Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

BAKALÁŘSKÁ PRÁCE

Interaktivní úlohy pro informatickou soutěž

Vypracoval: **Jan Kadlec**

Vedoucí práce: **doc. PaedDr. Jiří Vaníček, Ph.D.**

České Budějovice 2014

Čestné prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v úpravě vzniklé vypuštěním vyznačených částí archivovaných Pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Č. Budějovicích dne

Anotace

Tato bakalářská práce se zabývá přípravou, tvorbou a vyhodnocením jednotlivých interaktivních úloh pro soutěžní webovou aplikací iBobr – Bobřík informatiky. Dále se snaží objasnit a přiblížit technické a organizační problémy spjaté s nasazením těchto úloh a jejich uplatnění v soutěži.

Klíčová slova:

Soutěž, Bobřík informatiky, interaktivní, úlohy, flash, action script 2.0

Abstract

This work deals with the preparation, production and evaluation of the respective roles of interactive web applications for a contest iBobr - Bobřík informatiky. It also strives to clarify and bring technical and organizational challenges associated with implementing these tasks and their application in the real competition.

Keywords:

Informatics, contest, Beaver, Bobřík informatiky, interactivity, flash, action script 2.0

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan KADLEC**
Osobní číslo:	**P10334**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie ve vzdělávání**
Název tématu:	**Interaktivní úlohy pro informatickou soutěž**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Online soutěž žáků SŠ a ZŠ Bobřík informatiky vyžaduje kromě tradičních úloh s výběrovou odpovědí také úlohy interaktivní, realizované pomocí apletů Flash. Tyto úlohy musí kromě své vizuální funkčnosti komunikovat s prostředím testu (php + MySQL).
Úkolem studenta bude vytvořit sadu takových interaktivních úloh na daná témata a vyzkoušet jejich funkčnost během ostré soutěže. Student optimalizuje metodu implementace takových interaktivních úloh do prostředí testu včetně možností zobrazení situace, kterou soutěžící odeslal jako odpověď na soutěžní otázku, formou zpětné vazby pro žáka ve vzorovém řešení úlohy. Student vybere pestrou sadu problémů především z hlediska ovládání apletu a práce s různými typy objektů a dat, uplatní postupy, v nichž bude možno změnou vstupních dat úlohy modifikovat nebo vytvářet různé varianty či dokonce typy úloh v témže apletu.
Student následně vyzkouší vytvořené úloh v ostrém testu národního kola soutěže a zjištěné zkušenosti, příp. zpětnou vazbu od soutěžících či učitelů popíše.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **COMPUTERPRESS. Adobe Flash CS5 Professional - uživatelský manuál. Brno: Computer Press, 2012. ISBN 978-80-251-3224-1.**
2. **Adobe. Adobe Flash CS6 - Oficiální výukový kurz. Brno: Computer Press, 2013. ISBN 978-80-2513-802-1.**
3. **VANÍČEK, J. Bobřík informatiky, výběr úloh z národních kol soutěže 2008 a 2009. Praha: Výzkumný ústav pedagogický, 2009. 32 s. ISBN 80-87000-26-7.**
4. **HRUŠECKÁ, A., PEKÁROVÁ, A., TOMCSÁNYI, P., TOMCSÁNYIOVÁ, M. Informatický bobor - nová súťaž v informačných technológiách pre žiakov základných a stredných škôl. [online] Portál Česká škola, 21. 4. 2008. Dostupné z www http://www.ceskaskola.cz/ICTveskole/Ar.asp?ARI=104994&CAI=2129].**
5. **Informaciniu technologiu konkursas "BEBRAS" [online]. Vilnius: Matematikos ir informatikos insitutas [cit. 20. 11. 2008]. Dostupné z www http://www.bebras.org.**
6. **TOMCSÁNYI, P., VANÍČEK, J. Implementation of informatics contest Bebras in Czechia and Slovakia. In Mechlová, E. , Valchař, A. (eds.): Information and communication technology in education '09. Ostrava: Ostravská univerzita, 2008. s. 214 - 218. ISBN 80-7368-459-4.**
7. **PŘIBYL, V. Online soutěž v informatických znalostech pro žáky ZŠ a SŠ. Diplomová práce, 2012. Jihočeská univerzita, Pedagogická fakulta.**

Vedoucí bakalářské práce: **doc. PaedDr. Jiří Vaníček, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **24. září 2013**

Termín odevzdání bakalářské práce: **2. května 2014**

L.S.

Mgr. Michal Vančura, Ph.D.
děkan

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 24. září 2013

Poděkování

Tímto bych rád poděkoval doc. PaedDr. Jiřímu Vaníčkovi, Ph.D. za cenné rady a odborné vedení během vypracování této bakalářské práce.

Obsah

1 Úvod

1.1 Motivace

Téma práce jsem si vybral po seznámení se s online soutěží Bobřík informatiky a po přizvání do realizačního týmu jako programátor flashových úloh. Po prohlédnutí a vyzkoušení si testů z předchozích soutěží jsem došel k závěru, že mohu připravit testy novějšími a zajímavějšími formami.

Při vývoji úloh jsem mohl uplatnit své zkušenosti a blíže se seznámit s komunikací mezi apletem FLASH a programovacím jazykem PHP.

1.2 Cíle práce

Mým hlavním cílem je zpracovat jednotlivá zadání vybraná z mezinárodně schválených úloh pro soutěž Bebras do interaktivní podoby. Tyto úlohy musejí umět komunikovat s webovou aplikací iBobr podle daných metod, případně metody pro komunikaci doplnit. Úlohy budou obsahovat grafické prvky, které budou vhodné pro studenty základních a středních škol.

Bude proveden průzkum interaktivních úloh, které již byly v soutěži použity.

Úlohy budou vytvořeny s rozdílnou logikou řešení a jejich obsah lze nastavit, čímž bude možno úloze měnit obtížnost a opakovaně je používat pro různé obtížnosti a jiné ročníky soutěže.

U úloh, kde je to možné, bude umožněno zpětné znázornění postupu řešení.

Navrhnout a realizovat jednotnou metodu pro nastavování a editování obsahu. Tedy nebude zapotřebí úlohy opakovaně kompilovat při změně zadání.

Vytvořené úlohy budou otestovány v ostré soutěži a případně podměty ke změnám budou zapracovány.

1.3 Metoda práce

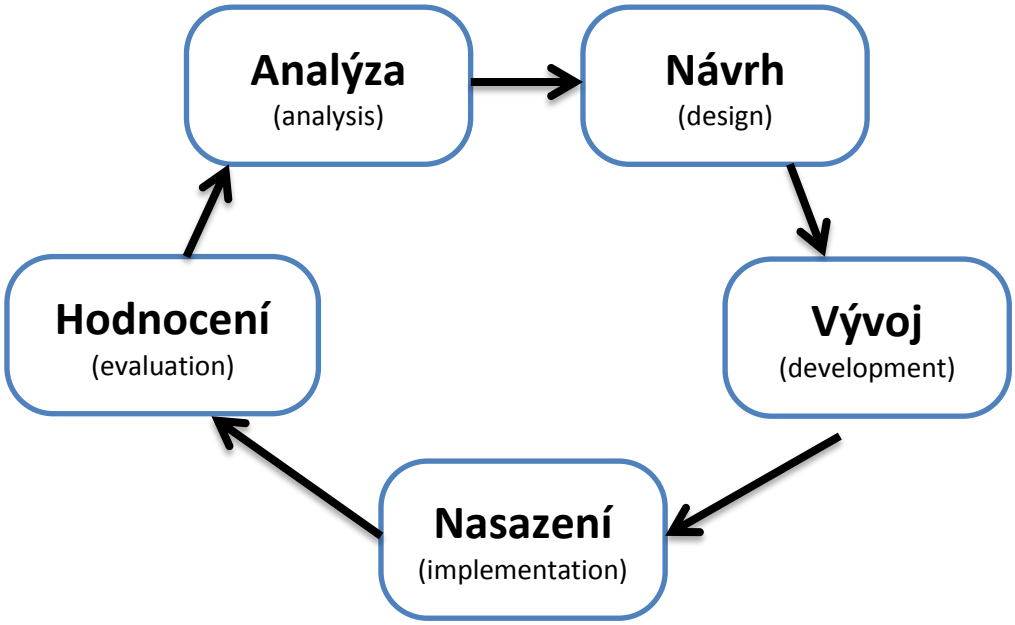
1.3.1 Studium materiálů o online soutěži Bobřík informatiky

Na prostudování soutěže Bobříka informatiky mi byly od vedoucího práce poskytnuty veškeré materiály z magisterské práce Vladimíra Přibyla, se kterým jsem rovněž konzultoval podrobnosti funkcí a nových možností řešení pro nasazování flashových úloh.

1.3.2 Tvorba interaktivních úloh

Testové úlohy byly vyvíjeny a testovány v souladu s modelem ADDIE.

Obrázek 1 znázorňuje vývojový cyklus této metody.



Obrázek 1: Schéma metody ADDIE

Fáze první: Analýza

Na základě údajů o možnostech webové aplikace iBobr bylo zjištěno, jaká pravidla pro komunikaci se serverem musejí výsledné flashové úlohy splňovat. Za předpokladu, že soutěžící mohou používat myš i klávesnici, mohou se v úlohách vyskytovat metody pro výběr či přetahování objektů myší a zároveň obsahovat textová pole, která lze vyplňovat, nebo se v nich může zobrazovat dynamický obsah v závislosti na dění či stavu úlohy.

Fáze druhá: Návrh

Na základě analýzy a požadavků ze strany realizačního týmu jsem vytvořil společný model pro všechny úlohy. Všechny úlohy si po spuštění načtou vlastní data se strukturou v XML s individuálním nastavením a inicializují si své proměnné. Po úspěšném načtení všech dat se spustí samotná úloha programovaná jazykem Action Script 2.0. Dokončení úlohy je závislé na jejím konkrétním zadáním, ale funkce odesílající odpověď a komunikující s webovou aplikací iBobr jsou ve všech úlohách shodné.

Součástí návrhu je rastrový model s výslednou grafikou pro konkrétní úlohy. Pomocí tohoto modelu jsem mohl odvodit maximální požadované rozměry samostatných úloh.

Fáze třetí: Vývoj

Vývoj samotných úloh je z větší části individuální na základě konkrétních zadání. Vycházel jsem z rastrového návrhu, jehož prvky jsem postupně převáděl do vektorových objektů. Ty jsem následně přetvářel na Movie Clipy s vlastní funkcionalitou. Spolu s vývojem jsem využíval program Adobe Photoshop pro vytváření rastrových modelů, které nebylo vhodné vytvořit pomocí vektorů.

Hotové objekty byly samostatně spouštěny a testovány na přijímání všech možných parametrů, které by mohly v budoucnosti získat na základě změn v zadání a případně byly ošetřeny pro případ nevhodných požadavků.

Fáze čtvrtá: Nasazení

Testování nasazení probíhalo ve webové aplikaci iDeveloper, která byla vytvořena v rámci online soutěže iBobr. Tato aplikace simuluje předávání a přijímání parametrů stejnými metodami, které jsou použity v aplikace iBobr a lze s nimi jednoduše manipulovat.

Samotné nasazení úloh do soutěžních testů proběhlo na základě testování funkčností úloh a komunikaci se serverem.

Úlohy byly do soutěžních testů nasazeny osobami, které mají oprávnění k editaci soutěže.

Fáze pátá: Hodnocení

Po proběhlé soutěži s použitými flashovými úlohami byly získány zpětné reakce na jejich funkčnost a provedení. Reakce byly veskrze pozitivní. Část dotázaných by ocenila lépe řešené zpětné zobrazení řešení jednotlivých úloh.

Fáze šestá: Doplnění funkcí na zpětnou rekonstrukci řešení

Na základě zpětných reakcí jsem vytvořil novou metodu odesílání dat, která předává jak výsledek úlohy, tak přesný postup řešení, který se ukládá spolu s výsledkem do databáze. O tyto hodnoty lze úlohou zpětně požádat a postup řešení rekonstruovat tak, jak jej soutěžící vymyslel.

Při opuštění podstránky s flashovou úlohou a opětovném vrácení na tuto podstránku se flashová úloha znovu načítá a pokud nalezne odeslané řešení, zrekonstruuje nejprve postup řešení s možností úlohu opět začít řešit od počátku.

1.4 Příloha práce

Vytvořené úlohy jsou vloženy mezi soutěžní otázky, které je možné nalézt na webové adrese Bobříka informatiky: http://www.ibobr.cz/test/archiv/.

Součástí tištěné verze bakalářské práce je CD, které obsahuje veškeré zdrojové soubory interaktivních úloh ve formátu FLA.

2 Problematika interaktivních úloh

2.1 Definice výrazu „Interaktivní úloha“

Vytvořené úlohy do soutěže Bobřík jsou interaktivní v případě, že soutěžícímu vracejí ihned a bez znovunačtení stránky informace o stavu jeho řešení. Mohou obsahovat prvky jako je přetahování objektu, možnost vložit příkaz a potvrzovací tlačítka, na základě kterých se ihned v rámci úlohy provede činnost sdělující soutěžícímu stav úlohy, nebo zobrazí informaci o správnosti aktuálního výsledku.

2.2 Online soutěž Bobřík informatiky

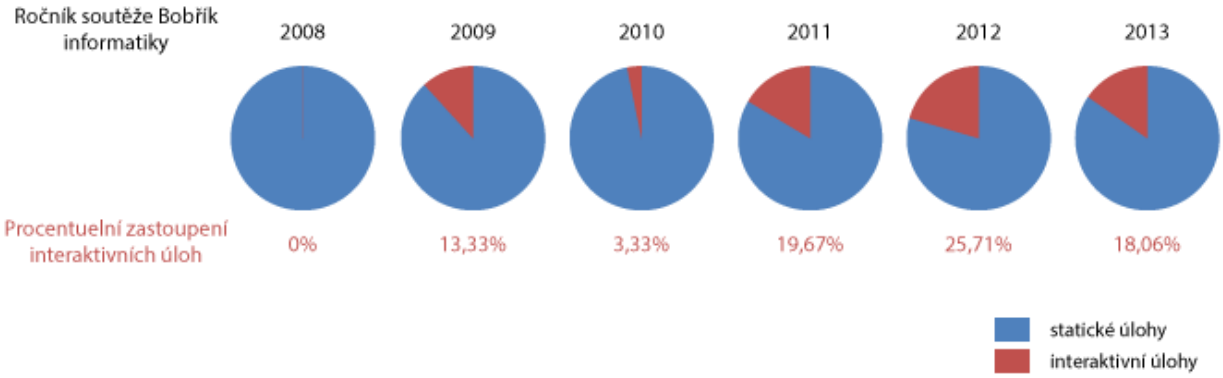
2.2.1 Interaktivní úlohy v soutěži iBobr

Soutěže informatiky iBobr probíhají již od roku 2008. Během této doby došlo k navýšení počtu kategorií ze tří na pět v roce 2013. Každá kategorie obsahuje 10 až 15 soutěžních otázek. V roce 2009 byly do soutěže úspěšně zapojeny první interaktivní flashové úlohy.

Jelikož ze začátku nebylo možno úlohy editovat pomocí externího souboru, bylo možno úlohy použít jen na jeden ročník soutěže, aby nedošlo k opakování stejného zadání. Jiné znění úlohy bylo možno provést pouze úpravou ve zdrojovém kódu a poté opětovnou kompilací úlohy v programu Flash.

V následujícím grafu jsou zobrazeny výsledky z průzkumu procentuálního zastoupení interaktivních a statických úloh pro soutěž Bobřík informatiky v rozmezí let 2008 – 2013.

***Obrázek 2:** Graf procentuálního zastoupení interaktivních a statických úloh v letech 2008 až 2013*



Z grafu je zřetelný nárůst zastoupení interaktivních úloh. K poklesu v roce 2013 došlo z důvodu malého počtu programátorů schopných připravovat interaktivní úlohy v programu Flash. Úlohy vytvořené v rámci této bakalářské práce by mohly zastoupení trvale navýšit, neboť je možné je pomocí externího souboru s nastavením upravovat do jiných podob.

2.2.2 Požadovaný formát publikovaných testů

Pro vytváření interaktivních aplikací, které lze prezentovat pomocí webových aplikací se nabízí více apletů než je Flash, a to například Silverlight, Java nebo přímo HTML za pomoci javascript. Flash je pro tuto situaci nejvhodnější (1) viz práce Vladimíra Přibyla, který vypracoval zabezpečení pro komunikaci mezi aplikacemi Flash a webovou aplikací iBobr (2). Dalšími výhodami apletu Flash je, že k jeho přehrávání je zapotřebí nainstalovaný modul pro internetové prohlížeče Adobe Flash PLayer, který je obecně rozšířený a na počítačích, kde probíhají soutěže zapojených škol, je instalován ve všech případech na rozdíl od modulu Silverlight.

Skriptovací jazyk se nabízí Action Script 2.0 a Action Script 3.0. Z důvodu rozšířenější dokumentace jazyka AS2.0 pro takovýto druh úloh a pro jednodušší implementaci vzorových knihoven a funkcí jsem zvolil právě AS2.0.

2.2.3 Požadované vstupy a výstupy pro komunikaci s webovou aplikací iBobr

Pro komunikace mezi flashovými aplikacemi a webovou aplikací iBobr jsou vypracovány a zdokumentovány metody (3) odesílající a přijímající data ze strany php a javascript aplikace iBobr.

Aby mohly být úlohy použity v soutěži bobřík informatiky, musí umět zpracovat parametry, které jim budou předány soutěžní aplikací. Tyto parametry budou také důležité při sestavování výsledků úlohy, které jsou potom odesílány na server k vyhodnocení.

Seznam a význam vstupních parametrů

- **otazka** - Číslo otázky Tato proměnná nemá vliv na chod samotné úlohy, ale je třeba ji předat jako identifikátor, až v dotazu při odesílání výsledků úlohy na server.
- **onlyshow** - Ovládací prvky úlohy Nabývá pouze hodnot true, nebo false. Udává, zda má úloha zobrazit (true), nebo skrýt (false), své ovládací prvky (tlačítka, informace o uložení údajů, …).
- **idata** - Obsahuje data s výsledky, která mohla být odeslána na server. Pomocí těchto údajů může být úloha rekonstruována (rozmístění objektů,…) do stavu v jakém ji soutěžící uložil. Před uložením výsledků, nebo pokud úloha žádná data neukládá, neobsahuje proměnná žádná data (null).

Seznam a význam výstupních parametrů odesílaných pomocí HTTP metodou POST

- **Otázka** - obsahuje identifikátor úlohy v rámci testu.
- **Typ** - může nabývat hodnot flash, nebo flashtext.
- **Odpověď** - objekt hodnot, obsahuje výsledek řešení úlohy.
- **flashdata** - volitelná hodnota objektu Odpověď, do které si může úloha uložit libovolné textové údaje, které mohou být opětovně získány.
- **Flash** - znamená, že je v proměnné odpoved odeslán rovnou celočíselný výsledek pro získání správnosti řešení, případně zrušení předchozího řešení.
- **Flashtext** - znamená, že je v proměnné odpoved uložen textový řetězec kódovaný v base64, jenž má být na serveru nejdříve vyhodnocen a v rámci správnosti určena odpověď k uložení.

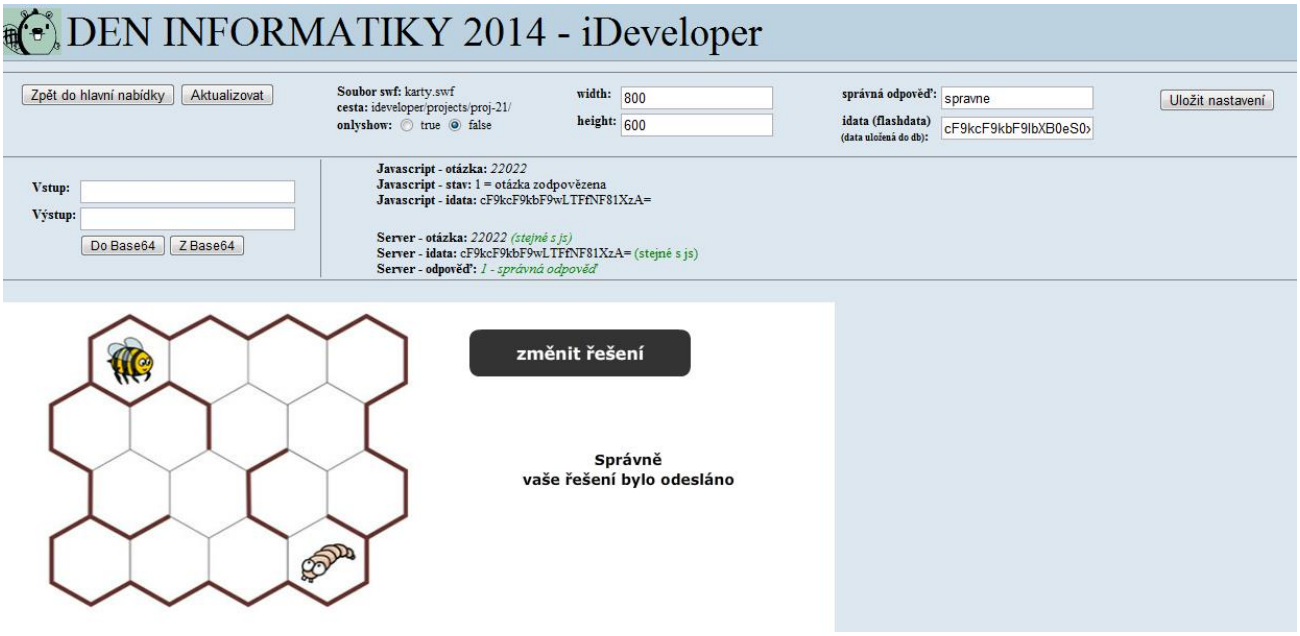
Rozměry úloh

Před přípravou grafických podkladů bylo zapotřebí určit maximální velikosti úloh, aby nedocházelo k jejich deformaci následným zmenšováním. Z prostoru, který je pro úlohy vyhrazen webovou aplikací iBobr vychází maximální šířka na 720 pixelů, výšku není zapotřebí omezovat, ale z důvodu přehlednosti jsem zvolil obdélníkový formát orientovaný na šířku.

Aplikace iDeveloper

V rámci projektu webové soutěže Bobřík informatiky mi byla poskytnuta testovací aplikace pro testování úloh programovaných v prostředí FLASH. Tato aplikace je schopna načítat všechna potřebná data pro běh úlohy a umožňuje je uchovávat a předávat stejným způsobem, jako soutěžní prostředí online soutěže Bobřík informatiky.

Simuluje komunikaci mezi flashovými úlohami a online soutěží a umožňuje dešifrovat data, která jsou přenášena zakódována v Base64.



Obrázek 3: ukázka testovacího prostředí iDeveloper

2.3 MC Tween: Open Source knihovna (4)

Použití této knihovny volím z důvodu její malé kapacity a poskytnutí potřebných užitečných funkcí, které v aplikacích uplatňuji.

Mc Tween je knihovna obsahující funkce pro rozpohybování, transformace a změny efektů nezávisle na časovou osu a to na Movie Clipy v programu Adobe Flash za použití skriptovacího jazyka Action Script 1.0 a AS2.0. Knihovna obsahuje 30 typů animace na rozdíl od funkcí „Ease“ (5) programu Adobe Flash.

Tyto funkce lze používat k rozpohybování statických Movie Clipů bez využití časové osy. Průběh animace je závislý pouze na vlastním časovači, který se po dokončení či přerušení animace automaticky odstraní. Nedochází tak ke kolizím při využívání základní funkce jazyka AS2.0 zvané: setTimer().

3 Průběh a výsledky práce

3.1 Výběr úloh

3.1.1 Zdroj a kritéria

Po konzultaci s vedoucím práce bylo vybráno šest úloh. Vybíráno bylo z úloh odsouhlasených pro mezinárodní soutěž Beaver. Výběr vycházel z předpokladu, že přidaná interaktivita pomůže soutěžícímu v pochopení a řešení úlohy. Dále že bude obsahovat odpovídající grafické prvky pro studenty základních škol.

3.1.2 Vybrané úlohy a jejich charakteristika

Hlavolam

Úloha, kde je cílem najít v matici 5x5 políček správnou cestu tak, aby se skrze všechna políčka prošlo pouze jednou. Tato matice může obsahovat překážky. Počátek si podle zadání může soutěžící určit sám, nebo je nastaven pevně daný.

Kamiony

Úloha představuje si situaci, kde jsou 2 nákladní vozidla a každé má danou maximální nosnost. Zároveň je připraven nastavený počet boxů o různé váze, které mohou, ale nemusí být všechny na nákladní vozidla naloženy. Cílem úlohy je vytvořit takové 2 skupiny nákladu, které nejvýhodněji využijí maximální nosnost nákladních vozidel.

Nahrazování

Simulace funkce „Nahradit“ podobné té, která bývá v textových editorech. Do úlohy jsou načteny řádky s textem a cílem soutěžícího je pomocí funkce „nahradit“ postupně modifikovat původní texty na texty shodné se zadáním.

Plotr

Úloha, která představuje simulaci frézovacího stroje, který do matice vybarví požadovanou cestu dle zadání. Soutěžící napíše sekvenci znaků definujících směr a počet opakování určitých sekvencí směrů. Tento řádek bude programem vykreslen a vyhodnocen, zda výsledný obrazec odpovídá zadání.

Roboti

V matici představující bludiště se nalézají dva roboti. Soutěžící připraví pro každého z nich sekvenci znaků vyjadřujících postupné kroky určitým směrem. Cílem úlohy je, aby si roboti prohodili své pozice, aniž by při cestě bludištěm nastala kolize s překážkami, nebo mezi roboty samotnými.

Doplňování řady

Je aplikace, která může zcela měnit svůj význam a obsah. Je závislá na zadání, ve kterém jsou napsány počáteční souřadnice objektů a obrázků, které se k nim načítají, a pořadí, ve kterém mají být tyto objekty složeny za sebou. Úlohou soutěžícího je pochopit vzor tvarů, barev, čísel nebo jiných prvků a podle tohoto vzoru pokračovat v řadě výběrem objektů a jejich správným seřazením a to pomocí myši a přetažením.

3.2 Společné prvky úloh

3.2.1 XML formát dat s nastavením

Každá z úloh načítá data s jejím nastavením. Tímto nastavením je možno měnit obtížnost úlohy, případně i její vyznění. Data jsou zapsána ve formátu XML. Zvolil jsem pro tyto soubory jednotný název **xmldata.xml**.

Zde bylo zapotřebí, aby byl úloze předáván parametr s cestou k souboru xmldata.xml, protože z adresářové struktury webové aplikace iBobr není jednoznačné, kde je soubor uložen. Tato funkcionalita byla tedy autorem Vladimírem Přibylem následně doplněna. Při načtení úlohy je tedy úloze předáván parametr s cestou, kde je XML soubor uložen.

Předávaný parametr **cesta** je generována webovou aplikací iBobr a představuje umístění souboru ve formě relativní cesty, která odpovídá hierarchickému modelu souborů webové aplikace.

XML data mají společnou formu, odpovídají standardu XML a vypadají následovně:

```
<?xml version="1.0" encoding="utf-8"?>
<nazev_ulohy>
        <vlastnost   hodnota_parametru1="…"  hodnota_parametru2="…"/>
        <jina_vlastnost>…   …</ jina_vlastnost >
        …
        …
        …
</nazev_ulohy >
```

3.2.2 Parser načítající xmldata.xml

První časový snímek všech úloh je vyhrazen pro definování statických proměnných a funkce načítající data s nastavením.

Po zpracování parametru s cestou je spuštěna funkce představující parser, která automaticky naplní potřebné proměnné.

```
…
//Definice statických proměnných (příklad1:Array příklad2:Number)
…
var data_xml:XML = new XML();                    // Deklarace objektu typu XML
data_xml.onLoad = function(succ:Boolean):Void  {
        if (succ) {
                for (var element:XMLNode = this.firstChild.firstChild; element != null;
element=element.nextSibling) {
                        if (element.nodeName == "vlastnost") {
                                var pom:Array = new Array();
                                pom [0] = Number(element.attributes.hodnota_parametru1);
                                pom [1] = Number(element.attributes.hodnota_parametru2);
                                příklad1.push(pom);
                        }
                        if (element.nodeName == " jina_vlastnost") {
                                příklad2 = Number(element.nodeName);
                        }
                }
                play();                          // funkce načetla požadované hodnoty
        } else {
                trace("XML data nenačtena");     // soubor nebyl pod předanou cestou nalezen
        }
};
data_xml.load(xmlSource);                        // volání parsovací funkce
```

3.2.3 Funkce odesílající výsledek řešení úlohy na server

Pro zpracování odpovědí jsou data odesílána dvěma formami.

sendAndLoad

Metoda, která odešle objekt s uloženými daty na server skriptu v souboru „test_rpc.php“. V těchto datech je uloženo číslo otázky, typ odpovědi (ve všech případech mnou vytvořených úloh se jedná o „flashtext“), odpověď jako zakódovaná sekvence znaků podle Base64 a data pro rekonstrukci úlohy (také kódována).

getURL

Tato metoda posílá příkaz javascryptové funkci „flashZodpovezeno()“ a předává informaci o čísle otázky, stavu odpovědi a data pro rekonstrukci úlohy. Slouží k aktualizaci zobrazení na straně uživatele ve webovém prohlížeči.

```
function odeslat(hodnoceni:String) {
	var scoreVars = new LoadVars();
	var saveResult = new LoadVars();
	scoreVars.otazka = _root.otazka;
	scoreVars.typ = "flashtext";
	scoreVars.odpoved = Base64.Encode(hodnoceni);
	scoreVars.flashdata = Base64.Encode(input.text);
	saveResult.onLoad = function(success) {
		if (success) {
			if (vysledek) {
				// zobrazení informace o úspěšném odeslání
			}
		}
	};
	scoreVars.sendAndLoad("test_rpc.php",saveResult,"POST");
	getURL("javascript:flashZodpovezeno(" + _root.otazka.toString() + "," + answerStatus.toString()
+ ",'" + scoreVars.flashdata + "');");
}
```

3.2.4 Test překrývání objektů – překrývání karet

Úvod

Při řešení úlohy „kamiony“ a „karty“ bylo zapotřebí řešit přesouvání objektů na předem definovaná místa. Aby byly tyto funkce pro soutěžícího uživatelsky příjemné, věnoval jsem této problematice více prostoru.

Teorie

Pro testování kolizí se nabízí využívání funkce AS2.0 hitTest()

```
public hitTest(target:Object):Boolean
```

Která informuje o kolizi objektů při jejich částečném překrytí.

Kolize může nastat pouze při manuálním přesunu objektu funkcí

```
startDrag(target:Object) : Void
```

Proto mohu testovat kolizi pouze v případě, kdy se pohybuje myší, čímž omezím náročnost aplikace.

Pro tuto situaci lze použít listener zvaný: Mouse.onMouseMove event.

Obrázek 4: Over při překrývání středu cíle

Výsledek

Na základě poznatků viz výše, jsem se rozhodl zjišťovat kolizi pouze při překrytí kartou střed cílové pozice, který je umístěn ve středu cíle jako samostatný movieClip, a mění tedy svou polohu v závislosti na změnách velikosti rodičovské cílové pozice. Pro zjištění, se kterým cílem vzniká kolize s kartou, testuji kolizi pro každý cíl zvlášť, viz následující kód.

```
this.onMouseMove = function() {
        if (pohybKarty) { // testuji pouze při přetahování karty
                var over:Boolean = false;
                for (var i:Number = 0; i < cile.length; i++) {
                        if (kartaPohyb.hitTest(cil.stred) && !cil.obsazen) {     //pokud se dotýká
                                                                                 neobsazeného cíle
                                over = true;
                                // funkce pro definování cíle, který je překrýván kartou
                                // funkce pro zvýraznění cíle
                        }
                        else {
                                // funkce pro odvýraznění cíle
                        }
                }
                if (!over) {          // pokud se karta nedotýká žádného cíle
                                // funkce pro odstranění informací o překrývaných cílech
                }
        }
};
```

Obrázek 5: eliminace problematiky kolize více objektů

3.3 Jednotlivé úlohy

3.3.1 Hlavolam

Obrázek 6: úloha Hlavolam

Zadání pro soutěžící

Úloha, kde je cílem najít v matici 5x5 políček správnou cestu tak, aby se skrze všechna políčka prošlo pouze jednou. Matice může obsahovat překážky. Pohyb vybraným směrem bude pokračovat do momentu, kdy narazí na překážku, okraj matice, nebo na políčko, kterým již bylo procházeno. Počátek si podle zadání může soutěžící určit sám, nebo je nastaven pevně daný.

Cíle

Cílem bylo vytvořit úlohu s graficky přívětivým prostředím, aby soutěžící neměl problém poznat ovládací prvky a intuitivně se jim přizpůsobil.

Motivovat k přemýšlení nad řešením, znevýhodnit metodu řešení pokus/omyl.

Výsledek

Nastavení úlohy

Struktura souboru a příklad nastavení úlohy xmldata.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<hlavolam_jednim_tahem>
        <rychlost_pohybu hodnota="0.2"/>
        <pocatek x="3" y="4"/>
        <prekazka x="3" y="1"/>
        <prekazka x="4" y="4"/>
</hlavolam_jednim_tahem>
```

Grafické zpracování

Z důvodu různorodosti mezi úlohami jak z pohledu interaktivity, tak z pohledu grafického, jsem pro tuto úlohu zvolil manuální grafický styl. To znamená, že prvky v úloze jsou kresleny jinak než elektronicky, a to za pomocí tužky na papír a následně oskenovány a až následně zpracovány v programu Photoshop CS6, kde jsou tyto grafické prvky mírně dobarveny.

Jako vektorová grafika je zde použito pouze podbarvení políček a blikajících prvků.

Obrázek 7: Situace během řešení úlohy Hlavolam

Funkcionalita úlohy

Pro znázornění cesty tahu jsou nakresleny drobné tečky ve směru tahu a pro konce tahu, tedy tam kde narazí na překážku, je zobrazen jeden větší bod. Z toho lze vizuálně rekonstruovat, kudy byl tah veden od začátku až do konce. Počáteční a poslední vyplněné políčko jsou znázorněny zeleně, překážky jsou zobrazeny červeně.

Vždy po dokončení tahu se zobrazí možnosti dalších tahů v podobě šipek, které se po najetí myši zvýrazní.

Úloha po každém tahu odesílá informaci o stavu výsledku na server. V momentě, kdy jsou všechna políčka vybarvena, je na server odeslána informace o úspěšném řešení a soutěžící je informován o správném řešení a že úloha byla uložena. Pokud nejsou všechna políčka vyplněna, je na server odeslána informace o špatném řešení.

Soutěžící může po prvním tahu vrátit hlavolam do původního stavu pomocí tlačítka „znovu“.

Závěr

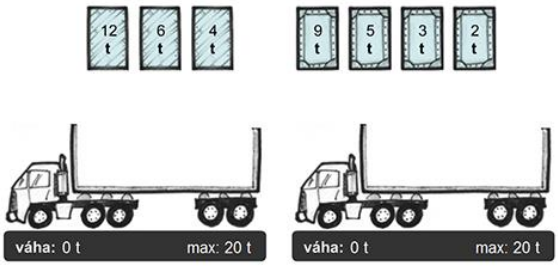
Použitím prodlevy mezi postupným vybarvováním tahů je dosaženo motivace, aby soutěžící dosáhl správného řešení spíše přemýšlením než zkoušením, protože jinak bude zbytečně ztrácet čas. Tato prodleva lze nastavovat v souboru xmldata.xml. Také lze vymýšlet varianty rozestavění překážek tak, aby se docílilo různé obtížnosti úlohy.

Úloha jak funkčností, tak od pohledu působí jako klasická logická hra.

3.3.2 Kamiony

Zadání pro soutěžící

Jsou dány 2 nákladní vozidla a každé má nastavenu určitou maximální nosnost. Zároveň je připraven nastavený počet boxů o různé váze, které mohou ale nemusí být všechny na nákladní vozidla naloženy.



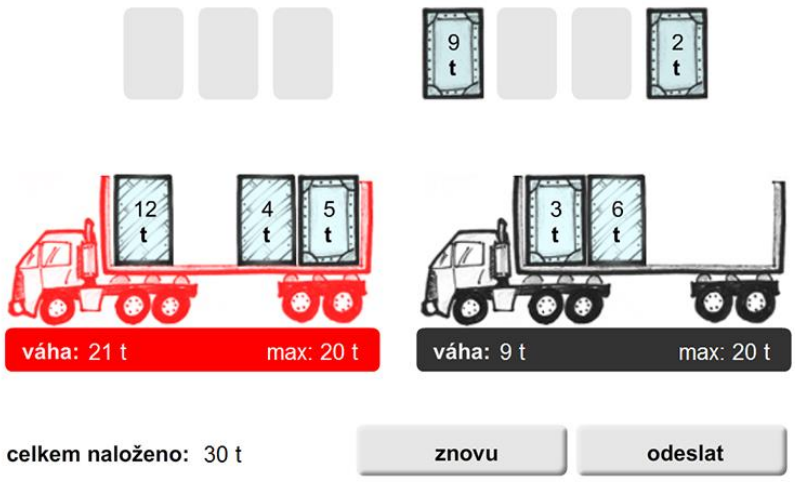
Obrázek 8: úloha Kamiony

Cílem soutěžícího je vytvořit takové 2 skupiny nákladu, které nejvýhodněji využijí maximální nosnost nákladních vozidel.

Teorie

Jelikož se zde jedná o přetahování objektů a jejich následné upouštění do předem daných cílových pozic, vycházel jsem ze znalostí viz kapitola 3.2.4 - Test překrývání objektů – překrývání karet.

Výsledek



Obrázek 9: úloha Kamiony - scénář přetížení kamionu

Chování úlohy

Po spuštění se načtou data nastavení ze souboru xmldata.xml. Obrázky jsou již implementovány v grafice, k jejich načítání nedochází. Boxy lze libovolně přetahovat na cílové pozice. Jejich upuštěním mimo cílovou pozici dojde k jejich automatickému návratu na původní souřadnice.

Po kliknutí na tlačítko odeslat dojde k výpočtu váhy naložených boxů a tato hodnota je odeslána na server, kde je porovnána se správným výsledkem. Soutěžící je informován o odeslání výsledku na server a má možnost pomocí tlačítka znovu boxy umístit do výchozích poloh a úlohu opakovat k jinému výsledku. Kliknutím na tlačítko znovu je serveru odeslána informace o zrušení řešení.

Pokud se soutěžící na úlohu vrátí, úloha si z dat uložených na serveru zjistí, zda již byla řešena a případně zobrazí odeslaný výsledek a možnost kliknout na tlačítko znovu.

Nosnost nákladních automobilů může být rozdílná, záleží na nastavení v souboru XML.

Boxy mohou být umístěny kdekoliv na korbě, řešení není jednoznačné.

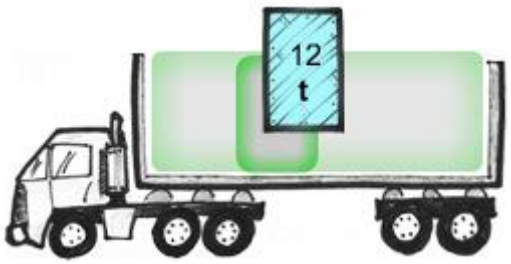
Grafické prvky

Veškerá grafika použitá v této úloze byla vytvořena pomocí tradiční kresby tužkou a perem a následně rastrově upravena a vyčištěna.

V prostředí FLASH byly vytvořeny jen podbarvující plochy a funkce upravující barvu objektů celkově.

Boxy

V nastavení je počet boxů neomezený, je jen zapotřebí počítat s tím, že se rovnají do řady vedle sebe zarovnané ke středu. Vzhledem k možnosti změny šířky aplikace jsem proto jejich počet neomezoval. V nastavení lze vybrat dva vizuálně rozdílné modely boxů. Po jejich uchopení se zvýrazní prostor na korbách nákladních aut, kam lze bedny přetáhnout. Po zvýraznění volné cílové pozice lze box upustit a ten se automaticky zarovná na cílovou pozici.



Obrázek 10: zvýraznění cílové pozice

Box je v úloze instancí nesoucí si unikátní identifikátor a obsahuje proměnné s původní polohou a momentálním stavem, tedy kde se právě nachází.

Nastavení

```
<?xml version="1.0" encoding="utf-8"?>
<kamiony>
	<nosnost kamion1="20" kamion2="20"/>
	<bedna hmotnost="12" typ="1"/>
	<bedna hmotnost="6" typ="1"/>
	<bedna hmotnost="4" typ="1"/>
	<mezera/>
	<bedna hmotnost="9" typ="2"/>
	<bedna hmotnost="5" typ="2"/>
	<bedna hmotnost="3" typ="2"/>
	<bedna hmotnost="2" typ="2"/>
</kamiony>
```

Element < nosnost >

- Udává maximální zatížení jednotlivých nákladních automobilů v tunách.

Element < bedna >

- Určuje váhu jednotlivých beden v tunách a jejich vizuální styl.
  V aplikaci jsou připraveny dvě grafické verze boxů.
- Počet elementů < bedna > je omezen pouze prostorem vyhrazeným pro aplikaci
- Bedny jsou řazeny posloupně podle pořadí v nastavení v souboru xmldata.xml.

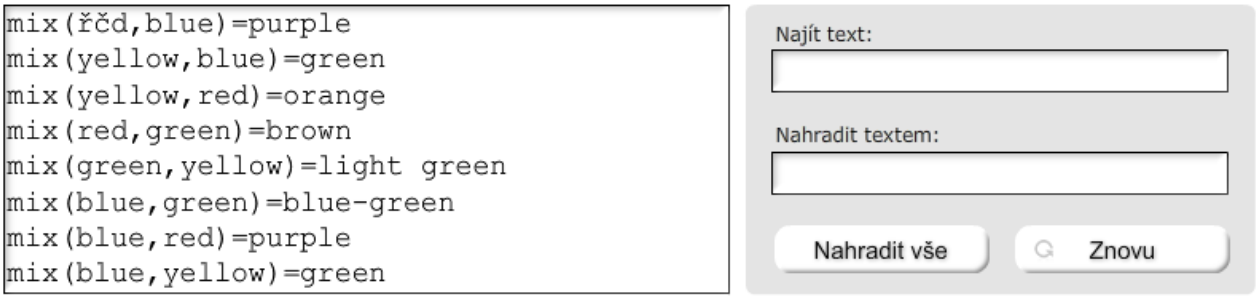
Element < mezera >

- Je prázdný nepovinný element, který nastavuje mezeru mezi ostatními bednami.

Závěr

Podařilo se mi vytvořit úlohu tak, že její ovládání je velmi intuitivní bez doplňujících popisků. Boxy jsou na korbu nákladních automobilů přichytávány až po kolizi se středem cílové pozice. Nejednoznačnost řešení vytváří dojem nejistoty, zda je vymyšlené řešení správné. Úloha může být pro soutěžícího velmi obtížná, protože je zapotřebí vytvářet úvahy o jiných možnostech naplnění nákladních automobilů.

3.3.3 Nahrazování



Obrázek 11: úloha Nahrazování

Zadání pro soutěžící

Soutěžící musí v několika krocích nahradit text, který je mu zobrazen na vzorový text, který je mu zobrazen externě v zadání úlohy. Soutěžící přitom používá klasické funkce nahrazování podobné nástroji „Nahradit“, jaký se nalézá v běžných textových editorech.

Cíle

Cílem je připravit úlohu, která postupně modifikuje původní texty na texty shodné se zadáním. Úloha je vyhodnocena jako správně, pokud jsou všechny upravované texty shodné s texty požadovanými. Dále aplikovat možnost zobrazení pro opětovný návrat na řešení úlohy ve webové aplikaci iBobr.

Východiska

Při vytváření této úlohy jsem se inspiroval společnými prvky nástroje nahradit v textových editorech TexPad, Poznámkový blok a Microsoft Word.

AS2.0 nemá standartní funkce pro procházení a nahrazování textu, ale pro práci s řetězci se nabízí funkce pro práci s polem: split() a join() a jejich následné převedení zpět do řetězce.

Výsledek

Funkce nahrazování textu

Rozšíření třídy String o funkci replace()

```
String.prototype.replace = function(searchStr, replaceStr):String  {
        return this.split(searchStr).join(replaceStr);
};
```

Chování úlohy

Úloha si po spuštění vyžádá data ze souboru xmldata.xml, kde je uložen jak původní text, který se bude měnit, tak text, se kterým se musí po úpravě shodovat.

Po každé úpravě je serveru odesláno vyhodnocení z porovnání řetězců, které je vyhodnoceno správné až v momentě, kdy se text shoduje se zadáním. V momentě, kdy je dosaženo shody textů, je serveru odesláno správné řešení a tato informace o správnosti je sdělena uživateli. Tlačítko pro další úpravy je skryto. Pokud jednou bylo dosaženo správného výsledku, nelze již s úlohou pracovat.

Při opětovném spuštění již vyřešené úlohy tato úloha soutěžícího informuje, že byla správně vyřešena a řešení odesláno na server. Pokud nebyla řešena správně, budou načtena původní data.

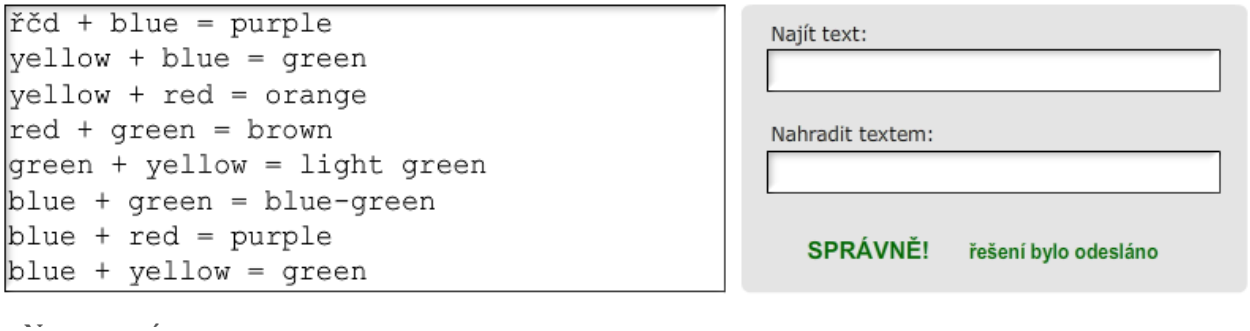
GUI

Grafické prvky úlohy jsou tvořeny čistě vektorovými tvary vytvořenými kreslícími nástroji vývojového prostředí Adobe FLASH.

Ošetření textového vstupu

Jelikož applet Flash player není ohrožen standartními útoky typu injection či jinými, je možno zadávat a nahrazovat zcela všechny znaky kódování UTF-8.

Omezení jsem vytvořil jen v délce zadávaného řetězce a to na 20 znaků.



Obrázek 12: správně vyřešena úloha Nahrazování

```
<?xml version="1.0" encoding="utf-8"?>
<nahrazovani>
        <radek_zadani>mix(řčd,blue)=purple</radek_zadani>
        <radek_zadani>mix(yellow,blue)=green</radek_zadani>
        <radek_zadani>mix(yellow,red)=orange</radek_zadani>
        <radek_zadani>mix(red,green)=brown</radek_zadani>
        <radek_zadani>mix(green,yellow)=light green</radek_zadani>
        <radek_zadani>mix(blue,green)=blue-green</radek_zadani>
        <radek_zadani>mix(blue,red)=purple</radek_zadani>
        <radek_zadani>mix(blue,yellow)=green</radek_zadani>

        <radek_spravne>řčd + blue = purple</radek_spravne>
        <radek_spravne>yellow + blue = green</radek_spravne>
        <radek_spravne>yellow + red = orange</radek_spravne>
        <radek_spravne>red + green = brown</radek_spravne>
        <radek_spravne>green + yellow = light green</radek_spravne>
        <radek_spravne>blue + green = blue-green</radek_spravne>
        <radek_spravne>blue + red = purple</radek_spravne>
        <radek_spravne>blue + yellow = green</radek_spravne>
</nahrazovani>
```

Element < radek_zadani >

Udává textový obsah jednotlivých řádků textu určeného k úpravám pomocí nahrazovací funkce.

Element < radek_spravne >

Udává výsledný řetězec znaků, se kterým se bude pozměněný text porovnávat.

Důležité

Je zapotřebí dodržet stejný počet řádků zadání s počtem řádků k porovnání, jinak nelze dosáhnout úspěšného řešení.

Počet řádků není omezen.

Závěr

Vytvořená úloha graficky i funkcionalitou představuje simulaci nástroje „Nahradit“ běžných textových editorů.

Pro následnou implementaci funkce k rekonstrukci dat jsem volil částečnou změnu návrhu inicializace proměnných po načtení dat ze souboru XML.

Za použití rozšířené nahrazovací funkce pro třídu String jsem dosáhl efektivních a snadných metod pro náhradu konkrétních textů.

3.3.4 Plotr

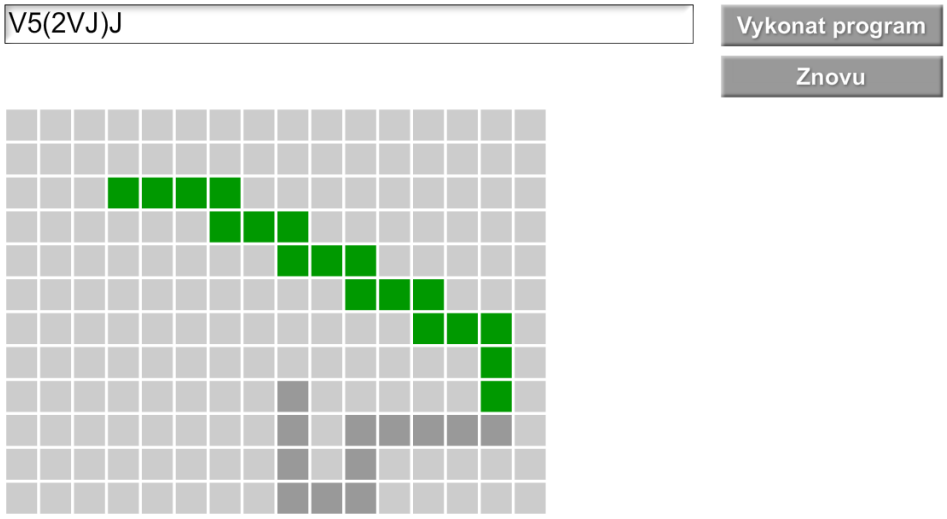
Zadání pro soutěžící

Soutěžící vytvoří kód, který bude sdělovat informaci o pohybu frézovacího stroje. Soutěžící použije znaky reprezentující světové strany a číslice, které sdělují kolikrát se následující směr, případně sekvence kódu uvnitř závorek bude vykonávat.

Cíle

Vytvořit aplikaci umožňující zadání viz výše, umožnit v jejím nastavení určení počtu políček a podle připomínek z testování do aplikace doplnit možnost zpětné rekonstrukce řešení.

Výsledky



Obrázek 13: úloha Plotr

Chování úlohy

Poté, co úloha načte data s nastavením, vygeneruje matici o rozměrech podle zadání a označí zeleně počáteční bod a šedivě požadovanou cestu, na kterou bude soutěžícím vytvářen kód. Pokud je v nastavení vyplněna nápověda, což je začátek požadovaného kódu, bude tento kód zobrazen v editovatelném textovém poli, úloha vypočítá cestu a vyznačí ji v matici políček zelenou barvou.

Pokud bude zadaný kód vyplněn tak, že vykreslená cesta vystoupí z požadované cesty, budou tato políčka označena odlišnou barvou.

Do textového políčka mohou být vyplněny pouze znaky představující světové směry (nezáleží, zda malými či velkými písmeny – písmena jsou automaticky přepisovány na velké znaky) a dále číslicemi, závorkami a mezerami. Mezery nijak neovlivňují funkcionalitu úlohy, jsou aplikací ignorovány. Jsou povoleny jen proto, aby mohl soutěžící snáze odlišovat cykly ve svém kódu, bude-li to v jeho zájmu.

Soutěžící vypíše takový kód, který naznačí cestu pro plotr a ten následně po kliknutí na tlačítko „Vykonat program“ cestu vykreslí a porovná se zadáním.

Kód je možno sestavit ze samotných směrů. Dále navíc umožňuje využití cyklů a to tak, že před směr je uvedena číslice a pohyb pro daný směr bude vykonán tolikrát, kolik číslice udává. Takto je možno vkládat i číslice před závorku, ve které je více znaků či číslic a celý tento blok je vykonán opakovaně viz hodnota zadaného čísla. Nelze vytvářet vnořené cykly, na což bude soutěžící při jejich vytvoření upozorněn a kód je následně vyhodnocen jako chybný.

Pokud je vytvořen kód, který obsahuje syntaktickou chybu, bude soutěžící upozorněn, o jakou chybu se jedná a kód bude vyhodnocen jako chybný.

Pokud bude kód vyhodnocen jako chybný, nebudou políčka aplikací vykreslena a na server je odeslána informace o chybném vyhodnocení.

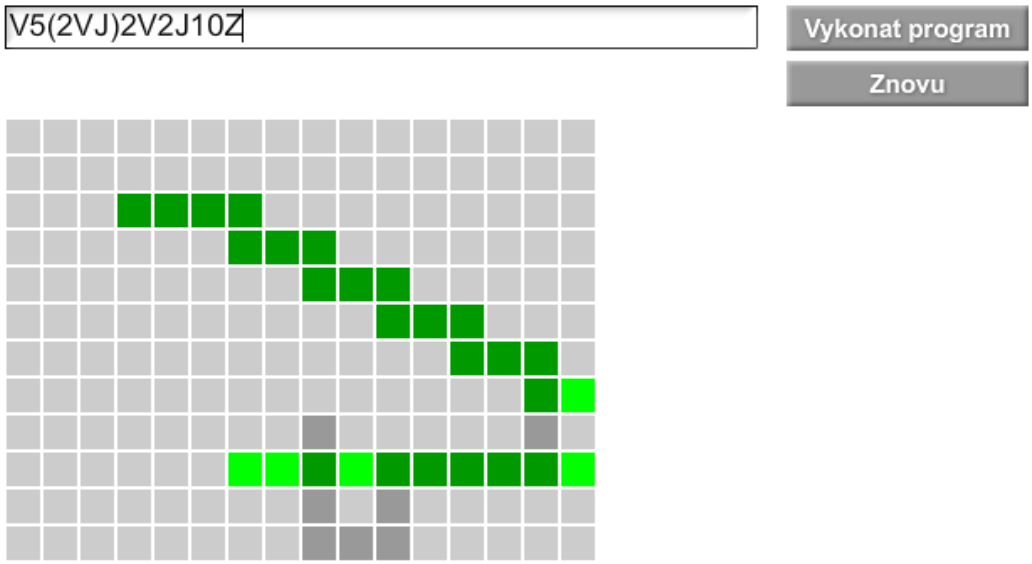
V případě, že je vytvořen bezchybný kód, pomocí kterého jsou vybarvena políčka podle zadání, je úloha vyhodnocena jako úspěšně splněna a výsledek je odeslán na server.

Aby byl soutěžící motivován využít zápis kódu s cykly, je v nastavení umožněno omezení počtu znaků kódu, protože s využitím cyklů je kód kratší.

Úloha umožňuje přijímat kód, který opouští hranice matice a navrací se v jiném místě, čímž simuluje chování plotru.

Rekonstrukce dat

Nežli soutěžící opustí úlohu, je na serveru uložen řetězec kódu, který byl naposledy testován tlačítkem „Vykonat program“. Tento kód je při dalším otevření úlohy načten místo nápovědy a automaticky vykonán. Soutěžící může navázat na své poslední řešení a může kód upravit, či dokončit.



Obrázek 14: úloha s použitým kódem, který opouští hranice matice a demonstruje rozdílné barvení nesprávně naprogramované cesty plotru.

GUI

Jelikož je možné nastavovat neomezené množství políček v matici, umožnil jsem v nastavení úpravu jejich velikosti. Lze proto vytvořit proporcionální rozložení matice vzhledem k počtu použitých políček.

Nastavení

```
<?xml version="1.0" encoding="utf-8"?>
<plotr>
        <zacatek y="2" x="3"/>
        <policko velikost="20"/>
```

```
        <input max="25" napoveda="V5(2VJ)J"/>
        <zadani vykreslovat="ano"/>
        <radek>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</radek>
        <radek>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</radek>
        <radek>0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0</radek>
        <radek>0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0</radek>
        <radek>0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0</radek>
        <radek>0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0</radek>
        <radek>0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0</radek>
        <radek>0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0</radek>
        <radek>0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0</radek>
        <radek>0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0</radek>
        <radek>0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0</radek>
        <radek>0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0</radek>
    </plotr>
```

V elementu *<radek>* představuje číslice 1 zadání pro výslednou vykreslenou cestu.

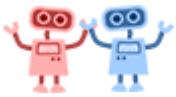
Závěr

Úloha umožňuje podle nastavení zobrazit libovolný počet políček.

Při vymýšlení úlohy vyvstala otázka, jak motivovat soutěžícího k použití cyklů v závorkách. Jelikož se tímto zápisem délka kódu značně zkracuje, umožnil jsem proto v úloze nastavení délky vkládaného řetězce, čímž nastane situace, kdy student musí takový zkrácený zápis použít, jinak by úloha nemohla být úspěšně vyřešena.

Tuto úlohu jsem jako první zpětně doplňoval o rekonstrukci dat při opětovném načtení a díky vhodnému objektovému návrhu úlohy mohly být funkce dopsány bez zbytečných úprav původního kódu.

3.3.5 Roboti



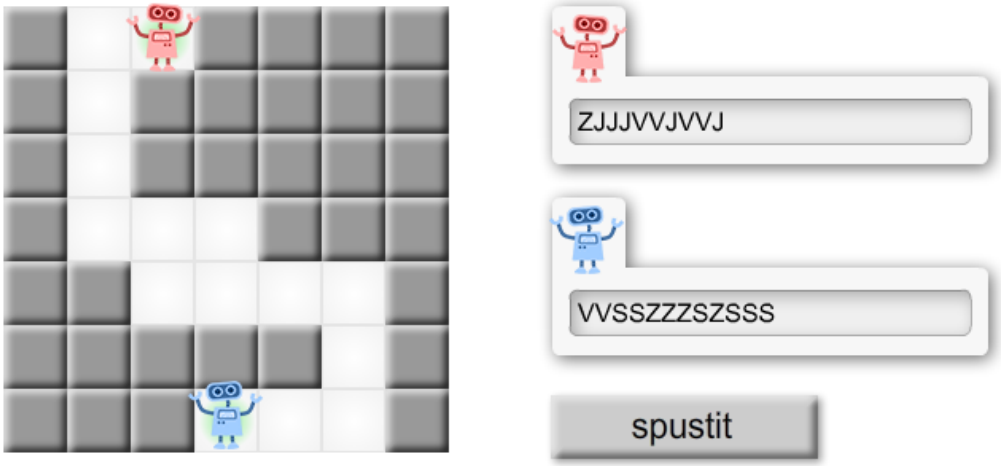
Zadání pro soutěžící

V matici představující bludiště se nalézají dva roboti. Soutěžící připraví pro každého z nich sekvenci znaků vyjadřujících postupné kroky určitým směrem tak, aby si roboti prohodili své pozice, aniž by při cestě bludištěm nenastala kolize s překážkami, nebo mezi roboty samotnými.

Cíle

Umožnit zadání viz výše, připravit úlohu graficky tak, aby její prostředí bylo příjemné a ovládání bylo intuitivní. Dále umožnit maximální volnost při vytváření zadání v souboru s nastavením úlohy.

Výsledky



Obrázek 15: počáteční stav úlohy Roboti

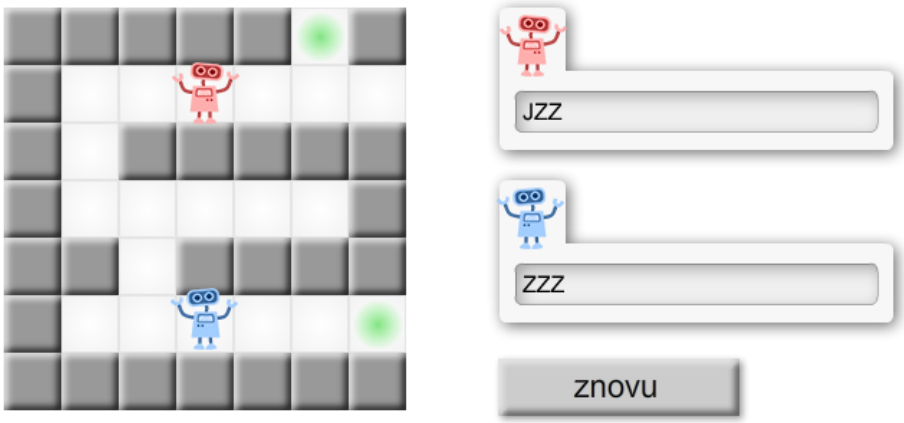
Chování úlohy

Úloha si po načtení souboru XML s nastavením vygeneruje pole s překážkami a umístí roboty na jejich startovní pozice. Každý robot má své vlastní textové pole pro vkládání kódu, které je omezeno pouze na znaky: S,J,V,Z; nezáleží, zda bude kód psán velkými, či malými písmeny, aplikace je přepisuje automaticky na velká písmena. Po spuštění dojde k animaci, kde se roboti pohybují podle sekvence zadaných znaků a po dokončení dojde k vyhodnocení úspěšnosti řešení a odeslání výsledku na server.

Výsledek je považován za úspěšný, když si roboti bez kolize prohodí svá počáteční místa. Při cestě nesmí vrazit jeden do druhého, ani do žádné překážky či opustit vyhrazené pole. Kód nemusí být jednoznačný, hodnotí se pouze výsledek, ne způsob cesty.

V případě kolize či nedokončení úspěšného prohození pozic robotů se soutěžícímu zobrazí tlačítko znovu, které po kliknutí navrátí roboty do jejich původních pozic.

Pokud je v nastavení dána nápověda, budou některé znaky od počátku zobrazeny v textových polích ovládajících roboty.



Obrázek 16: úloha roboti, další varianta

Rekonstrukce dat

Po následné úpravě je aplikace schopna získat informace o posledním řešení a řetězec, se kterým bylo dosaženo posledního výsledky před opuštěním úlohy je zobrazen v textových polích ovládajících roboty.

Kolize

V případě, že dojde ke kolizi některého z robotů, bude tento robot zvýrazněn blikáním, dokončí polovinu svého kroku a přestane nadále provádět svůj kód.

Obrázek 17: Kolize robota

Nastavení úlohy

```
<?xml version="1.0" encoding="utf-8"?>
<roboti>
	<napoveda_robot_1 retezec="ZJJJVVJVVJJZ"/>
	<napoveda_robot_2 retezec="VVSSZZZSZSSS"/>
	<rychlost_pohybu hodnota="0.25"/>
	<radek hodnota="1,0,0,1,1,1,1"/>
	<radek hodnota="1,0,1,1,1,1,1"/>
	<radek hodnota="1,0,1,1,1,1,1"/>
	<radek hodnota="1,0,0,0,1,1,1"/>
	<radek hodnota="1,1,0,0,0,0,1"/>
	<radek hodnota="1,1,1,1,1,0,1"/>
	<radek hodnota="1,1,1,0,0,0,1"/>
	<pocatek_robot1 x="2" y="0"/>
	<pocatek_robot2 x="3" y="6"/>
	<vystup1 retezec="SPRÁVNĚ. Odpověď byla odeslána"/>
	<vystup2 retezec="výborně!"/>
</roboti>
```

Element < napoveda_robot_1 > a < napoveda_robot_2 >

Definují, zda bude na počátku řešení úlohy soutěžícímu nabídnuta nápověda a pokud ano, jaké bude její znění.

Element < rychlost_pohybu >

Udává dobu ve vteřinách, za kterou roboti provedou jeden krok.

Element < radek >

Definuje řádek v matici a určuje, kde budou umístěny překážky. Hodnota 1 znamená překážku, hodnota 0 definuje volné políčko.

Element < pocatek_robot >

Určuje počáteční souřadnice robota. Počítá se od hodnoty 0 a od levého horního rohu.

Element < vystup >

Upravuje výsledné hlášení soutěžícímu v případě úspěšného řešení úlohy.

Závěr

V této úloze je možné nastavovat obtížnosti od velmi jednoduché po velmi těžkou. Vzhledem k možnosti úpravy rychlosti jednotlivých kroků lze celkové vyznění úlohy také zcela pozměnit.

Obrázky použité v této úloze jsem nakreslil bitmapově a nahradil jsem jimi předchozí obrázky reprezentující známé postavičky od Walta Disneye, ačkoli byly označeny jako Creative Commons, a pro nekomerční účely mohly být pravděpodobně použity, zvolil jsem jistější cestu.

Funkce zpětné rekonstrukce řešení byla doplněna.

3.3.6 Karty

Zadání pro soutěžícího

Soutěžícímu je představena logickoá posloupnost znaků, barev či objektů, podle které bude možno vydedukovat, jaké z nabízených karet umístit na určené cílové pozice.

Cíle

Pro úspěšné splnění musí soutěžící přesunout karty na správné pozice. Úloha může a nemusí o správnosti informovat v závislosti na nastavení.

Úloha bude načítat obrázkové pozadí a obrázky charakterizující jednotlivé karty.

Výsledek

Obrázek 18: Ukázka varianty úlohy Karty

Chování úlohy Karty

- Ve všech případech nastavení úlohy je společné, že po **každém uvolnění kartičky** dojde k vyhodnocení správnosti a výsledek je odeslán na server ke zpracování. Vyhodnocení o správném rozmístění kartiček je tedy odesíláno **pokaždé**, když jsou kartičky opravdu uloženy ve správném pořadí.
- Tlačítko „**znovu**“ ve všech případech odesílá serveru informaci o zrušení odpovědi a rozmístí kartičky do původního stavu, které zároveň i odemkne k opětovné manipulaci.
- Tlačítko „**odeslat**“ ve skutečnosti jen zamkne kartičky, aby nebyla omylem odeslána informace o špatném řešení následným pohybem kartiček, které o svém vyhodnocení a odesíláním dat na server soutěžícího neinformují.
- Pokud nebude úloha dokončena a soutěžící ji opustí, je serveru předána informace o stavu rozmístění karet, pomocí které ze úloha zrekonstruuje po jejím opětovném otevření. Soutěžící je informován, že úloha je v takovémto stavu uložena na serveru a má možnost navrátit karty do původních poloh a tím kartičky odemknout k další manipulaci.

Kartičky

- Každá karta má uloženou svou defaultní hloubku (překrývání) podle pořadí, ve kterém jsou nastaveny v souboru xmldata.xml. Tím lze dosáhnout snadného ovlivnění překrývání při návrhu obsahu úlohy. Tyto hloubky jsou vyšší než hloubka všech ostatních prvků v úloze. V momentě, kdy kartičku vybereme a přesouváme, je tato hloubka pozměněna na nejvyšší před ostatními kartičkami a po uložení na cílovou pozici, nebo po návratu na původní pozici se hloubka opět nastavuje na defaultní.
- Kartičku lze umístit do všech neobsazených cílových pozic. Tyto pozice se při překrytí jejich středu kartičkou zvýrazní zeleně a při uvolnění kartičky v tomto momentě se kartička uloží na označenou cílovou pozici. Při uvolnění kartičky nad obsazenou cílovou pozicí nebo kdekoliv jinde, se kartička automaticky navrátí na svou defaultní pozici dle nastavení v souboru xmldata.xml.

Nastavení zobrazení výsledku

Lze nastavit zobrazení výsledku (například „Správně“ nebo „Špatně“, podle vyplněného textu v nastavení). Aby úloha dávala smysl, jsou na tato rozdílná nastavení napsány různé scénáře.

Scénář 1

- <hotovo zobrazit="ano"/>; <vysledek zobrazit="ano"/>
- Soutěžící vyplní všechny pozice pro kartičky. Poté se mu zobrazí tlačítko odeslat, po jehož potvrzení se soutěžícímu zobrazí správnost jeho řešení. Kartičky se po kliknutí tlačítka odeslat zablokují z důvodu předejetí mylného pohybu kartičky (tedy automatického odesílání vyhodnocení po každém tahu). Zároveň se zobrazí tlačítko „znovu“, které by odeslalo informaci a zrušení odpovědi a umístilo kartičky do původního stavu před řešením úlohy.

Scénář 2

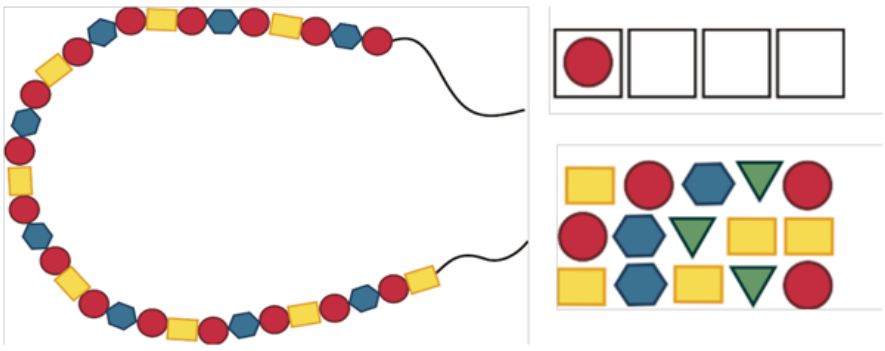
- <hotovo zobrazit="ano"/>; <vysledek zobrazit="ne"/>
- Stejné jako scénář 1 s rozdílem o zobrazení informace o správnosti řešení po kliknutí na tlačítko odeslat.

Scénář 3

- <hotovo zobrazit="ne"/>; <vysledek zobrazit="ano"/>
- Po vyplnění všech pozic pro kartičky je zobrazena informace o správnosti řešení. V případě špatného řešení lze kartičky přemisťovat. Pokud je výsledek vyhodnocen jako správný, zobrazí se zároveň s informaci o správnosti také tlačítko „změnit řešení“ a kartičky se uzamknou, aby se předešlo mylnému pohybu kartičky (tedy automatického odesílání vyhodnocení po každém tahu).

Scénář 4

- <hotovo zobrazit="ne"/>; <vysledek zobrazit="ne"/>
- Po umístění kartiček do všech pozic se zobrazí informace, že řešení bylo odesláno a tlačítko znovu. Kartičky jsou zablokovány v případě správného i špatného řešení.
- Pro správné vyhodnocení úlohy se předpokládá jediný postup řešení. Při návrhu obsahu je tedy zapotřebí zvážit a otestovat možnost nejednoznačnosti řešení.
- Úloha mí defaultní šířku 800 px a výšku 600 px. Tyto rozměry lze změnit v možnostech vystavení úlohy ve webové online aplikaci Bobřík informatiky.



Obrázek 19: Další možná varianta řešené úlohy

Nastavení úlohy

Úloha se nastavuje v souboru xmldata.xml, které má následující strukturu dat:

```
<?xml version="1.0" encoding="utf-8"?>
<karty>
	<pozadi x="35" y="8" width="361" height="481" soubor="obrazky/pozadi.jpg"/>
	<rozmer_karticky width="50" height="62"/>
	<znovu x="450" y="26" text="změnit řešení"/>
	<hotovo x="450" y="80" text="odeslat" zobrazit="ne"/>
	<vysledek x="450" y="140" text_spravne="Správně" text_spatne="Špatně" zobrazit="ano"/>

	<pozice x="7" y="417" kod="p" />
	<pozice x="75" y="417" kod="dp" />
	<pozice x="143" y="417" kod="dl" />
	<pozice x="279" y="417" kod="p" />

	<karta x="11" y="508" soubor="obrazky/sipka_p.png" kod="p" />
	<karta x="11" y="513" soubor="obrazky/sipka_p.png" kod="p" />
	<karta x="79" y="508" soubor="obrazky/sipka_l.png" kod="l" />
	<karta x="147" y="508" soubor="obrazky/sipka_dp.png" kod="dp" />
	<karta x="147" y="513" soubor="obrazky/sipka_dp.png" kod="dp" />
	<karta x="214" y="508" soubor="obrazky/sipka_dl.png" kod="dl" />
	<karta x="283" y="508" soubor="obrazky/sipka_np.png" kod="np" />
	<karta x="351" y="508" soubor="obrazky/sipka_nl.png" kod="nl" />
</karty>
```

Element < pozadí >

- udává cestu k souboru obrázku pozadí, čímž je veškerá grafika úlohy vyjma kartiček a prázdných pozic pro kartičky.
- Cesta je ovlivněna předávaným parametrem úloze serverem, který udává umístění souboru xmldata.xml vůči úloze v hiearchii souborů webové soutěže Bobřík informatiky. Ve výsledku je cesta k obrázkům dána relativní cestou od umístění souboru xmldata.xml.
- Parametry x a y nejsou povinné, defaultní hodnota je x=0, y=0.

- Parametry width a height nejsou povinné, defaultní hodnota je podle původní velikosti nahrávaného obrázku do pozadí.
- Lze přijímat obrázky ve všech standartních formátech pro web.
- Nahraný obrázek je vždy v pozadí vůči ostatním prvkům v úloze.

Element < rozmer_karticky >

- nastavuje velikost všech kartiček a jejich pozic (pozice jsou automaticky o 5px větší).

Elementy < znovu >, < hotovo > a < vysledek >

- udávají pozici, text a případně zobrazení tlačítek a textového pole pro výpis správnosti a stavu odeslání odpovědi na server.

*Elementy **< pozice >***

- nejsou omezeny počtem, udávají se souřadnice zleva odshora počínaje nulou. Parametr kód určuje, jakou kartičku musí obsahovat, aby byla úloha správně vyhodnocena.

*Elementy **< karta >***

- nejsou omezeny počtem, udávají se souřadnice zleva odshora počínaje nulou. Parametr kód určuje hodnotu, která musí být shodná s kódem pozice, aby byla úloha správně vyhodnocena.
- Parametr **soubor** udává cestu k obrázku reprezentujícího kartičku. Velikost tohoto obrázku může být libovolná. V případě, že je větší, bude obrázek zmenšen podle poměru kartičky a kratší stranou vycentrován na střed. V případě, že je obrázek menší, bude pouze vycentrován, aby nedošlo ke ztrátě kvality a jeho rozmazání.

Parametr kód

- v elementech **< pozice >** a **< karta >** nesmí obsahovat řetězec: „**empty**“, kterým jsou v databázi vyznačeny nevyplněné pozice pro karty a znaky: „**_**“ a „**-**“.

Závěr

Úloha umožňuje přetahování karet na pozice podle souřadnic v nastavení umožňuje načítat obrázková data a poskytuje několik scénářů zobrazení závislých podle nastavení v souboru XML.

Tato úloha byla vytvořena jako poslední, mohl jsem zde uplatnit poznatky z minulých úloh a vyvarovat se problémovým situacím z nasazených úloh v ostré soutěži. Již při návrhu jsem vytvořil model počítající s neomezenými počty prvků a s třídou načítající obrázky nezávisle na běhu programu. Důležité bylo již při návrhu počítat s implementací metod pro rekonstrukci dat.

Jelikož jsem analýzou a návrhem strávil více času, je její kód psán s jasným záměrem, a tudíž je snadno srozumitelný. Pokud nastanou požadavky a rozšíření funkcionalit, nebude se pravděpodobně jednat o složité úpravy ani pro programátora, který není srozuměn se zdrojovým kódem.

Předpokládám, že tato úloha bude mít v soutěži nejširší uplatnění, jelikož vzhledem k širokým možnostem při navrhování obsahu a jeho jednoduché realizaci v nastavení lze dosáhnout zcela rozdílných úloh.

4 Testování

Problematika určování cesty k souboru s nastavením

Na základě testování nasazení úloh do soutěže vznikla otázka, kam umisťovat soubor s nastavením XML a obrázky, pokud je úloha vyžaduje. Z prvotní analýzy a následného návrhu struktury jsem vycházel z předpokladu, že přidružené soubory budou umístěny ve stejném adresáři se souborem SWF. Jelikož se ale tyto soubory musejí nacházet jinde z důvodu dané hierarchie webové aplikace iBobr, je úloze předáván parametr „cesta“, který udává umístění k souborům v rámci webové aplikace, a následně se všechny cesty k obrázkovým souborům v souboru s nastavením XML mohou udávat pomocí relativní cesty.

Obrázkové soubory PNG

Při testování na straně metodiků vytvářejících zadání nastala situace, kdy některé obrázky ve formátu PNG nejsou aplikací akceptovány. Po podrobném průzkumu jsem došel k závěru, že se jednalo o obrázky PNG komprimované ve standardu PNG-8, nikoli běžným PNG-24. Tato hypotéza bohužel nemohla být potvrzena kvůli neznámému původu těchto obrázků a jejich nedochováním.

Zpětná rekonstrukce dat

Při běhu na ostré verzi soutěže často nastává situace, kdy si soutěžící vrátil zpět na již vyřešené úlohy, které vypadají, jako by řešeny ještě nebyly. Pro soutěžícího to bylo matoucí a nastalo několik případů, kdy byly již vyřešené úlohy řešeny znovu. Tomuto jsem zamezil zavedením funkcí pro zpětnou rekonstrukci dat, kde si soutěžící může prohlédnout, že úloha již byla vyplněna či odeslána, nebo v jakém zůstala stavu při jejím opuštění a v řešení poté pokračovat.

5 Závěr

Provedl jsem analýzu možností interaktivních úloh pro internetovou soutěž Bobřík informatiky a provedl průzkum již použitých interaktivních úloh a jejich uplatnění v soutěži v rozmezí let 2008 – 2013.

Vytvořil jsem šest kompletních a vzájemně odlišných úloh plně kompatibilních s online soutěží Bobřík informatiky. Vytvořené úlohy byly použity v ostré soutěži v ročníku 2012 a v ročníku 2013.

Grafické prvky úloh jsem připravil a nakreslil konkrétně pro dané úlohy. Nevztahovaly se na ně tedy žádná licenční ujednání.

Vytvořené interaktivní úlohy je možno upravovat pomocí externích dat ve struktuře XML. S těmito daty se snadno manipuluje a lze jimi úlohy zcela pozměnit, a to snadno a rychle.

Při testování komunikace úloh s webovou aplikací iBobr byly doplněny a nasazeny metody umožňující archivovat data potřebná ke zpětné rekonstrukci dat.

Jelikož se úlohy v soutěžích osvědčily a jsou nově doplněny o funkce zpětné rekonstrukce, lze je v nastavovaných obměnách používat i v následujících ročnících. První úlohy, které nebyly zcela navrhovány pro rozšíření o funkce zpětné rekonstrukce nebylo ve výsledku těžké pozměnit. Původní vývojový model ADDIE se při navrhování interaktivních úloh osvědčil a jsou na něm postaveny i všechny ostatní úlohy.

Software ani hardware škol, na kterých soutěže dosud probíhaly, neměly problémy se spuštěním úloh běžících za pomocí appletu Flash player, což bylo předpokládáno jako jedno z rizik ve vystavení těchto úloh.

Během zpracování této bakalářské práce probíhaly konzultace s realizačním týmem soutěže Bobřík informatiky a úlohy jsem postupně upravoval dle cenných rad a zkušeností ostatních členů, kteří mají v oblasti metodik a interaktivních úloh bohaté zkušenosti.

6 Literatura

1. **JANOŠÍK, Dušan.** Něco málo k technologiím WPF a Silverlight. *www.vyvojar.cz.* [Online] 13. 5 2007. http://www.vyvojar.cz/articles/473-neco-malo-k-technologiim-wpf-a-silverlight.aspx.

2. **PŘIBYL, Vladimír.** *Online soutěž v informatických znalostech pro žáky ZŠ a SŠ.* Č. Budějovice : Jihočeská univerzita v Českých Budějovicích, 2012. str. 62.

3. **PŘIBYL, Vladimír**. *Inteaktivní úlohy - dokumentace pro komunikaci mezi úlohami vytvořenými v programu Flash a aplikací iBobr.* [pdf] České Budějovice 2011.

4. **FERNANDO, Zeh.** MC Tween. *hosted.zeh.com.br.* [Online] 2006. http://hosted.zeh.com.br/mctween/.

5. **Adobe Creative Team.** *Adobe Flash CS4 Professional.* Brno : COMPUTER PRESS, 2010. 9788025123348.

7 Seznam příloh práce

CD s úlohami

- Úlohy jsou uloženy ve formátu FLA kompatibilním s aplikací Adobe FLASH CS6
- Počet kusů: 1