Plochy pre ďalšie farby vytvoríme prienikom pôvodných kružníc. Novovytvorené objekty sa budú vzájomne prekrývať (majú spoločnú časť plochy a obrysovej čiary).

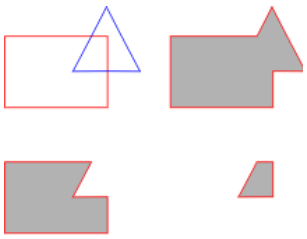


Obrázok 6: Výsledok logických operácií medzi vektorovými objektmi



Logické operácie v prostredí grafického editora CorelDraw.

Úloha 12	Vytvorte obraz RGB modelu podľa nasledovného vzoru. Poznámka: farba výplne šiestich malých trojuholníkov vznikla zmiešaním farieb príslušných častí pôvodných obdĺžnikov. 



zjednotenie, rozdiel, prienik dvoch útvarov

Kombinácia objektov a transformácie

Kombinácia objektov je jedna z metód, ako z viacerých objektov vytvoriť jeden. Výsledný objekt obsahuje toľko podciest (oddelených úsekov krivky) ako je počet kombinovaných objektov. V mieste, kde sa prekrýva párny počet objektov, dôjde k vystrihnutiu otvoru, v mieste s nepárnym počtom prekrývajúcich sa otvorov sa vytvorí výplň.

Transformácie sú ďalším z nástrojov pre zmenu vlastností objektov. Objekty môžeme transformovať zmenou pozície, otočením, zrkadlením, zmenou veľkosti a skosením. Použitie si ukážeme pri vytváraní modelu kocky, s vyrezaným otvorom v každej stene.

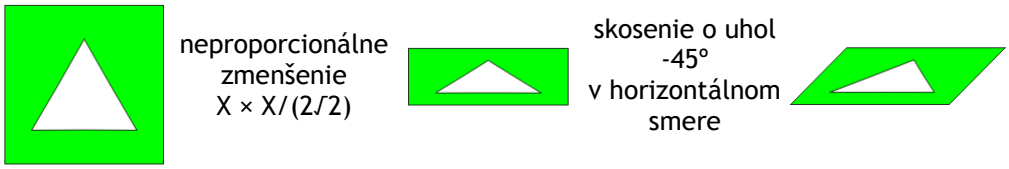
Výsledok kombinácie niekoľkých vzájomne pootočených elíps:

Najskôr si pripravíme tvar každej steny. Štvorec do ktorého vyrežeme niektorý z tvarov. Vo štvorci takto vznikne diera. Rovnaký efekt dosiahneme aj skombinovaním štvorca s daným tvarom.

Ďalším krokom je úprava stien budúcej „kocky“ tak, aby sme ich mohli zložiť do rovnobežného priemetu kocky. Predná a zadná strana kocky ostávajú bez zmeny. Ostatné steny musíme neproporcionálne zmenšiť a skosiť pod uhlom ±45°.

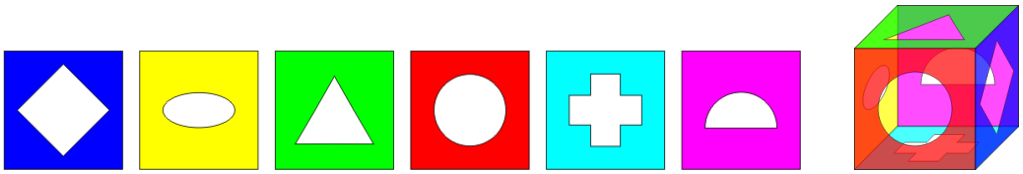
Detská hračka na učenie sa rozpoznávania tvarov.

Zdroj:
http://www.franhaus.cz/



Obrázok 7: Transformácie objektu

Upravené steny kocky vzájomne zarovnáme a umiestnime ich do správnych úrovní (objekty vpredu by mali prekrývať objekty za nimi).



Obrázok 8: Steny kocky a jej priemet

Ako ďalší efekt môžeme každej stene kocky pridať vlastnosť priehľadnosť. Kocka bude vyzerať, ako keby jej steny boli tvorené farebnou, priehľadnou fóliou.

Objekty nikdy nezarovnávajme ručne. Vektorový editor dokáže pracovať s presnosťou cca. 0,001 mm (je možné nastaviť aj inú presnosť). Takúto presnosť by sme voľnou rukou nikdy nedosiahli. Objekty zarovnáme využitím funkcie prichytávanie k objektom alebo pomocou nástroja zarovnanie.

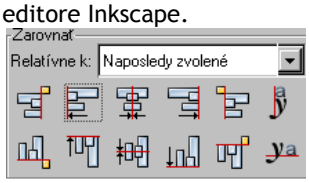
Úloha 13

Vytvorte obrázok dvoch očiek retiazky, ktoré sú vzájomne spojené.

Jednoduché 3D objekty

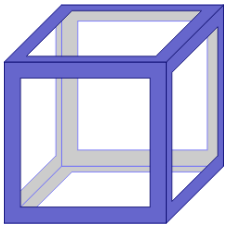
Podobne ako maliari boli schopní na plátno preniesť obraz trojrozmerného sveta, môžeme sa o to pokúsiť aj my. Jednoduchým postupom dokážeme vytvoriť efekt kocky v priestore.

Možnosti zarovnania objektov v grafickom editore Inkscape.

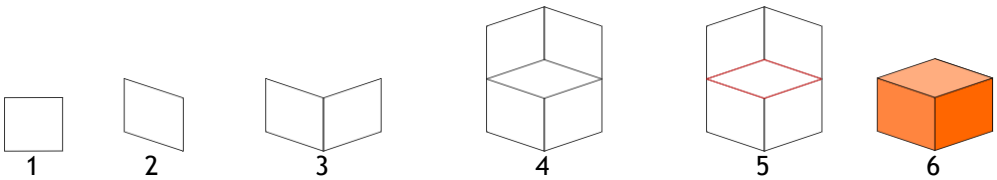


Začneme kresbou štvorca (1), ktorý postupne upravíme na steny kocky:

- štvorec najskôr skosíme (2) a vytvoríme z neho kosodĺžnik,
- vytvoríme zrkadlovú kópiu kosodĺžnika v horizontálnom smere (3), oba kosodĺžniky sa dotýkajú jednou stranou,
- vytvoríme zrkadlovú kópiu oboch kosodĺžnikov vo vertikálnom smere (4) tak, aby sa dotýkali len v dvoch bodoch,
- priestor ohraničený štvoricou kosodĺžnikov ručne dokreslíme (využijeme prichytávanie k bodom), (5),
- odstránime dva horné pomocné kosodĺžniky a tri plochy vyfarbíme, ak svetlo dopadá z ľavej hornej časti, horná časť bude najsvetlejšia, pravá predná naopak najtmavšia (6).

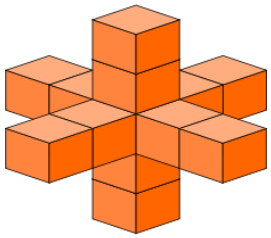


plast_kocky.svg

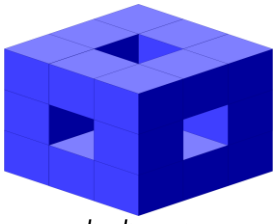


Obrázok 9: Postup tvorby obrazu kocky v priestore

Z výslednej kocky môžeme vytvoriť niekoľko kópií, ktoré uložíme vedľa seba. Pri ukladaní používajme prichytávanie k objektom alebo zarovnávanie objektov. Voľnou rukou by sme kocky umiestnili len približne. Takýto nepresný výsledok by len pokazil výsledný dojem.



Obrázok 10: Ilúzia tretieho rozmeru v dvojrozmernom editore



kocka.svg

Moderné vektorové editory ponúkajú nástroje pre tvorbu 3D efektov: Nástroj „Extrude“ v programe CorelDraw.

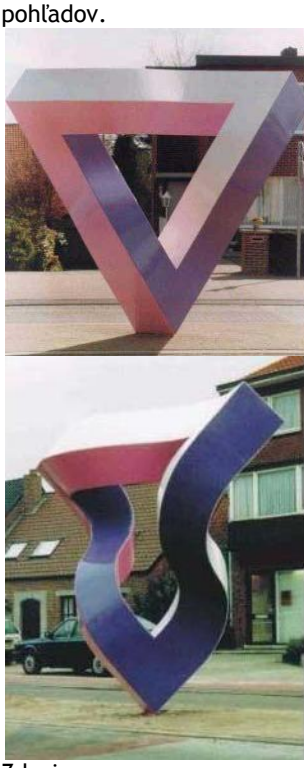
3D_text_coreldraw7.cdr

Neexistujúci 3D svet

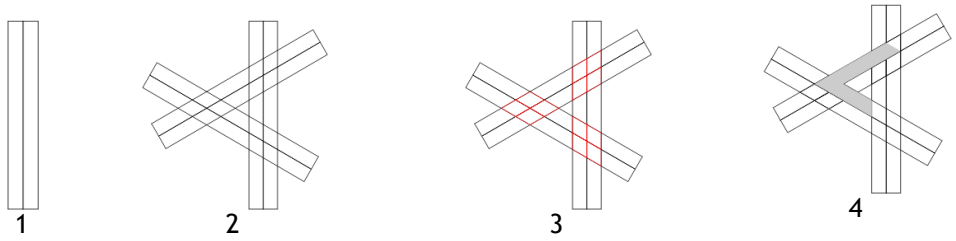
V predchádzajúcich častiach sme sa snažili vytvoriť ilúziu trojrozmerného sveta v prostredí dvojrozmerného grafického editora. Pri troche fantázie môžeme vytvoriť ilúziu sveta, ktorý reálne „nemôže“ existovať. Takáto realizácia však vyžaduje dobrú priestorovú predstavivosť.

Pokúsime sa vytvoriť obraz trojrozmerného triangla (obrázok na okraji), ktorý narúša naše vnímanie priestoru. Skôr ako začneme, pozrime sa bližšie, z čoho sa takýto triangel skladá. Každá jeho strana je zobrazená pomocou dvoch plôch (1). Dvojice plôch sa v rohoch triangla vzájomne pretínajú. Ilúziu dosiahneme vhodným zafarbením plôch a ich prienikov. Vytvoríme kópie dvojice plôch vzájomne pootočené o uhol 120° (2). Vytvoríme prieniky plôch v rohoch triangla (3). Tieto prieniky (útvary) nám pomôžu nakresliť tri plochy. Využijeme funkciu prichytávania k bodom a nepravidelný šesťuholník nakreslíme tak, aby sa jeho vrcholy prekrývali so zodpovedajúcimi vrcholmi objektov (4).

Tvrdiť, že niektoré veci nemôžu existovať je trochu unáhlené. Nasledujúce obrázky zobrazujú jeden a ten istý objekt z rôznych pohľadov.



Zdroj: http://visualfunhouse.com/



Obrázok 11: Postup tvorby „neexistujúce“ trojrozmerného triangla

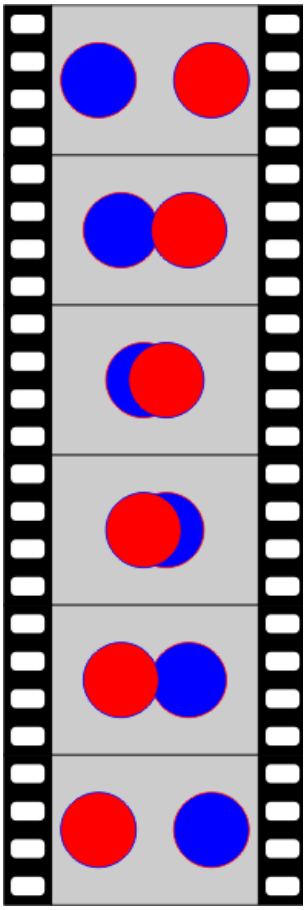
Výsledný útvar sa skladá z troch, rôzne zafarbených plôch (šesťuholníkov). Pomocné obdĺžniky a ich prieniky môžeme zmazať.

Obrázok 12: Na prvý pohľad neexistujúci útvar

Úloha 14

Vytvorte návrh poštovej známky pre Slovenskú poštu s vybranou tematikou.

V piatich krokoch sme postupne menili súradnice dvoch kruhov a medzivýsledky sme zaznamenávali. Výsledkom je animácia znázorňujúca zmenu pozície dvoch kruhov.



Úloha 15

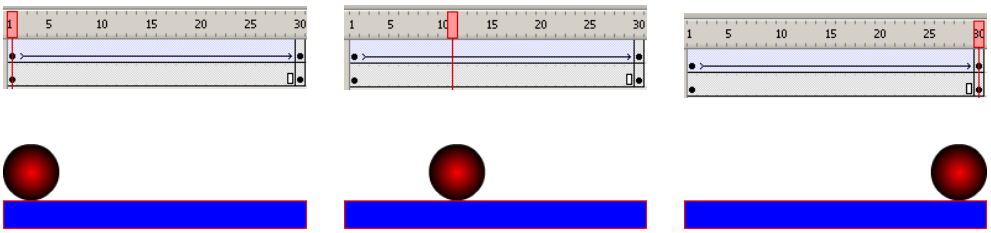
Vytvorte obrázok 3D ilúzie podľa vlastného návrhu.

4.4 Animované vektorové obrázky

V predchádzajúcich častiach sme sa oboznámili so spôsobom reprezentácie vektorovej grafiky. Vektorové objekty sú reprezentované postupom na ich zostrojenie a ich vlastnosti sú vyjadrené pomocou čísel. Zmena vlastností objektu spočíva v zmene číselnej hodnoty príslušnej vlastnosti. Ak túto zmenu budeme robiť v krokoch a objekt so zmenenými vlastnosťami postupne vykresľovať, výsledkom bude animácia. Animácia znázorňujúca postupnú transformáciu objektu.

Programy umožňujúce vektorové animácie môžu pracovať na základe dvoch princípov. Nasledujúce ukážky využívajú prostredie programu Adobe Flash CS4.

Pre používateľa jednoduchšia možnosť spočíva v definovaní začiatočného a koncového stavu objektu. Program potom postupne prepočítava zmenu vlastností objektu a objekt postupne zobrazuje (*lopta_animacia_cs4.fla*).



Obrázok 13: Animácia prechodu medzi dvoma kľúčovými snímkami

Na obrázku vyššie vidno dva stavy guľôčky. Začiatočný stav (ľavá časť) a koncový stav (pravá časť). Tieto pozície sa označujú ako kľúčové a snímky, v ktorých sme tieto pozície definovali, označujeme ako kľúčové snímky. Obrázok v strede ukazuje jeden z medzikrokov tak, ako ho prepočítal animačný program. Celá animácia pozostáva z 30 snímok. Ak by sme počet snímok zvýšili (resp. znížili), výsledná animácia bude plynulejšia (resp. menej súvislá). Pri každom spustení animácie program nanovo prepočítava každý medzikrok.

Druhá, o niečo náročnejšia možnosť spočíva v zmene vlastností objektu na základe programu vytvoreného používateľom. Rovnaký efekt ako v predchádzajúcom príklade by sme dosiahli na základe takéhoto programového kódu, skriptu (*lopta_skript_cs4.fla*):

```
function krok(udalost:TimerEvent) {
  lopta.x = lopta.x + dlzkaKroku;
}

var pocetKrokov = 30;
var dlzkaKroku = (podlozka.width - lopta.width) / pocetKrokov;
var casovac:Timer = new Timer(50,pocetKrokov);

casovac.addEventListener(TimerEvent.TIMER,krok);
casovac.start();
```

Nepísaným štandardom pre vektorové animácie sa stala technológia Adobe Flash (predtým Macromedia Flash). Skúšobnú verziu programu Adobe Flash môžeme prevziať z webovej stránky http://www.adobe.com/products/flash/?promoid=BPDEE.

Všimnime si, že pokiaľ vlastnosti objektov meníme programovo, musíme objekty najskôr pomenovať (v našom prípade `lopta` a `podlozka`). V kóde sme využili časovač (pozri napr. 3Prog4A), ktorý každých 0,05 s zavolá funkciu `krok`. Funkcia `krok` posunie loptu v smere osi x o určenú vzdialenosť.

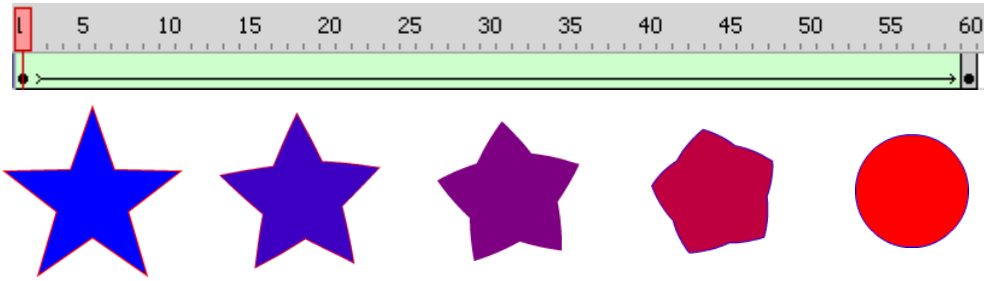
Aj keď ovládanie objektov (zmenou ich vlastností) skriptom je pre autora náročnejšie, táto možnosť ponúka širšie možnosti ovládania objektov.

Animácia typu „doplnenie tvaru“

Pri animácii typu „doplnenie tvaru“ Adobe Flash dokáže „prepočítavať“ postupnú zmenu tvaru jedného objektu na tvar druhého objektu. Do prvej snímky časovej osi našej animácie nakreslíme ľubovoľný útvar (Adobe Flash použijeme ako grafický editor). Do vybraného okienka (ktoré to bude záleží od toho, ako plynulý pohyb požadujeme) časovej osi vložíme prázdnu kľúčovú snímku a do plochy nakreslíme cieľový útvar.

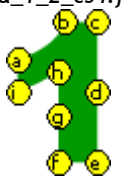
Posledným krokom je definovanie spôsobu zmeny prvého objektu na druhý. V prvej snímke na časovej osi vytvoríme animáciu typu „doplnenie tvaru“.

V našej animácii sme nechali zmeniť modrú hviezdu na červený kruh. Celá animácia prebehne v šesťdesiatich krokoch.



Obrázok 14: Animácia typu „doplnenie tvaru“

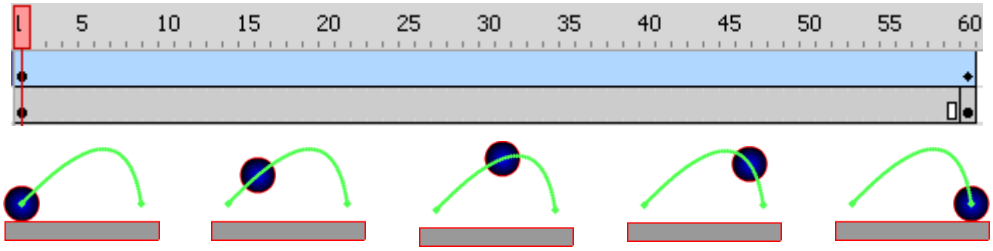
Ak sa nám nepáči spôsob, akým Flash mení tvar objektov, môžeme ho spresniť pridaným pomocných bodov do kľúčový snímok. Označené body prvého objektu sa zobrazia do rovnomenných bodov druhého objektu. (*animacia_1_2_cs4.fla*)



Animácia typu „doplnenie pohybu“

Ďalším typom animácie je „doplnenie pohybu“. Pri tomto type animácie môžeme meniť vlastnosti objektov ako napr. pozíciu, úroveň priehľadnosti, jasu a pod.

Do prvej snímky časovej osi našej animácie nakreslíme ľubovoľný útvar. Zároveň v tejto snímke vytvoríme animáciu typu „doplnenie pohybu“. Adobe Flash automaticky skonvertuje nakreslený útvar na symbol a umiestni ho do knižnice symbolov. V poslednej snímke animácie zmeníme polohu útvaru. Zároveň môžeme upraviť aj cestu pohybu.



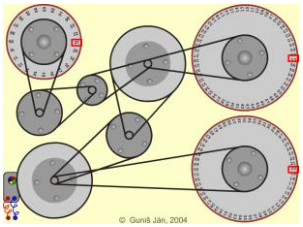
Obrázok 15: Animácia typu „doplnenie pohybu“

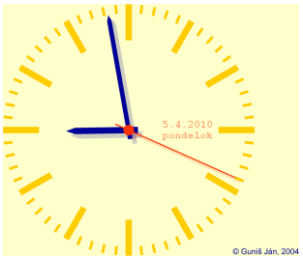
Animácia v animácii, symbol v symbole, pokročilejšie techniky

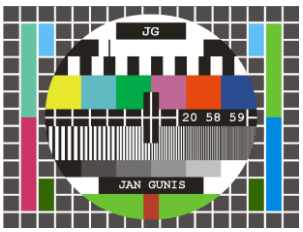
Animácie a objekty, ktoré vytvoríme, môžeme uložiť do knižnice symbolov. Uložený symbol môže byť typu filmový klip (napr. animácia), grafika (statický objekt) alebo tlačidlo (zvláštny typ symbolu schopný reagovať na zatlačenie myškou). Výhodou tohto riešenia je, že symbol z knižnice môžeme opakovane použiť. Stačí ho myškou pretiahnuť do plochy (vložiť inštanciu symbolu). Adobe Flash umožňuje aj ďalšie zaujímavé možnosti. Animovaným objektom môže byť aj inštancia symbolu - animácie z knižnice. Výsledný pohyb je potom zložený z viacerých pohybov. Ak zmeníme symbol v knižnici, zmenia sa aj všetky jeho inštancie. Viacnásobné použitie symbolu z knižnice nezvyšuje pamäťové nároky výslednej animácie, pretože symbol je definovaný len raz, v knižnici.

Animácii typu „doplnenie pohybu“ môžeme nastaviť aj ďalšie vlastnosti ako napr. nábeh/dobeh, otočenie objektu a pod.

Hodiny vytvorené vo Flash-i







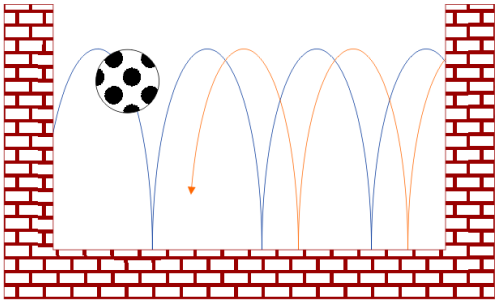
Zdroj: http://www.gjar-po.sk/~gunis/hodiny/

Obmedzený priestor tohto materiálu nám umožnil len letmý prehľad niektorých možností, ktoré Adobe Flash ponúka. Záujemcom odporúčame do pozornosti zoznam literatúry uvedený na konci tohto učebného textu..

Úloha 16

pre náročnejších

Vytvorte animáciu lopty donekonečna sa odrážajúcej od stien miestnosti.



Návod: Vytvorte animáciu, v ktorej bude lopta „skákať“ len vertikálne. Výslednú animáciu animujte v horizontálnom smere. Stenu umiestnite do ďalšej vrstvy.

4.5 Výhody a nevýhody použitia vektorovej grafiky

Bitmapová grafika	Vektorová grafika
obrázok je zložený z farebných grafických bodov	obrázok je zložený z geometrických objektov
každý bod má definovanú svoju farbu a umiestnenie	každý objekt vrátane jeho vlastností je popísaný návodom na jeho zostrojenie
pri manipulácii s obrázkom pracujeme s bodom alebo skupinou bodov	pri manipulácii s obrázkom pracujeme s objektom alebo skupinou objektov
nie je jednoduché pracovať so skupinou bodov, ktoré reprezentujú nejaký objekt, napr. tvár	ľahko sa pracuje so skupinou objektov
obrázky verne zachytávajú obrazy reálneho sveta (napr. fotografia)	obrázky zväčša zobrazujú zjednodušený obraz reálneho sveta (skica, náčrt, jednoduchý obrázok)
pri manipulácii s obrázkom dochádza často ku strate kvality obrázka	pri manipulácii s obrázkom nedochádza ku strate kvality
zmena proporcií obrázka podstatne ovplyvňuje veľkosť výsledného súboru na disku počítača	zmena proporcií obrázka a objektov v ňom len nepatrne ovplyvňuje veľkosť výsledného súboru na disku počítača
súčasná zmena (napr. pri špeciálnych efektoch) väčšieho počtu bodov môže byť časovo náročná	výpočet nových vlastností objektov pri zmene môže byť náročný
zložitosť obrázka obvykle nemá vplyv na náročnosť jeho spracovania	zložitosť obrázka je priamo úmerná náročnosti jeho spracovania
farebné body sa nemôžu prekrývať, nové body vždy nenávratne nahradia body pod nimi (nemyslíme funkciu krok späť)	objekty sa môžu prekrývať, nové objekty môžu prekrývať staršie objekty, poradie objektov sa dá zmeniť
nie je lepšia ako vektorová	nie je lepšia ako bitmapová
vzájomne sa dopĺňajú	

Čo sme sa naučili

- Vysvetliť princíp reprezentácie obrázkov vo vektorovej grafike.
- Porovnať reprezentáciu obrázkov pri rastrovej a vektorovej grafike.
- Určiť vhodnú reprezentáciu obrázku podľa situácie.
- Vysvetliť funkciu základných nástrojov vektorového grafického editora.
- Demonštrovať použitie základných nástrojov vektorového grafického editora.
- Analyzovať zložitejší obrázok a rozlíšiť použitie jednotlivých nástrojov.
- Vysvetliť princíp tvorby animácie reprezentovanej vektorovou grafikou.
- Popísať základné typy animácií reprezentovaných vektorovou grafikou.
- Demonštrovať použitie základných typov animácií reprezentovaných vektorovou grafikou.

Kapitola 5: Programovanie rastrových animácií

Rastrová animácia je reprezentovaná skupinou striedajúcich sa snímok (statických obrázkov). Niektoré programy na spracovanie rastrovej grafiky (napr. Logomotion) umožňujú vytvárať rastrové animácie z jednotlivých rastrových obrázkov. V animácii sa dá nastaviť pri každom obrázku doba trvania jeho zobrazenia a počet opakovaní celej animácie. Pomocou rôznych nástrojov animačného rastrového editora (priesvitky, generovanie animácie z jedného obrázka, generovanie animácie zmiešaním dvoch obrázkov atď.) vieme vytvárať rôzne aj absurdné animácie. Pri tvorbe animácii by sme mali dodržiavať určité pravidlá, napr. zohľadniť vlastnosti prostredia a objektov (pružnosť, zotrvačnosť, gravitáciu), ich vzájomnú prepojenosť (napr. pri animácii chôdze), v každom momente sa zamerať len na jeden aktívny objekt, na zvýraznenie deja využívať zveličovanie atď. [8]

Ak vieme matematicky popísať dráhu pohybu objektu, ktorý sa má pohybovať v súlade s fyzikálnymi, biologickými a inými zákonmi, je efektívnym spôsobom tvorby animácie jej naprogramovanie v niektorom z programovacích prostredí.

Vhodnou ukážkou naprogramovanej rastrovej animácie je pohyb loptičky v nejakom prostredí (napr. vzduchu) vyhodenej šikmo nahor pod uhlom $\propto$ so zadanou počiatočnou rýchlosťou v_0. Pri výpočte trajektórie šikmého vrhu (balistickej krivky) rozdelíme pohyb na malé časové intervaly, na ktorých rýchlosť, resp. zrýchlenie pohybu môžeme považovať za konštantné. Na začiatku vypočítame zložky rýchlosti:

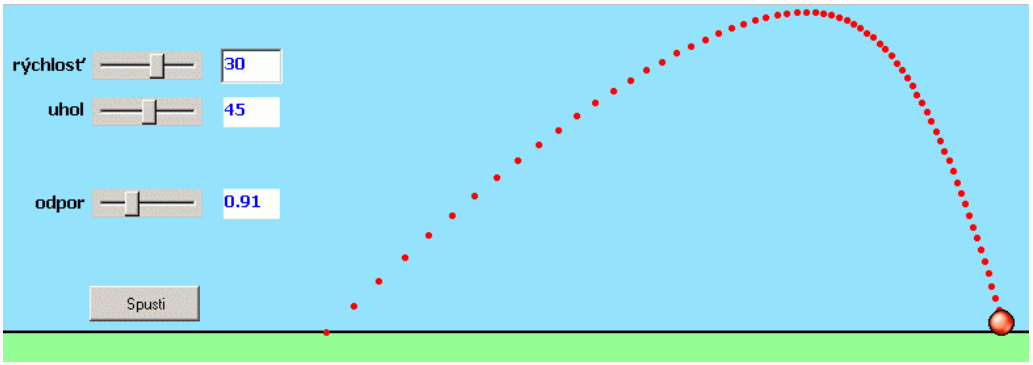
$$v_x = v_0 \cos\alpha \qquad v_y = v_0 \sin\alpha$$

V každom kroku vykresľovania trajektórie upravujeme vektor rýchlosti $[v_x, v_y]$ (vzhľadom na tiažovú silu a odpor prostredia) a pripočítavame príslušné prírastky súradníc za časový interval Δt ($v_x\Delta t$, resp. $v_y\Delta t$) k aktuálnym súradniciam loptičky $[x, y]$. Odporová sila $\vec{F}_o$ pri obtekaní telesa vzduchom bez vírov je priamo úmerná rýchlosti telesa a pôsobí proti smeru pohybu telesa a udelí mu zrýchlenie $\vec{a}_o = \frac{\vec{F}_o}{m} = \frac{-k\vec{v}}{m}$. Tiažová sila $\vec{F}_G$ udelí telesu tiažové zrýchlenie $\vec{g} = \frac{\vec{F}_G}{m}$. Za čas Δt sa zmení rýchlosť a súradnice loptičky nasledovne:

$$v_x \leftarrow v_x + a_{ox}\Delta t + g_x\Delta t = v_x - \frac{kv_x}{m}\frac{v_x}{\sqrt{v_x^2 + v_y^2}}\Delta t$$

$$v_y \leftarrow v_y + a_{oy}\Delta t + g_y\Delta t = v_y - \frac{kv_y}{m}\frac{v_y}{\sqrt{v_x^2 + v_y^2}}\Delta t - g\Delta t$$

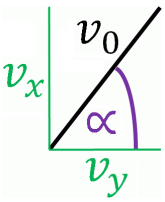
$$x \leftarrow x + v_x\Delta t \qquad y \leftarrow y + v_y\Delta t$$



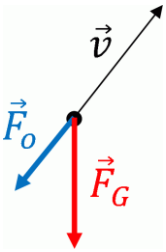
Obrázok 16: Naprogramovaná animácia šikmého vrhu s vykreslenou trajektóriou

Obidva rastrové obrázky sú rovnako náročné na spracovanie:

Pri vektorových obrázkoch to už neplatí:



Zo vstupných hodnôt počiatočnej rýchlosti v_0 a uhla $\propto$ vypočítame zložky rýchlosti v_x, v_y.



Na pohybujúci sa bod v nejakom prostredí (napr. vzduchu) pôsobí odporová sila $\vec{F}_o$ proti pohybu telesa a tiažová sila $\vec{F}_G$.

Ukážka programového kódu animácie šikmého vrhu s vykreslením trajektórie pohybu v programovacom prostredí Imagine Logo.

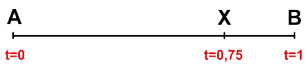
Aplikácia pre vykreslenie balistickej krivky je prístupná v LMS Moodle. (*balistika.imp*)

```
viem pohyb
 ;nastavenie počiatočných súradníc [x y]
 urobTu "x -150
 urobTu "y 0
 ;výpočet zložiek vektora rýchlosti [vx vy]
 urobTu "vx :v * (cos :u)
 urobTu "vy :v * (sin :u)
 ;vypočítanie súradníc [x y] a vykreslenie bodu
 kým [:y >= 0] [
   nechPoz (zoznam :x :y)
   bod 5
   urobTu "zx 1 / (sqrt 1 + :vy * :vy / :vx / :vx)
   urobTu "zy 1 / (sqrt 1 + :vx * :vx / :vy / :vy)
   urob   "vx :vx - :vx * :odpor * :zx * :dt
   urob   "vy :vy - :vy * :odpor * :zy * :dt - :g * :dt
   urobTu "x :x + :vx
   urobTu "y :y + :vy
 ]
koniec
```

Premenná `:odpor` zastupuje pomer $\frac{k}{m}$. Ak neuvažujeme odpor prostredia, tak premennú nastavíme na hodnotu 0, inak nastavíme nejakú nenulovú hodnotu.

Ak k predchádzajúcim úvahám zoberieme do úvahy aj vztlakovú silu (kolmú na vektor rýchlosti) a silu motora, môžeme naprogramovať jednoduchý letecký simulátor.

Pri vytváraní algoritmu riešenia týchto úloh môžeme využiť rovnicu úsečky: $X = A + t(B - A)$, kde A, B sú koncové body úsečky, t je parameter, pomocou ktorého vieme zobraziť vnútorné body úsečky $\overline{AB}$ pre $t \in < 0{,}1 >$.



Úloha 17	Vo vybranom programovacom prostredí naprogramujte animáciu zalievania kvetov pomocou hadice a ich rastu (resp. hru hádzanie do basketbalového koša). Animáciu môžete oživiť zvukom (pri zobrazení každého bodu trajektórie sa prehrá tón s výškou odpovedajúcou vertikálnej pozícii letiacej kvapky vody).
Úloha 18	Vo vybranom programovacom prostredí naprogramujte animáciu pripomínajúcu let vesmírom (t. j. z oblasti zo stredu obrazovky letia okolo nás stále zväčšujúce sa objekty).
Úloha 19	Vo vybranom programovacom prostredí naprogramujte animáciu premeny troj-uholníka na štvorec (resp. iniciálok jedného mena na iné iniciálky). 

Prostredie Imagine Logo umožňuje pri animácii využívať nielen zmenu pozície vykresleného objektu, ale aj zmenu záberu animovaného tvaru (napr. rozfázovaný zaber s pohybmi panáčika pohybujúceho sa vľavo a iný rozfázovaný záber pre panáčika pohybujúceho sa vpravo). Pri tvorbe animácie môžeme využiť tvar korytnačky vytvorený pomocou návodu na kreslenie, napr. pri animácii letiaceho šípu, pohybu sekundových a minútových ručičiek na hodinách.

Čo sme sa naučili

- Popísať základné pravidlá tvorby animácie.
- Vo vybranom programovacom prostredí (napr. Imagine Logo) poznať príkazy na zobrazenie animovanej grafiky.
- Naprogramovať jednoduché animácie pohybujúcich sa, resp. meniacich sa objektov pomocou využitia základných pohybových zákonov, rovnice úsečky, sčitovania vektorov atď.

Kapitola 6: Tvorba multimediálnych aplikácií

V rámci multimédií sa stretávame s pojmami, ako sú **multimediálna prezentácia a multimediálna aplikácia**. Rozumieme pod nimi interaktívne prepojenie súboru obrazoviek obsahujúcich text, obrázky, zvuk, animácie a videozáznamy.

Prezentáciu definujeme ako predloženie, predstavenie sa alebo prehliadku niečoho. Jej účelom je prezentovať, zaujať. Cieľom prezentácie je ponúkanie, presviedčanie, presadzovanie. Prezentácia sa na rozdiel od aplikácie snaží poukázať na určitý problém, prípadne objasniť príčinu jeho vzniku a ponúknuť možnosti riešenia.

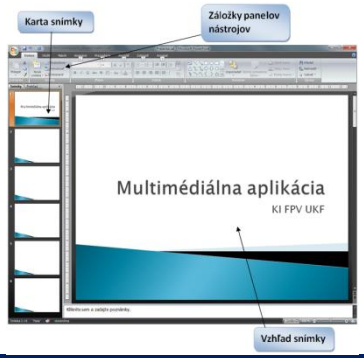
Výraz **aplikácia** pochádza z 80-tych rokov 20-teho storočia, kedy ho zaviedla spoločnosť Microsoft v rámci OS Windows. *Aplikáciou* v informatike nazývame softvér určený na vykonávanie špecifickej funkcie, ktorú požaduje používateľ počítača. Je to teda počítačový program pomáhajúci používateľovi pri uskutočňovaní činnosti určitého konkrétneho typu, napríklad pri manipulácii s textami, grafikou, videom, a pod.

Ak sa rozumie pod multimediálnou aplikáciou jeden, alebo viacero multimediálnych dokumentov, ku ktorým je vybudované používateľské rozhranie, v takom prípade je rozdiel medzi multimediálnou aplikáciou a multimediálnou prezentáciou zanedbateľný.

Úloha 20	Popremýšľajte, aký softvérový nástroj by ste mohli použiť k tvorbe multimediálnej aplikácie alebo prezentácie. Dokážete v ňom zabezpečiť skutočne interaktívne prepojenie mediálnych elementov?

6.1 MS PowerPoint ako jednoduchý nástroj na tvorbu multimediálnych aplikácií

Jedným zo základných nástrojov určených na tvorbu multimediálnych aplikácií, bez nutnosti používateľa vedieť programovať, je MS PowerPoint. Aj mierne pokročilý používateľ vie veľmi ľahko vytvoriť či upraviť už vytvorenú aplikáciu podľa svojich predstáv a aktuálnych potrieb. K pokročilejším zručnostiam pri tvorbe pomocou MS PowerPoint patrí vlastná tvorba nelineárnej multimediálnej aplikácie s použitím makier.

Úloha 21	Skúste navrhnúť šablónu multimediálnej aplikácie. Dbajte na to, aby ste použili farby, v ktorých sa nestráca podstata toho, prečo sa aplikácia vytvára - kvalitný obsah. Vložte ju na prvú snímku do programu MS PowerPoint. Príklad šablóny multimediálnej aplikácie vloženej do programu MS PowerPoint. 
Úloha 22	Vytvorte niekoľko snímok spolu s jednoduchým navigačným menu multimediálnej aplikácie a otestujte ho, či ste správne zabezpečili prechod na jednotlivé snímky iba pomocou navigačného menu aplikácie.

Ak vytvárame akékoľvek menu majme na pamäti, že vždy ide v podstate o hru s tlačidlami a závisí len na nás, aký spôsob usporiadania budeme aplikovať. Pri tvorbe horizontálneho alebo vertikálneho menu, musíme všetky časti presne rozmiestniť, aby nedošlo k nežiaducim medzerám medzi jednotlivými položkami menu.

Úloha 23	Šablónu navigačného menu môžete vytvoriť buď priamo v programe MS PowerPoint, alebo môžete použiť akýkoľvek voľne dostupný program na spracovanie vektorovej alebo rastrovej grafiky (napr. Gimp).

Animačným a zvukovým efektom určeným pre celé snímky, hovoríme prechody snímok. Prechodové efekty pre striedanie snímok môžeme nastaviť v záložke Animácie.

Obrázok 17: Záložka Animácie

Zobrazovanie jednotlivých snímok, či už majú nastavené prechodové efekty alebo nie, môžeme pri projekcii ovládať ručne alebo nechať vykonávať automaticky. Môžeme tak určiť, či má byť daná snímka vystriedaná pri projekcii klepnutím myši alebo automaticky po určitom počte sekúnd. Druhý spôsob sa hodí pre automatické projekcie bez prítomnosti rečníka. Je možné nastaviť i obe možnosti súčasne. Potom sa ďalšia snímka objaví podľa toho, k čomu z týchto dvoch možností príde skôr.

6.2 Zabezpečenie interaktivity v MS PowerPoint

Úloha 24	Do programu MS PowerPoint vložte GIF animáciu pampúcha, ktorý otvára a zatvára ústa. F5 spustite prezentáciu a pozorujte, ako sa daná animácia správa. Dokážete postavičku pampúcha ovládať (napr. určiť smer jeho pohybu)?

Zabezpečenie interaktivity v MS PowerPoint je značne problematické. Interaktívnu aplikáciu je preto potrebné zostaviť veľmi prehľadne a logicky, aby používateľ mohol sám pristupovať k údajom, ktoré sú pre neho potrebné.

Veľký dôraz sa kladie najmä na správne umiestnenie ovládacích prvkov v aplikácii. Interaktívne prvky v multimediálnej aplikácii vytvorenej pomocou MS PowerPoint pozostávajú najmä z tlačidiel a hypertextových odkazov.

Používať takéto rozdelenie je však veľmi všeobecné, keďže v praxi funkcia jedného alebo druhého prvku často splýva do jednej a tej istej veci, ktorou je nadefinovanie akcie alebo odkazu. Rozdiel je najmä v základnom vnímaní pojmu hypertextový odkaz, ako označenia pre činnosť vyvolanú kliknutím na text.

Pri tlačidle však akciu vyvoláme kliknutím na grafický objekt – zástupcu. Hypertextový odkaz sa prejaví farebným podčiarknutím, pri spustení, po klepnutí alebo pri prechode myšou nad odkazom. MS PowerPoint vytvára hypertextové odkazy s podčiarknutím automaticky, pri každom zápise e-mailovej adresy alebo webovej adresy URL do snímky.

Pri vytvorení hypertextového odkazu postupujeme nasledovne:

1. Vyberieme text alebo objekt, ktorému chceme hypertextový odkaz priradiť.
2. Príkazom *Hypertextový odkaz* (Prepojenie) v záložke *Vložiť* otvoríme dialógové okno *Vložiť hypertextový odkaz*. Tu do textového poľa *Adresa* napíšeme alebo vyberieme cieľ odkazu (konkrétnu snímku).

Okrem hypertextových odkazov môžeme použiť na zabezpečenie interaktivity aj už spomenuté tlačidlá. V prípade potreby nemusíme vždy vyrábať nové, ale môžeme použiť tie, ktoré sú implementované priamo v MS PowerPoint. Tlačidlá s akciou vložíme do snímky rovnakým spôsobom ako automatické tvary:

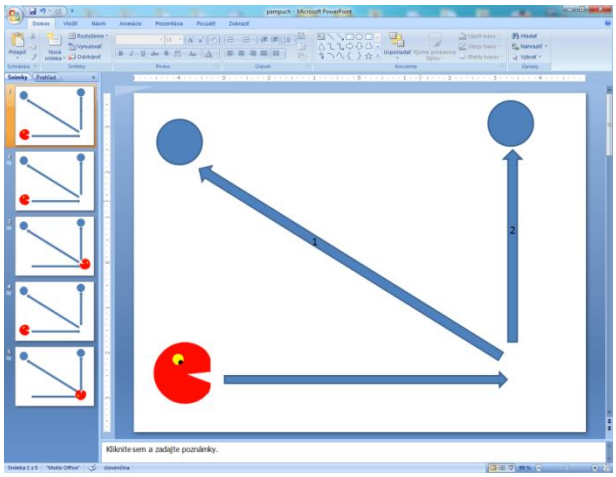
1. Na paneli nástrojov *Kreslenie* potvrdíme tlačidlo *Automatické tvary* a z ponuky vyberieme *Tlačidlá akcií*.
2. Po zvolení potrebného tlačidla sa presunieme na snímku, kde ťahaním myši určíme jeho tvar a veľkosť.

3. Automaticky sa zobrazí dialóg *Nastavenie akcie*, v ktorom sa už nachádza preddefinovaná akcia.
4. Pre zadefinovanie vlastnej akcie odstránime pôvodný text.

V prípade vytvorenia kvalitnej hypertextovej štruktúry pomocou odkazov alebo tlačidiel akcií, určíme v interaktívnej multimediálnej aplikácii sled jednotlivých snímok a teda nemá ju zmysel dopĺňať nahratým hovoreným komentárom. Prvky interaktivity tak môžeme dodať akémukoľvek objektu s výnimkou pozadia snímky.

Úloha 25

V programe MS PowerPoint vytvorte jednoduchú multimediálnu aplikáciu – hru *Pampúch*. Ako pohybujúci sa objekt použite postavičku pampúcha vytvoreného v programe LogoMotion, ktorý opakovane otvára a zatvára ústa. Pampúch sa bude pohybovať po Vami zvolenej trase.



Riešenie je celkom jednoduché. Ako je možné vidieť z návodu zadania úlohy, k dosiahnutiu potrebného výsledku je nutné vytvoriť 5 snímok, pričom každá z nich sa líši od predchádzajúcej len minimálne (premiestnením pampúcha alebo jeho pootočením). Ak chceme, aby sa pampúch pohyboval podľa nami zvolenej trase, musíme mu ju najskôr určiť. Šípky predstavujú smer pohybu pampúcha a taktiež slúžia ako hypertextové odkazy pre prechod na snímku, na ktorej sa odohrá pohyb pampúcha.

Postup:

a) Prvá snímka predstavuje akýsi základ (šablónu) pre vytvorenie pohybu. Šípka číslo 1 a 2 predstavujú hypertextové odkazy.
b) Na druhej snímke nastavte Pampúchovi trasu pohybu doprava a zaškrtnite voľbu pri prechode na ďalšiu snímku „Automaticky po...“
c) Na tretej snímke pootočte Pampúcha správnym smerom a nastavte potrebnú trasu pohybu. Prechod na ďalšiu snímku už nezabezpečujte (nezaškrtnuté položky „Po kliknutí myšou“ a („Automaticky po...“) – pre návrat na prvú snímku môžete použiť tlačidlo *Domov*, ktoré sa nachádza v ponuke Tvary.
d) Na štvrtej snímke je trasa Pampúcha úplne totožná s trasou pohybu, ktorá sa vykonáva na druhej snímke, preto postačí, ak si všetky objekty prekopírujete (ak ste to ešte doteraz neurobili) alebo iba skopírujete Pampúcha (trasa pohybu sa kopíruje spolu s ním).
e) Na piatej snímke realizujte postup z kroku c).

Úloha 26

Do hry Pampúch doplňte ďalšie mediálne elementy (zvuk, text). Vytvorte Prvú snímku spolu s navigačným menu, ktoré bude obsahovať položky: O aplikácii, Kontakt, Pomoc, ... Zabezpečte správne hypertextové prepojenie medzi jednotlivými snímkami a objektmi, ktoré sa na nich nachádzajú.

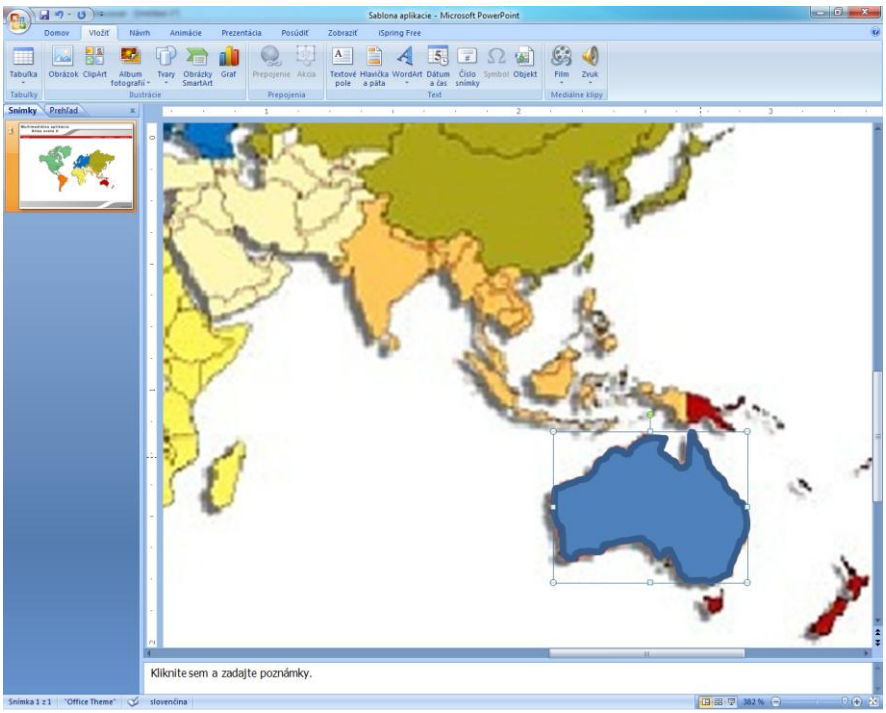
6.3 Vytváranie atraktívneho menu a rôznych tvarov tlačidiel v programe MS PowerPoint

Úloha 27	Vytvorte šablónu multimediálnej aplikácie pre podporu výučby predmetu Geografia podľa predlohy. Multimediálna aplikácia bude pozostávať okrem horizontálneho menu aj z tlačidiel rôzneho tvaru (mapa sveta).

Tlačidlá rôznych tvarov vytvoríme pomerne jednoducho. Ako vidieť zo šablóny v zadaní úlohy, je potrebné najskôr všetky krajiny, ktoré budú predstavovať jednotlivé tlačidlá, navzájom od seba oddeliť – orezať. Po vložení obrázka (v tomto prípade mapy) do programu MS PowerPoint ako pozadia pre prvú snímku vyberieme nástroj „*Čarbanice*“ (Voľnou rukou) z ponuky *Tvary*, ktoré sa nachádzajú na karte *Vložiť* v skupine *Ilustrácie*.

Pomocou tohto nástroja pomerne jednoducho môžeme vytvoriť tlačidlo akéhokoľvek tvaru. Postačí, ak obkreslíme okraje vybraného objektu, v tomto prípade krajiny – napr. Austrálie. Aby sme získali správny tvar, je potrebné pracovať s veľkou presnosťou.

Po označení oblasti sa vyplní tvar a taktiež okraje preddefinovanou farbou, preto ich musíme nastaviť na priehľadnú farbu (pri okrajoch môžeme použiť funkciu – *Bez výplne*) kliknutím pravého tlačidla myši na tvar a výberu položky *Formátovať tvar* z kontextového menu. V položke *Vyplniť* následne nastavíme *Priehľadnosť* na 100%.



Obrázok 18: Príklad vytvárania rôznych tvarov tlačidiel

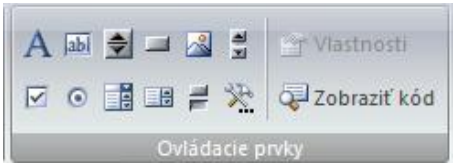
6.4 Integrácia Adobe Flash animácií do programu MS PowerPoint

Ako už bolo spomenuté, formát SWF patrí v súčasnosti medzi štandard v oblasti animovanej grafiky, keďže (okrem rastrovej a vektorovej grafiky) umožňuje využívať aj text, audio a hlavne interakciu.

Ak sme pomocou programu napr. Adobe Flash alebo Adobe Captivate vytvorili animáciu a uložili sme ju ako súbor s príponou SWF, môžeme ju v prípade potreby prehrať aj v programe MS PowerPoint. Musíme však do vytváranej prezentácie či aplikácie vložiť príslušný ovládací prvok ActiveX a vytvoriť z neho prepojenie na súbor s príponou SWF.

Ovládacie prvky ActiveX sa používajú na webových lokalitách a v aplikáciách v počítači, pričom nepredstavujú samostatné riešenia. Môžu byť spustené v programoch ako Internet Explorer alebo programoch balíka MS Office. Aj keď to možno na prvý pohľad nevyzerá, sú veľmi účinné, keďže ide o objekty typu COM (Component Object Model), ktorý využíva neobmedzený prístup k počítaču.

Ak nás však práve uvedená skutočnosť neodradila od nášho zámeru a predsa chceme vložiť animáciu takýmto spôsobom, musíme v MS PowerPoint zobraziť záložku *Vývojár*, cez *Možnosti programu PowerPoint*. Potom v bloku *Ovládacie prvky* karty *Vývojár* klikneme na položku *Ďalšie ovládacie prvky* (je to tlačidlo s ikonou kladiva a maticového kľúča).

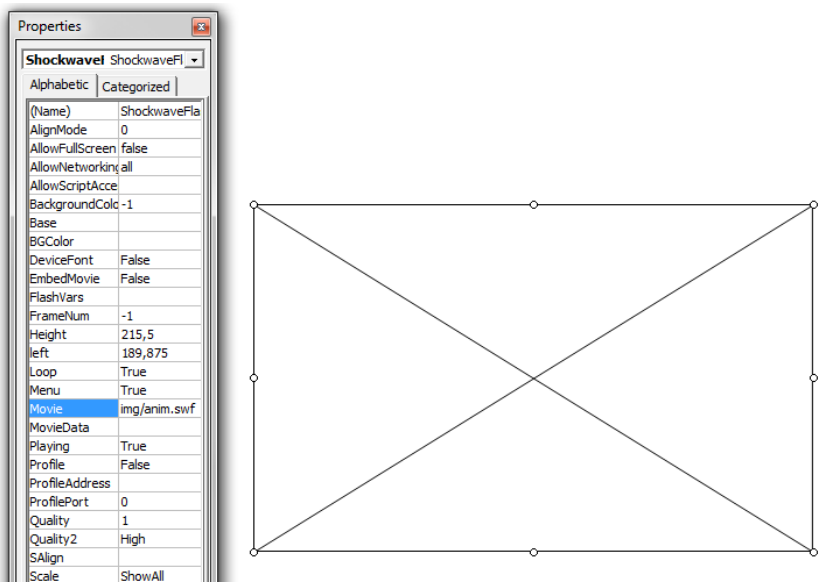


Obrázok 19: Ovládacie prvky

Z ponuky ďalej vyberieme *Shockwave Flash Object*. Kurzorom myši naznačíme na ploche veľkosť nášho zobrazovaného Flash objektu. Klikneme naň pravým tlačidlom myši a vyberieme z kontextového menu položku *Vlastnosti*. Do položky *Movie* na karte *Alphabetic* zadáme relatívnu cestu objektu napr. *img\anim.swf*. Týmto krokom zabezpečíme načítanie objektu do programu MS PowerPoint. Adresár „img“ je umiestnený v hlavnom adresári, kde sa nachádza aj naša prezentácia, pričom v adresári img sa nachádzajú naše flash animácie.

Na profesionálnu tvorbu multimediálnych aplikácií a jednotlivých mediálnych elementov (napr. interaktívne animácie), sa používa program Adobe Flash, ktorý má v sebe implementovaný skriptovací jazyk (tzv. ActionScript).

Tento program je platený, jeho trial verziu si môžete stiahnuť zo stránky spoločnosti Adobe: http://www.adobe.com/products/flash



Obrázok 20: Vloženie flash animácie do MS PowerPoint

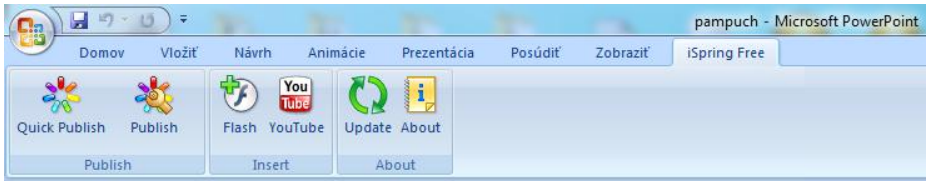
Druhou možnosťou, ako vložiť animáciu do prezentácie programu MS PowerPoint, je použitie špeciálnych programov, ktorých je však našťastie dostatok a sú navyše aj zadarmo. Jedným z nich je napr. *Swiff Point Player*, ktorý integruje voľný plugin do prezentácie vytvorenej pomocou MS PowerPoint, pričom na rozdiel od ActiveX nevidíme iba prázdny „kríž“ ale aj náhľad samotnej prezentácie.

6.5 Export multimediálnych aplikácií a interaktívnych animácií z formátu PPT do formátu SWF

Už vieme, že formát SWF predstavuje štandard pri tvorbe animovanej a interaktívnej grafike, ktorá v súčasnosti dominuje na webových stránkach. Existuje veľa softvérových nástrojov, ktoré umožňujú konverziu formátu PPT do formátu SWF, ale väčšina z nich je plateného charakteru, alebo má obmedzené funkcie.

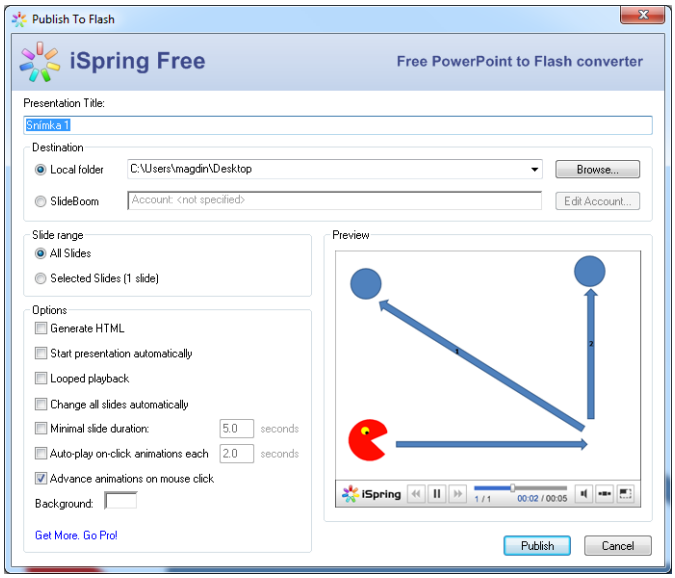
Doplnok Flash iSpring pre MS PowerPoint je určený na rýchlu a jednoduchú tvorbu

interaktívnych flash animácií alebo multimediálnych aplikácií z programu MS PowerPoint pri zachovaní kvality aj po konverzii.



Obrázok 21: Program iSpring implementovaný do MS PowerPoint

iSpring dokáže nielen vyexportovať formát SWF, ale ho dokáže do prostredia MS PowerPoint aj vložiť. Celý export závisí najmä od celkového počtu snímok, na ktorých sa nachádza animovaná grafika a iné mediálne elementy.



Obrázok 22: Okno pre nastavenie parametrov pre export do formátu SWF

Čo sme sa naučili

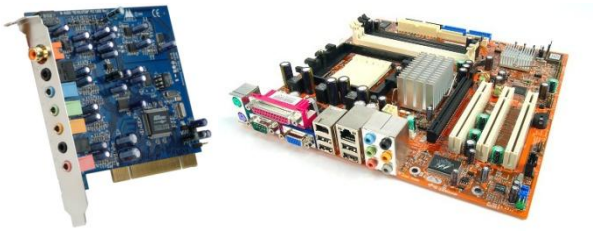
- Popísať rozdiel medzi multimediálnou aplikáciou a multimediálnou prezentáciou.
- Navrhnúť dizajn, ako aj štruktúru multimediálne aplikácie a pomocou nástrojov rastrovej alebo vektorovej grafiky, vytvoriť šablónu pozadia multimediálnej aplikácie.
- Využiť a vhodne implementovať vytvorené mediálne prvky do návrhu multimediálnej aplikácie.

Kapitola 7: Spracovanie zvukov

Zvuk je mechanické vlnenie, ktoré vnímame svojim sluchom. Rozlišujeme jeho výšku, intenzitu a zafarbenie. Na zaznamenanie a prehrávanie zvuku ľudia využívali rôzne zvukové nosiče (voskové valčeky, vynilové platne, magnetické pásky, optické disky) a rôzne prístroje na prehrávanie zvukov (Edisonov Phonograph, gramofón, magnetofón, diskman, mobilný telefón, stolný počítač). S nástupom počítačov sa zvýšila kvalita nahrávania a prehrávania zvukov a vďaka špecializovaným zvukovým programom aj značne rozšírili možnosti digitálneho spracovania zvukov.

7.1 Prehrávanie zvukov

Aby sme si vedeli prehrať a vypočuť zvuk uložený na počítači (digitálny zvuk, uložený vo zvukovom súbore) potrebujeme mať v počítači (samostatnú alebo na matičnej doske integrovanú) zvukovú kartu, na ňu pripojené reproduktory resp. slúchadlá a patričný softvér na prehrávanie zvukov (napr. Windows Media Player, Winamp, Realplayer, QuickTime Player).



Obrázok 23: Samostatná a integrovaná zvuková karta.
Zdroje: http://members.driverguide.com/device/index.php?id=19
http://www.hardwarezone.com/articles/view.php?cid=6&id=1757&pg=10

Nepočujete na počítači prehrávaný zvuk? Skúste skontrolovať nasledovné:

Máte zapojené reproduktory resp. slúchadlá do zvukovej karty (do zeleného konektora resp. pri 5+1 kanáloch tiež oranžového, čierneho a sivého konektora)?

Nemáte na slúchadlách resp. na reproduktore stiahnutý gombík úrovne prehrávania na minimum?

Nemáte zlomený resp. nalomený kábel na slúchadlách? Počujete prehrávané zvuky na slúchadlách z iného počítača?

Máte v operačnom systéme vybrané správne zariadenie na prehrávanie zvukov (integrovaná zvuková karta, samostatná zvuková karta, mikrofón na webkamere)?

Nie je vybrané zvukové zariadenie nastavené na nízku úroveň prehrávania?

Lusknite prstami. Ak stále nepočujete, navštívte ušného lekára. :-)

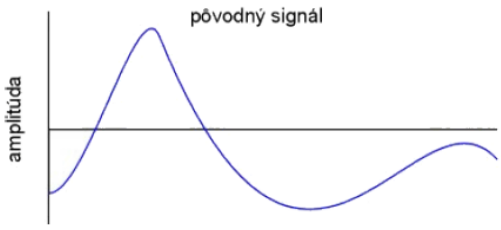
Úloha 28

Nájdite na vlastnom počítači zvukové súbory (napr. s príponou WAV a MP3) a prehrajte ich v niektorom zo zvukových prehrávačoch.

7.2 Nahrávanie zvukov

Pri nahrávaní zvuku do počítača prevádzame analógový zvukový signál na digitálny záznam (digitalizácia zvuku). Pri prehrávaní zvukov ide naopak o prevedenie digitálnej podoby zvuku do analógovej. Na zabezpečenie oboch smerov prevodov sa na zvukovej karte stará analógovo-digitálny a digitálno-analógový (AD/DA) prevodník.

Proces digitalizácie zvuku pozostáva z viacerých krokov.



Obrázok 24: Analógová podoba zvuku.

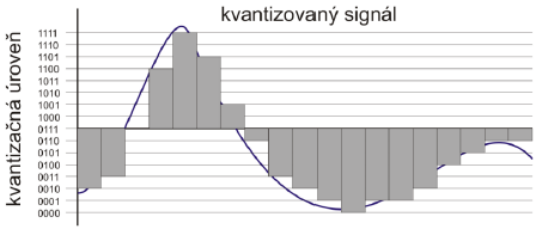
Úroveň vstupného signálu zvuku sa meria v pravidelných intervaloch. Ak pri nahrávaní jednej sekundy zvuku použijeme 44100 meraní, použili sme vzorkovaciu frekvenciu 44,1 kHz. Proces rozdelenia záznamu zvuku na rovnaké intervaly nazývame **vzorkovaním**. **Vzorkovacia frekvencia** by mala byť aspoň dvojnásobkom frekvencie pôvodného signálu (Nyquistov teorém).

Obrázok 25: Vzorkovanie zvukového záznamu.

Príklady parametrov nahrávaní. Niektoré typické sú označené menom:

telefónna kvalita
11 025 Hz, 8 b, mono
rádiokvalita
22 050 Hz, 8 b, mono
CD kvalita
44 100 Hz, 16 b, stereo
DVD kvalita
192 000 Hz, 24 b, 5.1 surround 5 reproduktorov a subwoofer.

Po rozdelení pôvodného zvukového signálu na malé časové intervaly sa pridelí intenzite každému záznamu hodnota najbližšia kvantizačnej úrovne. Počet kvantizačných úrovni sa nastavuje pri nahrávaní zvuku pomocou parametra **rozlíšenie vzorky**. Tento proces nazývame **kvantizácia**.



Obrázok 26: Vzorkovanie zvukového záznamu.

Napokon zakódujeme pre každé meranie jeho kvantizačnú úroveň do postupností núl a jednotiek. Tento krok digitalizácie nazývame **kódovanie**.

Celkovo pri nahrávaní rozlišujeme tieto parametre:

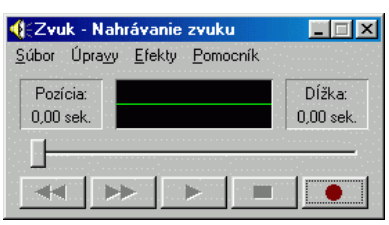
- Vzorkovacia frekvencia (napr. 8000, 11025, 22050, 41100, 48000, ... 192000 Hz)
- Rozlíšenie vzorky (8, 16, 32 b)
- Počet kanálov (1, 2, ... 5+1)

Aký je súvis medzi digitalizáciou obrazu a zvuku? Vidíte analógiu medzi veľkosťou rastra a farebnou hĺbkou rastrového obrázka a vzorkovacou frekvenciou a rozlíšením vzorky zvuku?

Úloha 29	Skontrolujte si v Ovládacom paneli zvuky priradené rôznym udalostiam operačného systému. Nahrajte vlastné zvuky a nastavte ich pre vybrané udalosti (napr. príjem elektronickej pošty, vysypanie koša)
Úloha 30	Zaspievajte čo najčistejšie jeden tón a nahrajte ho so vzorkovacou frekvenciou 44,1 KHz. Z nahratej vzorky vypočítajte frekvenciu nahraného tónu.
Úloha 31	Vypočítajte koľko bitov v operačnej pamäti počítača zaberie 1 sekunda vzorky nahraná so vzorkovacou frekvenciou 44100 Hz, 16-bitovým rozlíšením a stereo kanálmi. Aké rozmery by mal mať bitmapový obrázok s 24-bitovou farebnou hĺbkou zaberajúci rovnakú kapacitu pamäte?

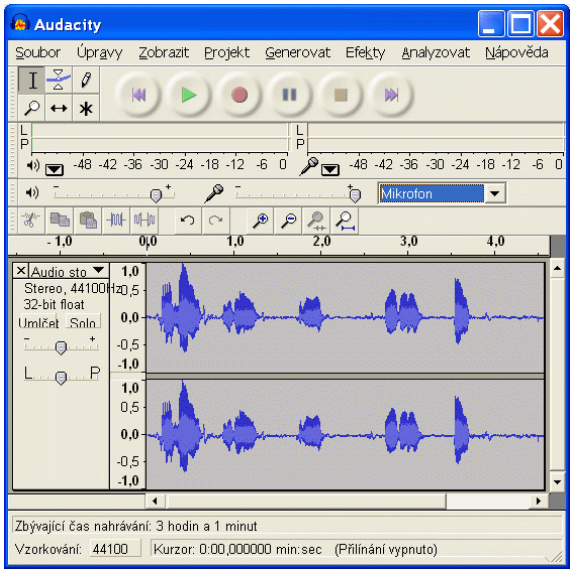
7.3 Spracovanie zvuku

Nahratý zvuk často potrebujeme ďalej spracovať. V operačnom systéme MS Windows je k dispozícii jednoduchý program Nahrávanie zvukov, pomocou ktorého vieme nastaviť parametre nahrávania zvuku, upraviť intenzitu nahraného zvuku, zmeniť rýchlosť, pridať echo, uložiť do patričného formátu atď.



Obrázok 27: Program Nahrávanie zvukov

Pre pokročilejšie možnosti digitálnej úpravy zvuku je vhodný napr. program Audacity. Tento umožňuje spracovávať zvuky vo viacerých stopách, orezávať nepotrebné časti vzorky, meniť intenzitu vzorky po častiach (nábehy a dobehy), odšumovať zvuky, pridávať rôzne efekty (reverb, tremolo, kvákadlo ...)



Obrázok 28: Freeware Audacity na spracovanie zvukov

Úloha 32	Naspievajte a nahrajte dva krátke úryvky pesničiek a upravte ich tak, aby mali rovnakú hlasitosť a aby mali na začiatku a na konci efekt zosilňovania a zoslabovania hlasitosti.
Úloha 33	Uložte nahrané úryvky 2 pesničiek z predchádzajúcej úlohy vo formáte MP3 s rôznymi bitovými tokmi (128, 160, 192, 320 kb/s) a tiež vo formáte OGG. Jednotlivé vzorky porovnajte podľa ich kvality a veľkosti, ktorú zaberajú na disku.
Úloha 34	Zisťovanie výšok hlasov žiakov v triede. Postupne nahrávajte do počítača reč jednotlivých žiakov (napr. nech vyslovia to isté slovo - „mama“). Z takto nahraných vzoriek vypočítajte výšku hlasu jednotlivých žiakov v triede.
Úloha 35	Overenie Dopplerovho javu. Pomocou nejakého zariadenia (napr. diktafónu, mobilného telefónu) zaznamenajte zvuk okoloidúceho auta, uložte ho do počítača a preskúmajte frekvencie zvukov auta pri jeho približovaní a vzďaľovaní od nás.

7.4 Prehrávanie a tvorba skladieb – formát MIDI

Podobne ako pri obraze existujú dva základné popisy obrázkov - bitmapový a vektorový, aj pri zvuku existujú dva základné popisy zvuku. Jednému z nich - vzorkami - sme sa venovali v predchádzajúcej časti. Druhý spôsob sa označuje **MIDI** (Musical Instrument Digital Interface). Ten sa využíva hlavne na popis hudobných skladieb - kedy a na ako dlho sa v skladbe zahrá tón danej výšky, ktorý bude simulovať zvuk daného hudobného nástroja). Pomocou MIDI sa nedajú zaznamenávať zvuky, reč a spev.

Na prehrávanie MIDI môžeme použiť niektorý z bežných prehrávačov zvukov (napr. WinAmp, Windows Media Player). Súbory MIDI môžu byť obohatené o text skladieb, takéto skladby označujeme **karaoke**. Príkladom freewarového programu

na prehrávanie karaoke súborov je vanBasco's Karaoke Player. V tomto programe vieme okrem prehrávania skladieb s textami nastaviť aj vlastné tempo a výšku skladby.

Obrázok 29: Freeware vanBasco Player na prehrávanie MIDI a karaoke súborov

Úloha 36	Stiahnite a inštalujte freewarový program (napr. vanBasco Karaoke Player http://www.vanbasco.com/) na prehrávanie karaoke súborov. Nájdite a spustite karaoke niektorej z vašich obľúbených skladieb. Prispôsobte v programe výšku a tempo piesne, aby ste ju mohli zaspievať.

Programy na tvorbu a spracovanie hudobných skladieb vrátane stôp so spevom sa nazývajú **sekvencery** (typickými predstaviteľmi sú CakeWalk a Cubase). Ďalšie programy – **notátory** (napr. NoteWorthy Composer) dokážu tiež zapisovať noty a tlačiť partitúry MIDI skladieb, iné spolupracujú so skenerom a vedia rozpoznať noty.

Úloha 37	Stiahnite a inštalujte program Anvil Studio (http://www.anvilstudio.com/) a vytvorte v ňom karaoke verziu známej pesničky (napr. Jedna druhej riekla).

Čo sme sa naučili

- Vysvetliť princíp reprezentácie zvukov v počítači (vzorky zvukov, popisy skladieb).
- Popísať spôsoby prehrávania zvukov pomocou patričného softvéru a hardvéru, vedieť vyriešiť problémy spojené s prehrávaním zvukov na počítači.
- Popísať základné parametre nahrávania zvukových vzoriek (vzorkovacia frekvencia, rozlíšenie vzorky, počet kanálov) a základné kroky digitalizácie zvuku (vzorkovanie, kvantizácia, kódovanie).
- Vo vybranom zvukovom editore vedieť nahrať zvuk s danými parametrami, orezať nepotrebné úseky, meniť úroveň hlasitosti, pridávať efekty, odšumovať nekvalitné časti nahrávky, ukladať zvuky do vybraného zvukového formátu s určitou mierou kompresie.
- Poznať základné parametre MIDI súborov, vedieť ich prehrať pomocou patričných programov a prípadne vedieť vytvoriť jednoduchú MIDI skladbu.

Kapitola 8: Spracovanie videa

Videozáznam je zachytenie udalostí reálneho sveta v určitých časových okamžikoch vo forme snímok, ktoré sa podobne ako pri animácii, rýchlo za sebou opakujú. Pri zázname využívame nedokonalosť ľudského oka (jeho rozlišovacej schopnosti), ktoré tieto rýchlo sa striedajúce obrázky vníma ako súvislý pohyb (v skutočnosti však ide len o ilúziu pohybu). Ľudské oko nedokáže tieto obrázky vnímať individuálne a tak vníma projekciu plynulo.

Snímka = frame.

Obrázok 30: Ukážka sekvencie obrázkov

Pri nahrávaní sa štandardom stáva tzv. širokouhlý formát záznamu, kde je pomer strán 16:9. Tomuto formátu sa už prispôsobujú aj LCD na videokamerách, monitory a televízory.

Pri spracovaní videozáznamu je potrebné uvedomiť si, z čoho chceme výsledné video „vyskladať“. V zázname sa môže nachádzať:

Kamery však umožňujú aj konvenčný formát záznamu v pomere 4:3.

- obrazová stopa,
- hudobná stopa,
- zvuková stopa,
- prechody medzi snímkami,
- titulky,
- efekty (obrazové, zvukové).

Napriek rozdielne vyzerajúcim prostrediam video editorov je filozofia spracovania videozáznamu vo všetkých softvéroch identická. Rozlišujú sa len možnosťami a používateľskou podporou.

Existujú rôzne nástroje na spracovanie videozáznamov, napr.: Otvorený softvér VirtualDub, Windows Movie Maker (ako súčasť OS Windows), Apple iMovie Proprietárny softvér Revelation Sight and Sound (RSS), Adobe Priemer, Sony Vegas.

Postup pri spracovaní videozáznamu (niektoré činnosti sa samozrejme prelínajú):

- Vybrať použitý obsah a sústrediť ho do jedného priečinka. Do toho istého priečinka uložiť súbor z videoeditora (zvyšuje sa prehľadnosť a prenositeľnosť).
- Vytvoriť obrazový scenár – predstavu o obrazovej stope, o slede obrázkov a videa. Zmenou obrazovej stopy v budúcnosti môže viesť k zbytočnej úprave ostatných stôp (zvuky, efekty apod.).
- Vybrať hudobný podklad pre výsledné video - výber vhodnej hudby je dôležitý na celkové dotvorenie dojmu, niekedy je potrebné podriadiť hudbe aj časovanie obrazu - je vhodné aby výrazne zvukové prechody boli sprevádzané aj prechodmi obrazovými.
- Ukladaním obsahu videa na časovú os vytvoriť obrazovú stopu, nastaviť parametre pre každé políčko.
- Doplniť videozáznam o textové komentáre (titulky v obrazovej stope alebo prekrývajúce titulky), pozor na časovanie.
- Doplniť videozáznam o vhodné efekty a prechody.

RSS umožňuje vytvárať efekt panoramatického pohľadu na klasické fotografie.

Pri spracovaní videa, môžeme okrem videozáznamu použiť aj fotografie, teda statické obrázky. Pomocou programov na spracovanie videa z nich môžeme vytvoriť zaujímavú dynamickú prezentáciu.

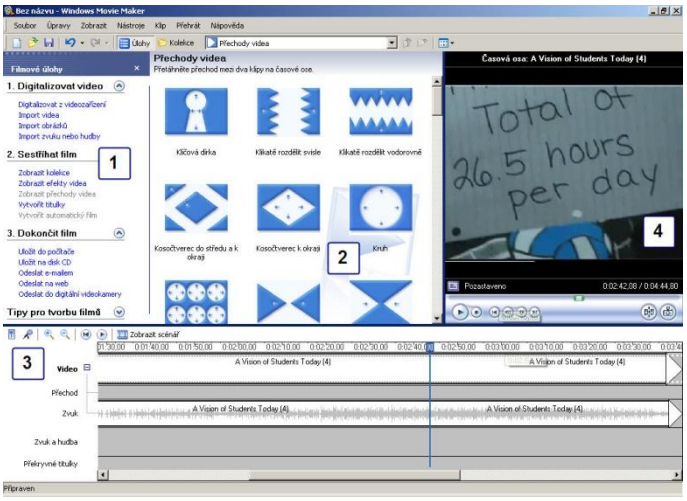
Úloha 38	Stiahnite si s kurzu videozáznam v súbore *Analyzuj.zip* a diskutujte o jeho jednotlivých častiach. Popíšte čo je obrazová, hudobná stopa, zvuková stopa. Nájdite prechody ako i efekty. Vyjadrite sa, ako na Vás záznam pôsobí a čo by ste urobili inak.

8.1 Windows Movie Maker

Program Windows Movie Maker (WMM) je súčasťou operačného systému Windows. Jeho možnosti postačujú na jednoduchú, amatérsku úpravu. Pri profesionálnej práci treba siahnuť po kvalitnejších, platených, softvéroch.

Hlavné okno WMM sa skladá z týchto častí:

1. Filmové úlohy (digitalizácia, strih, efekty, prechody, titulky, ukladanie).
2. Ponuka prechodov, efektov a kolekcií importovaných zdrojov.
3. Časová os (scenár).
4. Náhľad videa.



Obrázok 31: Prostredie videoeditora Windows Movie Maker

Na začiatku práce je nutné vytvoriť si kolekciu zdrojov (import videa, obrázkov a zvukov). Tieto zdroje sa potom ťahaním presúvajú vo vhodnej postupnosti do obrazovej stopy. Môžeme pracovať v dvoch režimoch:


zobraziť scenár


zobraziť časovú os

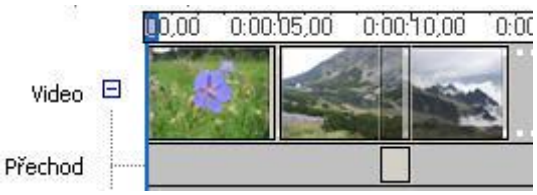
Režim **Scenár** – jednoduchší a názornejší spôsob vytvárania obrazovej stopy, kde je každá snímka prezentovaná samostatne. Pri scenári sa na video lepšie kladú efekty a prechody.

Odporúčame prepínať sa medzi režimami. Práca v každom z nich má svoje výhody.

Obrázok 32: Vytváranie obrazovej stopy v režime scenára

Režim **Časová os** – podrobnejší prehľad o každej snímke a všetkých doplnkoch na nej aplikovaných, spolu so zobrazením titulkovej, hudobnej a zvukovej stopy. V tomto režime môžeme zmeniť aj dĺžku zobrazenia snímky (najmä ak používame statické fotografie). Stačí ťahať hranicu medzi snímkami. Pozor ale na to, že tým sa posúva len obrazová stopa s efektmi a prechodmi, pričom vložený zvuk alebo titulky ostanú na starých pozíciách.

Zmena trvania vybraného úseku sa dá urobiť systémom ťahaj a pusť.



Obrázok 33: Vytváranie obrazovej stopy v režime časovej osi

Ikony indikujúce prechod:


bez prechodu


použitý prechod typ, zhrnú do stredu

Prechody – animovaný efekt zobrazenia snímok. Prechod funguje tak, že sa „ukrojí“ z času pre zobrazenie jednej snímky a času pre zobrazenie snímky nasledujúcej, a určí sa akým spôsobom sa majú na definovaný čas zobraziť súčasne, respektíve akým spôsobom prechádza jedna snímka do druhej. Na vloženie prechodu sú potrebné vždy dve snímky. Pri vložení ďalšieho prechodu na to isté miesto sa starý prechod nahradí novým.

Efekty - slúžia na zvýšenie „zaujímavosti“ aktuálnej snímky (spomalenie, zrých-

lenie, čiernobiely režim a pod.). Efekt sa aplikuje vždy len na vybranú snímku, pričom na ňu môžeme aplikovať aj viac efektov súčasne. Pri vložení ďalšieho efektu sa ich účinok zlúči. Treba však upozorniť, že pri používaní efektov platí, že menej je viac.

Ikony indikujúce použitie efektov:

 žiadny efekt

 jeden efekt

 viac efektov

Ak chceme zmazať vložený snímok, prechod alebo efekt, stačí ho označiť (najlepšie v režime Scenár) a stlačiť Delete.

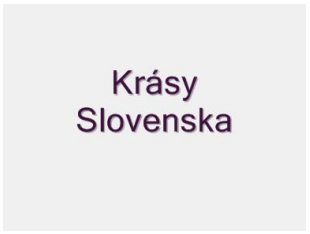
Obrázok 34: Ukážka prechodu (zhrnúť do stredu) a efektu (vodová farba)

WMM ponúka možnosti na jednoduchý strih videa. Stačí kliknúť na ikonu  v dolnej časti náhľadu (je potrebné byť prepnutý v režime Časová os) a záznam sa rozdelí na dva celky. V prípade, ak chceme aplikovať efekt len na vybranú časť videa, je potrebné odlíšiť celky, ktoré budú ponechané bez efektu od tých, kde je efekt žiadaný. To dosiahneme práve strihom záznamu a aplikovaním efektu na novovzniknutú časť.

Kliknutím na ikonu  môžeme aktuálny „frame“ uložiť ako samostatný obrázok. Môžeme teda vytvárať „fotografie“ z videozáznamu.

Titulky - pri vkladaní titulkov máme dve možnosti. Prvá možnosť je vložiť ich do obrazovej stopy. Titulky sa stávajú jej súčasťou ako samostatný sled snímok, ktoré zároveň predĺžia dĺžku trvania videozáznamu. Druhá možnosť je vložiť ich do tzv. priehľadnej vrstvy (prekrývajúce titulky). Titulky nepredlžujú videozáznam, iba sa zlúčia s obrazovou stopov presne v mieste ich vloženia na dobu, ktorú sme titulku určili.

 kliknutím na túto ikonu spustíme prezentáciu videa, do ktorého môžeme priamo nahovoriť komentár. Ten sa uloží ako zvuková stopa záznamu.



Obrázok 35: Ukážka titulkov vložených v obrazovej stope a v stope priehľadnej

720x576 je rozlíšenie stále používaného televízneho signálnu **PAL**. Najnovší systém HDTV (High Definition TV) používa rozlíšenia 1920x1080 (**Full HD**).

Hudba a zvuky - vkladanie zvukov a hudby je najmenej prepracovaná možnosť WMM. Pokiaľ do projektu importujeme videozáznam, ktorý už zvukovú stopu obsahoval, tak dôjde k rozdeleniu obrazu a zvuku do samostatných stôp (video a zvuk), ktoré stále ostávajú zlúčené. Ak dôjde k strihu záznamu, tak sa spolu s ním strihá aj pridružený zvuk. Na videozáznam môžeme potom aplikovať napr. efekt dvojnásobného zrýchlenia, pričom hudba sa automaticky zrýchli spolu s ním. Ak však máme videozáznam bez hudby a chceme ju pridať, tak ju vložíme do stopy *Zvuk a hudba*. Ďalšia editácia takto vloženej hudby je obmedzená len na ponuku: zosilniť, zoslabiť, stlmiť a nastaviť hlasitosť. Navyše, ak dôjde k aplikácii nejakého efektu ovplyvňujúce napr. rýchlosť videa, tak k zmene v samostatne stojacej hudobnej stope nedôjde. Nevýhodou je tiež skutočnosť, že nemôžeme prehrať viac zvukových stôp súčasne (napr. dážď a hromy).

Kodeky je potrebné nahrať do počítača, väčšinou vo forme balíčkov (napr. All In One, K-lite Codec Pack).

Ukladanie – pri ukladaní spracovaného videozáznamu je vhodné premyslieť si na aké účely bolo video vytvorené, kde ho budeme publikovať a tomu prispôsobiť rozmery, dátový tok ako aj formát výstupu.

Rozmer obrazu (často aj rozlíšenie) – počet bodov v horizontálnom a vertikálnom smere. Existujú rôzne štandardné rozmery so skratkami napr. PAL, SECAM, HD Ready, Full HD apod.

Kompresia = zníženie veľkosti záznamu na úkor jeho kvality.

Frekvencia snímok (Framerate, fps – frame per second) – udáva, z koľkých obrázkov sa skladá jedna sekunda záznamu. Štandardom je 25fps, alebo 29,97fps.

Dátový tok (Bitrate) – udáva počet bitov za sekundu, ktoré musí prehrávač pri prehrávaní videa spracovať (dekomprimovať, udáva sa obraz a zvuk súčasne).

Zaznamenáva sa v Kbit/s alebo Mbit/s. (Blue Ray disk má dátový tok takmer 54 Mbit/s, DVD približne 9,8 Mbit-s, bežne sa však používa aj oveľa nižší dátový tok, napr. 218 Kbit/s pre Pocket PC) Všeobecne platí, že čím vyšší je dátový tok, tým kvalitnejší záznam získame. Existujú dva typy dátového toku:

CBR (constant bitrate) – konštantný tok, počas celého záznamu sa prenáša rovnaké množstvo bitov.

VBR (variable bitrate) – premenlivý dátový tok, počas záznamu sa prenáša rozdielne množstvo bitov. Pri náročných, dynamických scénach je tok väčší, pri pomalých alebo statických scénach je zase tok menší.

Pri spracovaní digitálneho videa sú nároky na pamäť a diskový priestor vyššie. Ak by sme totiž uchovávali napr. 1 hodinu záznamu v rozlíšení 720x576 (PAL) potrebovali by sme približne 111GB diskového priestoru. Riešením je kompresia, zmenšenie dátového objemu obrázkov. Kompresných algoritmov je ale veľa, preto nie je možné všetky „pribaliť“ do komprimovaného videa. Preto existuje medzičlánok medzi súborom (videom) a programom, ktorý ho prehráva, tzv. **kodek** (angl. **CO**mpression **DEC**ompression, MPEG-2, MPEG-4, DivX, XviD atď.). Komprimovaný videosúbor má tak uloženú len informáciu o type kompresie a použitom kodeku. Moderné kamery na domáce použitie už ukladajú záznam v komprimovanej podobe, preto je možné na 40 GB HDD uložiť až 47 hodín záznamu.

Úloha 39	Najčastejšie videoformáty sú avi, mpeg, wma, flv, 3gp, mov, ra. Diskutujte o tom, kde je možné vymenované formáty vhodne použiť. Kde ste sa s nimi stretli?
Úloha 40	Využite pripravené video materiály a obrázky v kurze a vytvorte z nich video s rôznymi prechodmi a efektmi, doplnené vhodnou hudbou a titulkami.

8.2 Publikovanie videa na webe

Prenášanie audiovizuálneho materiálu po interne sa nazýva webcasting. Aby používateľ nemusel dlho čakať na stiahnutie veľkého multimediálneho súboru, je vhodné prenášať médium v nepretržitom prúde a tie časti, ktoré už boli prenesené jednoducho prehrať. Tomu sa hovorí streaming, alebo tzv. prúdové vysielanie.

Úloha 41	Diskutujte o konkrétnych príkladoch streaming-u v praxi. Skúste popísať na akom princípe a kde sa môžeme stretnúť s on-line prúdovým vysielaním a tzv. videom na požiadanie (video on demand).

Programy na sťahovanie videí z YouTube:

keepvid.com
www.dvdvideosoft.com

Častým problémom bežného používateľa tvoriaceho amatérske videá je nájsť spôsob, ako pri nízkych kapacitných možnostiach emailových správ, poslať cez internet pomerne veľký súbor, resp. ako ho publikovať na stiahnutie. Jednou z možností je práve publikovať video formou streamu, napr. na YouTube. Takto uložené video je možné okamžite zdieľať s kýmkoľvek. Druhá strana má zase možnosť s použitím konkrétnych nástrojov ako napr. KeepVid alebo Free YouTube Download uložiť video do počítača (kvalita originálu a kópie však už nemusí byť rovnaká). Všeobecný popis ako stiahnuť ľubovoľný prúdový kanál však neexistuje.

O YouTube píšeme z dôvodu jeho komplexnosti, ponúka totiž aj materiál edukačného charakteru. Treba však dať pozor na autorské práva záznamov.

Úloha 42	S prihliadnutím na definíciu CBR a VBR (o stranu skôr), diskutuje o tom, ktorý typ dátového toku je vhodný na stream, ktorý na DVD záznam, satelitný prenos apod. Porozmýšľajte, ktorým spôsobom dosiahneme zníženie veľkosti výsledného súboru, pri rovnakej kvalite záznamu.

Ak nechceme videozáznam prezentovať v prúde, tak je vhodné využiť služby portálov ako je napr. RapidShare, UlozTo a GoogleDocs, kde je možné zdieľať ľubovoľné typy súborov. V prípade prvých dvoch menovaných je možné bezplatné zdieľanie súboru len s obmedzeným počtom pozvaných používateľov. GoogleDocs

však ponúka ako jednu z najnovších možností zdieľanie súboru pre neobmedzený a neautorizovaný zoznam používateľov.

Úloha 43	Neopakovateľným a zaujímavým projektom je kampaň švédskej firmy s názvom The Hero. Ide o film (prúd, kanál), ktorý je vytváraný na základe používateľovho vstupu. Pripravte si vlastnú fotografiu (najlepšie portrét), nahrajte ju do http://en.tackfilm.se/, upravte veľkosť a počkajte na výsledok.
Úloha 44	Skombinujte vedomosti z posledných úloh. Nájdite na YouTube záznam „A Vision of Students Today“ (http://www.youtube.com/watch?v=dGCJ46vyR9o). Uložte ho na disk vo formáte vhodnom na ďalšie spracovanie vo Windows Movie Maker. Pred film vložte titulok v ktorom uvediete, kedy ste stiahli film a adresu, na ktorej bolo umiestnené. Preložte anglické texty, otitulkujte film (aspoň čiastočne) a uložte ho na disk.

Čo sme sa naučili

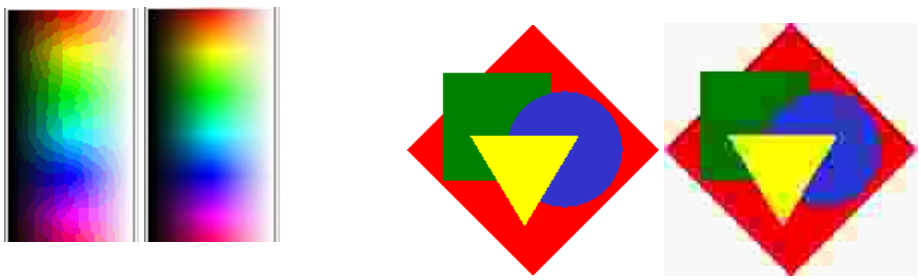
- Chápať z akých elementov sa skladá videozáznam.
- Navrhovať video scenár s využitím rôznych multimediálnych súborov.
- Vytvárať a strihať videozáznam.
- Dopĺňať videozáznamy vhodnými efektmi, prechodmi a titulkami.
- Rozlišovať medzi formátmi a ukladať súbory v týchto formátoch.

Kapitola 9: Grafika na webe

Grafika výrazne ovplyvňuje chuť používateľa prijímať jemu predkladané informácie. Grafika na webe je takmer všade – pozadia stránok, ovládacie prvky ako rôzne tlačidlá, šípky, piktogramy alebo len obyčajné obrázky. Napriek tomu, že na webe je možné používať vektorovú grafiku (SVG, MS SilverLight, Adobe Flash), rastrová grafika je momentálne rozšírenejšia. Najčastejšími formátmi sú GIF a JPG (JPEG).

GIF – statický alebo animovaný obrázok, ktorý dokáže uchovávať 256 farieb. Je preto vhodný na uchovanie obrázkov s jednoduchou farebnou schémou alebo obrázkov s ostrými prechodmi (napr. hrany). Jednu z farieb môžeme označiť za priesvitnú (transparentnú). Táto farba sa pri reálnom použití na webovej stránke nahradí farbou, ktorá je pod obrázkom. Používa bezstratovú kompresiu.

JPEG – statický obrázok, ktorý dokáže uchovávať 16,7 milióna farieb. Je preto vhodný na uchovanie fotografií, alebo obrázkov s komplexnou farebnou schémou. Používa stratovú kompresiu.



Obrázok 36: Ukážka formátu GIF (v ľavo) a JPEG (v pravo) pri komplexnej a jednoduchej farebnej schéme

Formát PNG výrazne znižuje rozdiely medzi obidvoma formátmi (bezstratový formát uchovávajúci milióny farieb s možnosťou navoliť transparentnú farbu).

Pri tvorbe grafiky na web je veľmi dôležité zvoliť správnu kombináciu farieb. Je

Viete že:
Všeobecne sa odporúča priebežne ukladať prácu vytváranú na PC. Pri ukladaní do komprimovanej podoby, napr. jpeg sa však pri každom uložení použije algoritmus komprimácie a tak sa kvalita vždy o nejaké percento zníži. Je preto vhodné editovať obrázky v nekomprimovanom formáte a uložiť do jpeg. len raz.

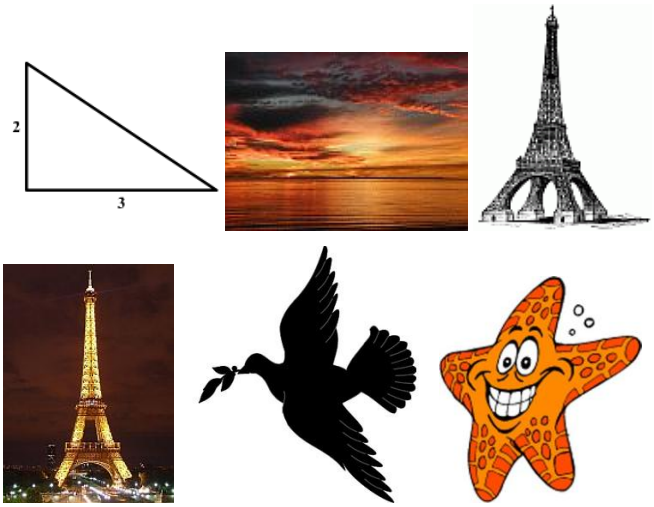
dôležité myslieť na rôzne typy zobrazovacích zariadení (inak sa zobrazuje grafika na klasickom monitore a inak na displeji mobilných zariadení). Zároveň je potrebné myslieť aj na ľudí s farbosleposťou. Pomôcku pri výbere farebných schém nám môžu pomôcť napr.:

http://colorschemedesigner.com/
http://www.colorschemer.com/online.html
http://colors.napcsweb.com/colorschemer/

Poznámka:
Publikovanie fotografií na webe v kvalite akú sme dosiahli pri fotografovaní je zbytočné. Obrázkom je potrebné napr. zmenšiť rozmer. Nie je vhodné len zmeniť atribút <img height =““ wight = „“>, vtedy dochádza len k „softvérovému“ prepočítaniu a zobrazeniu v požadovanom rozmere. Zobrazenie tak môže byť aj mnohonásobne dlhšie ako pri fyzickej zmene rozmerov.

Úloha 45

Ondrej dostal za úlohu rozhodnúť aký formát je vhodnejšie použiť pre nasledujúce obrázky:



Monitory zobrazujú obraz buď v rozlíšení 72 dpi (Mac OS X), alebo 96 dpi (Win), preto by mali mať všetky obrázky určené iba pre prezeranie v PC rozlíšenie 72 dpi. Obrázky, ktoré sa budú tlačiť by mali mať rozlíšenie aspoň 300 dpi (minimálne rozlíšenie tlačiarne).

Jednou z vlastností, ktorú si môžeme pri publikovaní na webe všímať je rozlíšenie **dpi** - vyjadruje pomer medzi rozmermi obrázku v centimetroch (alebo palcoch, 1 palec = 2,54cm) a pixeloch.

Úloha 46

Stiahnite si z kurzu obrázky v súbore *GIFvsJPEG.zip* a porovnajte ako sa zmenil rozmer ako i veľkosť na disku pri použití rôzneho typu, rozmeru a dpi.

Čo sme sa naučili

- Chápať rozdiel medzi orientačnou, informačnou a estetickou funkciou grafiky na webe.
- Rozlišovať medzi formátmi a ukladať súbory vo vhodných formátoch.
- Chápať pojem rozlíšenie a vhodne ho používať v závislosti od rôznych foriem výstupu (web, web +tlačiareň).
- Rozlišovať medzi rozmermi v px a cm v závislosti od zvoleného rozlíšenia.
- Správne voliť kombináciu farieb pri publikovaní obrázkov.

Kapitola 10: Čo sme sa naučili v tomto module

Oboznámili sme sa s významom pojmu multimédia. Ukázali sme si základné princípy a postupy pri tvorbe statických a animovaných obrázkov a zvukov. Poznáme princíp reprezentácie obrázkov pomocou rastrovej a vektorovej grafiky a z toho vyplývajúce dôsledky vieme využiť pri voľbe správneho grafického formátu. Vieme, ako je reprezentovaný zvuk v počítači. Oboznámili sme sa nie len s postupmi tvorby, ale aj so základmi programovania multimediálneho obsahu. Poznáme a vieme aplikovať pravidlá a odporúčania pre tvorbu multimediálneho obsahu a jeho publikovanie na webe.

Preverenie výstupných vedomostí

Záverečné zadanie	Vytvorte jednoduchú multimediálnu aplikáciu (interaktívna Adobe Flash animácia, video, naprogramovanie multimediálnej aplikácie a pod.) podľa inštrukcií lektora.

Literatúra a použité zdroje

[1] Refsnes Data (1999) SVG Tutorial. Dostupné na internete: <http://www.w3schools.com/svg/>

[2] Refsnes Data (1999) Flash Tutorial. Dostupné na internete: < http://www.w3schools.com/flash/>

[3] Adobe Systems Incorporated (2010) Adobe Flash CS4 Professional : Používání programu Adobe Flash CS4 Professional. Dostupné na internete: <http://help.adobe.com/cs_CZ/Flash/10.0_UsingFlash/WS816BB12E-70DE-48c7-9C6C-4735B11BC9E9.html>

[4] Oddelenie didaktiky informatiky a podporných technológií (2010) Typy a námety pre nepovinnú informatiku v 3. ročníku : Corel Draw. Dostupné na internete: <http://di.ics.upjs.sk/informatika_na_zs_ss/studijny_material/grafika/coreldraw/namety.htm>

[5] Oddelenie didaktiky informatiky a podporných technológií (2003) Zaspievajme si spolu : FLASH. Dostupné na internete: <http://di.ics.upjs.sk/informatika_na_zs_ss/studijny_material/grafika/flash/>

[6] Oddelenie didaktiky informatiky a podporných technológií (2010) Typy a námety pre informatiku v 3. ročníku (IKT) : Zoner Callisto. Dostupné na internete: <http://di.ics.upjs.sk/informatika_na_zs_ss/studijny_material/grafika/zoner/namety.htm>

[7] Šnajder, Ľ, Kireš, M. (2007) Informatika pre stredné školy – Práca s multimédiami. 2. vyd. Bratislava : SPN - Mladé letá, s. r. o., Bratislava, 2007, ISBN 80-10-01224-4

[8] De Stefano, Ralph A. (1987) The Principles of Animation. Dostupné na internete: <http://www.evl.uic.edu/ralph/508S99/contents.html>

[9] Magdin, M., Turčáni, M., Burianová, M., Vrábel, M. (2009) Projektovanie multimediálnych aplikácií. 1. vyd. FPV UKF : edícia Prírodovedec, Nitra, 2009, ISBN 978-80-8094-626-5

[10] CTER. University of Illinois. What's the difference between a GIF and a JPEG? Dostupné na internete: http://cter.ed.uiuc.edu/gif-jpeg/#

[11] Mak, J. (2006) Monitory, rozlíšenie a veľkosť obrazu. Dostupné na internete: http://photoshop.szm.com/photoshop-manual/monitory.html

[12] Žára, J., Beneš, B., Felkel, P. (1998) Moderní počítačová grafika. 1. vyd. Brno : Computer Press

Tento študijný materiál vznikol ako súčasť národného projektu Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika v rámci Aktivity „Ďalšie vzdelávanie kvalifikovaných učiteľov informatiky na 2. stupni ZŠ a na SŠ“.

Autori © RNDr. Martin Homola
RNDr. Zuzana Kubincová, PhD.
Mgr. Ján Guniš
Mgr. Martin Cápay, PhD.
PaedDr. Martin Magdin
RNDr. Ľubomír Šnajder, PhD.

Ilustrácie v časti Webové technológie a publikovanie na webe 2
Michaela Danišová

Reprodukcie ilustrácií z publikácie "A walk in the hay-fields", H. & E. Phinney, 1822. Voľne dostupné na www.openclipart.org, vyhotovil používateľ johny_automatic.

Názov Ďalšie vzdelávanie učiteľov základných škôl a stredných škôl v predmete informatika

Podnázov Web, Multimédiá

Študijný materiál prešiel recenzným pokračovaním.

Recenzenti doc. Ing. Ľudovít Trajteľ, PhD.
PaedDr. Roman Hrušecký
PaedDr. Ján Beňačka, PhD.
doc. Ing. Peter Fabián, CSc.

Počet strán 68

Náklad 400 ks

Prvé vydanie, Bratislava 2010

Vydal Štátny pedagogický ústav, Pluhová 8, 830 00 Bratislava, v súčinnosti s Univerzitou Pavla Jozefa Šafárika v Košiciach, Univerzitou Komenského v Bratislave, Univerzitou Konštantína Filozofa v Nitre, Univerzitou Mateja Bela v Banskej Bystrici a Žilinskou univerzitou v Žiline

Vytlačil BRATIA SABOVCI, s r.o., Zvolen

ISBN 978-80-8118-051-4