POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Iwona Arcimowicz

Nr albumu 7792

Język Logo w procesie zdalnego nauczania

Praca magisterska napisana
pod kierunkiem:

dr inż. Mariusz Trzaska

Streszczenie

Praca dotyczy problemu nauczania i uczenia się języka programowania Logo przy wykorzystaniu e-learningu. Przedstawiono w niej koncepcję zdalnego nauczania opartą o interpreter języka działający w przeglądarce internetowej oraz zestaw materiałów edukacyjnych. Do budowy interpretera wykorzystano technologię Javascript, a przy tworzeniu pozostałych materiałów posłużono się edukacyjnym system Moodle i elementami animacji flash.

Spis treści

1. Wstęp

Decyzja o temacie pracy została podjęta na podstawie własnych doświadczeń autorki i pewnego sentymentu jakim darzony jest przez nią język Logo. Obecnie język ten jest wykorzystywany do nauki podstaw algorytmiki i programowania w klasach gimnazjalnych. w polskim Internecie istnieje obecnie wiele materiałów edukacyjnych dotyczących podstaw programowania z wykorzystaniem języka Logo przygotowywanych głównie przez nauczycieli gimnazjów. Autorka nie spotkała się jednak z wykorzystaniem w tych materiałach działającego interpretera online tego języka w polskiej wersji językowej jaka jest używana obecnie w polskich szkołach. Wykonany projekt jest skierowany głównie do młodzieży zainteresowanej nauką podstaw języka Logo, ale może być także wykorzystany przez nauczycieli zajmującej się tym zagadnieniem.

Zasadniczą częścią projektu jest interpreter. Jego działanie na stronie www ma zagwarantować dostęp z dowolnego urządzenia posiadającego przeglądarkę internetową co powinno być pomocne dla uczniów. Obecnie używane w szkołach oprogramowanie jest oprogramowaniem komercyjnym co zmniejsza jego dostępność dla uczniów, ponadto jest to oprogramowanie wyłącznie dla systemu Windows. Pierwotnie interpreter miał być wykonany w formie apletu Java, jednak ze względu na obserwowane obecnie odejście od tej technologii wybrane zostało wykorzystanie JavaScript i Html 5.

Praca składa się z sześciu rozdziałów: wstępu, rozdziału dotyczącego teorii nauczania online, rozdziału poświęconego językowi Logo i jego historii, rozdziału zawierającego opis zrealizowanej propozycji nauczania online, rozdziału dotyczącego wykorzystanego oprogramowania oraz rozdziału poświęconego wykonanej implementacji.

2. Nauczanie programowania online

W rozdziale tym przedstawiono podstawowe elementy teorii e-learningu, informacje na temat istniejących kursów e-learningowych dotyczących programowania oraz istniejących interpreterów języka Logo działających na stronach www.

2.1. Teoria eLearningu

Wykorzystując technologie internetowe nauczyciele realizują swoje działania, świadomie lub nie, na podstawie istniejących teorii pedagogicznych, które w praktyce przenikają się. Każda teoria pedagogiczna bazuje na innych założeniach psychologicznych i filozoficznych. Podstawowe teorie to: behawioryzm, kognitywizm i konstruktywizm.

2.1.1. Behawioryzm

Behawioryzm to teoria, która określa proces uczenia się jako rezultat reakcji na bodźce. Powtarzające się bodźce powodują powstanie uwarunkowania, czyli wytworzenie się automatycznych reakcji. Efekty uczenia się są mierzone jako zmiany w zachowaniu [10]. Bodźcami mogą być wybrane metody i środki nauczania, a zmianą zachowania może być poprawa wyników testów na skutek wyćwiczenia pamięci.

Teoria ta zakłada, że wiedza jest zbiorem obiektywnych informacji zgromadzonych przez naukowców na skutek procesów badawczych. Pomija się w niej znaczenie rozumowania. Uczący się jest metaforycznie pustym naczyniem.

Przy zastosowaniu tej teorii technologia informacyjna jest wykorzystywana głównie jako metoda przekazywania informacji które mają być przyswojone oraz jako narzędzie przeprowadzania testów - w tym zwłaszcza testów wyboru.

2.1.2. Kognitywizm

Innym podejściem do teorii uczenia się jest kognitywizm - psychologia poznawcza. w tym przypadku uwaga zwrócona jest na procesy interakcji uczącego się ze środowiskiem, to jest procesy zapamiętywania informacji i przypominania informacji. Zgodnie z tą teorią powstał modułowy model pamięci złożony z rejestru sensorycznego, pamięci krótkotrwałej i pamięci długotrwałej (szczegółowy opis tego modelu można znaleźć w [10]).

Wykorzystanie technologii informacyjnych przy nauczaniu z wykorzystaniem kognitywizmu opiera się na założeniu, że należy usprawnić procesy przyswajania informacji. Zaowocowało to

projektami kursów mających określoną konstrukcję składającą się zazwyczaj z wprowadzenia, treści zasadniczych, konkluzji i oceny. Kursy takie nastawione są najczęściej na pracę samodzielną kursanta.

2.1.3. Konstruktywizm

Konstruktywizm to teoria zakładająca, że procesy uczenia się nie polegają na przepływie informacji, a na procesach budowania wiedzy opartego na doświadczeniu uczącego się. Wiedza więc nie jest odwzorowaniem rzeczywistości tylko rezultatem interakcji uczącego się ze środowiskiem, natomiast uczeń nie jest tylko odbiorcą ale i twórcą swojej wiedzy [9].

Pierwsze poważne założenia konstruktywizmu można znaleźć w pracach Alfreda Bineta, który wskazywał na znaczenie aktywności dziecka w konstruowaniu swojej wiedzy. Fundamentalne znaczenie dla ukształtowania się teorii konstruktywizmu miały badania Jeana Piageta oraz Lwa Wygotskiego. Jean Piaget „wywarł wielki wpływ na rozwój psychologii poznawczej w XX wieku" [9]. Badał on mechanizmy tworzenia się struktur poznawczych w umyśle człowieka, a nie to za pomocą jakich oddziaływań można te procesy przyspieszać.

Jednym z najbardziej znanych i najwybitniejszych współpracowników Piageta jest Seymour Papert. Seymour Papert (ur. 1 marca 1928 w Pretorii, założyciel laboratorium sztucznej inteligencji na MIT) jest twórcą konstrukcjonizmu - nurtu konstruktywizmu. Konstrukcjonizm kładzie nacisk na trzy aspekty rozwoju: mentalny (procesy myślowe), społeczny (współpraca między uczniami) oraz materialny (wytworzony produkt). Konstrukcjonizm głosi, że uczenie się (przez tworzenie nowych idei) jest skuteczne najbardziej wtedy gdy uczniowie są zaangażowani w budowanie „artefaktów". Takim artefaktem może być robot, poezja, przedmiot albo program komputerowy [9].

Seymour Papert jest także twórcą języka Logo, który miał być środowiskiem dydaktycznym umożliwiającym aktywne uczenie się ważnych umiejętności z różnych dziedzin, nie tylko informatyki. Obecnie język Logo jest wykorzystywany głównie jako środek do nauki podstaw programowania.

2.2. Istniejące kursy e-learningowe

Większość kursów e-learningowych dotyczących programowania ma podobny układ. Są to wykłady uzupełnione zadaniami do wykonania i czasami elementami pokazowymi. z reguły nie różnią się one niczym od innych kursów e-learningowych i w większości przypadków nie ma potrzeby by występowały zasadnicze różnice między tymi kursami. Trzeba zaznaczyć, że zazwyczaj odbiorcom tych kursów (którymi są najczęściej ludzie dorośli oraz młodzież) standardowe podejście w zupełności wystarcza.

Ciekawym projektem jest platforma YoungCoder [11] przeznaczona głównie do nauczania online języków c, c++, Pascal i Java. Poza gotową bazą wiedzy, możliwością przeprowadzania testów

online, edycji istniejących lekcji przez zarejestrowanych nauczycieli, organizowaniem konkursów itp. umożliwia sprawdzanie poprawności gotowych rozwiązań zadań (czyli programów). Niestety sprawdzenie poprawności nie spowoduje wyświetlenia się działania programu na żywo a tylko informacji czy jest poprawny. Platforma YoungCoder to ciekawa propozycja dla uczniów i nauczycieli, ma jednak jedną poważną wadę – jest płatna. Zasadniczo też dostęp do platformy jest zarezerwowany dla grup uczniowskich. Konto na platformie zakłada nauczyciel deklarując ilu uczniów będzie miał w grupie. Każde konto uczniowskie jest płatne, konto prowadzącego grupę nauczyciela bezpłatne. w innych przypadkach (np. próby utworzenia konta dla jednej osoby, lub dużej grupy) zasady dostępu (i cennik) ustala się indywidualnie.

Innym ciekawym projektem jest "Młodzieżowa akademia informatyczna" [12] (main.edu.pl) portal edukacyjny działający przy ogólnopolskiej Olimpiadzie Informatycznej stworzony przez jurorów olimpiady w 2005 roku.[1] Serwis zawiera aktualne informacje dotyczące konkursów informatycznych (głównie związanych z algorytmiką i programowaniem), kursy programowania w języku c++ i w języku Pascal oraz bardzo rozbudowaną bazę zadań ćwiczeniowych pochodzących z wielu konkursów oraz olimpiad informatycznych. Ponadto rozwiązania zadań można przetestować wysyłając je przez platformę i otrzymując informację o poprawności. Założenie konta w serwisie pozwala śledzić własne postępy i wysyłać rozwiązania.

Jeśli chodzi język Logo istnieje w polskim Internecie sporo stron prowadzonych głównie przez nauczycieli jako pomoc do własnych zajęć, ale nie mających charakteru regularnego kursu. Propagatorem wykorzystania języka Logo w szkołach jest Ośrodek Edukacji Informatycznej i Zastosowań Komputerów w Warszawie, który jest także dystrybutorem między innymi tego oprogramowania. Na stronach OEIIZK wydzielono specjalny serwis dotyczący Logo promujący Logomocję Imagine. Można tam obejrzeć przykłady pokazów multimedialnych tworzonych w Logomocji. Oprócz tego na stronach OEIIZK uruchomiono serwis eKlasa (oparty na Moodle) z kursami Logo dla szkół podstawowych i gimnazjów. Kursy te są dostępne dla chętnych uczniów w określonych odstępach czasu, wymagają zapisów i akceptacji prowadzących (obecnie prowadzącymi jest trzech nauczycieli). w kursach oprócz informacji teoretycznych czyli lekcji, wykładów, jako elementy interaktywne wykorzystano pokazy tworzone w Logomocji.

2.3. Ciekawe implementacje interpreterów Logo online

Poniżej przedstawiono niektóre ciekawsze według autorki realizacje interpreterów języka Logo działających na stronach WWW.

[1] informacje ze strony http://main.edu.pl/pl

2.3.1. RLogo

Autorem tej wersji Randall Embry, amerykański programista. RLogo (Rysunek 1) jest napisane w Javie i udostępnione na stronie domowej projektu w formie apletu, razem z niewielkim tutorialem i kilkoma przykładami procedur. Udostępnione są źródła na licencji GNU PL

Zgodnie z informacjami umieszczonymi na stronie w 2001 roku projekt został przetłumaczony na język arabski, a w 2004 na hebrajski. Po roku 2004 nie był rozwijany.

W rLogo zrealizowana mocno okrojona wersja języka Logo:

- podstawowe operatory arytmetyczne (+,-,*,/) i logiczne (>, <, =, oraz * jako and, + jako Or, ~ jako not),
- podstawowe komendy grafiki żółwia: **forward**,fd (distance) – naprzód, **back**,bk (distance) – wstecz, **right**,rt (angle) – prawo, **left**,lt (angle) – lewo, **write** "(message)" wypisywanie komunikatu, **home** – odpowiednik wróć, **hideturtle**,ht – schowajżółwia (sż), **showturtle**,st – pokaż żółwia (pż), **clearscreen**,cs – czyszczenie ekranu, **penup**,pu – podnieś (pod) pisak, **pendown**,pd – opuść (opu), **setpencolor**,setpc (red) (green) (blue) – zmiana koloru pisaka, **setbackcolor**,setbc (red) (green) (blue) – zmiana koloru tła
- instrukcje sterujące: repeat (powtórz), ifelse (jeśli), while (odpowiednik *dopóki* w polskiej wesji)
- słowa kluczowe umożliwiające zdefiniowanie własnej procedury przez użytkownika, zadeklarowanie zmiennej

Nie są zaimplementowane procedury operujące na słowach i listach, nie ma możliwości napisania procedury nie związanej z grafiką żółwia.

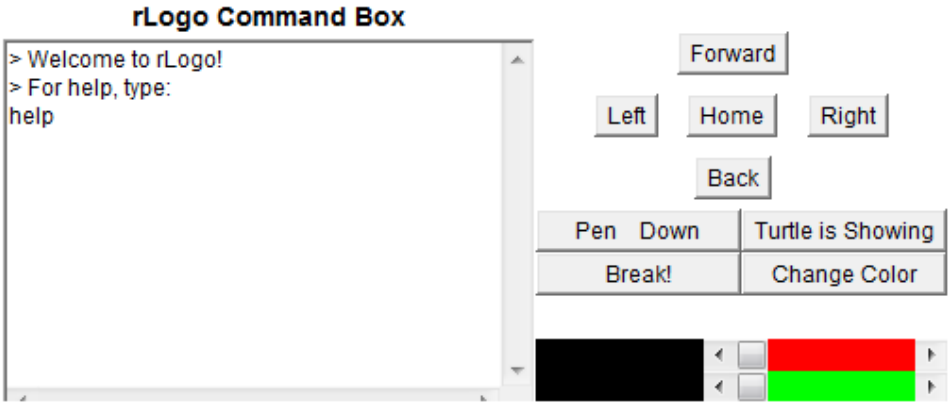
Biorąc pod uwagę samą grafikę żółwia w rLogo zaimplementowano podstawowe operacje, nie ma możliwości wypełniania kształtów (w polskiej wersji polecenie *zamaluj*).

Zaimplementowany w tym projekcie podzbiór języka Logo nie odbiega od innych jego realizacji poza jednym szczegółem – w przypadku definiowania procedury z parametrami przez użytkownika należy parametry umieścić po dodatkowym słowie kluczowym params w formie listy ograniczonej nawiasami kwadratowymi, a nie tak, jak jesteśmy przyzwyczajeni zaraz po nazwie procedury poprzedzone znakami dwukropka.

Poniżej (Listing 1) zaprezentowano przykład procedury rysującej prostokąt o bokach długości x i y działającej w interpreterze rLogo. Jak widać parametry są wypisane po słowie „params” w nawiasie kwadratowym co jest specyficzne dla tej wersji interpretera i nie spotykane w standardzie języka Logo.

Listing 1. Przykład procedury w rLogo.

```
to rectangle
params [ x y ]
repeat 2 [forward :x left 90 forward :y left 90]
end
```



Rysunek 1. Okno interpretera rLogo.

Strona domowa http://www.embry.com/rLogo/.

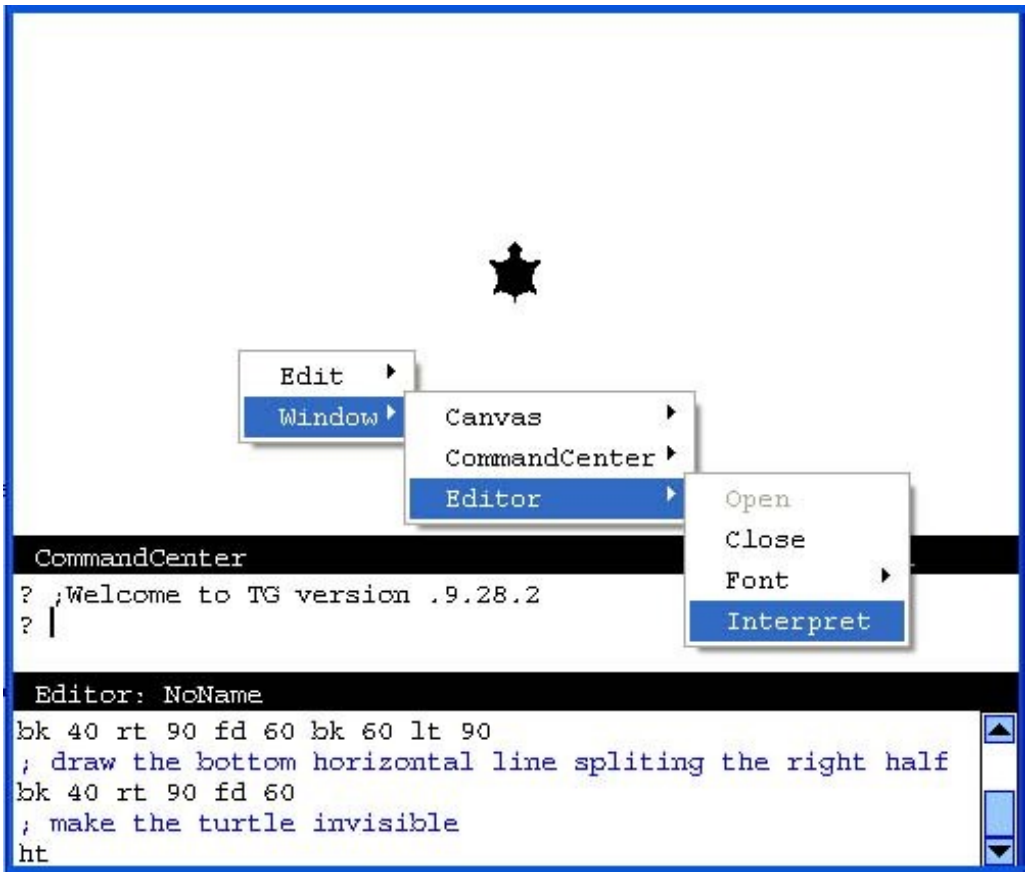
2.3.2. JLogo w projekcie umieszczonym na stronie BFOIT (Berkeley Foundation for Opportunities In Information Technology)

Na stronie BFOIT umieszczono pełny kurs „Wstęp do programowania" oparty o język Logo oraz aplet interpretera nazwany TG. Jest to bardzo rozbudowany interpreter, zawierający dodatkowe procedury ułatwiające początkującemu programiście tworzenie interakcji użytkownika z programem w języku Logo, o bardzo prostym na pierwszy rzut oka interfejsie złożonym z okna graficznego i okna komend. Interfejs zawiera menu kontekstowe umożliwiające dostęp do edytora tekstowego, zapisywania plików, otwierania plików, korzystania ze schowka i ustawień ekranu (np. czcionki).

TG (Rysunek 2) jest wykorzystywany w kursie podstaw programowania z użyciem języka Logo, oraz powiązanym kursie wstępu do programowania w Java. Kurs składa się z wielu, ciekawych projektów programistycznych uporządkowanych według wzrastającej trudności (od rysowania kolorowych domków do gry mastermind i kalkulatora).

TG napisany został w Javie. Część kodu źródłowego obejmująca podstawowe procedury grafiki żółwia jest udostępniona na stronie domowej w kursie wstępu do programowania w Java. Całość nie jest dostępna, nie ma także informacji o licencji. Chętni zainteresowani, zgodnie z informacją na stronie, mogą ubiegać się o udostępnienie u autora.

Projekt jest rozwijany i prowadzony do dnia dzisiejszego. Strona domowa http://www.bfoit.org/itp/



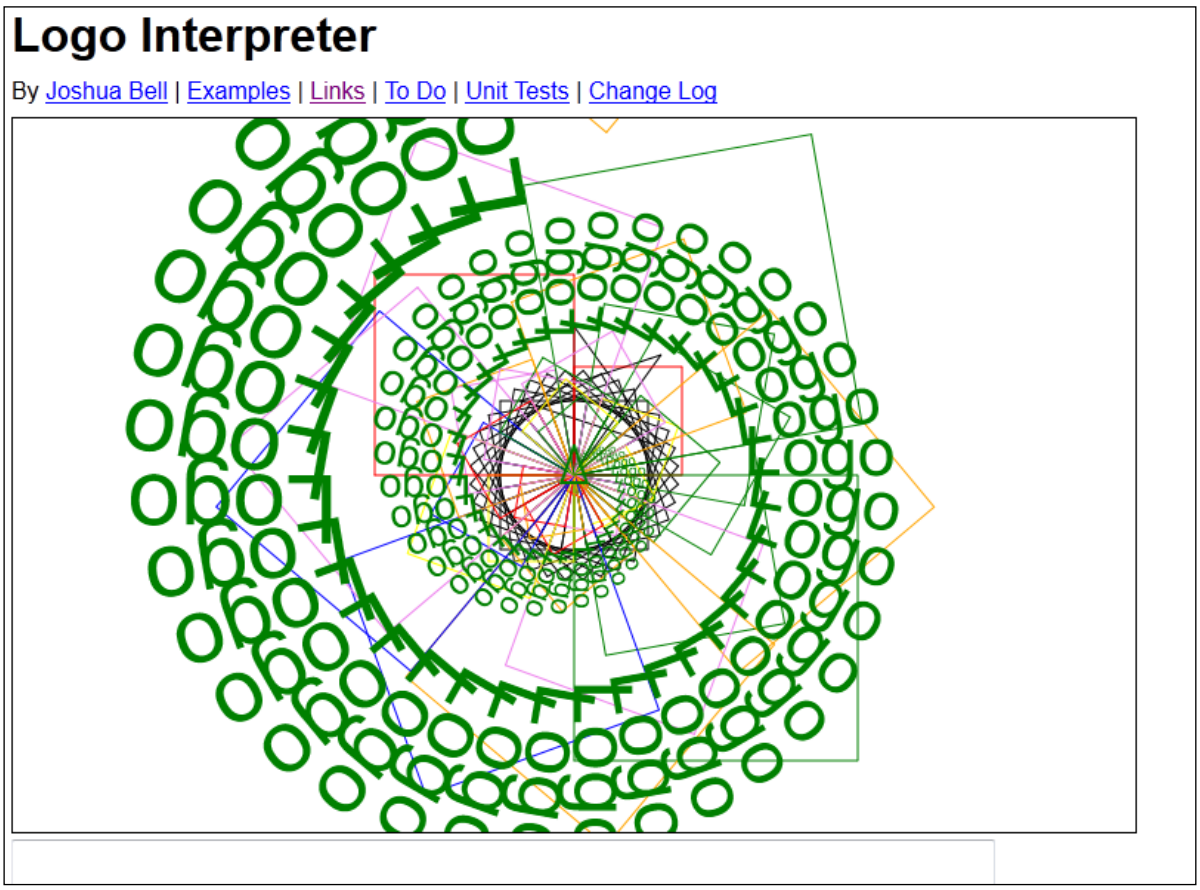
Rysunek 2. JLogo-TG widok interpretera z otwartym menu kontekstowym

2.3.3. Carolmen Logo interpreter

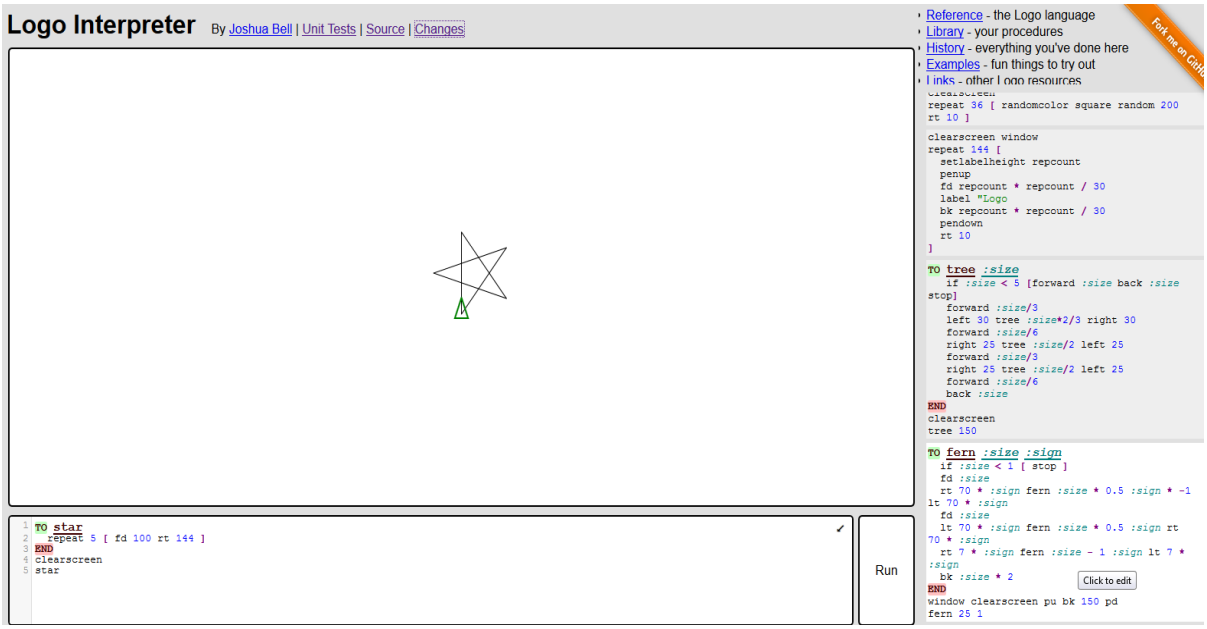
Interpreter Logo bazujący pod względem realizacji języka na opisie UCBLogo, napisany w javascript przez Joshuę Bella. Projekt obsługuje duży podzbiór języka Logo, to jest: większość komend grafiki żółwia, tworzenie procedur, wyrażenia i funkcje arytmetyczne, słowa i listy. Interfejs tej wersji jest bardzo prosty i składa się tylko z okna żółwia i edytora.

Projekt jest rozwijany od 2008 roku do chwili obecnej. w rejestrze zmian na stronie domowej umieszczone są aktualne informacje obejmujące także ostatnie dni. Na początku 2013 roku zmieniono interfejs strony głównej interpretera i usprawniono uruchamianie testowych procedur - kliknięcie na kod jednej z nich po prawej stronie okna przeglądarki natychmiast przenosi go do pola edytora, a przycisk „run" uruchamia wykonywanie kodu. Na rysunku 3 przedstawiono widok interpretera z uruchomionym przykładem z końca 2012 roku, na rysunku 4 widok obecny.

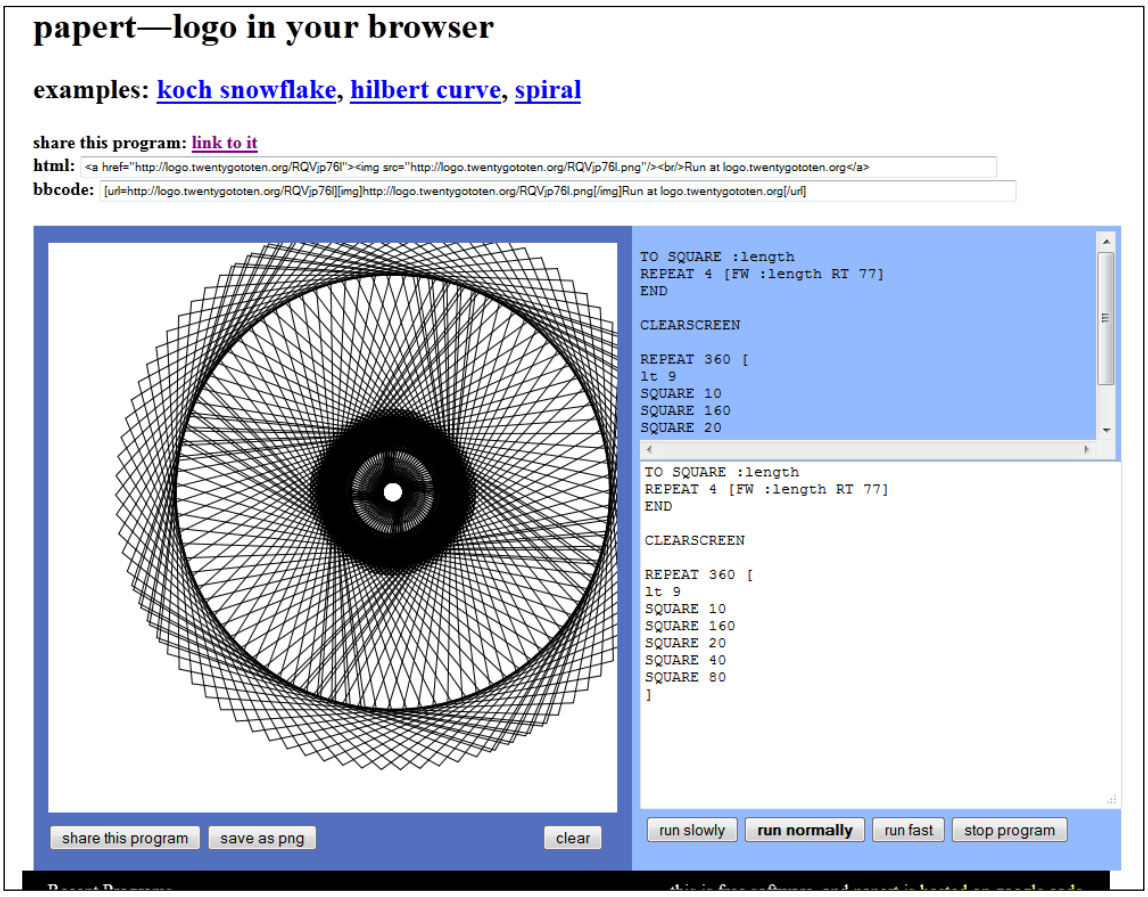
Strona domowa tego interpretera http://www.calormen.com/Logo/ nie posiada obudowy dla uczącego się użytkownika. Dostarczonych jest tylko kilka przykładów oraz linki do związanych z Logo stron.



Rysunek 3. Carolmen Logo interpreter widok z lutego 2013



Rysunek 4. Carolmen Logo interpreter widok interfejsu czerwiec 2013



Rysunek 5. Papert - Logo interpreter. Widok interpretera z uruchomionym przykładem procedury.

2.3.4. PapertLogo

PapertLogo to interpreter online zrealizowany w javascript. Wyróżnia się funkcjonalnym interfejsem (Rysunek 4), w którym uwzględniono bardzo szybkie i intuicyjne uruchamianie przykładów. Zrealizowano w nim dość duży podzbiór języka Logo, jednak projekt wygląda na niedokończony. Kod źródłowy projektu jest udostępniony na licencji MIT (Open Source Initiative OSI - The MIT License) . Ostatnich modyfikacji dokonano w 2010 roku. w projekcie brało udział kilka osób, jednak obecnie wygląda na porzucony.

Interpretera nie obudowano dodatkowymi pomocami i materiałami. Na rysunku 5 poniżej przedstawiono widok interpretera z uruchomioną procedurą przykładową.

Strona domowa http://Logo.twentygototen.org/

3. Wprowadzenie do języka Logo

W tym rozdziale została krótko omówiona historia języka Logo i typowe dla niego konstrukcje.

3.1. Historia języka

Pierwsza wersja języka Logo powstała w 1967 roku w Stanach Zjednoczonych w Laboratorium Sztucznej Inteligencji MIT (Massachusetts Institute of Technology) i była wynikiem pracy grupy uczonych pod kierownictwem matematyka Seymoura Paperta [1].

W pracy nad językiem wykorzystano wyniki badań szwajcarskiego pedagoga i psychologa, profesora Jeana Piageta (wcześniejszego współpracownika Paperta z Genewy), który przez kilkadziesiąt lat badał procesy uczenia się u dzieci. Podstawą była teoria konstruktywizmu.

W latach siedemdziesiątych Logo było rozwijane w MIT oraz kilku innych ośrodkach naukowych w Edynburgu (Szkocja) oraz w Tasmanii (Australia).

Szersze wykorzystanie Logo zaczęło się pod koniec lat siedemdziesiątych kiedy powstały pierwsze wersje języka dedykowane na dwie maszyny: Apple II i Texas Instruments TI 99/4. Wersje te wykorzystane zostały w projektach edukacyjnych prowadzonych w szkołach amerykańskich, a następnie ewoluowały do wersji komercyjnych.

W 1980 roku powstało Logo Computer Systems, Inc. (LCSI). Prezesem LSCI został Seymour Papert. w LCSI powstała wersja Apple Logo.

Od roku 1980, po publikacji książki Paperta Mindstorms, język Logo zdobył uznanie i popularność na całym świecie. Powstały nowe wersje Logo przetłumaczone na wiele języków.

W roku 1985 LCSI wyprodukowało wersję LogoWriter – z rozbudowanym interfejsem graficznym. w podobnym okresie powstała także wersja LEGO Logo umożliwiająca sterowanie robotem złożonym z klocków Lego, oraz wersja PCLogo dla DOS i później dla Windows wyprodukowana przez Harvard Associates.

Między innymi z powodu braku nowszych wersji LogoWriter w Stanach część ludzi nauki uznała Logo za projekt przestarzały, jednak w innych krajach rozwijał i rozwija się nadal.

w kilku krajach język wszedł do programów nauczania w szkołach (np. Ameryka Łacińska, Wielka Brytania, Japonia, Polska).

W Polsce w użyciu obecnie jest Logo Komeniusz wyprodukowany na Słowacji i spolszczony przez Andrzeja Walata, a dystrybuowany przez OEIIZK (Ośrodek Edukacji i Zastosowań

Komputerów w Warszawie) oraz zastępująca go Logomocja (polska wersja Imagine następcy Komeniusza, również dystrybuowana przez OEIIZK). Obydwie wersje są dziełem zespołu pracowników Uniwersytetu Komeniusza w Bratysławie: Ivana Kalaša, Petera Tomcsányi, Andreja Blaho i Lubomira Salanci [13]. Żadna z nich nie jest darmowa. Także uczniowie i nauczyciele danej szkoły muszą opłacić kopię dla siebie jeśli chcą z niej korzystać legalnie w domu.

Przyglądając się polskojęzycznym wersjom języka Logo, z którymi autorka miała czynienia osobiście (AC-Logo w latach dziewięćdziesiątych, Logo Komeniusz, Logomocja) można pokusić się o stwierdzenie, że rozwój nie idzie w stronę nacisku na algorytmikę, czy naukę pewnego rodzaju myślenia u młodzieży, a uatrakcyjniania środowiska. Oczywiście powstanie nowych wersji dla nowszych systemów operacyjnych jest zrozumiałe. Dobrym kierunkiem jest też rozszerzanie Logo o elementy programowania obiektowego. Jednak głównie dodawane są, w zasadzie niepotrzebne w nauce programowania, elementy typu tworzenie animacji i projektów multimedialnych, a mniej poszerza się możliwości samego języka (np. o więcej instrukcji sterujących; w Logo jest odpowiednik while ale nie ma do-while). w Logo Komeniusz można już stworzyć animowanego żółwia, w Logomocji tworzyć bardziej rozbudowane animacje. Biorąc pod uwagę fakt, że istnieją na rynku o wiele doskonalsze i bardziej atrakcyjne dla młodzieży narzędzia do tworzenia prezentacji multimedialnych uważam, że nie jest to dobry kierunek.

3.2. Opis języka Logo

Język Logo powstał jako dialekt języka Lisp [1]. Jest językiem wysokiego poziomu, proceduralnym. Konstrukcje języka z założenia są bardzo proste ponieważ Logo powstało jako język do zastosowań edukacyjnych. Podstawowym założeniem była łatwość opanowania go przez dzieci i młodzież.

Jako podstawę przy opisie elementów języka przyjęto wersję Logo Komeniusz wyprodukowaną w 1996 roku przez A. Blaho, I. Kalas P. Tomcsanyi z Comenius University w Bratysławie, spolszczoną przez Andrzeja Walata.

W 2001 roku ten sam zespół, który tworzył Komeniusza, utworzył nową wersję Imagine (nazywaną w Polsce Logomocją). w Imagine nie zmieniono podstawowych cech języka natomiast został on rozszerzony o elementy obiektowości oraz większe możliwości multimedialne (jest możliwe sterowanie zdarzeniami, tworzenie obiektów – obiektami mogą być żółwie lub obrazy, pokazów, tworzenie animacji, obsługa plików multimedialnych, rozpoznawanie mowy)

3.2.1. Podstawowe zasady

W języku Logo nie określamy typu zmiennych. Zasadniczo rozróżniane są słowa (w tym liczby czyli „słowa numeryczne" według nazewnictwa Komeniusza) i listy elementów. Wymienia się także wśród typów danych obrazy.

Słowa, stosowane jako parametry procedur lub stałe, poprzedzane są jednym znakiem cudzysłowu ("*słowo*). Listą natomiast, jest zestaw elementów oddzielonych spacjami umieszczony w nawiasach kwadratowych. Elementami list mogą być listy ([A B C] jest listą, ale także np. [[A B] C D]).

Procedury wbudowane nazywa się pierwotnymi, w odróżnieniu od zdefiniowanych przez użytkownika. Logo działa w trybie interpretera. Zbiory procedur użytkownika nazywane są projektami.

Procedury pierwotne w Logo Komeniusz dzielą się między innymi na: liczbowe, słowne i listowe, geometrii żółwia, operowania ekranem, sterujące i warunkowe, tworzenia i stosowania zmiennych, przetwarzania obrazów, zarządzania pamięcią i zarządzania plikami. Wszystkich procedur pierwotnych jest około 300 [2].

Część procedur pierwotnych jest przeciążonych. Np. *pierw* w zależności od tego czy parametrem będzie słowo, liczba czy lista daje w wyniku pierwszą literę słowa, cyfrę liczby lub pierwszy element listy.

Procedury pierwotne należące do kategorii geometrii żółwia są najliczniej reprezentowane. Obejmują, oprócz komend dotyczących położenia żółwia na ekranie także procedury zmiany parametrów pisaka, tła, a także samego żółwia. w wersji Komeniusz można posługiwać się kilkoma żółwiami jednocześnie.

Funkcje liczbowe obejmują podstawowe operacje arytmetyczne (+,-,*,/), a ponadto: *abs* (wartość bezwzględna), *arctan* (arctg), *cos*, *sin*, *tan* (tg), *pwk* (pierwiastek), *ent* (część całkowita), *losowa* (losowa liczba całkowita), *startlos* (inicjowanie generatora liczb losowych), *reszta*, zaokr, ln (logarytm naturalny), exp (exponent), *iloczyn*, *iloraz*, *ilorazc* (iloraz całkowity), *suma*, *różnica*.

Parametry funkcji podajemy po spacji. Jeśli parametrem jest zmienna poprzedzamy ją dwukropkiem.

Część z funkcji liczbowych posiada postać „zachłanną". Np. *iloczyn* standardowo ma dwa parametry oddzielone spacją np. *iloczyn* 2 3, można jednak tę funkcję zapisać także używając nawiasu okrągłego i większej liczby parametrów, np.: (*iloczyn* 2 3 4)

Ciekawą kategorię (zwłaszcza z powodu częstego zapominania o nich przy potocznym myśleniu o Logo) stanowią funkcje służące do operowania na słowach i listach. Należą do nich: porównania (ze względu na traktowania liczb także jako słowa: <,<=,>,>=,<>,=), *ASCII* (kod litery lub pierwszego znaku słowa, które będzie parametrem), *bezpierw / bp* (słowo lub lista bez pierwszego elementu), *bezost / bo*, *bezelementu / be* (słowo lub lista bez wskazanego jako pierwszy parametr elementu), *długość*, *element* (parametry numer i lista lub słowo, wynikiem jest element o podanym numerze), *element?* (czy dany element zawiera się w liście lub słowie), *nak* (dodanie elementu na koniec do listy lub słowa), *nap*, *liczba?* (czy dane słowo jest liczbą), *lista* (tworzenie listy z podanych parametrów, posiada także postać zachłanną), *lista?*, *los* (losowo wybrany element z listy podanej jako parametr), *numel* (numer elementu na liście), *odelementu* (fragment drugiego parametru zaczynający się od pierwszego parametru – o ile występuje), *pierw*, *ost*, *puste?*, *równe?*, *słowo*, *słowo?*, *znak* (znak którego kod jest parametrem), *zdanie / zd*.

Do spójników i stałych logicznych należą: *fałsz*, *prawda*, *i*, *lub*, *nie*. Przy tym i oraz *lub* posiadają postać zachłanną (z nawiasami przy większej niż dwa liczbie parametrów), a poza tym parametry dla nich podaje się tak jak i dla np. iloczynu z odstępami (np. *lub par1 par2*) .

3.2.2. Instrukcje sterujące i warunkowe

Procedury sterujące i warunkowe wymagają trochę dokładniejszego omówienia. Należą do nich między innymi (przykłady pochodzą w większości z systemu pomocy Logo Komeniusz 1.1.040 [2]):

- **`powtórz`** - parametry liczba i lista poleceń (w nawiasach kwadratowych). Jeśli liczba jest nie całkowita jej część ułamkowa zostanie pominięta. Działaniem jest powtórzenie wykonania określoną ilość razy poleceń wypisanych w liście. w procedurze nie ma zmiennej sterującej jak np. w Pascalu w pętli for, ale można użyć wśród poleceń w liście funkcji numpow / npw dającej w wyniku numer aktualnego powtórzenia. np.

  wynik działania: *powtórz 3 [ps npw]* to :
  1
  2
  3

- **`dla`** – postać : ***dla*** *słowo [n1 n2] [lista poleceń]* lub ***dla*** *słowo [n1 n2 n3] [lista poleceń] ,* gdzie n1, n2, n3 są liczbami, słowo to nazwa zmiennej.Jeśli drugim parametrem są dwie liczby n1 i n2 oraz n1<n2, to polecenia z listy zostaną wykonane dla wartości zmiennej od n1 do n2 włącznie, przy tym wartość zmiennej będzie wzrastać o 1, jeśli wystąpi także n3, to o n3.

Przykład: *dla " j [1 3] [pokaż :j]*

1

2

3

dla "j [1 3 2] [pokaż :j]

1

3

- **`dlakażdego`** – postać: *dlakażdego słowo parametr [lista poleceń]* , dla każdej wartości (podanej jako parametr, który może być konkretną wartością lub listą wartości) zmiennej o nazwie podanej parametrem słowo wykonywana jest lista poleceń.

  Przykład: *dlakażdego "w [Ola Ela Tola] [ps dlugość :w]*

  3

  3

  4

- **`dopóki`** – postać: *dopóki wyrażenie [lista poleceń]*, wykonywanie instrukcji zawartych na liście poleceń tak długo jak długo wyrażenie daje w wyniku wartość prawda (odpowiednik pętli *while*)

  Przykład:

  ```
  przyp "h kierunek
  np 10 pw 5
  dopóki [:h <> kierunek] [np 10 pw 5]
  ```

  (wynikiem będzie narysowanie kółka zakładając, że początkowy kierunek był równy 0; ponieważ *przyp* zmiennej :h przypisał aktualny kierunek żółwia – czyli kąt, a następnie żółw się obraca aż osiągnie kierunek początkowy)

- **`jeśli`** – postać: *jeśli wartość logiczna [lista poleceń]* lub *jeśli wartość logiczna [lista poleceń] [lista poleceń2]* , jeśli wartość logiczna daje w wyniku prawdę wykonywana jest lista poleceń, jeśli fałsz lista poleceń 2 (o ile występuje). Co ciekawe, całość, aby zadziałała, powinna być w Logo Komeniusz umieszczona w jednej linii. Uwaga: w Imagine jeśli ma tylko jedną listę poleceń, aby można było na drugiej liście wpisać działanie przy nie spełnionym warunku należy użyć **jeżeli**.

- **`test`** – postać: *test wartość logiczna jeślitak [lista poleceń] jeślinie [lista poleceń]* , działanie podobne do jeśli.

- **`czekaj`** – parametrem jest liczba milisekund, a wynikiem zatrzymanie na dany czas

- **`stop`** – zatrzymanie działania aktualnej procedury
- **`wy`** – wynik funkcji (odpowiednik *return* z języka c)

Definiowanie procedur przez użytkownika jest bardzo proste. Każda procedura zaczyna się słowem *oto*, po którym podajemy nazwę procedury i ewentualne parametry. Zakończenie procedury określa się używając słowa *już*.

Przykład procedury „kwadrat” (Listing 2)

Listing 2. Procedura „kwadrat”.

```
oto kwadrat :a
powtórz 4 [ np 100 pw 90 ]
już
```

Definiowanie funkcji zwracającej jakąś wartość wygląda podobnie (powinno wystąpić w niej użycie instrukcji *wy / wynik*). Na listingu 3 przedstawiono przykład procedury logicznej sprawdzającej czy dana liczba jest parzysta.

Listing 3. Procedura „parzysta?”.

```
oto parzysta? :a
jeśli reszta :a 2 = 0 [wy "prawda][wy "fałsz]
już
```

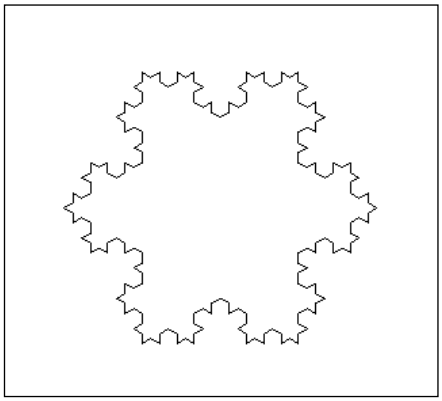
W przypadku funkcji rekurencyjnych może to wyglądać tak jak na listingu 4 przedstawiającym procedurę rysującą płatek Kocha:

Listing 4. Procedura bok rysująca rekurencyjnie jedną trzecią płatka Kocha, oraz procedura gwiazda tworząca całość.

```
oto bok :a :n
jeśli :n = 0 [np :a stop]
 bok :a / 3 :n - 1
 lw 60
 bok :a / 3 :n - 1
 pw 120
 bok :a / 3 :n - 1
 lw 60
 bok :a / 3 :n - 1
już
oto gwiazdka :a :n
 powtórz 3 [bok :a :n pw 120]
```

```
już
```

Na Rysunku 6 przedstawiono wynik działania powyższej procedury.



Rysunek 6. Wynik działania procedury "gwiazdka" dla parametrów a = 200 i n = 3

4. Propozycja zdalnego nauczania języka Logo

Najważniejszym elementem proponowanej koncepcji zdalnego nauczania języka Logo jest wykorzystanie interpretera zrealizowanego jako aplikacja działająca w przeglądarce internetowej połączonego ze standardowym kursem języka.

Podstawowe założenia:

- wykorzystanie interpretera online umożliwiającego nie tylko wykonywanie przykładów ale i własną naukę przez eksperymentowanie w myśl teorii konstruktywizmu
- dostępność kursu bez potrzeby logowania do serwisu
- możliwość stosowania poziomów użytkowników
- brak ograniczeń czasowych przy dostępie do kursów
- zakres kursu obejmujący elementy algorytmiki zawarte w programie nauczania gimnazjum

4.1. Kurs e-learningowy zrealizowany w pracy

Kursy e-learningowe zrealizowane w ramach pracy mają być dostępne dla każdej osoby przeglądającej stronę bez konieczności logowania czyli w roli gościa. Takie podejście ułatwi chętnym korzystanie z serwisu, pozwoli na wyszukanie potrzebnych informacji osobom, które nie mają potrzeby przechodzić regularnego kursu, może też zachęcić niezdecydowanych do założenia konta. Gość jednakże nie ma dostępu do wszystkich funkcjonalności serwisu.

Użytkownik zalogowany może wypełniać testy, śledzić swoje postępy, komunikować się z prowadzącym, pisać na forum, używać wszelkich innych dostępnych funkcjonalności (np. blog)

Zalogowany nauczyciel może zgłosić administratorowi serwisu chęć prowadzenia własnego kursu, w takim przypadku będzie mógł korzystać z bazy pytań do testów wspólnej dla całego serwisu i tworzyć własne bazy, a także z gotowych elementów kursu (np. lekcji które można zaimportować do nowego kursu).

Kursy nie są ograniczone czasowo mimo, że zastosowana platforma umożliwia dodanie takiego ograniczenia, dostęp do każdego z kursów otwartych jest nielimitowany, a kolejność lekcji nie jest wymuszona. Jest możliwość wprowadzenia regularnego kursu, ograniczonego czasowo,

z ograniczonym dostępem dla użytkowników i wymuszoną kolejnością lekcji jeśli użytkownik kursu będący jednocześnie nauczycielem będzie potrzebował takiej funkcjonalności.

Serwis edukacyjny jest stroną internetową opartą o system Moodle zawierającą zestawy złożone z teorii i ćwiczeń na poziomie od najbardziej podstawowego (zrozumiałego dla dzieci z ostatnich klas szkoły podstawowej i gimnazjalistów) do bardziej złożonych. Jego dokładniejszy opis został umieszczony w rozdziale 6.1.

Dzięki zastosowaniu Moodle w prosty sposób można dodawać kolejne kursy, rozbudowywać bazę pytań testowych dostępną dla całego serwisu, administrować użytkownikami, którzy mogą mieć różne poziomy dostępu, tworzyć lekcje interaktywne, korzystać z dostępnych modułów (test, kalendarz, forum, czat, ankiety, przechowywanie plików i inne).

Zakres materiału kursów obejmuje: podstawy geometrii żółwia, instrukcje sterujące `powtórz`, `jeśli`, `jeżeli`, pojęcie procedury, parametry procedur, procedury rekurencyjne, operacje na słowach i listach, procedury operujące na słowach i listach.

Założona jest możliwość przetestowania przykładów oraz próbowania wykonywania ćwiczeń bezpośrednio na stronie z kursem także w krótkich aplikacjach flaszowych oprócz interpretera zrealizowanego w Javascript.

4.2. Interpreter online

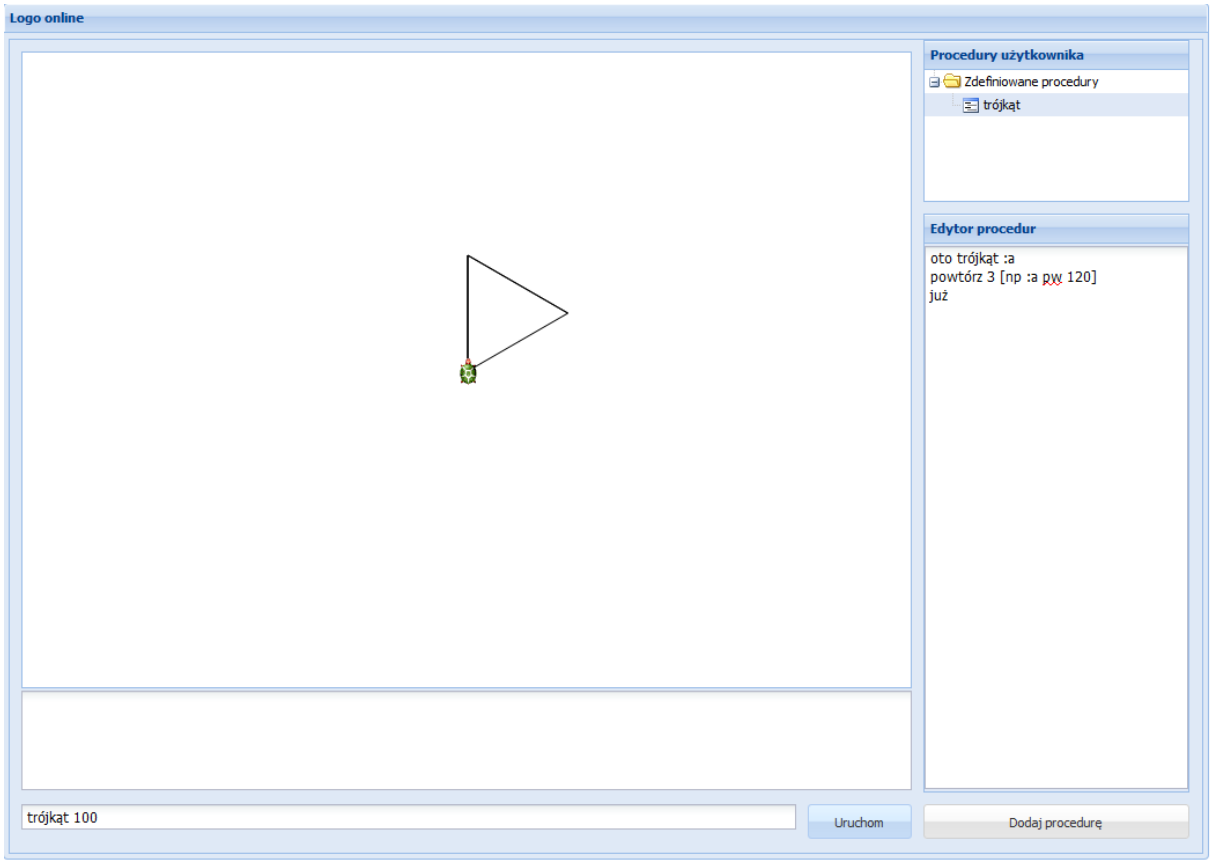
Interpreter online zrealizowany w pracy ma umożliwiać wykonywanie przykładów umieszczonych w kursie, wykonywanie procedur najczęściej używanych na lekcjach w szkołach, a także własną naukę myślenia algorytmicznego przez eksperymentowanie.

Działanie interpretera w oknie przeglądarki ma umożliwiać zabawę z żółwiem na każdym urządzeniu posiadającym przeglądarkę internetową oraz mogącym wyświetlać obraz w odpowiedniej rozdzielczości. w ten sposób uczący się nie jest uzależniony na przykład od odpowiedniej wersji systemu operacyjnego (obecne polskojęzyczne wersje środowisk języka Logo są aplikacjami dla systemu Windows) i nie jest zmuszony do kupna oprogramowania.

Interfejs graficzny użytkownika (Rysunek 7), złożony jest z okna żółwia, pola służącego do wyświetlania wyników procedur i komunikatów, pola służącego do wpisywania poleceń, edytora i listy procedur użytkownika wyświetlanej w postaci drzewa. Ponieważ w interfejsach Logomocja Imagine oraz Logo Komeniusz w oknie „pamięć” można edytować w danej chwili zawsze tylko jedną procedurę, podobnie zrealizowane zostało to w pracy.

Pracując z interpreterem można wpisywać całe listy poleceń bezpośrednio w polu komend oddzielając poszczególne polecenia spacjami, a następnie klikając przycisk *uruchom* wykonać je.

Aby dodać nową procedurę należy wpisać jej kod w oknie edytora, a następnie nacisnąć przycisk poniżej. Dodana procedura pojawia się na liście procedur użytkownika. Przycisk dodawania procedury w przypadku gdy okno edytora jest puste powoduje wyświetlenie się okna dialogowego z pytaniem o nazwę nowej procedury, następnie w oknie edytora umieszcza kod - szablon nowej procedury składający się z nagłówka procedury (`oto nazwa :parametr`), nowego wiersza jako miejsca na treść i słowa `już` kończącego procedurę.



Rysunek 7. Interfejs interpretera

Na wyświetlanej liście procedur znajdują się procedury wprowadzone przez użytkownika. Kliknięcie na nazwę którejkolwiek z nich powoduje przeniesienie jej kodu do edytora i umożliwienie edycji. w przypadku gdy istnieje już procedura o podanej nazwie kliknięcie przycisku *Dodaj procedurę*, umieszczonego pod edytorem, spowoduje aktualizację treści istniejącej procedury.

Zaimplementowany podzbiór języka Logo ma umożliwiać realizację typowych zadań spotykanych w nauce podstaw programowania z wykorzystaniem tego języka w gimnazjum. Wiąże się to z implementacją: interpretowania wyrażeń arytmetycznych, podstawowych komend dotyczących

grafiki żółwia, funkcji matematycznych, procedur operujących na słowach. Szczegółowy spis zaimplementowanych komend został umieszczony w rozdziale 6.2.3.

Interpreter ma także umożliwiać dodawanie procedur użytkownika. Procedury użytkownika mogą posiadać parametry, mogą też zwracać wartości. w czasie działania interpretera procedury użytkownika przechowywane są w pamięci.

Ważnym zagadnieniem w przypadku programowania z wykorzystaniem języka Logo jest rekurencja. Język Logo jest bardzo dobrym narzędziem do nauki podstaw myślenia rekurencyjnego ze względu na prostotę zapisu. Realizacja typowych przykładów rysunków tworzonych rekurencyjnie przy pomocy Logo jest dużo prostsza niż w innych językach programowania, gdzie nie ma zdefiniowanej „grafiki żółwia" i trzeba operować bezpośrednio współrzędnymi ekranowymi. Bardzo dobrym tego przykładem jest krzywa Kocha i złożony z takich krzywych płatek Kocha. Realizacja płatka Kocha za pomocą Logo w zaimplementowanym interpreterze została pokazana na Listingu 3 i Rysunek 6 w poprzednim rozdziale. Zdarza się, że elementy grafiki żółwia są implementowane w innych językach właśnie w celu uproszczenia rysowania takich grafik.

W zaimplementowanym interpreterze założone jest działanie procedur rekurencyjnych w tym takich, w których może się pojawić więcej niż jedno wywołanie rekurencyjne - tak jak w przytoczonym wyżej płatku Kocha.

Struktura kodu interpretera pozwala łatwo dodawać nowe, jeszcze niezaimplementowane, funkcje, zwłaszcza np. typowe funkcje matematyczne.

Podczas wyboru zbioru zaimplementowanych instrukcji języka Logo celowo pominięto te związane z możliwościami multimedialnymi oferowanymi przez istniejące środowiska Logo dla Windows tj. tworzenie animacji, edytowanie kształtu żółwia itd. z jednej strony to oczywiście uprościło implementację, z drugiej jednak według autorki nie są to elementy konieczne przy nauce myślenia algorytmicznego i podstaw programowania, a nawet mogą „rozpraszać uwagę".

5. Omówienie wykorzystanego oprogramowania

W tym rozdziale krótko przedstawiono oprogramowanie wykorzystane podczas pisania pracy.

5.1. Platforma Moodle

Platforma Moodle została wykorzystana do wykonania portalu zawierającego kursy programowania w Logo.

Moodle (Modular Object-Oriented Dynamic Learning Environment) jest systemem darmowym (licencja GNU PL) mającym duże możliwości i konkurującym z podobnymi platformami płatnymi. Obecnie korzysta z niego wiele szkół i instytucji (np. OKE), a także część uczelni (np. AGH w Krakowie).

Moodle został zaprojektowany przez pedagoga i informatyka Martina Dougiamas'a, który jest jego głównym developerem. Ma budowę modułową i intuicyjny interfejs. Napisany jest w PHP i wykorzystuje bazę danych MySQL lub PostgreSQL [4].

Moduły platformy Moodle:

- Zadania – możliwość przesyłania plików przez uczniów na platformę
- Czat – możliwy do włączenia dla uczestników kursu, w zależności od ustawień może być zapisywana historia czatu lub nie
- Głosowanie – umożliwia przeprowadzenie szybkiej ankiety, której wyniki są widoczne dla uczestników
- Dialog – komunikacja nauczyciel – student lub student – student
- Forum – możliwość założenia nawet kilku for dyskusyjnych w obrębie kursu. w forum można ustawiać uprawnienia użytkowników (jak i na całej platformie), włączać opcje wysyłania plików, wyświetlania treści w formie zwykłego tekstu lub html
- Słownik – można wykorzystać jako słownik terminów, może być listą pojęć encyklopedią itd.
- Lekcja – jeden z ważniejszych modułów, umożliwia umieszczanie lekcji w kursie. Lekcja może zawierać pytania a odpowiedzi na nie mogą sterować wyświetlaniem kolejnych ekranów, ponadto dla każdej lekcji można ustawić widoczność, dostępność, dostępność

lekcji może być uzależniona od innych elementów kursu (np. przerobienia poprzednich lekcji)

- Quiz – kolejny bardzo przydatny moduł – pozwala tworzyć testy z pytaniami należącymi do kilkunastu kategorii, bazy pytań mogą być dostępne dla innych testów w obrębie całej platformy, pytania można importować i eksportować do kilku formatów
- Zasób – umożliwia umieszczenie na stronie dowolnego pliku z danymi np. lekcji w formie pdf lub innej. Są też zasoby prywatne umożliwiające użytkownikom dodawanie własnych plików nie widocznych dla innych
- Ankieta – gotowe narzędzia do analizy (COLLES, ATTLS)
- Warsztat – możliwość wzajemnej oceny przez uczestników swoich projektów i dokumentów
- Kalendarz – oprócz wyświetlania aktualnej daty pozwala wprowadzać terminy z podziałem na cztery rodzaje (kursu, użytkownika, grupy, globalne)
- Oceny – możliwość wprowadzenia własnej skali ocen (przeliczania z procentów), przeglądania własnych ocen przez użytkowników oraz ocen całej grupy w kursie przez prowadzącego kurs
- Wiadomości – prywatne wiadomości wysyłane między użytkownikami. Wiadomości są trzymane na serwerze do momentu przeczytania przez odbiorcę, w wypadku nieobecności odbiorcy online są wysyłane na mail użytkownika. Można także w tym module zarządzać tworzonymi przez siebie grupami dyskutantów

Ponadto (między innymi):

- Możliwość podziału studentów na grupy tak by widzieli lub nie widzieli (w zależności od ustawień) uczestników z innych grup
- Kilka możliwości uwierzytelniania użytkowników i sterowanie dostępem do każdego elementu kursu i każdej aktywności (rozbudowany system zezwoleń)
- Możliwość tworzenia kopii kursów, importu i eksportu
- Możliwość przeglądania aktywności studentów (dostępna dla administratora strony i kursu) – podane ip Logowania i rodzaj aktywności np. upload plików, wejście na element kursu

Z platformy według danych na stronie moodle.org (Rysunek 8) obecnie korzysta ponad 70 milionów użytkowników w 236 krajach.

Rysunek 8. Statystyki Moodle na stronie moodle.org

5.2. Javascript

Javascript jest językiem skryptowym wykonywanym po stronie przeglądarki przeznaczonym do dodawania funkcjonalności do elementów interaktywnych na stronach www. Język został wymyślony i po raz pierwszy zastosowany przez Netscape w latach dziewięćdziesiątych.

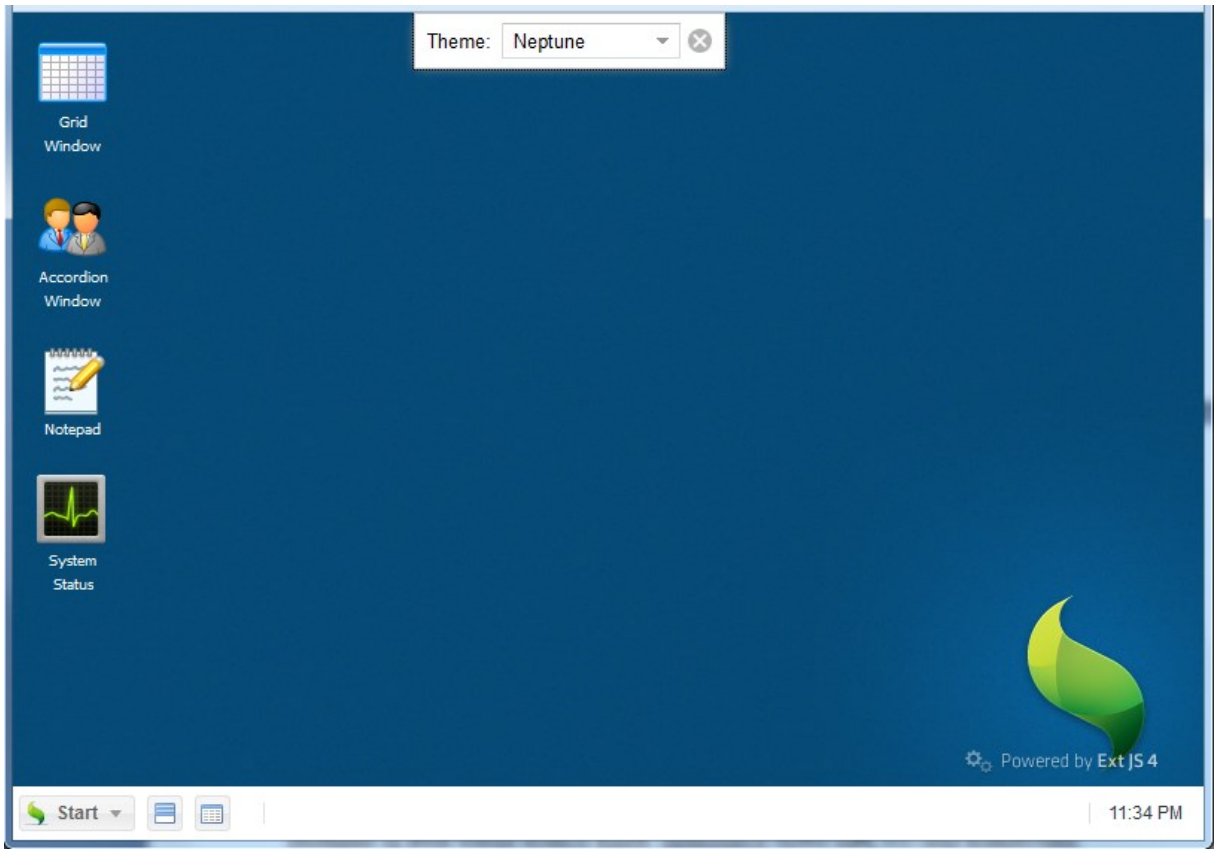
Podstawowe cechy Javascript: jest zorientowany obiektowo (ale opiera się na funkcjach i prototypach, nie ma pojęcia klas i dziedziczenia w hierarchii klas), właściwości i metody mogą być dołączane dynamicznie do obiektów, nie ma konieczności deklaracji zmiennych ani ustalania ich typu (ale należy pamiętać o zasięgu takich zmiennych), nie ma możliwości deklarowania sposobu przekazywania parametrów dla funkcji (zawsze przekazywane są przez wartość) [6].

Pisząc skrypty Javascript należy pamiętać, że mogą być różnie interpretowane przez różne przeglądarki.

5.3. Framework ExtJS

ExtJS jest rozbudowanym frameworkiem Javascript umożliwiającym tworzenie skomplikowanych interfejsów użytkownika dla aplikacji webowych z wykorzystaniem technologii Ajax [8]. Zestaw kontrolek wbudowanych ExtJS jest bardzo bogaty i pozwala tworzyć aplikacje wyglądające jak pulpit (Rysunek 9) w systemie operacyjnym (możliwe do wykorzystania na przykład przy tworzeniu aplikacji typu wirtualnego komputera), interfejsy graficzne dla aplikacji działających

na urządzeniach mobilnych oraz standardowych rozbudowanych aplikacji działających na www. Framework ten wspiera budowanie interfejsów z wykorzystaniem programowania obiektowego i zdarzeniowego, umożliwia współpracę z wieloma zewnętrznymi bibliotekami (np. JQuery) i dobrze współpracuje z wieloma przeglądarkami. Posiada rozbudowaną dokumentację obejmującą oprócz wszystkich funkcji także zestaw przykładów dostępnych online oraz dużą społeczność użytkowników. Dostępne są dwie wersje licencji ExtJS: opensource (GNU GPL v3) oraz komercyjna.



Rysunek 9. Wersja demonstracyjna aplikacji w stylu Web Desktop na bazie ExtJS

W pracy ExtJS został wykorzystany do utworzenia interfejsu interpretera języka Logo.

5.4. Camstudio

Camstudio [14] jest darmowym prostym programem do nagrywania aktywności pulpitu przydatnym przy tworzeniu różnego rodzaju tutoriali. Jak na niewielki darmowy program ma wystarczające możliwości. Pozwala na nagrywanie całego pulpitu lub wybranego jego fragmentu, nagrywanie dźwięku z mikrofonu i zapis powstałej animacji w formacie AVI lub Flash.

Camstudio wykorzystano w pracy do utworzenia krótkich prezentacji działania procedur i krótkich filmów instruktażowych

5.5. Selteco Alligator Flash Designer

Alligator Flash Designer (Rysunek 10) to bardzo prosty w obsłudze program przeznaczony do szybkiego tworzenia prezentacji i banerów flash. Umożliwia szybkie utworzenie prezentacji zawierającej kształty, obrazy, elementy aktywne typu przyciski i inne. Każdemu elementowi można przypisać w prosty sposób animacje dla trzech rodzajów zdarzeń (wejście, czas trwania ramki, wyjście), a także inne akcje łącznie z możliwością wykorzystana języka ActionScript. Obsługa jest intuicyjna, przypominająca działanie MS PowerPoint. Język ActionScript zaimplementowany w programie jest okrojony ale wystarczający do prostych zastosowań, w tym tworzenia niewielkich aplikacji flash [15].

Program został wykorzystany do utworzenia kilku prostych aplikacji do testowania poleceń żółwia w pierwszej części kursu.

Rysunek 10. Okno Alligatora pokazane na stronie domowej http://www.flashdesigner.pl/

6. Implementacja

W tym rozdziale omówione zostały zastosowane w pracy rozwiązania, umieszczony opis implementacji prototypu interpretera oraz oprogramowanych funkcjonalności. w rozdziale 6.1 omówiono wykonany serwis edukacyjny a w rozdziale 6.2 interpreter. Umieszczono także przykłady działania interpretera i zakres zaimplementowanego w nim języka.

6.1. Serwis edukacyjny

Poniżej został opisany, wykonany na potrzeby pracy, serwis edukacyjny oparty na platformie Moodle, jego konstrukcja oraz zastosowane rozwiązania.

6.1.1. Opis serwisu edukacyjnego

Rysunek 11. Widok głównego okna serwisu z poziomu administratora

Rysunek 11 i Rysunek 12 przedstawia widok głównej strony serwisu do nauki języka Logo wykonanego na platformie Moodle 2.2.3 z poziomu zaLogowanego i nie zaLogowanego użytkownika. w celu zachęcenia nowych użytkowników, a także spełnienia wymagań osób, które z różnych powodów nie chcą zakładać kont w serwisie, większość elementów jest dostępna bez potrzeby zakładania konta w serwisie to jest:

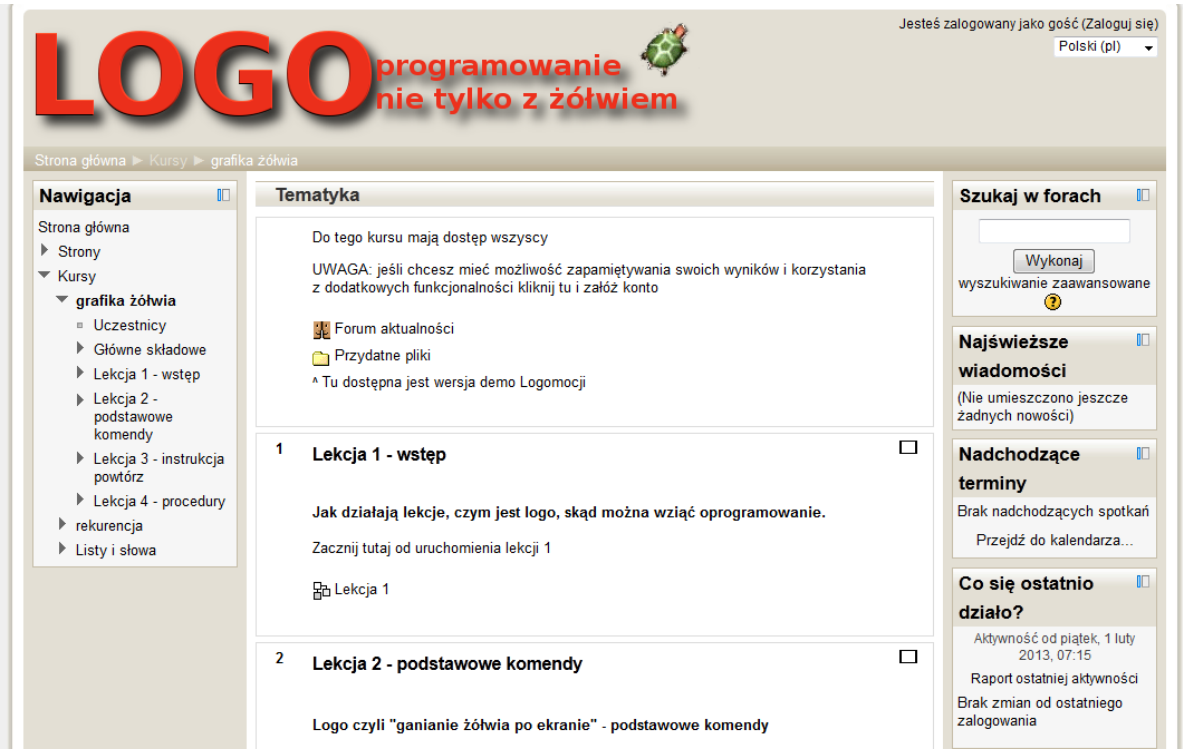
- przeglądanie listy istniejących kursów,
- dostęp do podstawowych kursów i umieszczonych w nich lekcji oraz przykładów,
- interpreter działający w oddzielnym oknie,
- czytanie informacji umieszczonych na forach kursów.

Rysunek 12. Widok głównego okna serwisu z poziomu gościa

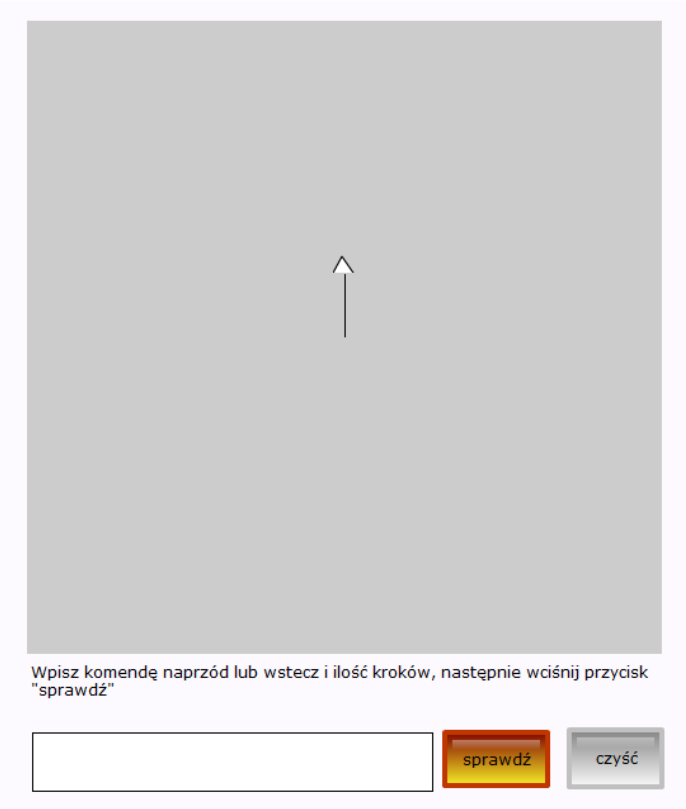
Zalogowany użytkownik może dodatkowo wypełniać testy i śledzić swoje postępy dzięki uzyskiwanym w ten sposób ocenom. Jest też możliwość udostępnienia chętnym nauczycielom założenia własnego kursu z wolnym dostępem dla gości lub zamkniętego.

W serwisie wprowadzone zostały trzy kursy dostępne publicznie z poziomu gościa: "Logo - grafika żółwia" (Rysunek 13), "Logo – rekurencja", "Logo – listy i słowa". Zakres kursu "Logo - grafika żółwia" – obejmuje podstawy pracy z żółwiem w Logo, a także pracę z podstawowymi instrukcjami sterującymi (*powtórz*, *jeżeli*) i tworzenie własnych procedur użytkownika. Kurs został uzupełniony o dodatkowe elementy interaktywne: proste aplikacje flash (Rysunek 14), prezentacje instruktażowe flash (Rysunek 15) oraz testy.

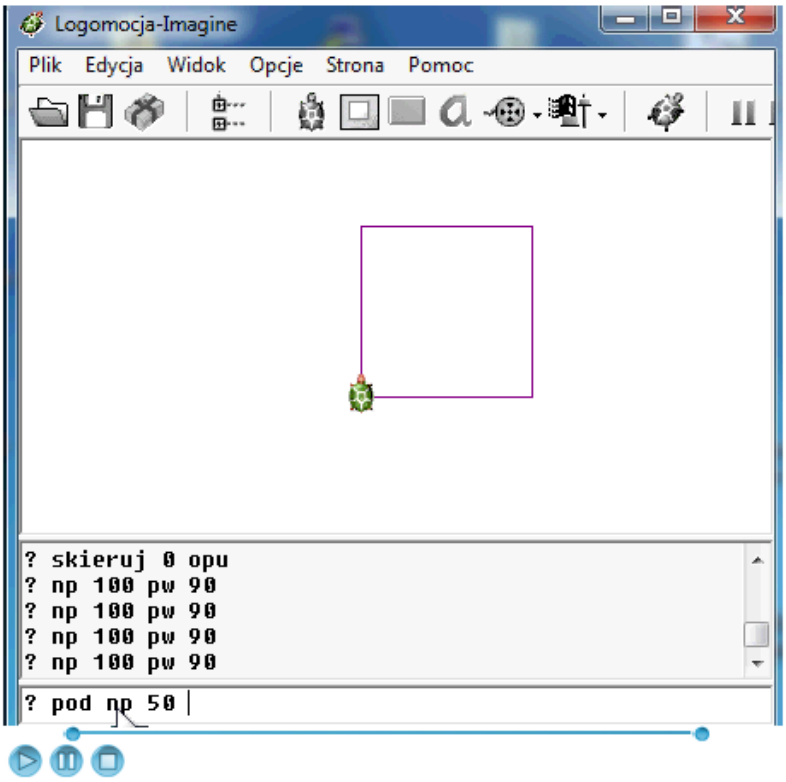
Kurs "Logo – rekurencja" zawiera podstawy rekurencji i standardowe przykłady graficznych procedur rekurencyjnych (płatek Kocha, dywan Sierpińskiego, drzewa). w każdej lekcji tego kursu jest umieszczona animacja pokazująca działanie przykładów.



Rysunek 13. Widok okna kursu "grafika żółwia" z poziomu gościa



Rysunek 14. Prosta aplikacja flash do testowania pojedynczych poleceń (umieszczona w lekcji 2 kursu)



Rysunek 15. Jedna z animacji instruktażowych umieszczonych w kursie

W kursie "Logo – listy i słowa" omówione zostało działanie procedur operujących na listach i słowach. Przykłady procedur tego typu, zastosowanie rekurencji w tego typu procedurach.

6.2.1. Implementacja serwisu edukacyjnego

Na wykonanie serwisu będącego obudową edukacyjną dla interpretera składają się następujące czynności:

- instalacja Moodle w wersji 2.2 na serwerze nazwa.pl pod adresem Logo.prog.waw.pl z użyciem bazy danych MySQL 5.5,
- dostosowanie interfejsu portalu do potrzeb projektu poprzez ręczną modyfikację elementów graficznych,
- utworzono trzech podstawowych kursów języka Logo,
- skonfigurowanie serwisu tak, by każdy użytkownik miał dostęp do lekcji w kursach bez potrzeby zakładania konta w serwisie,
- nałożenie wymagania założenia konta w przypadku, gdy użytkownik zechce skorzystać z: udostępnionych testów mających na celu sprawdzenie wiedzy, forum dołączonego do kursów, komunikacji z innymi użytkownikami, przechowywania plików w serwisie.

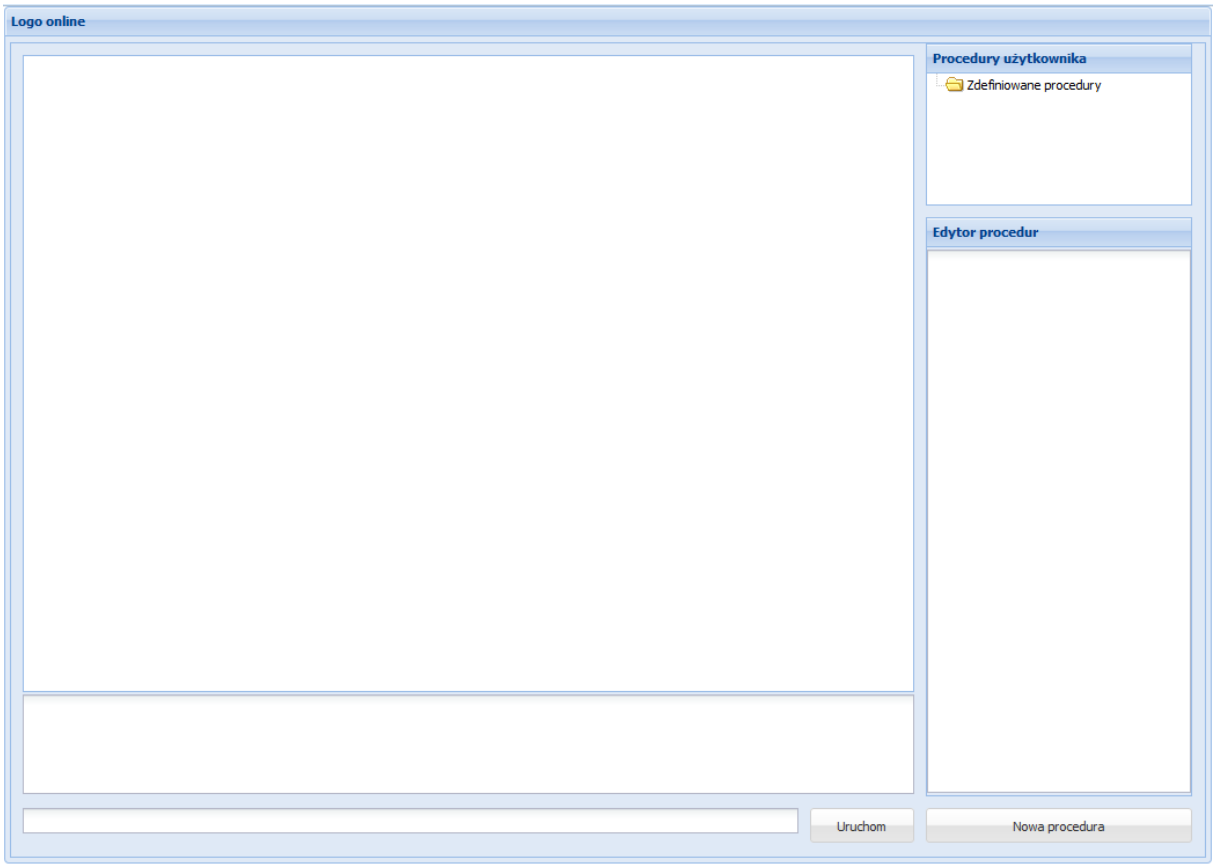
6.2. Interpreter

W podrozdziale tym umieszczono opis działania interpretera oraz implementacji zastosowanych w nim rozwiązań.

6.2.1. Opis interpretera

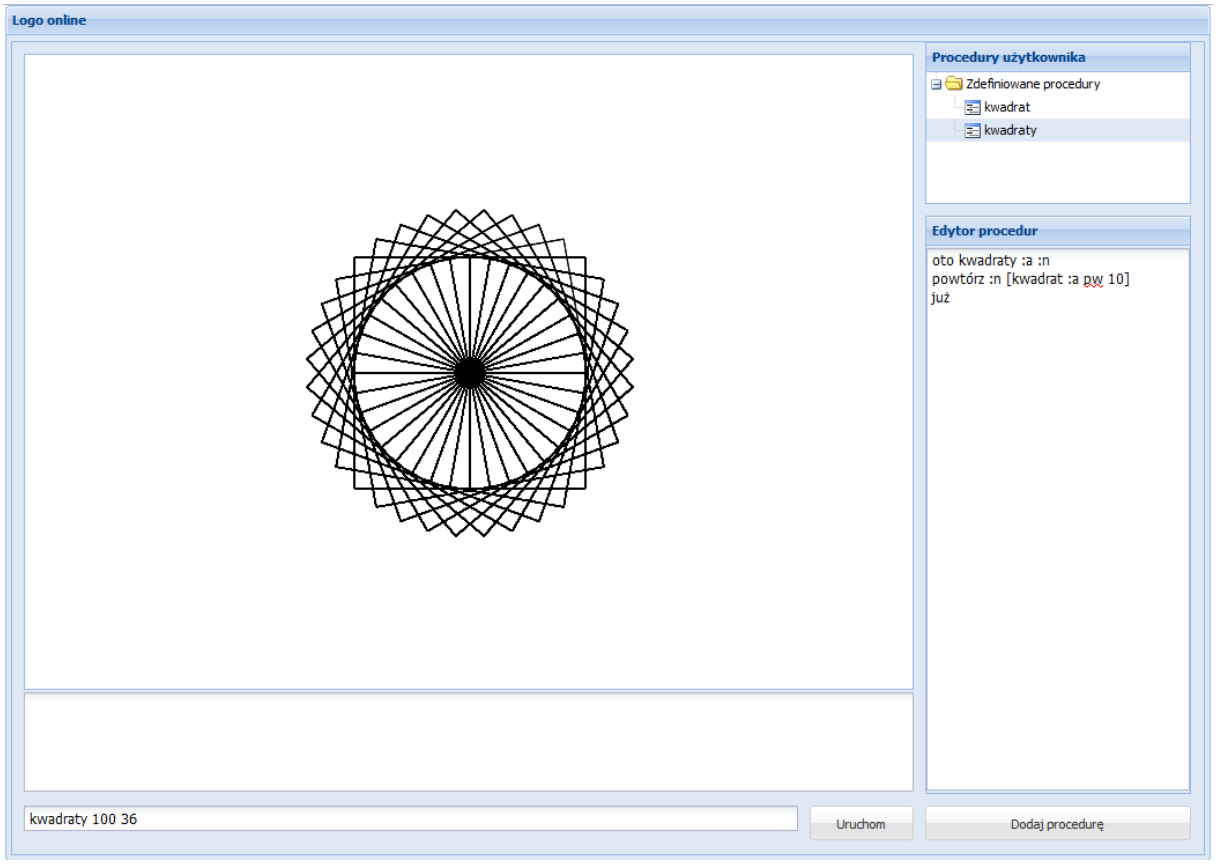
Interpreter dostępny z każdego miejsca kursu, działający w osobnym oknie pozwala na wykonywanie "na żywo" poleceń Logo oraz tworzenie własnych procedur. Interfejs pozwala na dodanie nowej procedury oraz wpisywanie poleceń. z okna interpretera dostępna jest pomoc dla języka. Interfejs graficzny interpretera został wykonany z wykorzystaniem frameworka ExtJS.

W oknie interpretera (Rysunek 16) nie piszemy całych programów w sensie kodu zawierającego wywołania procedur pierwotnych oraz definicje nowych procedur. Każda procedura użytkownika jest tworzona w oddzielnym oknie edytora i dodawana do pamięci za pomocą odpowiedniej funkcji wywoływanej przyciskiem przez użytkownika. Lista dodanych procedur użytkownika jest wyświetlana w oknie interpretera. Dodanie nowej procedury o nazwie już istniejącej powoduje zamianę istniejącej procedury na nową wersję.



Rysunek 16. Okno interpretera

Interfejs aplikacji interpretera został podzielony na pięć części. Największy obszar zarezerwowany został na okno żółwia (wstawiony został tam element Canvas z HTML5), pod nim znajduje się mniejszy obszar, w którym wyświetlane są wyniki działania funkcji oraz informacje o błędach, poniżej umieszczona została kontrolka umożliwiająca wpisywanie poleceń języka i obok przycisk uruchamiający wykonywanie poleceń. Po prawej stronie w kolejności od góry umieszczone zostały: okno do wyświetlania listy procedur wprowadzonych przez użytkownika (w postaci obiektu typu drzewo - treepanel z ExtJS), nieduże okno pozwalające na edycję procedur i przycisk dodający procedurę.



Rysunek 17. Działanie przykładowej procedury w interpreterze

Kliknięcie w istniejącą procedurę wyświetloną na liście procedur powoduje pojawienie się jej kodu w oknie edytora (Rysunek 17), takie działanie ułatwia edytowanie już istniejących procedur użytkownika. Naciśnięcie przycisku "Nowa procedura" w przypadku pustego okna edytora powoduje pojawienie się okna dialogowego z pytaniem o nazwę procedury, a następnie wpisanie do edytora szablonu procedury z wpisaną już jej nazwą. Naciśnięcie tego samego przycisku w wypadku, gdy w edytorze mamy treść procedury powoduje dodanie procedury do pamięci i wyświetlenie jej nazwy na liście (o ile już takiej nazwy nie było, w takim przypadku następuje zamiana istniejącej procedury o takiej samem nazwie na nową). w momencie dodawania procedury nie są raportowane użytkownikowi błędy wynikające z niepoprawność kodu procedury a jedynie poprawności "szkieletu" tj. czy

procedura ma wpisaną poprawnie nazwę i parametry. Ewentualne błędy w kodzie procedury są wyświetlane w momencie próby uruchomienia procedury. Taki sposób działania jest wykorzystywany w istniejących, działających w szkołach, aplikacjach Imagine i Logo Komeniusz.

6.2.2. Implementacja Interpretera Logo wykorzystanego w serwisie

Interpreter został zbudowany z kilku modułów odpowiadających etapom przetwarzania kodu źródłowego na wynik. Całość została wykonana w języku Javascript. z tekstu źródłowego usuwane są najpierw znaki białe, następnie tekst źródłowy najpierw dzielony jest na pojedyncze słowa (przy zasadzie, że znakiem rozdzielającym jest spacja, znaki operatorów oraz znaki nawiasów), a następnie identyfikowane są poszczególne "tokeny" to jest, elementy tekstu są dzielone na typy: zmienne, teksty, liczby itd. Następnie lista otrzymanych w ten sposób tokenów jest analizowana przez parser, a w dalszej kolejności wykonywana przez interpreter.

Skaner (omówiony w punkcie 6.2.2.1), parser (omówiony dokładnie w punkcie 6.2.2.2) i interpreter (omówienie w 6.2.2.3) zostały napisane od podstaw bez wykorzystania gotowych modułów tego typu, natomiast do budowy interfejsu użytkownika wykorzystany został framework ExtJS.

6.2.2.1. Skaner

Skaner odpowiedzialny jest za pierwszą część pracy, czyli podział tekstu źródłowego na "tokeny". Jest złożony z funkcji `textToWrd`, oraz `listToTokens`. Funkcja `textToWrd` dzieli podstawowy tekst na oddzielne słowa. Parametrem tej funkcji jest zwykły tekst, a wynikiem tablica słów. Przy tym: usuwane są znaki białe, oraz komentarze (komentarz na początku linii ma znak średnika i jest zawsze do końca linii), spacja, operatory arytmetyczne i logiczne oraz nawiasy są znakami rozdzielającymi słowa pod warunkiem, że nie są wewnątrz słowa (słowo zaczyna się od jednego znaku cudzysłowu) lub zdania (zdanie w tekście jest oznaczone na początku i końcu znakiem pionowej kreski)

Funkcja `listToTokens` na wejście otrzymuje wynik funkcji `textToWrd` i identyfikuje każdy token jako jeden z typów: operator, liczba, zmienna, słowo, zdanie, nawias, inne. w "inne" mogą się zawierać nazwy procedur pierwotnych (czyli wbudowanych języka) lub użytkownika lub teksty błędne. Na tym etapie nie jest sprawdzana poprawność zapisu języka. O poprawności syntaktycznej decyduje parser.

Każdy token reprezentowany jest jako obiekt Javascript zawierający pola: typ, wartość, argument1 i argument2. w polu typ umieszczana jest informacja z jakiego typu elementem mamy do czynienia, w polu wartość wpisywane jest słowo pobrane z tekstu wejściowego, argumenty są zarezerwowane dla dalszego wykorzystania przez parser. Lista tokenów zapisywana jest w tablicy.

Przykład:

Tekst wejściowy: `np 20 pw 40 powtórz 3 [np 40 pw 30]`

Wynik działania funkcji `textToWrd`: np,20,pw,40,powtórz,3,[,np,40,pw,30,]

Wynik działania funkcji `listToTokens`: nn np, num 20, nn pw, num 40, nn powtórz, num 3, na [, nn np, num 40, nn pw, num 30, na]. Przy tym na liście pierwszy element to typ a drugi wartość dla każdego tokena.

6.2.2.2. Parser

Parser ma za zadanie przeanalizować listę tokenów i utworzyć listę instrukcji, które będą gotowe do wykonania.

Parser został wykonany jako obiekt zawierający: listę nazw procedur analizowanego języka wraz z liczbą ich parametrów (zrealizowaną jako tablica asocjacyjna, w której indeksem jest nazwa procedury a wartościami są ilości parametrów), listę nazw procedur użytkownika (zrealizowaną w podobny sposób) wraz z liczbą parametrów, listę synonimów procedur (w języku Logo część procedur ma nazwy skrócone i pełne) oraz kilka funkcji odpowiedzialnych za tworzenie listy wynikowej.

Listy nazw procedur zawierające też ilości parametrów są konieczne ponieważ w języku Logo parametry procedur oddzielane są spacjami i nie są ujęte w nawiasy jak w większości języków programowania. w związku z tym na etapie analizy syntaktycznej byłoby niemożliwe podjęcie decyzji jaka ma być kolejność wykonywania funkcji.

Rozważmy przykład: `fun1 :a fun2 :b :c`

Taki zapis można interpretować jako: fun1(a,fun2(b,c)) lub fun1(a), fun2(b,c) lub fun1(a,fun2(b),c). Nie da się tego rozstrzygnąć nie znając liczby parametrów obu funkcji. Dodatkowo muszą być w momencie parsowania znane także ilości parametrów dla funkcji użytkownika.

Parser w wyniku tworzy listę tokenów, które reprezentują instrukcje języka. Przy tym każdy element listy może być drzewem.

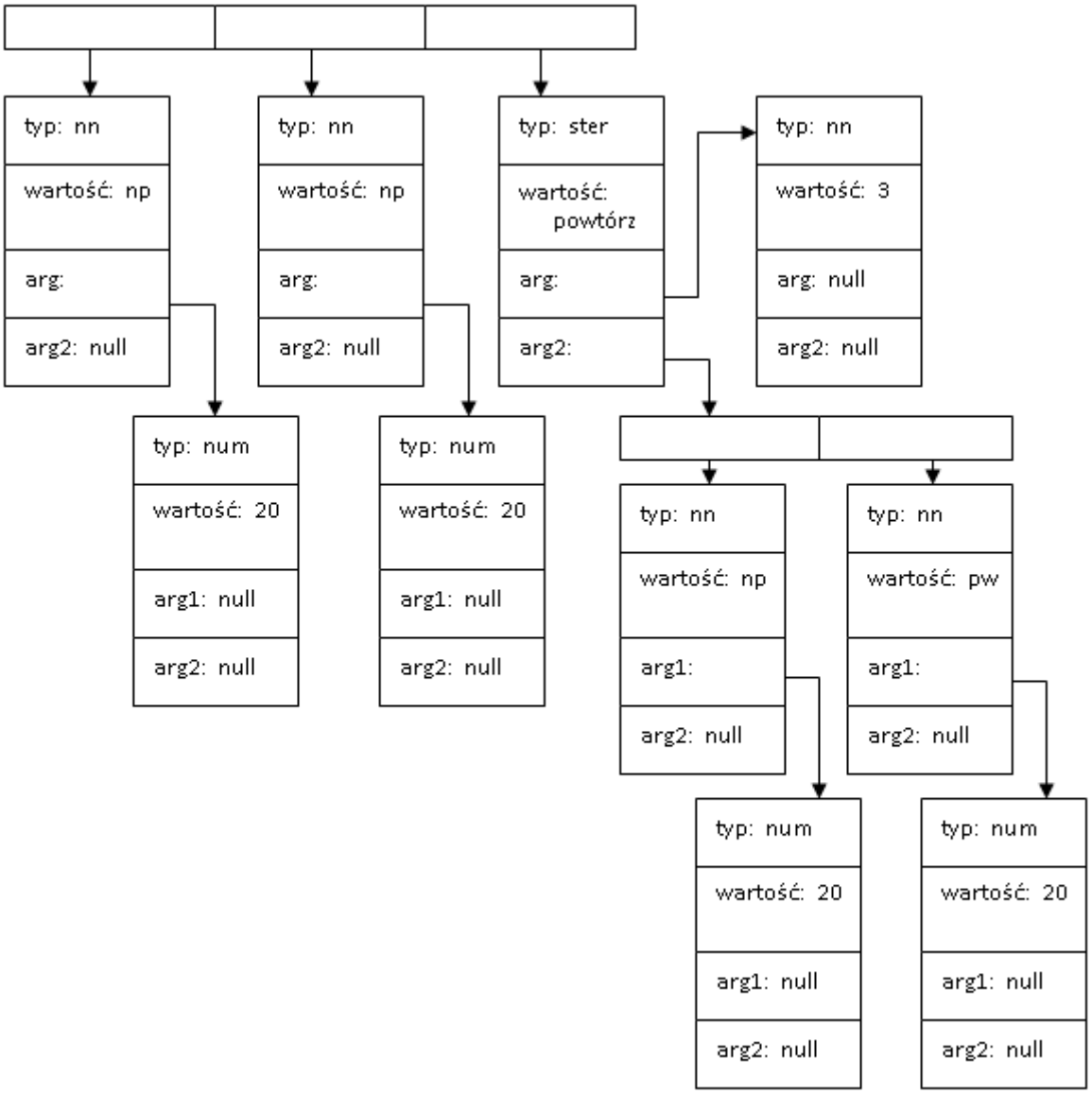
Weźmy tekst wejściowy z przykładu wcześniejszego pokazywanego przy skanerze:

`np 20 pw 40 powtórz 3 [np 40 pw 30].`

Po przetworzeniu przez skaner a następnie parser otrzymamy wynik, który jest listą zawierającą kolejne tokeny (Rysunek 18):

((typ - nn, wartość - "np", arg - (typ num, wartość 20), arg2 - null), (typ - nn, wartość - "pw", arg - (typ num, wartość 40), arg2 - null), (typ - ster, wartość - "powtórz", arg - (typ num, wartość 3),

arg2 - ((typ - nn, wartość - "np", arg - (typ num, wartość 40), arg2 - null), (typ - nn, wartość - "pw", arg - (typ num, wartość 30), arg2 - null))).



Rysunek 18. Schemat struktury danych powstałej na wyjściu parsera dla podanego przykładu

6.2.2.3. Interpreter

W Interpreterze przechowywana jest lista funkcji języka Logo ładowana w momencie tworzenia nowego obiektu typu interpreter, lista procedur użytkownika, oraz pamięć. Ponadto zdefiniowana jest funkcja, której zadaniem jest wykonanie instrukcji ze struktury dostarczonej przez parser oraz obsługę błędów. Przez oddzielną funkcję ładowane do pamięci są także procedury użytkownika.

W tej wersji interpretera nie ma możliwości napisania programu składającego się z kilku procedur oraz poleceń jako jednego ciągu tekstu. w jednym oknie edytora może być edytowana

w danej chwili tylko jedna procedura. Zasada ta jest zgodna z zasadą obsługi środowisk używanych w polskich szkołach - Logomocji i Komeniusza.

Funkcja wykonująca instrukcje (`processTokens`) wykonuje czynności: rozpoznanie kolejnego tokena, pobranie jego argumentów i wykonanie funkcji oraz instrukcji sterujących. Jeżeli argumentem jest lista instrukcji jest dla niej ponownie wywoływana ta sama funkcja (rekurencyjnie). Wykonanie różnego rodzaju instrukcji, ze względu na przejrzystość zapisu zostało podzielone na dodatkowe funkcje (`processControl` -wykonanie instrukcji sterujących, `runFun` - uruchomienie procedury).

6.2.2.4. Graficzny interfejs użytkownika

Interfejs został wykonany z użyciem frameworka ExtJS 4.1.1a. Ze względu na to, że interfejs generalnie jest prosty i składa się z jednego głównego panelu, całość kodu związanego z nim, łącznie z obsługą zdarzeń, została umieszczona w jednym pliku o nazwie skrypt.js. w pliku tym zdefiniowane zostało główne aplikacji okno jako panel, na którym umieszczone zostały pozostałe kontrolki to jest: duży panel z tagiem Canvas, pola tekstowe do wyświetlania wyników i wpisywania poleceń typu textfield i textareafield, dwa przyciski: jeden służący do uruchamiania poleceń, drugi, służący do dodawania procedur oraz treepanel reprezentujący drzewo do wyświetlania listy procedur i pole tekstowe służące do edycji procedur. Panel z tagiem Canvas, pola tekstowe do wprowadzania poleceń i wyświetlania wyników oraz przycisk uruchamiający polecenia umieszczone zostały po lewej stronie głównego panelu, pozostałe elementy tj. drzewo wyświetlające wprowadzone przez użytkownika procedury oraz pole służące do edycji procedur wraz z przyciskiem do dodawania nowych procedur pod nim - po prawej stronie.

Fragmenty kodu interfejsu:

Funkcja onReady uruchamiana w momencie załadowania okna - listing 5.

Listing 5. Funkcja onReady.

```
Ext.onReady(function() {
var okno= new MyPanel;
okno.render('box');
z=new zolw("zolw1","canvas","canvas1");
interpreter=new Interpreter(z);});
```

Na Listingu 6 widoczny jest element treepanel służący do wyświetlania listy procedur wraz z kodem obsługi zdarzenia kliknięcia na liście powodującego załadowanie treść istniejącej procedury do edytora:

Listing 6. element treepanel

```
items: [
          {
              xtype: 'treepanel',
              x: 790,
              y: 0,
              height: 140,
              width: 230,
                  name: 'tree',
                  id: 'tree',
              title: 'Procedury użytkownika',
            root:{
            text: 'Zdefiniowane procedury',
            expanded: true
            },
            listeners: {
            itemclick: {
            fn: function(view, record, item, index, event) {
            var nazwa=record.get('text');
            var ui=interpreter.findUInstr(nazwa);
            if (ui!=null)
            Ext.getCmp('proc').setValue(ui.source);
            Ext.getCmp('button2').setText("Dodaj procedurę");
            }
    }
    }
},
```

Na listingu 7 widoczny jest fragment kodu obsługi przycisku "Uruchom" - uruchamiającego polecenia wpisywane przez użytkownika, pole do wpisywania komend i okno żółwia ze wstawionym tagiem Canvas.

Listing 7. Przycisk „Uruchom”, pole komend i okno żółwia z tagiem canvas.

```
    {
          xtype: 'button',
          x: 690,
          y: 660,
```

```
                height: 30,
                width: 90,
                text: 'Uruchom',
                id: 'button1',
                name: 'button1',
                handler: function() {
                var s= Ext.getCmp('text').getValue();
                var wynik=interpreter.processText(s);
                Ext.getCmp('wynik').setValue(wynik);
                                        }
                      },
          {
                xtype: 'textfield',
                x: 10,
                y: 660,
                width: 670,
                name: 'text',
                id: 'text',
                fieldLabel: ''
             },
            {
                xtype: 'panel',
                x: 10,
                y: 10,
                height: 550,
                width: 770,
                title: '',
                borders: false,
                plain: true,
                id: 'myCanvas',
                html:       '<canvas       id="canvas"       width="770"
height="550"></canvas>'}
```

6.2.3. Opis elementów języka Logo w zakresie używanym w projekcie interpretera

Poniżej przedstawiono listę zaimplementowanych elementów języka Logo wraz z opisem ich działania procedur z podziałem na kategorie.

1) Operatory:

Zrealizowano obsługę operatorów arytmetycznych: `*`, `/`, `+`, `-` (z uwzględnieniem kolejności wykonywania operatorów), operatorów: `<`, `>`, `>=`, `<=`, `=` oraz nawiasów zwykłych („okrągłych") w wyrażeniach arytmetycznych.

2) Funkcje:

- `cos` - cosinus kąta podanego po spacji
- `sin` - sinus kąta podanego po spacji
- `pwk` (`pierwiastek`) - pierwiastek z liczby podanej po spacji
- `reszta` - reszta z dzielenia pierwszego parametru przez drugi
- `liczba?` - funkcja sprawdzająca czy parametr jest liczbą
- `słowo?` - funkcja sprawdzająca czy parametr jest słowem (zaczyna się od cudzysłowu)
- `długość` - liczba liter w słowie lub liczba cyfr liczby
- `puste?` - sprawdzanie czy parametrem jest puste słowo
- `element?` - funkcja sprawdzająca czy pierwszy parametr występuje w drugim (litera w słowie, cyfra w liczbie)

3) Procedury geometrii żółwia (w nawiasach podano drugie - zazwyczaj dłuższe - wersje poleceń):

- `np` (`naprzód`) - polecenie powodujące rysowanie linii o podanej po spacji długości w kierunku wskazywanym przez parametr kąt wskazujący odchylenie żółwia od pionu w stopniach
- `ws` (`wstecz`) - polecenie powodujące rysowanie linii o podanej po spacji długości w kierunku odwrotnym od wskazywanego przez parametr kąt
- `lw` (`lewo`) - zmiana kierunku żółwia o podany po spacji parametr w stopniach przez odjęcie parametru od aktualnego kąta, inaczej skręt w lewo (w miejscu)
- `pw` (`prawo`) - zmiana kierunku o podany po spacji parametr w stopniach przez dodanie parametru od aktualnego kąta, inaczej skręt w prawo (w miejscu)
- `pod` (`podnieś`) - podniesienie pisaka żółwia - czyli wyłączenie rysowania linii podczas ruchu żółwia
- `opu` (`opuść`) - opuszczenie pisaka żółwia

- `cs` (`czyść`) - czyszczenie ekranu i ustawienie żółwia w pozycji zerowej (na środku)
- `pisak` - zwraca aktualny stan pisaka wypisując informację w oknie wyników
- `upis` (`ustalpisak`) - ustala aktualny stan pisaka na opu, pod lub ścier
- `sż` (`schowajMnie`) - wyłącza widoczność żółwia
- `pż` (`pokażMnie`) - włącza widoczność żółwia
- `ścier` (`ścieranie`) - włącza tryb ścierania (rysowanie kolorem tła)
- `ukp` (`ustalKolPis`) - ustalanie koloru pisaka (jeden z 16 kolorów podstawowych wpisywanych jako liczba lub jako nazwa koloru)
- `ugp` (`ustalGP`) - ustalanie grubości pisaka

4) Instrukcje sterujące:
   - `powtórz` - powtarzanie instrukcji podanych w nawiasach kwadratowych podaną (po spacji) ilość razy
   - `jeśli` - wykonanie instrukcji podanych w nawiasach kwadratowych pod warunkiem, że warunek podany po instrukcji jeśli ma wartość prawda
   - `jeżeli` - wykonanie instrukcji podanych w nawiasach kwadratowych pod warunkiem, że warunek podany po instrukcji jeżeli ma wartość prawda, lub instrukcji podanych w następnych nawiasach jeśli wartość warunku jest fałsz
5) Funkcje operujące na słowach:
   - `pierw` (`pierwszy`) - pierwsza litera słowa lub pierwsza cyfra liczby
   - `ost` (`ostatni`) - ostatnia litera słowa lub ostatnia cyfra liczby
   - `nap` - dodanie pierwszego parametru na początku drugiego parametru (słowa lub liczby)
   - `nak` - dodanie pierwszego parametru do końca drugiego parametru (słowa lub liczby)
   - `bo` - słowo bez ostatniej litery (lub liczba bez ostatniej cyfry)
   - `bp` - słowo bez pierwszej litery (lub liczba bez pierwszej cyfry)
   - `słowo` - słowo złożone z dwóch parametrów (słów) lub liczba złożona ze sklejenia dwóch liczb

- `element` - pierwszy parametr to liczba, drugi słowo (lub liczba), wynikiem jest litera stojąca na danym miejscu w słowie (lub cyfra liczby)

6) Inne:

- `ps` (`pisz`) - procedura powodująca wypisanie parametru, którym może być tekst, liczba lub wartość funkcji
- `stop` - instrukcja zatrzymująca działanie aktualnie wykonywanej procedury
- `wy` - instrukcja zwracająca wynik działania funkcji

6.2.4. Przykłady działania interpretera

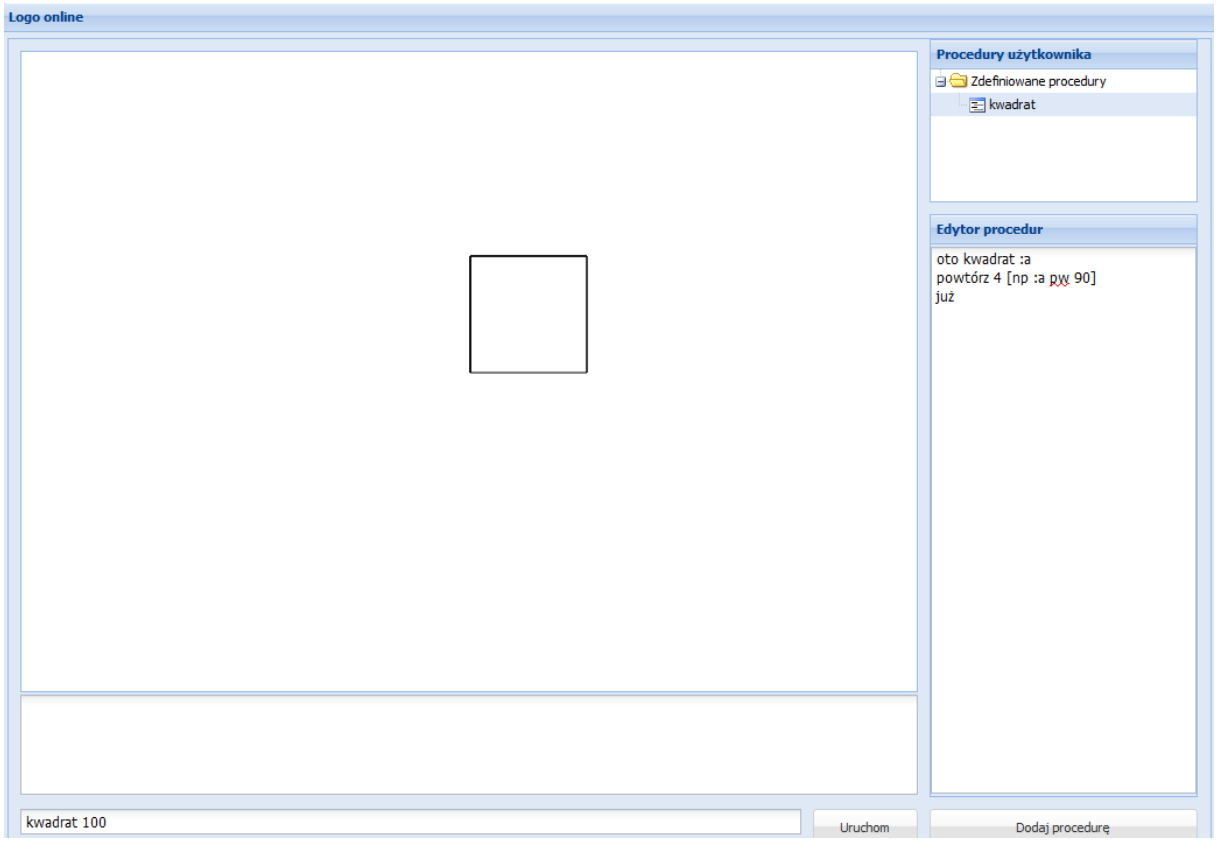
W tym podrozdziale przedstawiono przykłady procedur zrealizowanych w interpreterze online.

Rysowanie prostych figur geometrycznych:

Rysunek 19 i listing 8 przedstawiają procedurę rysującą kwadrat:

Listing 8. Procedura kwadrat

```
oto kwadrat :a
powtórz 4 [np :a pw 90]
już
```

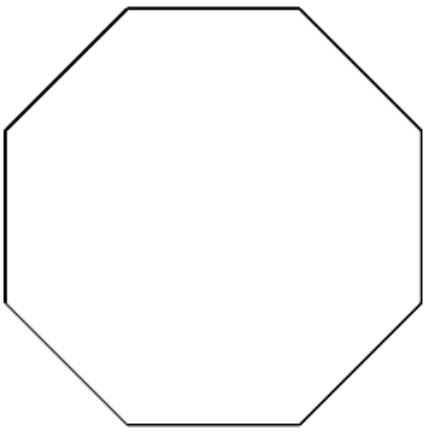


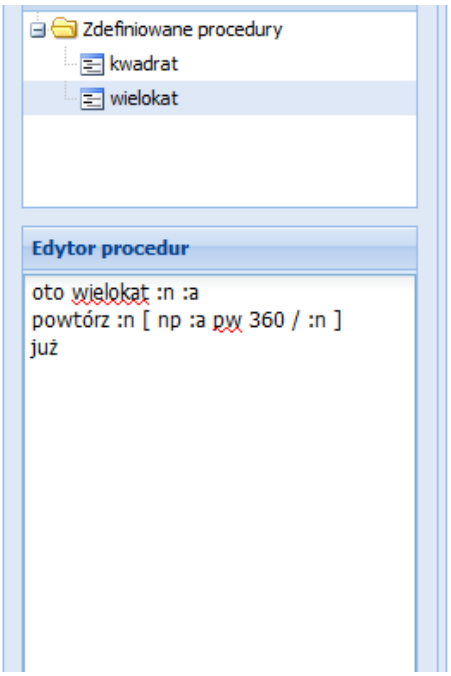
Rysunek 19. Procedura kwadrat

Procedura rysująca wielokąt o podanej liczbie boków i długości boku listing 8 i Rysunek 20:

Listing 9. procedura wielokąt o ilości boków :n i długości boku :a

```
oto wielokat :n :a
powtórz :n [ np :a pw 360 / :n ]
już
```



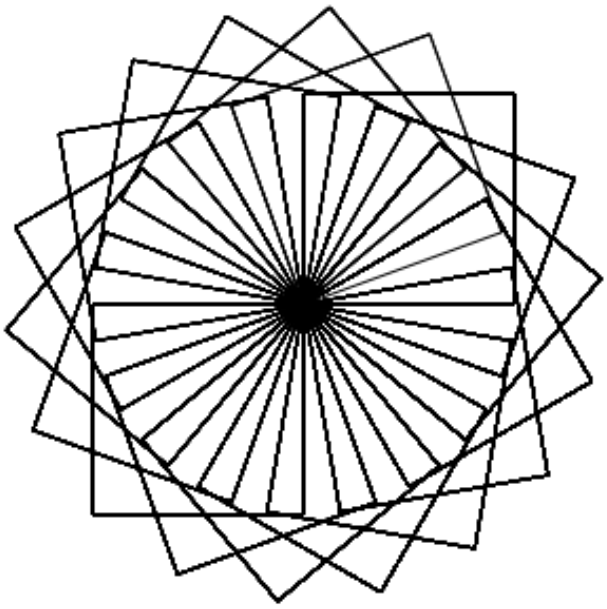


Rysunek 20. Procedura wielokąt dla parametrów 8 100

Procedura rysująca rozetę z kwadratów (listing 10 i Rysunek 21):

Listing 10. Procedura kwadraty

```
oto kwadraty :a :n
powtórz :n [kwadrat :a pw 360 / :n]
już
```



Rysunek 21. Wynik działania procedury kwadraty dla parametrów o wartościach 100 i 18

Procedury rekurencyjne:

Procedura rysująca łańcuch n kwadratów z których każdy następny ma bok mniejszy o połowę od poprzedniego (listing 11 i Rysunek 22).

Listing 11. Procedura rekurencyjna kwadraty1.

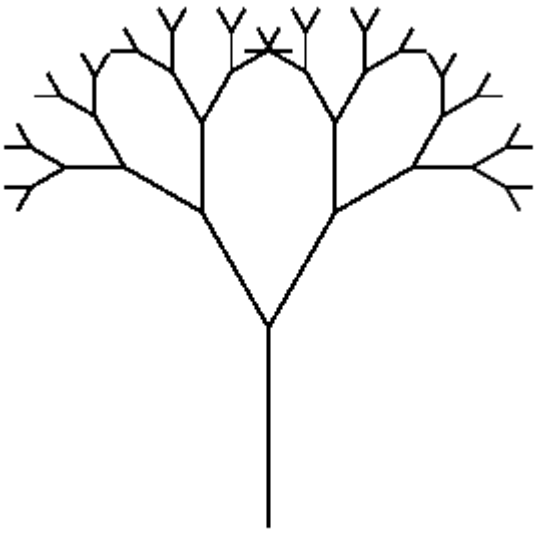
```
oto kwadraty1 :a :n
jeśli :n = 0 [ stop ]
kwadrat :a pw 90 np :a lw 90
kwadraty1 :a / 2 :n - 1
już
```

Rysunek 22. Wynik działania procedury *kwadraty1* dla parametrów 100 i 4.

Procedura rekurencyjna rysująca drzewo binarne (listing 12 i Rysunek 23).

Listing 12. Procedura rysująca drzewo

```
oto drzewko :a :n
jeśli :n = 0 [ stop ]
np :a lw 30
drzewko 2 * :a / 3 :n - 1
pw 60
drzewko 2 * :a / 3 :n - 1
lw 30
ws :a
już
```



Rysunek 23. Wynik działania procedury ***drzewko*** dla parametrów 100 i 4

Funkcje:

Funkcja licząca moduł liczby (listing 13):

Listing 13. Funkcja licząca wartość modułu z liczby a.

```
oto modul :a
jeżeli :a >= 0 [ wy :a ]
 [ wy -1 * :a ]
już
```

Aby sprawdzić działanie funkcji moduł należy wpisać komendę używając instrukcji `ps` (pisz) np.: `ps moduł -10` (Rysunek 24).

Rysunek 24. Wynik działania funkcji *moduł*.

7. Podsumowanie

W tym rozdziale autorka zamieściła podsumowanie zrealizowanej pracy przedstawiając jej wady i zalety oraz propozycje planu rozwoju.

7.1. Wady i zalety rozwiązania

Podstawową zaletą zastosowanego rozwiązania jest wykorzystanie elementów interaktywnych, zwłaszcza interpretera online. Umożliwiają one naukę przez własne eksperymentowanie czyli realizację założeń konstruktywizmu.

Zrealizowany interpreter pracuje bezpośrednio na stronie WWW co sprawia, że można go uruchomić na dowolnym sprzęcie obsługującym przeglądarki internetowe. Jego rozszerzanie o kolejne procedury w większości przypadków nie jest skomplikowane. Na przykład w przypadku rozszerzenia o dodatkowe instrukcje reprezentujące funkcje matematyczne jest to kwestia dodania kodu tylko w jednym miejscu w interpreterze.

Materiały elearningowe dołączone do projektu mogą być dowolnie rozszerzane, jak każde materiały tego typu, o większą ilość artykułów, przykładów, testów i zadań. Ich umieszczenie na popularnym systemie edukacyjnym Moodle jest korzystne z punktu widzenia użytkowników, którzy prawdopodobnie w większości kiedyś się już z nim zetknęli - dotyczy to zwłaszcza nauczycieli.

Niewątpliwą wadą samej aplikacji pracującej w przeglądarce opartej na Javascript jest brak jej odporności na odświeżanie widoku strony - w tym momencie traci się zapisane w pamięci procedury użytkownika.

Wadą zastosowanego rozwiązania może być przełamywanie przyzwyczajeń użytkowników, którzy często poszukują w kursach tego typu gotowych rozwiązań do zastosowania, a nie narzędzi służących do znajdowania tych rozwiązań.

7.2. Proponowany plan rozwoju

Elementy, które można rozszerzyć lub dodać:

- ponieważ nie zostały zaimplementowane wszystkie instrukcje języka Logo występujące w używanej obecnie w szkołach Logomocji pierwszą ścieżką rozwoju byłoby dodanie kolejnych poleceń [3] np.

- `dlaKażdego` - instrukcja sterująca, będąca odpowiednikiem pętli foreach występujących w PHP czy Java,
- `piszTekst` - instrukcja służąca do wpisywania tekstu na ekranie za pomocą żółwia,
- `czcionka` - ustalanie czcionki pisanego przez żółwia tekstu,

- rozszerzenie działania instrukcji `nap`, `nak`, `bo`, `bp` o obsługę list elementów; tworzenie procedur użytkownika operujących na listach rzadko pojawia się w szkole (np. w pozycji [5] nie ma ani jednego takiego przykładu) ale bywa na konkursach dla uczniów
- dobrym pomysłem byłoby dołączenie do interpretera biblioteki przykładów, które można by było łatwo ładować do pamięci i wykonywać,
- do interpretera można dodać obsługę wielu żółwi

7.3. Wnioski końcowe

Zrealizowany projekt pozwala na rozpoczęcie samodzielnie lub pod kierunkiem nauczyciela przygody z podstawami programowania w języku Logo, a także, przy tej okazji, z podstawami myślenia algorytmicznego i podstawami rekurencji. Nie ma przy tym już konieczności kupowania licencjonowanego oprogramowania, a pracować z materiałami można na każdym sprzęcie umożliwiającym uruchomienie przeglądarki internetowej.

Nie zostały zaimplementowane wszystkie instrukcje występujące w polskiej wersji Logo - Komeniusz i Logomocji, ale obecny zbiór pozwala na wykonanie typowych zadań algorytmicznych, zwłaszcza związanych z grafiką żółwia, spotykanych w szkole.

Większość z niezaimplementowanych instrukcji została pominięta celowo. Chodzi tu zwłaszcza o możliwości multimedialne - to jest odtwarzanie animacji, edycja żółwi itp. Stało się tak ponieważ autorce zależało na istocie języka, a nie możliwościach multimedialnych, które nie są implementowane na ogół także w innych wersjach interpreterów. Ponadto, porównaniu do przeznaczonych tworzenia multimediów narzędzi, możliwości obecnych środowisk Logo nie są imponujące.

8. Bibliografia

[1] Logo Foundation http://el.media.mit.edu/Logo-foundation/Logo/index.html, Dostęp: luty 2013

[2] Dokumentacja Logo Komeniusz ver. 1.1.040 OEIIZK dołączona do programu

[3] Dokumentacja Logomocja Imagine Demo ver 2.0 OEIIZK dołączona do programu

[4] Dokumentacja platformy Moodle http://docs.moodle.org/23/en/?lang=en, Dostęp: luty 2013

[5] Jacek Lewicki. *Programowanie w języku Logo od podstaw* Informatyka w Szkole, Oficyna Edukacyjna Krzysztof Pazdro, 1998, ISBN 83-85751-53-X

[6] Michael Morrison, *Head First JavaScript. Edycja polska*, Helion, 2009, ISBN 978-83-2461-548-3

[7] Alfred V.Aho, Ravi Sethi, Jeffrey D. Ullman *Kompilatory*. *Reguły, metody i narzędzia.* WNT, Warszawa 2002, ISBN 83-204-2656-1

[8] Dokumentacja ExtJS http://docs.sencha.com/ext-js/4-1/, Dostęp: luty 2013

[9] Andrzej Walat *Zarys dydaktyki informatyki.* Warszawa OEIIZK, 2007, ISBN 978-83-922946-2-7

[10] Gerd Mietzel *Psychologia kształcenia.* Gdańskie Wydawnictwo Psychologiczne, 2002, ISBN 8389120186

[11] Platforma *{α}youngcoder* http://www.youngcoder.eu/, Dostęp: maj, 2013

[12] *Młodzieżowa Akademia Informatyczna* http://main.edu.pl/pl, Dostęp: czerwiec 2013

[13] Serwis poświęcony Logomocji prowadzony przez OEIIZK http://logo.oeiizk.waw.pl/, Dostęp: czerwiec 2013

[14] Strona domowa programu Camstudio http://camstudio.org/, Dostęp: czerwiec, 2013

[15] Strona domowa programu Alligator Flash Designer http://www.flashdesigner.pl/, Dostęp: czerwiec 2013

9. Spis ilustracji