UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Vizuálny programovací jazyk pre guľového robota Sphero - aplikácia pre Android

Bakalárska práca

2015 JURAJ KARLUBÍK

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Vizuálny programovací jazyk pre guľového robota Sphero - aplikácia pre Android

Bakalárska práca

Študijný program: Aplikovaná informatika

Študijný odbor: 2511 Aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci: Mgr. Pavel Petrovič, PhD.

Bratislava, 2015 JURAJ KARLUBÍK

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Juraj Karlubík
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vizuálny programovací jazyk pre guľového robota Sphero - aplikácia pre Android
Visual Programming Language for round robot Sphero - Android Application

Cieľ: Práca nadväzuje na predchádzajúcu bakalársku prácu, v rámci ktorej vznikla prvá verzia vizuálneho programovacieho jazyka pre mobilné zariadenia Android na programovanie robota Sphero. Úlohou študenta je vylepšiť existujúcu aplikáciu, prípadne vytvoriť celkom novú - programovacie prostredie pre robota Sphero pre mobilné zariadenia Android. Očakáva sa výrazné vylepšenie jazyka z hľadiska jeho vyjadrovacej sily a prehľadnosti spracovania informácií zo senzorov. Študent by mal koordinovať úsilie s paralelne prebiehajúcou prácou zameranou na PC.

Literatúra: Marek Melicherčík: Výukové prostredie na programovanie mobilného guľového robota, bakalárska práca, FMFI UK, Bratislava, 2014.
Dan Danknick: Gen 1 orbBasic Interpreter, doc.rev.1.07, Orbotix Inc., 2013.

Kľúčové slová: mobilný robot, sphero, vizuálny programovací jazyk

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 30.09.2014

Dátum schválenia: 04.11.2014

doc. RNDr. Mária Markošová, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestné prehlásenie

Čestne prehlasujem, že túto bakalársku prácu, ktorá je jej výsledkom som vypracoval samostatne s použitím uvedenej literatúry a za odbornej pomoci vedúceho bakalárskej práce.

Bratislava 29.5.2015

.....

Poďakovanie

Chcel by som sa poďakovať mojej rodine, ktorá mi bola oporou pi písaní bakalárskej práce a takisto aj môjmu vedúcemu Mgr. Pavlovi Petrovičovi, PhD. za jeho odbornú pomoc, cenné rady a trpezlivosť pri čase strávenom na stretnutiach.

Abstrakt

KARLUBÍK, Juraj: Vizuálny programovací jazyk pre guľového robota Sphero - aplikácia pre Android [Bakalárska práca]. Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. Vedúci bakalárskej práce: Mgr. Pavel Petrovič, PhD. Bratislava: FMFI UK, 2014, 49 s.

Táto práca využíva a vylepšuje program vytvorený v predchádzajúcej verzii. Obsahuje nové grafické programovacie prostredie, ktoré nie je založené na textových príkazoch ako doteraz, ale názornejších grafických blokoch, ponúka väčšiu paletu základných príkazov a hlavne nenúti používateľa, aby požíval interné premenné jazyka OrbBasic. Funkcionalitu a vyjadrovaciu silu nového vytvoreného programovacieho jazyka úspešne demonštrujem na niekoľkých príkladoch, ktoré z hračky Sphero vytvárajú hodnotnú učebnú pomôcku vhodnú na vyučovanie základov programovania.

Kľúčové slová: grafické programovacie jazyky, mobilná aplikácia, guľový robot, OrbBasic, Android

Abstract

KARLUBÍK, Juraj: Visual Programming Language for round robot Sphero - Android Application [Bachelor work]. Comenius University. Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics. Lead by: Mgr. Pavel Petrovič, PhD. Bratislava: FMFI UK, 2014, 49 s.

This work builds on and improves the application that was developed in the previous version. The contains a new graphical programming environment. In contrast to the previous version, it is not based on textual statements. Rather it uses graphical blocks, offers greater selection of built-in commands and most importantly, it does not require the user to use the internal OrbBasic variables. The functionality and the expressive power of the programming language that I have designed is successfully demonstrated on several examples. They show that the toy Sphero has turned into a valuable educational tool that is even suitable to teach programming fundamentals.

Keywords: graphical programming language, mobile application, spherical robot, OrbBasic, Android

Obsah

1 Úvod

Každé jedno obdobie našich dejín je niečím výnimočné. 16. storočie bolo charakterizované ako obdobie objavov nových kontinentov a vynájdenie kníhtlače. Obdobie 17.-18. storočia charakterizovala hlavne priemyselná revolúcia a vynájdenie parného stroja. Naopak 20. storočie sa stalo symbolom veľkých svetových vojen, hospodárskeho úpadku ,strachu a neistôt. Čo ale vystihuje 21. storočie?

Naše storočie by sa dalo charakterizovať ako obdobie rozvoja a rozmachu informačných technológii. Či si to uvedomujeme alebo nie, táto moderná doba obklopuje a ovplyvňuje nás všetkých. Životný štýl v dnešnej dobe sa radikálne zmenil hlavne kvôli rozvoju technologických výrobkov. Viac-menej všetci už poznáme a využívame televízor, mobilný telefón, počítač alebo notebook. Nehovoriac o tom, aký veľký dopyt je po rôznych mobilných či počítačových aplikáciách, vlastných webových stránkach či rôznych školských alebo výukových programoch. Avšak len málo z nás naozaj vie, ako si takéto softvéry naprogramovať. Hlavne z týchto dôvodov sú veľmi žiaduce okrem znalosti matematiky (s ktorou sa stretávame) aj aspoň základná znalosť programovania.

Programovanie nie je len o znalostiach, ale je hlavne o logickom myslení. Preto je potrebné ho trénovať už od útleho veku, aby dieťa mohlo v budúcnosti pracovať v tejto oblasti. Avšak ako to vo všetkých vedných oblastiach býva, potrebujeme sa najprv naučiť a pochopiť základné princípy programovania. Bez nich by sme neboli schopní pochopiť náročné algoritmy, štruktúry a syntax programovacích jazykov. Preto sa vyvíjajú programy, ktoré používajú zjednodušenú syntax a prívetivé grafické prostredie na zaujatie dieťaťa. Namiesto zdrojových kódov sa často využívajú grafické prvky, ktoré "zapadnutím do seba" zjednodušene ilustrujú, čo má daný program vykonať.

Avšak aby dieťa programovanie ešte viac zaujalo, začali sa vytvárať jednoduché programovacie jazyky na ovládanie robotov. Práve kvôli tomuto faktu som sa aj ja rozhodol prispieť svojím dielom práce k vytvoreniu vizuálneho programovacieho jazyka na ovládanie guľového robota Sphero pomocou mobilnej aplikácie.

2 Prehľad problematiky

V súčasnosti neexistuje veľa grafických programovacích jazykov pre mobilné aplikácie. Preto som začal pátrať po grafických programovacích jazykoch určených pre počítače, ktorých je oveľa viac. Pri ich pozorovaní som sledoval, aké sú medzi nimi rozdiely a ktorá grafická štruktúra s danou funkcionalitou by najviac vyhovovala grafickej štruktúre určenej pre moju aplikáciu. Zameral som sa hlavne na tie programy, v ktorých celé programovanie prebieha na báze spájania rôznych útvarov do logických celkov.

Pre mobilnú aplikáciu je potrebné, aby útvary neboli veľmi malé ale ani veľmi veľké. Keďže mobilné zariadenia a tablety majú rôznu veľkosť tak je potrebné, aby dané útvary boli aj škálovateľné a aby bolo dobre vidieť spoje medzi útvarmi a celkami.

Na programovanie aplikácie som si vybral operačný systém Android, pretože je to najrozšírenejší operačný systém, ktorý využíva väčšina vlastníkov smartfónov na Slovensku aj vo svete.

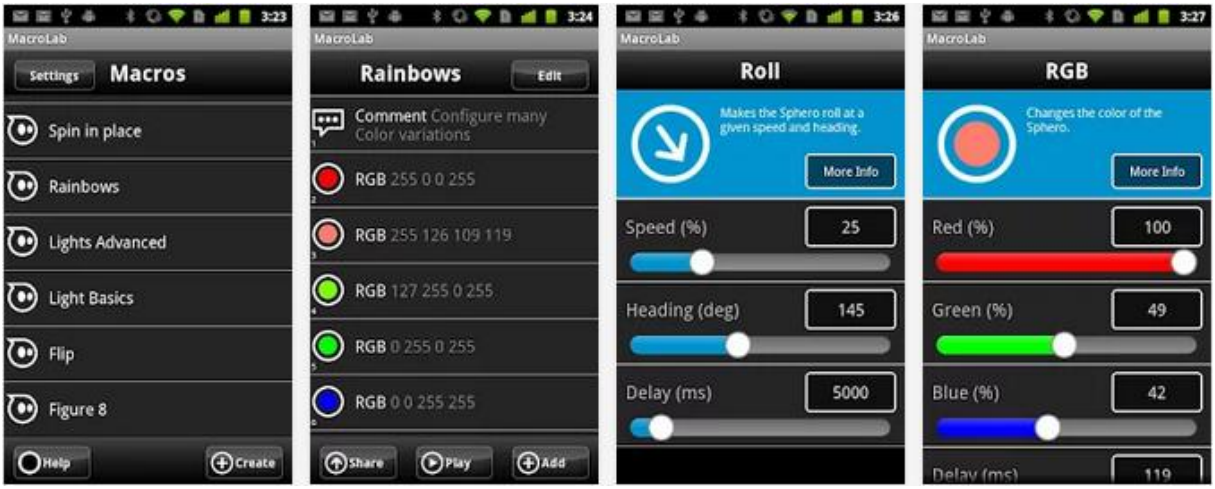
2.1 Grafické programovacie jazyky

Hlavný obchod s mobilnými aplikáciami - Google Play ponúka veľmi málo riešení grafických mobilných aplikácii. Z nich som do porovnávania vybral iba 1, nakoľko viacej riešení som nebol schopný nájsť. Preto som sa zameral hlavne na grafické programovacie jazyky pre počítače. Vyskúšal som viacero grafických programovacích jazykov a každý mal svoje špecifiká. Niektoré využívali výhradne spôsob drag-and-drop (presun a polož), niektoré iné mali v sebe implementovaný malý príkazový riadok, kde bolo okrem iného možné písať jednoduché a aj zložitejšie príkazy vytvárajúce útvar na grafickej ploche. Taktiež bolo možné naraziť na aj na kombináciu týchto prvkov.

Všetky nasledovné podkapitoly budú o programovacích jazykoch, ktoré slúžia na programovanie nejakého robota, nejakého virtuálneho počítačového robota (korytnačka), alebo na graficky zjednodušené programovanie mobilných Android aplikácii.

2.1.1 MacroLab - Sphero

MacroLab je aplikácia od tvorcov guľového robota Sphero. Táto aplikácia obsahuje súbor preddefinovaných makier (macros) a využíva programovací jazyk OrbBasic, ktorý je založený na programovacom jazyku BASIC.



Obr. 1 Programovacie prostredie aplikácie Macrolab [1]

Ako si môžeme všimnúť, tento programovací jazyk využíva možnosti grafického programovania na sekvenčnom prístupe. To znamená že si zvolíme, aké udalosti budú nasledovať za sebou a podľa toho sa zoradia na obrazovke pod sebou. Kým jedna skončí, tá pod tým začne a takto až po posledný príkaz.

Z hľadiska vyjadrovacej sily jazyka toho Macrolab veľmi nedokáže. Nie je tu možné zadávať premenné iba ako parametre k rôznym funkciám. Všetky premenné k funkciám sú zapísané v celých číslach, a percentá sa automaticky zaokrúhľujú na celé čísla. Čo sa týka textov, je možné ich použiť v komentároch.

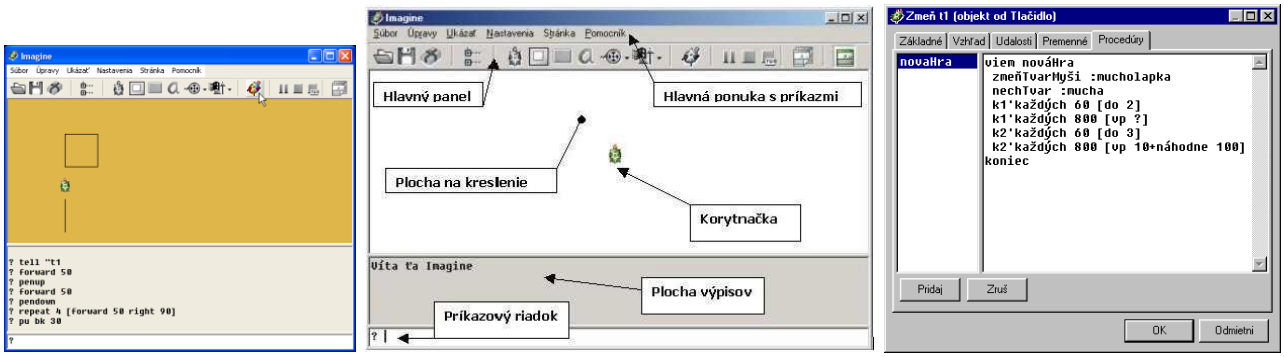
Neumožňuje pracovať ani s poliami, pretože žiadna vopred naprogramovaná funkcia ho nepoužíva. Takisto o paralelizme sa tiež nedá hovoriť, pretože vykonáva presne jednu inštrukciu za druhou. Nepodporuje ani podmienky ani while cyklus, maximálne sa tu dá použiť niekoľkonásobné opakovanie nejakých funkcií.

Táto aplikácia je dostupná v Google Play aj v iTunes Store (mobilné aplikácie pre iPhone). Osobne si myslím, že daný programovací jazyk je veľmi dobre navrhnutý na prácu s guľou pre malý počet príkazov, avšak pri zložitejších aplikáciách sa mi zdá byť nepoužiteľný z

dôvodu neprehľadnosti príkazov. Väčšina príkazov sa nezmestí na obrazovku a tým pádom nie je dobre vidieť štruktúru programu. Avšak na tvorbu jednoduchých aplikácii pre netechnických používateľov je perfektný.

2.1.2 Imagine Logo

Je to jazyk, ktorý vznikol v roku 2001. Je nepriamym nasledovníkom Comenius Loga. Je to kompletne objektový jazyk pre počítače, ktorý je riadený udalosťami. Podporuje paralelné programovanie a tiež má prepracovanú ideu obrázkových tvarov korytnačiek [2].



Obr. 2 Programovacie prostredie Imagine Logo

Tento grafický programovací jazyk je určený pre študentov a učiteľov základných a stredných škôl a momentálne je na našich slovenských školách ešte stále celkom populárny.

Môžeme si všimnúť, že tento grafický programovací jazyk funguje takmer iba na základe príkazov, ktoré sa zadávajú v príkazovom riadku alebo definujú v procedúre.

Z hľadiska vyjadrovacej sily jazyka využíva Imagine veľkú množinu možností. Je tu možné zadať ľubovoľné premenné a tým pádom je ich dostatočný počet a pri väčších programoch nie je problém si pomenovať premenné podľa udalosti ktorá sa programuje. Premenné je možné zadať v rôznych typoch od Integer-u, Real-u až po String.

S poliami sa tu bohužiaľ pracovať nedá (je tu možné využiť zoznam) ale za to čo sa týka vnútorných cyklov a rekurzií, dokáže Imagine vykresľovať rekurzívne funkcie do najmenších detailov. Rozhodne tu nie je žiadny ostrý limit, ktorý by vadil pri danom programovaní.

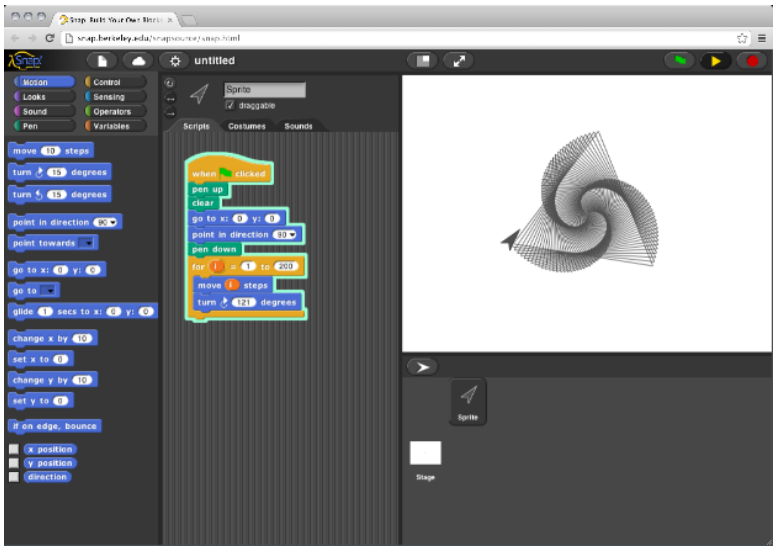
Výhodou je, že podporuje paralelné programovanie.

S týmto programom mám dlhoročnú skúsenosť zo strednej školy. Osobne môžem povedať, že daný spôsob programovania je výborný začiatok na rozvíjanie algoritmického myslenia. V

programe je možné kresliť na základe korytnačky, ktorá kreslenie realizuje. Takisto je tu možné rôzne prefarbovať prostredie, dokonca je možné aj vytváranie hier (a z vlastnej skúsenosti aj rozprávok :)). Avšak myslím, že tento štýl grafického programovacieho jazyka nie je vhodný na použitie v mobilnej aplikácii.

2.1.3 Snap!

Snap!(predtým BYOB) je vizuálny, drag-and-drop programovací jazyk. Jedná sa o rozšírenú reimplementáciu Scratch (projekt Lifelong Kindergarten Group na MIT Media Lab), ktorý vám umožní vytvoriť si vlastné bloky. Je tiež vybavený zoznamami a procedúrami. Tieto pridané funkcie sú dobrým základom do programovania na strednej alebo vysokej škole. [3]



Obr. 3 Programovacie prostredie jazyka Snap! [3]

V tomto programovacom jazyku sa programuje výhradne v blokoch. Všetky kompatibilné bloky do seba v programe pekne zapadajú.

Výhodou Snapu je, že sú blokovo štruktúrovane pekne farebne odlíšené, vytvárajú prehľadnú štruktúru kódu. Bloky sú pekne graficky spracované, aby sa aj dieťa vedelo naučiť na základe blokov programovať. Program takisto ako v Imagine Logo obsahuje grafickú plochu, kde korytnačku reprezentuje šípka.

Nevýhodou je, že tu nie je možné programovať pomocou príkazového riadku. Takisto pri zložitejších implementáciách môže vznikať v kóde zmätok, keďže môžu byť bloky rôzne cez seba poprehadzované.

Z hľadiska vyjadrovacej sily jazyka toho dokáže Snap naozaj veľa. Je tu možné zadať ľubovoľné premenné a tým pádom je ich dostatočný počet a pri väčších programoch nie je problém si pomenovať premenné podľa udalosti ktorá sa programuje. Premenné je možné zadať v rôznych typoch od Integer-u, Real-u až po String.

Takisto je tu možné pracovať s poliami, ktoré sú dynamické a kde jednoduchým kliknutím sa nám vyčlení nové miesto v poli. Čo sa týka vnútorných cyklov a rekurzí, Snap dokáže vykreslovať rekurzívne funkcie do najmenších detailov. Rozhodne tu nie je žiadny ostrý limit, ktorý by vadil pri danom programovaní.

Avšak ako jeden z mála grafických programovacích jazykov, podporuje paralelizmus čím ďalekosiahle zvyšuje efektivitu pri počítaní a vykresľovaní náročných rekurzívnych obrázkov.

Celkovo je Snap! veľmi dobrou variantou na oboznámenie sa so základmi programovania a myslím, že štruktúra blokov je celkom dobrý spôsob, ktorý by sa dal aplikovať aj v mobilnej aplikácii.

2.1.4 MIT App Inventor

MIT App Inventor je blokovo založený programovací nástroj, ktorý umožňuje každému, dokonca aj začiatočníkom, aby začali programovať a vytvárať plne funkčné aplikácie pre zariadenia so systémom Android. Nováčikovia na App Inventore môžu mať ich prvé aplikácie naprogramované do 1 hodiny. Pomocou MIT App Inventoru je jednoduchšie vytvárať zložitejšie aplikácie a za podstatne kratšiu dobu, než s tradičnými, textovo založenými jazykmi [4].

Obr. 4 Programovacie prostredie jazyka MIT App Inventor

MIT App Inventor je veľmi dobrou voľbou pre každého, kto si chce naprogramovať Android aplikáciu rýchlo a jednoducho bez zbytočného sťahovania rôznych nástrojov na vytváranie Android aplikácií.

Výhodou tohto softvéru je, že je iba blokovo založený a teda že je užívateľsky veľmi prívetivý. Čo je ale veľmi vhodné, obsahuje kopec rôznych automaticky naprogramovaných tlačidiel, vďaka ktorým sa veľmi jednoducho a rýchlo dá pozrieť rozloženie aplikácie na virtuálnej ploche mobilného telefónu. Tiež je tu rýchla možnosť otestovania aplikácie na mobilnom zariadení pomocou káblu.

Nevýhodou je, že je problematické počítať v programe rôzne zložité funkcie a že sme relatívne dosť obmedzení dizajnom tlačidiel a ďalších komponentov. Chýba tu možnosť vytvárania a pridávania nových komponentov k aplikácii. Taktiež ako pri Snap!-e, je pri zložitejších programoch veľmi ťažké sa v blokoch vyznať.

Z hľadiska vyjadrovacej sily jazyka sa dá v MIT App Inventore používať veľa možností. Je tu možné deklarovať ľubovoľné premenné a aj premenné je možné zadať v rôznych typoch od Integer-u, Real-u až po String.

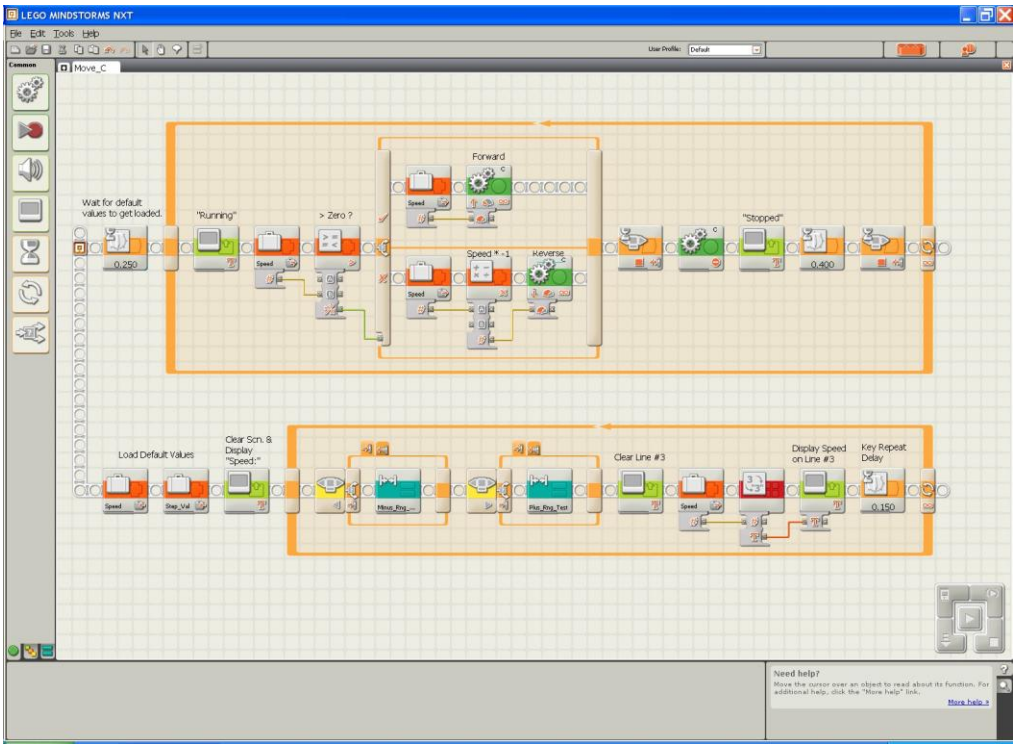
Pracovať s poliami tu nie je možné, ale dá sa použiť list prípadne list listov, ktoré fungujú podobne ako pole. Čo sa týka vnútorných cyklov a rekurzí, MIT App inventor nemá obmedzenia na cykly, mierne obmedzuje hĺbku rekurzie.

Čo sa týka paralelizmu, je tu taktiež podporovaný.

Myslím že MIT App Inventor je jeden z najlepších nástrojov na rýchle vytváranie Android aplikácií, a určite si ho každý začiatočník do programovania Android aplikácií obľúbi.

2.1.5 NXT-G

NXT-G je programovací jazyk určený na programovanie robota LEGO Mindstorms NXT. Robot je postavený zo stavebníc LEGO a je dodávaný s výbavou niekoľkých senzorov: dotykový, zvukový, svetelný a ultrasonický. Jednotlivé funkcie programovacieho jazyku NXT-G sú reprezentované grafickými blokmi, na ktorých je ikona funkcie, ktorá reprezentuje to čo by mala vykonávať. Na bloku funkcie sú taktiež všetky vstupné premenné, ktoré sú tak isto reprezentované ikonkami [5].



Obr. 5 Prostredie LEGO Mindstorms NXT

Všetky funkcie sú v editore v ľavom menu, v ktorom sú pekne farebne odlíšené a zaradené do jednotlivých kategórií, podľa ktorej im je pridelená farba. Kód je v konečnom dôsledku veľmi prehľadný z hľadiska grafického riešenia cyklov a vetvenia v podmienkach. Jednotlivé časti, vetvy kódu sa do seba nezlievajú a nie je problém nájsť, kde nejaká funkcia začína a končí. Nevýhodou však je, že kód ide dosť do šírky a jednotlivé bloky sú dosť veľké, čo znamená, že sa ho veľa na pracovnú plochu, obrazovku nezmestí.

Z hľadiska vyjadrovacej sily má tento jazyk tri typy premenných: číslo, logická hodnota a text. Premenné je možné vytvoriť podľa potreby a používať ich na ukladanie hodnôt, alebo aj aritmetické, logické výpočty alebo spájanie textov. Premenné dobre slúžia na výmenu údajov medzi paralelnými procesmi, procedúrami alebo rôznymi časťami programu.

V NXT-G je možné zapisovať/čítať do/zo súboru a vytvárať tak záznam meraní, pripraviť si konfiguračné parametre alebo ukladať pracovné, vstupné a výstupného hodnoty.

Pracovať s poliami tu tiež nie je umožnené. Čo sa týka vnútorných cyklov a rekurzí, rekurzia sa tu nedá využiť, je možné tu použiť podmienky a iné rozvetvovanie od rôznych senzorov. Čo sa týka cyklov, je možné opakovať dokiaľ nenastane požadovaný stav na nejakom senzore, prípadne využiť vlastné naprogramované logické podmienky využívajúce

premenné, konštanty, alebo hodnoty zo senzorov, ktoré určia podmienku ukončenia opakovania cyklu.

V NXT-G je možné používať paralelizmus, jeden z príkladov je na Obr. 5.

2.1.6 Porovnanie grafických jazykov

Tabuľka 1 Porovnanie grafických jazykov

	Macrolab	**Imagine Logo**	**Snap!**	**MIT App Inventor**	**NXT-G**
Premenné	nie	áno	áno	áno	Nie
Polia	nie	Nie (má zoznam)	áno	nie (list áno)	Nie
Podmienka IF	nie	áno	áno	áno	áno + senzory
Cykly	repeat	while, repeat	while, for	while, for	Repeat
Rekurzia	áno (iba celého programu)	áno	áno	áno	Nie
Hĺbka vnoreného cyklu/ rekurzie	max 4/ neobmedzená	neobmedzená/ neobmedzená	neobmedzená/ neobmedzená	neobmedzená/ obmedzená	neobmedzená/-
Paralelizmus	nie	áno	áno	áno	áno

2.2 Operačný systém Android

Na programovanie mobilnej aplikácie som sa musel rozhodnúť, ktorý operačný systém zvoliť. Existujú totiž iba dva operačné systémy, ku ktorým je vytvorené API na programovanie robota Sphero. Buď iOS od spoločnosti Apple, alebo Android od spoločnosti Google.

Pri výbere operačného systému som sa zameral na jeho obľúbenosť, dostupnosť zariadení s daným systémom a programovací jazyk, v ktorom sa robia dané aplikácie. Na základe týchto dôležitých faktorov som sa jednoznačne rozhodol pre Android. Je to aj vďaka tomu, že vlastním iba Android zariadenia.

Operačný systém Android nájdeme na mobilných zariadeniach takmer každého výrobcu a väčšinou aj na všetkých jeho modelových radách. iOS nedisponuje takým množstvom úžívateľov ako Android. Na druhej strane, používatelia s iOS sú ochotní za

aplikácie zaplatiť rádovo viac ako používatelia Androidu. Ja som dal prednosť Androidu, pretože mi ide skôr o čo najväčší dopyt po aplikácii.

Na iOS ma odradilo tiež to, že nemám absolútne žiadne skúsenosti s týmto softvérom. Android obsahuje obrovské množstvo návodov, kníh a e-bookov, ktoré nám môžu pomôcť pri programovaní. Tak isto môžeme nájsť obrovský počet fór, kde sa diskutuje o rôznych problémoch pri vytváraní Android aplikácií.

Na záver chcem dodať, že by bolo veľmi obťiažne naväzovať na bakalársku prácu pod iným operačným systémom, pod akým bola pôvodne vypracovaná.

3 Robot Sphero

Sphero je revolučná magická žiariaca guľa od spoločnosti Orbotix vo veľkosti tenisovej loptičky. Obsahuje gyroskop, ktorý jej zaručuje možnosť pohybu.

Ovládanie robotickej gule prebieha nakláňaním, dotýkaním a vlnením pomocou smartfónu či tabletu s operačným systémom Andoid alebo iOS. Sphero predstavuje novú dimenziu zábavy, v ktorej sa ponoríte do sveta neuveriteľne zábavných hier. Sphero vyniká kompaktnými rozmermi, vďaka ktorým sa pohodlne zmestí do každej tašky či do vrecka bundy a môžete ho tak nosiť všade so sebou [6].

Sphero bol navrhnutý a založený v roku 2010 dvojicou amerických inžinierov Ianom Bernsteinom a Adamom Wilsonom. Cena sa v dnešnej dobe v pohybuje okolo 100-115€. Sphera je možné dostať aj cez slovenské internetové obchody.

3.1 Ovládanie

Sphero sa so zariadením spojí pomocou technológie Bluetooth, vďaka ktorej spoľahlivosti je potom pripojenie vždy citlivé a plynulé, a to dokonca až na vzdialenosť 30 metrov. Sphero je možné používať aj vo vode. Vďaka vysoko odolnej polykarbonátovej škrupine zvládne aj plávať na hladine vody.

Sphero je vybavené mnohofarebnou LED technológiou, ktorá umožňuje meniť jeho farbu. O napájanie sa stará Lí-Pol batéria, ktorá poskytuje na jedno nabitie viac ako hodinu prevádzky.

3.2 Stavba robota a jeho parametre

Na základe [1] vyzerá stavba Sphera nasledovne:

Obr. 6 pohľad na robota zvonku aj zvnútra

A) Nabíjacia stanica

Na nabíjanie Sphera sa používa nabíjacia stanica využívajúca indukčný systém na prenos elektrického prúdu do 2 lítium-polymérových batérií robota. Nabíjanie trvá 3 až 4 hodiny. Batéria vydrží približne 75 minút počas aktívneho používania robota.

B) Kolesá

Robot obsahuje 2 kolesá nezávisle od seba poháňané dvoma motormi. Pomocou nich je možné otáčanie a pohyb do priestoru s maximálnou rýchlosťou až 2 metre za sekundu. Kolesá sú pogumované aby bránili treniu a prešmykovaniu s obalom.

C) Horné ložisko

Slúži na stabilizáciu robota v obale. Bez tohto ložiska by sa mohol robot voľne pohybovať a tým aj poškodiť základnú dosku. Taktiež je ložisko dôležité na to, aby boli kolesá v neustálom kontakte s obalom robota.

D) Plošné spoje

[1] Melicherčík, M.: Výukové prostredie na programovanie mobilného guľového robota, bakalárska práca, FMFI UK, Bratislava, 2014, str. 16-17

Tu procesor zbiera dáta z 3 osí akcelerometra a dáta z gyroskopu. Tie slúžia následne na vypočítavanie presného otočenie robota a taktiež detekciu kolízii. Z výsledkov týchto výpočtov môžeme naprogramovať správne reakcie robota.

E) Bluetooth vysielač a prijímač

Tento systém umožňuje bezdrôtovú komunikáciu s robotom. Dosah je približne 20 až 30 metrov v ideálnych podmienkach a slúži na komunikáciu robota so zariadením.

F) Farebné LED

Robot môže žiariť jednou z kombinácie intenzity trojice RGB farieb (červená, modrá, zelená). Vďaka tomu sme schopný zobraziť ľubovoľnú farbu rôznej intenzity jasu. Nájdeme tu však aj jednu modrú LED diódu, ktorá slúži najmä na kalibráciu smeru robota.

Parametre:

- ovládanie cez Bluetooth – dosah 30 metrov
- rýchlosť – Sphero 1.0 -> cca 1 m/s
  Sphero 2.0 -> cca 2 m/s
- počet farieb – 16 miliónov
- dĺžka nabíjania – 3 hodiny
- výdrž na 1 nabitie – 1 hodina
- vode odolné
- odolné voči vonkajším vplyvom

3.3 Príslušenstvo

a) Sphero Nubby Cover

Sphero Nubby Cover je vodeodolný kryt pomocou ktorého je možné jazdiť so Spherom v ťažkých terénoch ako je štrk, voda či betón. Nubby Cover je dostupný v modrej, zelenej, oranžovej, čiernej alebo priehľadnej farbe [7].

b) Sphero Turbo Cover

Sphero Turbo Cover chráni guľu pred nepriaznivým počasím. Hladký povrch obalu umožňuje Spheru dynamickejšiu jazdu. Poskytuje neprekonateľnú priľnavosť na všetkých typoch terénu. Chráni pred poškriabaním a odieraním. Je dostupný v červenej, čiernej a tyrkysovej farbe [7].

c) Sphero Chariot

Sphero Chariot je dvojkolesový plastový vozík. Obsahuje držiak mobilného telefónu, do ktorého môžete umiestniť ľubovoľný telefón s fotoaparátom alebo kamerou. Povrch vozíka je kompatibilný so stavebnicami LEGO a preto je možné použiť zaujímavú nadstavbu. S vozíkom môžete jazdiť ako s autom na diaľkové ovládanie [7].

d) Sphero Terrain Park

Sphero Terrain Park je súprava rámp a iných skateboardových terénov určený na trénovanie až piatich trikových konfigurácii. Je upravovateľný a slúži na trénovanie ovládania robota [7].

3.4 OrbBasic

OrbBasic je programovací jazyk určený výhradne na programovanie robotickej gule Sphero. Je to celkom dobre obsiahnutý jazyk a dokážeme v ňom tvoriť relatívne zložité programy. Skladá sa z niekoľkých riadkov. Každý riadok je očíslovaný číslom násobku 10-tky, následne medzera a za ním už daný príkaz [8].

OrbBasic obsahuje celkovo 82 premenných, z ktorých 32 sú systémové slúžiace primárne na čítanie a zvyšných 50 je voľne dostupných. Z voľne dostupných premenných je 25 priamo dostupných a 25 dostupných pomocou indexu:

- Priamo dostupné premenné sú označené písmenami A..Y (alebo a..y ak preferujete)
- Z (alebo z) je indexová premenná a je možné jej priradiť hodnotu len vtedy, ak je určená hodnota indexu v Y ktorá zavolá premennú Z(Y). Keďže Basic neobsahuje žiadne nulové indexy, tak rozsah premennej Y pri prístupe premennej Z je 1..25.

```
Zapísanie hodnoty 255 do indexovej premennej Z(8)
10 Y = 8
20 Z = 255
```

Obr. 7 Príklad z OrbBasica

Všetky tieto premenné sú 32-bitové celočíselného typu *Integer* v rozsahu od -2,147,483,647 do +2,147,483,647. OrbBasic nepodporuje premenné s desatinnými číslami, ale je možné použiť aritmetiku s pevnou desatinnou čiarkou[2] [8].

Nie je tu podpora pre reťazce (string-y) inak ako v dočasnom prístupe cez príkaz PRINT.

Prehľad všetkých premenných uvádzam v nasledujúcej tabuľke.

Tabuľka 2 Prehľad premenných jazyka OrbBasic [9]

Názov	**Prístup**	**Veľkosť**	**Popis**
A...Y (a...y)	r/w	32-bit signed value	bežné premenné na zapisovanie a čítanie
Z(Y=1)...Z(Y=25) (z(Y=1)...z(Y=25))	r/w	32-bit signed value	indexové premenné na zapisovanie a čítanie
timerA, timerB, timerC	r/w	16-bit positive value	časovače
ctrl	r/w	Boolean	stav kontrolného systému
speed	r	8-bit unsigned value	rýchlosť
yaw	r	16-bit unsigned value	otočenie robota yaw
pitch	r	16-bit unsigned value	otočenie robota pitch
roll	r	16-bit unsigned value	otočenie robota roll
accelX, accelY, accelZ	r	16-bit signed value	akcelerometer v osiach
gyroX, gyroY, gyroZ	r	16-bit signed value	gyroskop v osiach
Vbatt	r	16-bit unsigned value	napätia batérie
Sbatt	r	16-bit unsigned value	stav batérie
cmdroll	r	Boolean	detekcia roll príkazu
spdval	r	8-bit unsigned value	posledná rýchlosť
hdgval	r	16-bit unsigned value	posledné otočenie
cmdrgb	r	Boolean	detekcia RGB príkazu
redval, grnval, bluval	r	8-bit unsigned value	posledné farby
isconn	r	Boolean	stav pripojenia robota
dshake	r	Boolean	detekcia dvojitého zatrasenia
accelone	r	16-bit unsigned value	akceleračný vektor
xpos, ypos	r	16-bit signed value	pozícia
Qzero, Qone, Qtwo, Qthree	r	16-bit signed value	quaternion

Cykly je možné v OrbBasicu definovať dvoma spôsobmi:

- konštrukciou: *" for <loop variable> = <start value> to <end value> step <step value> ... next <loop variable> "*

[2] http://en.wikipedia.org/wiki/Fixed-point_arithmetic

- pomocou príkazov: *"goto/gosub <line number>"*

Príkaz FOR je použitý teda pri klasickom for-cykle. Táto konštrukcia má jednu nevýhodu, pretože nepustí viac ako 4 vnorené cykly. Kvôli väčšej bezpečnosti pri práci s programom, používame iba 3 úrovne cyklu, kde premenné počítadla nazývame i, j a k.

GOTO príkaz slúži na skočenie do určitého riadku. Na rozdiel od GOSUB, GOTO neobmedzuje maximálnu možnú úroveň vnorenia. GOSUB sa vie vnoriť maximálne 4x. Preto v programe používame iba konštrukciu GOTO. Príkaz GOTO je užitočný najmä na definovanie funkcie WHILE a IF, pretože v OrbBasicu neexistuje príkaz na WHILE alebo IF. Z tohto dôvodu sú tieto funkcie naprogramované pomocou príkazu GOTO.

Pri práci s premennými a dosadzovaním hodnôt do funkcií, cyklov a podmienok môžeme využívať aj výrazy. Ich prehľad je v nasledujúcej tabuľke.

Tabuľka 3 Prehľad operátorov jazyka OrbBasic [9]

Operátor	Priorita	Popis
(	vysoká	začína nový podvýraz
)	vysoká	ukončuje podvýraz
*****	medium	celočíselné násobenie
/	medium	celočíselné delenie
%	medium	zvyšok po celočíselnom delení
{	medium	binárny posun doľava
}	medium	binárny posun doprava
+	low	sčítanie
-	low	odčítanie
&	low	bitový AND
|	low	bitový OR

OrbBasic takisto disponuje rôznymi funkciami. Obsahuje matematické funkcie, funkcie na ovládanie robota a funkcie, ktoré pracujú priamo s aplikáciou v zariadení. Prehľad všetkých funkcii si môžete prezrieť v nasledujúce tabuľke.

Tabuľka 4 Prehľad funkcií jazyka OrbBasic [9]

Funkcia	Prístup	Veľkosť	Popis
sqrt x	r	32-bit unsigned value	odmocnina
rnd x	r	32-bit unsigned value	náhodná hodnota
abs x	r	32-bit unsigned value	absolútna hodnota
print	-	-	vypísanie správy
delay x	param	16-bit unsigned value	čakanie programu
end	-	-	ukončenie programu
RGB r,g,b	param	8-bit unsigned values	farba RGB osvietenia

LEDC x	param	8-bit unsigned value	farba osvietenia
backLED x	param	8-bit unsigned value	zadná LED dióda
goroll h,s,g	param	16,8,8-bit unsigned value	pohyb robota
random	-	-	spustenia rand. generátora
heading h	param	16-bit unsigned value	otočenie robota
raw *Lmode, Lspeed, Rmode, Rspeed*	param	4,8,4,8-bit unsigned value	pohyb ľavého a pravého motora
locate x,y	param	32-bit signed value	nastavenie polohy
data d1{,d2,d3}	param	32-bit signed values	množina Integerov
rstr	-	-	nastavenie indexu čítania množiny na začiatok
read X{,Y...}	param	32-bit signed values	číta hodnotu z množiny
tron	-	-	zapnutie sledovania riadkov
troff	-	-	vypnutie sledovania riadkov
reset	-	-	znova spúšťa program
sleep *duration, macro, line_number*	param	16,8,32-bit unsigned value	uspáva robota
macrun x	param	8-bit unsigned value	spúšťa makro
mackill	-	-	zastavuje makro
macstat	r	32-bit unsigned value	vracia informácie o vykonávanom makre

Nevýhoda OrbBasic-u je aj v tom, že tu nefunguje paralelizmus (nemôže sa paralelne vykonávať 2 a viac vetiev).

Vysvetlenie hlavných funkcii OrbBasica:

print (print "Hello world") – vypíše danú textovú správu do zariadenia. Nie je možné vypisovať premenné [8].

delay (delay ms) – počká zadaný čas a až po ňom sa presunie na ďalší príkaz. Obsahuje 1 parameter označujúci počet milisekúnd [8].

end – ukončí celý program. Funguje rovnako ako príkaz "exit" v Pascale [8].

RGB (RGB r,g,b) – nastaví farbu osvetlenia robota na hodnotu RGB. Má tri parametre, číselné hodnoty od 0...255 reprezentujúce intenzitu svietenia 3 farieb (červená, zelená, modrá) RGB spektra [8].

goroll (goroll h, s, g) – funkcia slúžiaca na pohyb robota. Prvý parameter je smer. Je v rozmedzí 0...359 a určuje smer robota na pohyb vzhľadom na vykalibrované natočenie. Druhý parameter je rýchlosť, hodnota medzi 0...255. Posledný parameter môže mať tri hodnoty. Pri 0 robot zastavuje. Ak je hodnoty 1 tak sa pohybuje normálne a ak 2 tak má prudkejšie pohyby najmä v zákrutách [8].

heading (heading h) – otáčanie robota do daného smeru. Parameter je číslo v rozmedzí 0...359 reprezentujúce žiadané výsledné natočenie robota [8].

goto (number) – funkcia umožňujúca skákať po riadkoch programu. Vďaka nej je možné programovať podmienky aj cykly pod platiacou podmienkou [8].

raw (Lmode,Lspeed,Rmode,Rspeed) – pomocou tejto funkcie je možné prevziať pohyb robota do vlastných rúk bez pomoci stabilizátora. Pomocou tejto funkcie môžeme robotovi nastaviť smer pohybu ľavého a pravého motora a takisto aj príslušnú rýchlosť obidvoch motorov. Tento príkaz je veľmi vhodný na crazy funkcie ako skákanie či otáčanie sphera vysokou rýchlosťou na mieste. Takisto sa dá pomocou tohto príkazu ľahšie naprogramovať kruh [8].

locate (x,y) - nastavuje robotovi základné súradnice. Ak robotom pohneme, tak novo získané súradnice môžme získať z premenných **xpos** a **ypos** [8].

```
10 locate 0,0
20 delay 100
30 R = sqrt (xpos*xpos + ypos*ypos)
40 if R < 20 then LEDC 1 else LEDC 2
50 goto 20
```

Obr. 8 Ukážka programu v jazyku OrbBasic [8]

4 Analýza predchádzajúcej verzie riešenia

Základom mojej aplikácie je riešenie minuloročnej bakalárskej práce [9] z dôvodu čo najlepšieho rozšírenia grafického programovacieho jazyka a čo najlepšie využitie OrbBasic funkcii. V nasledujúcich podkapitolách nájdete krátke zhrnutie predchádzajúcej práce, ktoré je dostupné v [9].

4.1 Pripojenie sphera k aplikácii

Ako základ aplikácie sa vychádzalo z Android SDK, ktoré pripravila spoločnosť Orbotix pre vývojárov. Je možné ju stiahnuť na [10].

Obsahom balíka je knižnica napísaná v Jave. Obsahuje návod na obsluhu robota, kompletnú dokumentáciu všetkých jej funkcií a tried a niekoľko jednoduchých aplikácií na pochopenie každej technológie vytvárania aplikácie.

Na to, aby sme mohli komunikovať s robotom, potrebujeme od aplikácie povolenie používať bluetooth. Táto požiadavka sa zapisuje do manifestu aplikácie.

```
<manifest ... >
  <uses-permission android:name="android.permission.BLUETOOTH" />
  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
  ...
</manifest>
```

Obr. 9 Požiadavka na používanie bluetooth v AndroidManifest.xml [9]

Sphero Android API obsahuje komponent, ktorý sa stará o pripojenie robota k ovládaciemu zariadeniu. Nazýva sa SpheroConnectionView a zobrazí všetky dostupné roboty cez bluetooth a kliknutím na príslušný názov sa nadviaže spojenie s robotom. Pri pridávaní tohto komponentu nám stačí vložiť krátky kód do layoutu aplikácie:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff888888" >

    <orbotix.view.connection.SpheroConnectionView
        android:id="@+id/sphero_connection_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#FFF" />

</LinearLayout>
```

Obr. 10 pripájanie SpheroConnectionView do layoutu [9]

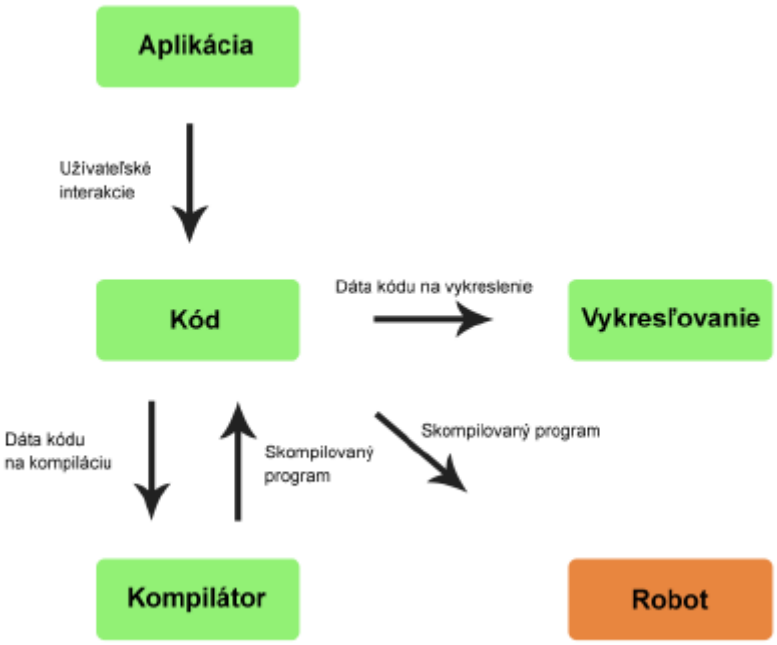
Po pripojení robota a jeho priradení do premmennej už s ním môžeme pracovať.

```
mSpheroConnectionView = (SpheroConnectionView) findViewById(R.id.sphero_connection_view);
mSpheroConnectionView.addConnectionListener(new ConnectionListener() {
        @Override
        public void onConnected(Robot robot) {
                sphero = (Sphero) robot;
        }
        @Override
        public void onConnectionFailed(Robot sphero) {
                // pripojenie zlyhalo ...
        }
        @Override
        public void onDisconnected(Robot sphero) {
                mSpheroConnectionView.startDiscovery(); //spustime vyhladavanie zariadeni
        }
});
```

Obr. 11 Inicializácia robota v logike aplikácie [9]

4.2 Štruktúra zdrojového kódu

Aplikácia z hľadiska komunikácie a toku dát funguje na tomto princípe:

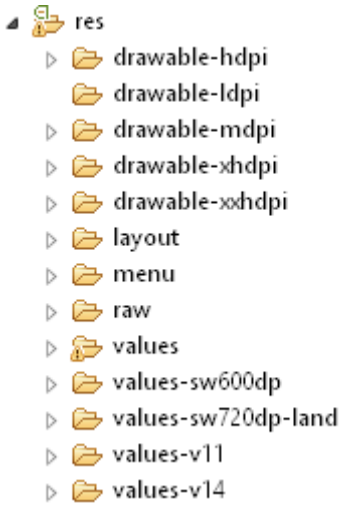


Obr. 12 Diagram z hľadiska toku dát a komunikácie [9]

Ako si môžeme všimnúť, z aplikácie sa pomocou užívateľských interakcii vytvára kód ktorý sa automaticky vykresľuje na plochu aplikácie. Pri spustení programu sa pošle kód do kompilátoru, kde sa daný kód prevedie do Orbbasic-u. Orbbasic-ový skompilovaný program sa naspať pošle do kódu a odtiaľ sa priamo pošle do robota, ktorý daný kód vykoná.

Štruktúra aplikácie je postavená na Android frameworku-u. Je rozdelená do niekoľkých priečinkov, knižníc a súborov ktoré dávajú dohromady 1 celok aplikácie (Obr. 13).

Obr. 13 Štruktúra aplikácie



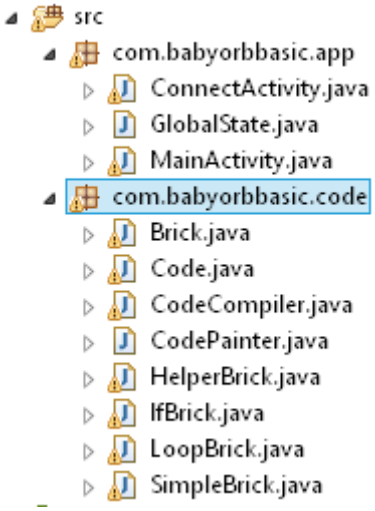
Obr. 14 Štruktúra priečinka /res

Veľmi dôležitým priečinkom je priečinok /res (Obr. 14). Obsahuje v sebe všetky dôležité obrázky použité v aplikácii pre rôzne rozlíšenia displeja v priečinkoch /drawable-... . Takisto obsahuje /layout , v ktorom sú zhrnuté všetky screeny v aplikácii a o každom pridanom elemente si pamätá layout jeho pozíciu, typ a veľkosť. Ďalším používaným priečinkom je /menu obsahujúce XML súbory slúžiace na tvorbu menu položiek. Posledným dôležitým prvkom je priečinok /values v ktorom sa uchovávajú hlavne hodnoty reťazcov ale aj Android tém.

Potom sa tu nachádza priečinok /libs, v ktorom sú uložené všetky potrebné knižnice na prácu s robotom.

Najviac nás ale bude zaujímať priečinok /src, v ktorom sa nachádza kompletný zdrojový kód chodu aplikácie. Ako si môžeme všimnúť (Obr. 15), tak zdrojový kód je rozdelený do 2 balíkov:

- *com.babyorbbasic.app* (predtým *com.melichercik.app*)
- *com.babyorbbasic.app* (predtým *com.melichercik.code*)



Obr. 15 Zdrojový kód v priečinku /src

4.2.1 Balík com.babyorbbasic.app (com.melichercik.app)

V tomto balíku sa nachádzajú triedy aktivít, ktoré sa starajú o všetky užívateľské interakcie s aplikáciou a o nadviazanie spojenia so Spherom.

ConnectActivity slúži na vytvorenie spojenia s robotom pomocou zavolania triedy z knižnice *SpheroConnectionView* a po pripojení na nastavovanie preddefinovaného smeru pohybu robota. Takisto pomocou tejto triedy je možné uspať robota.

GlobalState slúži iba na definovanie globálnej premmennej *robot*. Je to preto, aby sa nám počas práce s robotom neprerušovalo bluetooth spojenie pri používaní aplikácie.

MainActivity je hlavná aktivita aplikácie starajúca sa o všetky užívateľské interakcie v okne programovacieho jazyka. Reaguje na všetky stlačenia ktoré sa vykonávajú pri tvorení užívateľského kódu. Súčasťou je aj trieda *RenderView* pomocou ktorej sa spúšťa vlákno starajúce sa o neustále vykresľovanie plochy z dôvodu možného škálovania vytvoreného kódu v aplikácii.

4.2.2 Balík com.babyorbbasic.code (com.melichercik.code)

Tento balík obsahuje triedy, ktoré sa týkajú vytvárania, upravovania, kompilácie a vykresľovania kódu do aplikácie.

Code je hlavná trieda tohto balíka. Ukladá v sebe celú štruktúru grafického kódu a spravuje všetky volania z *MainActivity.* Táto trieda zároveň obsluhuje všetky ostatné triedy v balíku.

Brick je trieda, ktorá reprezentuje 1 celý príkaz v grafickom kóde. Obsahuje informáciu o svojom type príkazu a takisto aj súradnice z plochy, kde je daný príkaz vykreslený. Z tejto triedy sú odvodené ďalšie 4 triedy: *SimpleBrick*, *IfBrick*, *LoopBrick* a *HelperBrick* **.**

SimpleBrick je podtriedou triedy *Brick*. Slúži na uchovávanie informácie o klasickom príkaze a aj jeho vykresľovanie.

IfBrick je podtriedou triedy *Brick*. Obsluhuje informácie o príkaze podmienky a stará sa aj o vykresľovanie daného príkazu.

LoopBrick je podtriedou triedy *Brick*. Stará sa o informácie príkazu cyklu a stará sa aj o vykresľovanie daného príkazu.

HelperBrick je podtriedou triedy *Brick* stará sa hlavne o prehľadné vykresľovanie podmienky a cyklu while.

CodePainter zastrešuje vykresľovanie kódu. Prechádza všetky grafické príkazy a postupne ich vykreslí.

CodeCompiler slúži na preloženie grafického kódu do OrbBasic kódu na základe ktorého už robot presne vie, čo má robiť.

Viac informácii o zdrojovom kóde aplikácie nájdete v podkapitole štruktúra aplikácie v [9].

4.3 Analýza aplikácie

V predchádzajúcej bakalárskej práci bolo cieľom vytvoriť základné programovacie prostredie, v ktorom sa budú dať programovať základné príkazy na ovládanie robota.

Aplikácia umožňovala ovládať robota týmito funkciami:

ROLL(h, s) – slúži na pohyb robota ktorému sa určil nejaký smer a rýchlosť pohybu. Príkaz sa kompiloval do OrbBasic-ovej funkcie "*goroll h,s,*1". Na dĺžku trvania pohybu slúži ďalší príkaz **DELAY(ms)**.

STOP() – zastaví robota v pohybe. Príkaz sa kompiloval do OrbBasic-ovej funkcie "*goroll 0,0,2*".

COLOR(r, g, b) - nastaví farbu Sphera podľa RGB farby, všetky 3 parametre sú v rozmedzí 0-255. Príkaz sa kompiloval do OrbBasic-ovej funkcie "*RGB r,g,b*".

LED(intensity) - nastavuje intenzitu svietenia zadnej LED diódy, ktorá slúži väčšinou na nastavovanie smeru pohybu robota. Príkaz sa kompiloval do OrbBasic-ovej funkcie "*backLED intensity*".

DELAY(ms) - slúži na predĺženie určitej činnosti na niekoľko milisekúnd (napr. pohyb robota, dĺžku svietenia...). Príkaz používa OrbBasic-ovú funkciu "*delay ms*".

PRINT(message) - vypíše zadanú textovú správu do ovládacieho zariadenia. Príkaz používa OrbBasic-ovú funkciu "*print "message"* ".

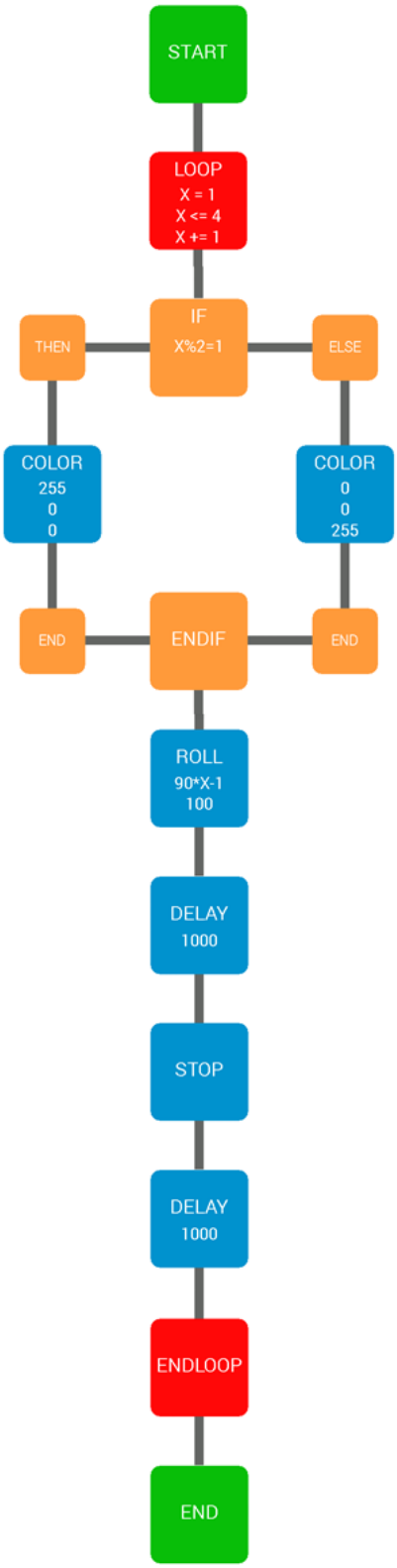
SOUND() – prehrá notifikačný zvuk na ovládacom zariadení.

VAR(X = value) – deklarovanie nejakej premennej na určitú hodnotu. V OrbBasic-u je to klasicke priradenie "*X = value*".

IF condition THEN ELSE– podmienka, slúži na rozvetvenie programu. Používa OrbBasic-ovú podmienku "if expresion then goto line else goto anotherline".

LOOP X = value to value step value - jednoduchý cyklus, ktorý zabezpečí určený počet opakovaní.

WHILE condition - klasický cyklus opakujúci sa pokým platí podmienka zadaná vstupným parametrom.



Obr. 16 Ukážka programu štvorec v predchádzajúcej bakalárskej práci [9]

Ako si môžeme v programe všimnúť, výhodou programu je, že typy funkcii sú pekne farebne odlíšené čo dodáva programu veľmi dobrú prehľadnosť. START a END sú zafarbené na zeleno, LOOP a WHILE na červeno, podmienka IF a jeho rozvetvovanie na oranžovo, deklarovanie premennej VAR na fialovo a ostatné funkcie na modro. Pri skončení programu a

opätovnom spustení program načíta predošlý naposledy zadaný program a zobrazí ho. V programe je taktiež možné ukladať a načítavať iné v minulosti naprogramované programy.

Nevýhodou tohto programu je, že niektoré funkcie sú troška nešťastne naprogramované.

Príkaz ROLL(heading, speed) musí v podstate vždy následne volať príkaz DELAY(ms), aby sa zabezpečilo iba určité trvanie príkazu. Oveľa výhodnejšie by bolo spraviť príkaz ROLL(heading, speed,delay), kde by delay=0 označoval nezastavovanie a iná hodnota v milisekundách by určovala trvanie ROLL príkazu. Takisto tu chýba funkcia na poslanie robota do presne určenej vzdialenosti. Avšak na to by bola potrebná hĺbková analýza, ktorá by musela obsahovať aj nejaký koeficient typu povrchu aby bol daný príkaz naozaj presný. Taktiež je trocha škoda, že ROLL príkaz neukazuje v programe smer pohybu nejakou grafickou šípkou ale iba číselnou reprezentáciou. Pre detský programovací jazyk je to veľmi dôležitý aspekt programu.

Je dobré, že príkaz COLOR obsahuje všetky farby spektra, avšak deti farebnému modelu RGB veľmi nerozumejú. Preto by bolo vhodné mať tam príkaz na zadanie nejakej klasickej farby ako red, purple, yellow... Na to by sa mohla jednoducho využiť funkcia z OrbBasica LEDC(number), ktorej by na základe klasickej farby našlo svoje number a sphera zafarbila.

Deklarácia premenných (VAR) je takisto veľmi užitočný nástroj, avšak z hľadiska dieťaťa tento príkaz pravdepodobne ťažko pochopí. Preto sa to skôr zíde pre skúsenejších používateľov.

Chýba tu takisto veľa zaujímavých a užitočných príkazov z OrbBasicu, ktoré by sa dali použiť v aplikácii.

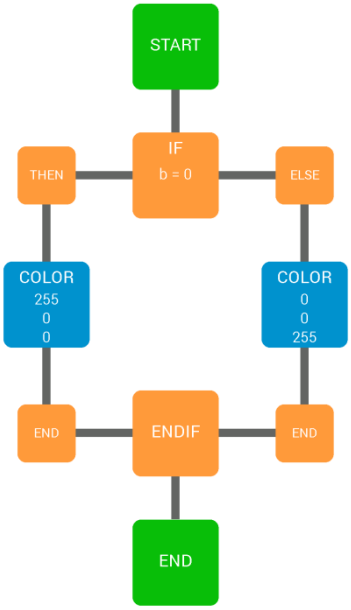
Avšak asi najväčším nedostatkom programu je to, že nemá uložené nejaké zaujímavé funkcie (hotové podprogramy) ktoré by stačilo zavolať a robili by požadovanú akciu. Do programu by sa určite zišli funkcie ako JUMP, kde by robot skákal na mieste, prípadne detekcie zo senzorov či je robot vo vzduchu prípadne či narazil do steny a podobne.

5 Návrh riešenia

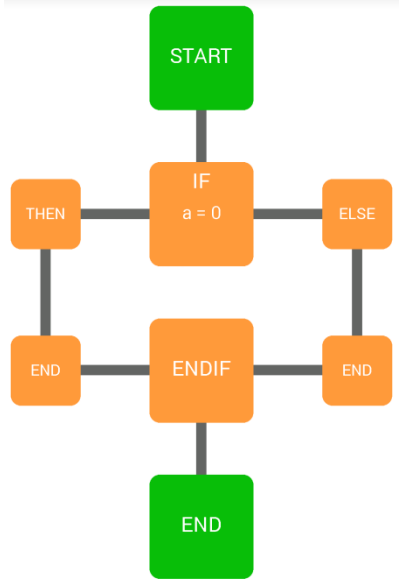
5.1 Načítanie podprogramu

Jednou z hlavných pridaných funkcii aplikácie bude načítanie už vopred uloženého kódu ako podprogramu do nového vytváraného programu. Malo by to prebiehať nasledujúcim spôsobom:

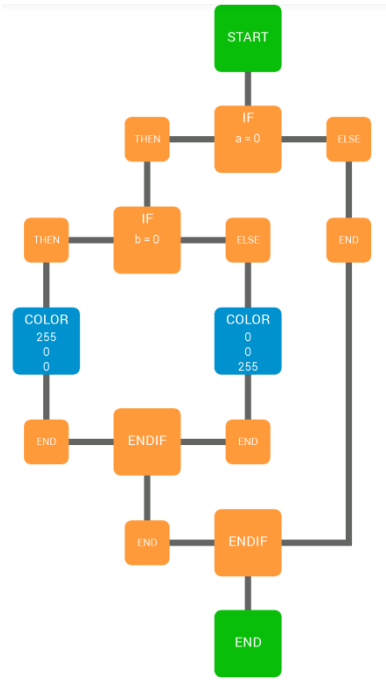
1. Najprv si vytvoríme určitý program a uložíme si ho (Obr. 17).
2. Vytvoríme nový program, do ktorého by sme chceli vložiť podprogram (Obr.18)
3. Kliknutím na ľavý "then" podľa Obr. 18 zvolíme "Add subprogram", vyberieme uložený program z Obr. 17 a náš program bude vyzerať ako na Obr. 19.



Obr. 17



Obr. 18



Obr. 19

5.2 Menu výberu grafickej funkcie

V pôvodnom riešení bolo možné robiť kliknutím na grafickú funkciu 3 možnosti:

Add after **-** slúžil na pridanie príkazu za danú grafickú funkciu

Edit **-** slúžil na zmenu parametrov grafickej funkcie

Delete **-** vymazal danú funkciu

V novej aplikácii budeme mať na výber zo 6-tich možností:

Edit **-** zmení parametre grafickej funkcie

Add command **-** pridá jednoduché príkazy, ktoré slúžia na prácu s robotom.

Add sensor **-** pridá do grafickej plochy vybranú senzorovú podmienku

Add effect **-** pridá do grafického kódu zvolenú efektnú funkciu

Delete **-** vymaže daný príkaz

Add subprogram **-** pridá za danú funkciu už v minulosti uložený program

5.3 Funkcie

Hlavným zlepšením predchádzajúcej aplikácie bude pridanie nových príkazov, senzorov a efektov.

5.3.1 Príkazy

Náš program by mohol obsahovať tieto nové príkazy:

PRINT(variable) - príkaz ktorý vypíše hodnotu premennej do telefónu

LEDC(color) - tento príkaz by mal uľahčiť výber farby sphera. Po kliknutí na príkaz si užívateľ bude môcť vybrať jednu z 8-mych základných farieb.

Heading(h) - bude nastavovať smer pohybu sphera (teda smer zadnej LED diódy). Tento príkaz bude používaný najmä s LED príkazom.

RAW(Lmode, Lspeed, Rmode, Rspeed) - príkaz, pomocou ktorého budeme môcť ovládať robota bez stabilizačného systému. Dá sa tu nastaviť smer rolovania každého z 2 bočných motorov a aj ich patričná rýchlosť. Vďaka tomu že tu nie je stabilita, dajú sa s týmto príkazom robiť veľmi zaujímavé až divoké veci.

Locate(x,y) - taktiež veľmi potrebná funkcia, slúži na nastavenie počiatočných súradníc robota. Dá sa využiť na meranie prejdenej vzdialenosti pomocou využitia systémových premenných *xpos* a *ypos*.

Fade(r,g,b) - funkcia, ktorej zadáme r, g, b hodnoty a táto farba bude počas cca 2 sekúnd blednúť.

Brighten(r,g,b) - funkcia, ktorá zo zhasnutého sphera bude robota postupne rozsvecovať počas 2 sekúnd až do dosiahnutia zvolenej r, g, b farby.

5.3.2 Senzory

V programe by mali byť implementované tieto senzory:

- Senzor pádu (**FREEFALL**) - je to senzor ktorý zistí či je Sphero padá (je vo vzduchu)
- Senzor dotyku (**TOUCH**) - senzor, ktorý detekuje akýkoľvek náraz
- Senzor zatrasenia (**SHAKE**) - senzor detekujúci poriadne zatrasenie so Spherom

5.3.3 Efekty

Budú to preddefinované programy, ktoré spravia nejaký zaujímavý efekt s robotom.

V aplikácii by mali byť použité tieto efekty:

- **TRIANGLE -** robot svojím pohybom opíše trojuholník
- **SQUARE -** robot opíše štvorec
- **CIRCLE -** robot spraví kruh
- **JUMP -** pomocou tejto funkcie by mal robot po dobu 2 sekúnd skákať
- **CRAZY BALL -** pomocou tejto funkcie sa bude Sphero točiť a svietiť na mieste ako šialený

5.4 Grafický návrh

Grafický návrh predchádzajúcej verzie je veľmi prehľadný a jednoduchý. Preto v aplikácii z hľadiska grafického dizajnu nastanú iba minimálne zmeny a to v grafických príkazoch.

Jednou zmenou bude príkaz ***ROLL***. Namiesto čísla, ktorý ukazuje otočenie o konkrétny počet stupňov bude vykresľovať šípku do daného smeru pohybu. Malo by to malým žiakom uľahčiť a sprehľadniť očakávaný pohyb robota.

Druhá zmena bude v grafike senzorov. Každý z 3 dostupných senzorov (FREEFALL, TOUCH, SHAKE) bude mať svoju vlastnú ikonku senzora.

5.5 Začiatočnícky a pokročilý mód aplikácie

Súčasťou aplikácie by mal byť aj Easy/Advaced mód.

V Easy móde budú dostupné iba niektoré jednoduchšie základné funkcie.

V Advanced móde budú dostupné všetky funkcie vrátane tých v Easy móde. Advanced mód bude obsahovať aj funkcie, na ktoré je potreba poznať štruktúru OrbBasic jazyka.

5.6 Motivácia

Jednou z hlavných motivácii je naprogramovať v mojej aplikácii program, pomocou ktorého bude schopný Sphero prejsť bludisko podľa pravidla pravej ruky.

6 Implementácia

6.1 Implementácia načítania podprogramu

Funkcia načítania podprogramu do aplikácie prebiehala celkom dobre. Bolo potrebné do zdrojového kódu *MainActivity* pridať novú funkciu *rawLoadSubprogram*. Tá bola kópiou funkcie *rawLoadProgram* až na to, že sa nevytváral nový kód (*code = new Code()*) ale sa len pozmenila funkcia načítania kódu od iného indexu: *code.restoreCode(c.code, addBrickIndex-1);* . Potom už len bolo potreba pridať položku do menu výberu grafickej funkcie aby načítalo daný podprogram a už to fungovalo tak ako to je spomenuté v návrhu.

6.2 Nové funkcie

Nové funkcie boli implementované a rozdelené do tých 3 kategórii, ako to bolo uvedené v návrhu.

6.2.1 Implementované príkazy

PRINT_VAR(variable)

> Príkaz na zobrazenie hodnoty premennej v telefóne. Na vstupe dostáva 1 parameter a to hodnotu premennej. Do OrbBasicu sa daná funkcia prekladá ako *"print variable"*.

PRINT_MSG(message)

Je to premenovaná funkcia PRINT(message) z predchádzajúcej verzie aplikácie

PRINT (line)

Zložitejší príkaz na vypisovanie textu do telefónu. Príkaz dostáva na vstupe riadok, ktorý môže byť tvorený reťazcami spolu s premennými. Na vypísanie textu treba použiť úvodzovky. Formát by mohol vyzerať takto:

"správa1" premenná1 "správa2" premenná2

Do OrbBasicu sa daná funkcia prekladá ako *"print line".*

Viac informácii o formáte textu je v OrbBasicovej funkcii ***print*** [2].

LEDC(color)

Tento príkaz poskytuje rýchly výber z 8 základných farieb.

K dispozícii sú tieto farby:

1. *červená(red)*
2. *zelená(green)*
3. *modrá(blue)*
4. *oranžová(orange)*
5. *fialová(purple)*
6. *biela(white)*
7. *žltá(yellow)*
8. *žiadna(off)*

V OrbBasicu daná funkcia vyzerá takto: *"LEDC num_color".*

Heading(h)

Príkaz nastavujúci smer pohybu Sphera (smer zadnej LED diódy). Obsahuje 1 parameter *h* označujúci stupeň otočenia podľa hodinových ručičiek. Stupeň je hodnota medzi 0..359 kde hodnota 0 predstavuje sever. Do OrbBasicu sa daná funkcia prekladá ako *"heading h".*

RAW(Lmode, Lspeed, Rmode, Rspeed)

Príkaz, pomocou ktorého môžeme ovládať robota bez stabilizačného systému. Obsahuje 4 parametre, kde *LMode* a *RMode* sú typy pohybu ľavého a pravého motora a môžu nadobúdať tieto hodnoty:

- Vypnuté(Off)
- Dopredu(Forward)
- Dozadu(Reverse)
- Brzda(Brake)
- Ignorovaný(Ignore)

Lspeed a *Rspeed* označujú rýchlosť ľavého a pravého motora. Hodnota je v rozmedzí 0..255. V OrbBasicu daná funkcia vyzerá takto: *"raw num_Lmode, Lspeed, num_Rmode, Rspeed".*

Locate(x,y)

Príkaz slúžiaci na nastavenie počiatočných súradníc robota. Obsahuje 2 parametre, X-ovú a Y-ovú súradnicu. Obidve premené sú v rozsahu -32768..32767. Dá sa využiť na meranie prejdenej vzdialenosti pomocou využitia systémových premenných *xpos* a *ypos*. Do OrbBasicu sa daná funkcia prekladá ako *"locate x,y".*

Fade(r,g,b)

Zaujímavý príkaz ktorý spôsobí efekt blednutia farby po dobu 2 sekúnd. Má 3 parametre - r, g, b čo sú vlastne farby RGB spektra v rozmedzí 0..255. Do OrbBasicu sme danú funkciu preložili ako vidieť na Obr. 21 . Premenné var1, var2, var3 sú hodnoty r, g, b:

```
10 Y = 25
20 Z = 255
30 if Z > 0 then goto 40 else goto 80
40 RGB Z*var1/255,Z*var2/255,Z*var3/255
50 delay 8
60 Z = Z - 1
70 goto 30
80 Z = 0
```

Obr. 20 OrbBasicový kód funkcie Fade(r, g, b)

Brighten(r,g,b)

Opačný príkaz ktorý spôsobí efekt rozsvecovania farby po dobu 2 sekúnd. Má 3 parametre - r, g, b čo sú vlastne farby RGB spektra v rozmedzí 0..255. Do OrbBasicu sme danú funkciu preložili ako vidieť na Obr. 21. Premenné var1, var2, var3 sú hodnoty r, g, b:

```
10 Y = 25
20 Z = 0
30 if Z < 256 then goto 40 else goto 80
40 RGB Z*var1/255,Z*var2/255,Z*var3/255
50 delay 8
60 Z = Z + 1
70 goto 30
80 Z = 0
```

Obr. 21 OrbBasicový kód funkcie Brighten(r, g, b)

Všetky tieto príkazy sú reprezentované modrou grafickou funkciou s názvom danej funkcie. Do ich dátovej štruktúry sa všetky ukladajú pod typom *SimpleBrick.* Príkazy sa zobrazia po kliknutí na "Add command".

6.2.2 Implementované senzory

Do aplikácie boli implementované tieto senzory:

FREEFALL

Senzor pádu, ktorý detekuje či sa Sphero znáša vo vzduchu alebo nie.

TOUCH

Senzor dotyku, ktorý detekuje či nenastala nejaká kolízia so stenou alebo nejakým iným predmetom. Bohužiaľ tento senzor je dosť nepresný, keďže kolíziu detekuje na základe intenzity zahrkania a preto niektoré nárazy nemusí detekovať.

SHAKE

Tento senzor zistí, či bolo so Spherom poriadne zahrkané. Na podobnom princípe funguje zobudenie robota zo spánku.

Všetky tieto senzory sú reprezentované oranžovou grafickou funkciou s príslušným obrázkom danej funkcie. Do ich dátovej štruktúry sa všetky ukladajú pod typom *IfBrick*, ktorá

rozvetvuje program na 2 časti a senzory využívajú túto grafickú stavbu podmienky. V OrbBasicu sa hodnoty odrazu zisťujú v premennej *accelone*, na základe ktorej bolo možné vytvoriť tieto 3 senzory. Príkazy sa zobrazia po kliknutí na "Add sensor".

6.2.3 Implementované efekty

TRIANGLE

Funkcia pomocou ktorej Sphero opíše svojím pohybom trojuholník. Pri každej strane trojuholníka svieti Sphero náhodnou farbou. V OrbBasicu vyzerá daný program približne takto:

```
10 Y = 25
20 Z = 0
30 if Z < 3 then goto 40 else goto 110
40 RGB rnd 255,rnd 255,rnd 255
50 goroll Z*120+30,100,1
60 delay 1000
70 goroll Z*120, 0, 2
80 delay 1000
90 Z = Z + 1
100 goto 30
110 Z = 0
```

Obr. 22 OrbBasicový kód funkcie Triangle

SQUARE

Táto funkcia urobí s robotom pohyb v tvare štvorca. Pri každej strane štvorca svieti Sphero náhodnou farbou. V OrbBasicu by to vyzeralo asi takto:

```
10 Y = 25
20 Z = 0
30 if Z < 4 then goto 40 else goto 110
40 RGB rnd 255,rnd 255,rnd 255
50 goroll Z*90,100,1
60 delay 1000
70 goroll Z*90, 0, 2
80 delay 1000
90 Z = Z + 1
100 goto 30
110 Z = 0
```

Obr. 23 OrbBasicový kód funkcie Square

CIRCLE

Funkcia, ktorá bude hýbať robota po obvode kruhu pričom bude robot blikať náhodnými farbami. OrbBasic-ový kód by mohol vyzerať nasledovne:

```
10 Y = 25
20 Z = 0
30 goroll Z,50,2
40 RGB rnd 255,rnd 255,rnd 255
50 delay 50
60 Z = Z + 5
70 if Z < 360 then goto 30
80 Z = 0
90 goroll Z,0,2
```

Obr. 24 OrbBasicový kód funkcie Circle

JUMP

Táto funkcia urobí so sphera skákajúcu loptu, ktorá sa bude pohybovať do vopred neznámeho smeru. Skákanie trvá 2 sekundy a počas toho Sphero bude svietiť 1 náhodnou farbou. V OrbBasicu vyzerá kód nasledovne:

```
10 RGB rnd 255,rnd 255,rnd 255
20 raw 1,255,1,255
30 delay 2000
40 raw 0,0,0,0
50 ctrl = 1
```

Obr. 25 OrbBasicový kód funkcie Jump

CRAZY BALL

Funkcia, vďaka ktorej urobí sphero na mieste naozaj zaujímavý efekt. Využíva svietenie zadnej LED diódy kombináciou s extrémne častého a náhodného menenia farby Sphera. Táto funkcia celkom šikovne využíva pomalé menenie zmeny smeru pomocou *heading* príkazu. V nasledujúcej ukážke vidno OrbBasic-ový kód, ktorý celkom nekorešponduje s praxou, pretože :

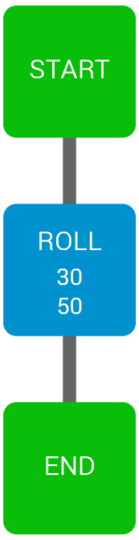
```
10 Y = 25
20 Z = 0
30 backLED 255
40 if Z > 359 then goto 100
50 heading Z
60 RGB rnd 255,rnd 255,rnd 255
70 delay 50
80 Z = Z + 1
90 goto 40
100 Z = 0
110 backLED 0
```

Obr. 26 OrbBasicový kód funkcie Crazy ball

6.3 Implementácia grafických funkcii

ROLL

V grafickej funkcii ROLL sa zmenilo zobrazenie smeru pohybu. Predtým to bolo zobrazené daným stupňom otočenia, teraz je to zobrazené príslušnou šípkou. Na obrázkoch môžete vidieť a nové riešenie grafickej funkcie:

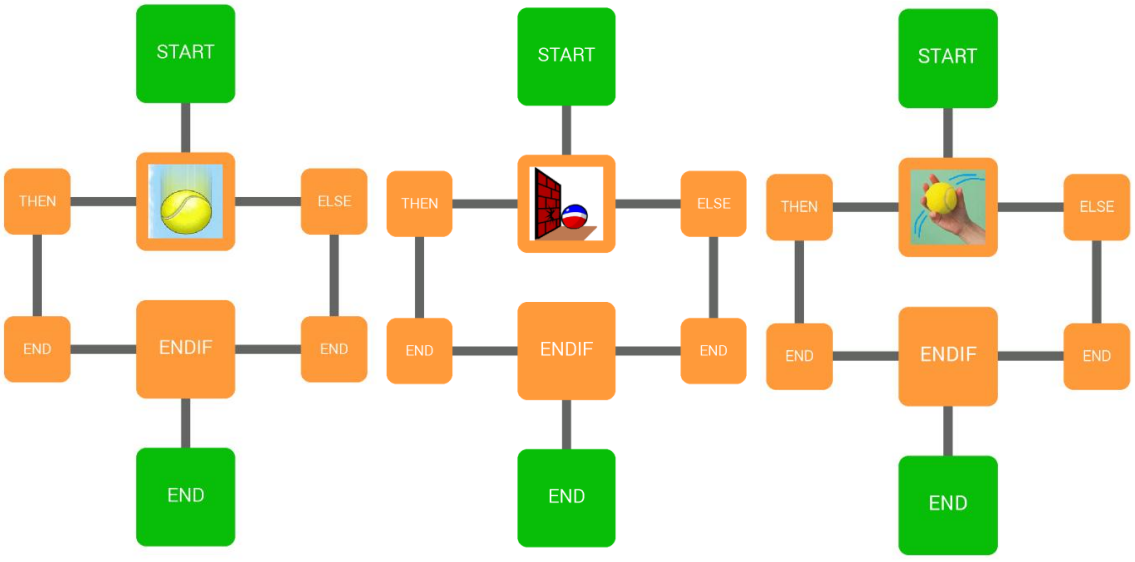


Obr. 27 Pôvodné riešenie

Obr. 28 Nové riešenie

Obrázky senzorov

V týchto grafických funkciách sa použila štruktúra príkazu IF. Zmena je iba v tom, že namiesto príkazu je tam implementovaný obrázok.



Obr. 29. Porovnanie príkazov FREEFALL, TOUCH a SHAKE

6.4 Easy/Advanced mód

Easy a Advanced mód majú spoločné tieto funkcie:

ROLL, STOP, LEDC, COLOR, LED, HEADING, DELAY, PRINT_MES, PRINT_VAR, SOUND, VAR, FADE, BRIGHTEN, JUMP, TRIANGLE, SQUARE, CIRCLE a CRAZY BALL

V Advanced móde je možné používať aj tieto funkcie:

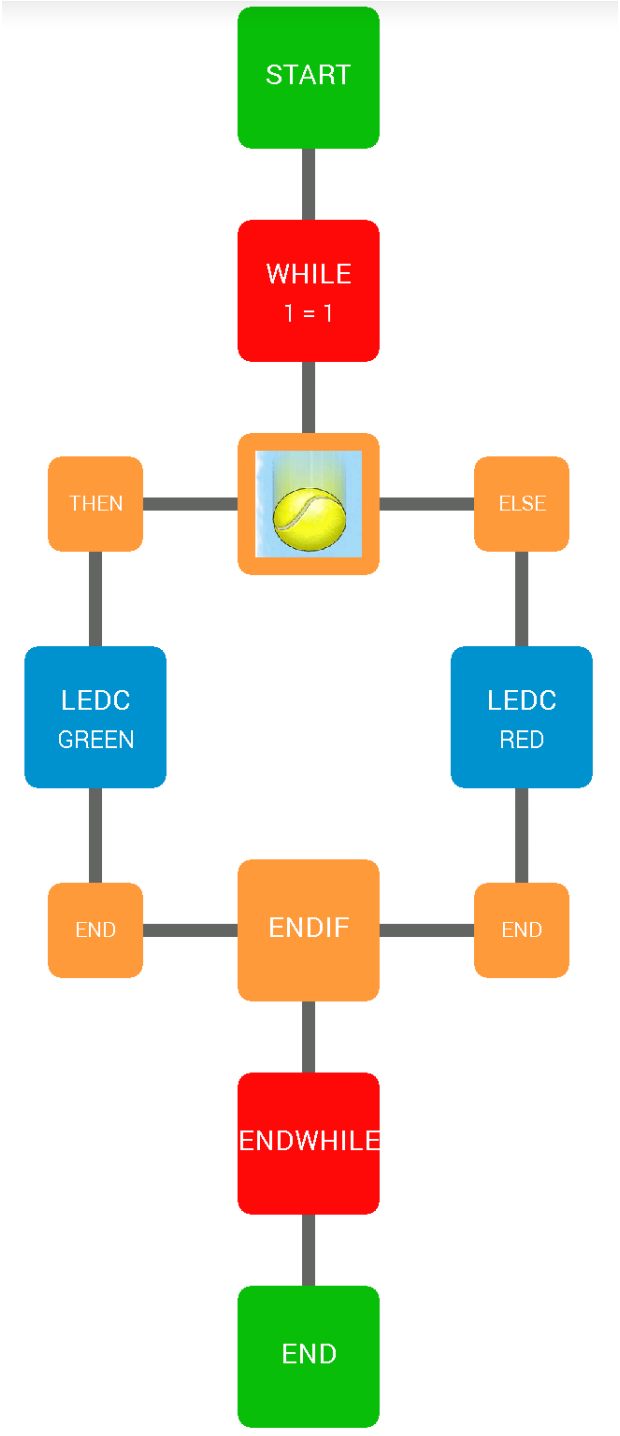
PRINT, LOCATE a RAW

7 Ukážka programov

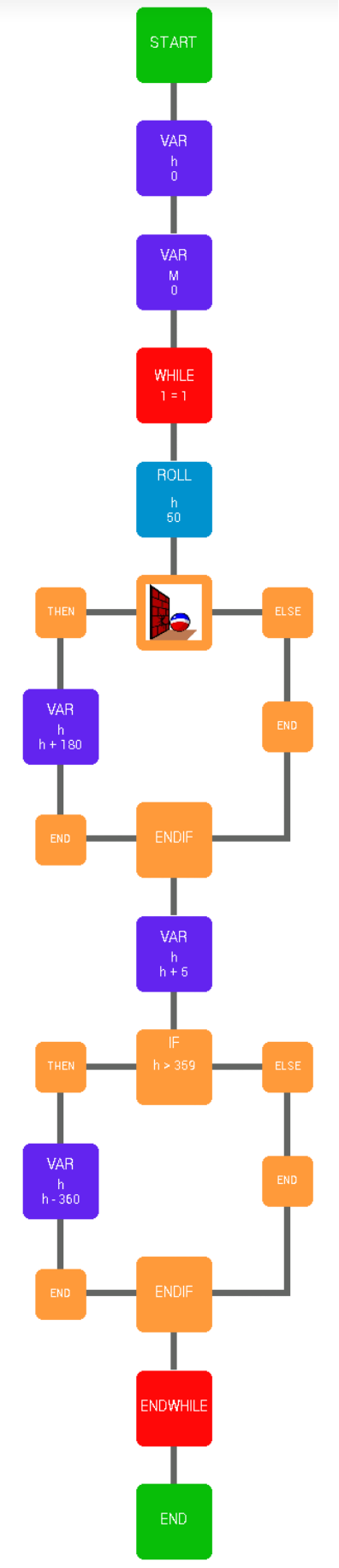
V nasledujúcej časti bude popísané 2 grafické programy: *Horúce vajce* a *Bludisko*.

Horúce vajce - je program, ktorý využíva nový implementovaný senzor FREEFALL. Pri spustení programu bude Sphero svietiť na červeno. Avšak ak ho vyhodíte, tak zmení svoju farbu na zelenú.

Bludisko - je to program, pomocou ktorého sa mi podarilo prejsť bludisko s väčšími uličkami. Program prechádza bludisko podľa pravidla pravej ruky. V programe sa robot pohybuje po kružnici a pri každom náraze do steny urobí opačný oblúk. To umožňuje lemovať pravú stenu uličky bludiska a tým pádom aj prejsť bludisko podľa pravidla pravej ruky. Na detekovanie nárazu využíva nový implementovaný senzor TOUCH.



Obr. 30 Ukážka programu "Horúce vajce"



Obr. 31 Ukážka programu "Bludisko"

Záver

Cieľom tejto bakalárskej práce bolo vylepšiť grafický programovací jazyk pridaním senzorov, aplikovaním všetkých užitočných funkcii z OrbBasica, zjednodušením zadávania grafických príkazov, pridaním potrebných obrázkov k senzorom a implementovanie grafickej reprezentácie smeru pohybu. Takisto tu bola snaha o vytvorenie zaujímavých efektov a malých programov, ktoré zaručia zaujímavú a prehľadnú funkcionalitu aplikácie najmä z hľadiska malých žiakov.

Zo začiatku bola potrebná analýza dostupných grafických programovacích jazykov a ich následné porovnanie. Zistilo sa, že mobilných programovacích jazykov nie je veľa a preto bola potrebná analýza hlavne grafických jazykov pre PC. Po analýzach grafických programovacích jazykov vrátane riešenia minuloročného riešenia na rovnakú tému som sa rozhodol túto aplikáciu vylepšiť o všetky potrebné a dostupné OrbBasicové funkcie.

Základné problémy s ktorými som sa musel popasovať boli naučenie sa pochopiť štruktúru tvorenia Android aplikácie, naučiť sa a vyskúšať všetky dostupné funkcie OrbBasica z čoho vychádzal aj návrh nových funkcii pre robota a taktiež pochopiť štruktúru a prepojenia aplikácie, na ktorú nadväzujem.

Najväčším problémom bol samotný OrbBasic. Najprv som prišiel na to, že hlavná aplikácia pre Google Play nefunguje, tak som bol nútený hľadať iné riešenie. Našťastie v Android SDK bola spravená základná OrbBasic aplikácia. Ďalej som zistil, že premenné *redval, grnval* a *bluval*, ktoré by si mali pamätať hodnoty posledného zadaného RGB príkazu si ho nepamätajú. Taktiež príkaz ROLL by mal pri nezadaní príkazu DELAY ísť jedným smerom teoreticky donekonečna. U mňa sa z neznámych dôvodov po krátkom čase zastaví čo mi znemožnilo implementovať funkciu na rolovanie do danej vzdialenosti. Taktiež som vychádzal z mylného predpokladu v minuloročnej práci, kde bolo uvádzaných len 25 užívateľských premenných. Zistil som že je možné použitie ďalších 25 premenných pomocou indexu Y v premennej Z.

Motiváciou bolo aj naprogramovať v aplikácii program, ktorý prejde bludisko pomocou pravidla pravej ruky a následné otestovanie robota na súťaži ISTROBOT. Bohužiaľ vzhľadom na úzke uličky a klzký povrch nevedel Sphero dobre detekovať náraz do steny a tým pádom bolo jeho chovanie nepredvídateľné. Avšak pri väčších rozmeroch bludiska a dobrom povrchu (napr. koberec) sa nakoniec podar ilo s robotom bludisko prejsť.

Na koniec chcem dodať, že výsledná aplikácia má veľmi dobrú funkčnosť. Nefunkčnosťou OrbBasic aplikácie v Google Play by sa mohla stať dočasne veľmi plnohodnotnou náhradou tejto oficiálnej aplikácie. Funkčnosť aplikácie je na takej úrovni, že by sa mohla používať za účelom výučby programovania na základných školách. Takisto by svojím grafickým prostredím mohla zaujať veľké množstvo ľudí, ktorí sa s robotom iba zoznamujú a nemajú záujem pracovať s textovým programovacím prostredím.

Zdroje

[1] Sphero, Orbotix, 15.9.2014

Macrolab 2.4.2, dátum prístupu: 2.5.2015

https://play.google.com/store/apps/details?id=com.orbotix.macrolab

[2] BLAHO, Andrej, 13.4.2009

Imagine Logo, dátum prístupu: 4.5.2015

http://imagine.input.sk/popis.html

[3] HARVEY, B. - MöNIG, J.

Snap! (Build Your Own Blocks) 4.0, prístupu: 7.5.2015

https://snap.berkeley.edu/

[4] The MIT App Inventor Group.

MIT App Inventor 2.0, *dátum prístupu: 9.5.2015*

http://appinventor.mit.edu/explore/about-us.html

[5] The LEGO Group

LEGO Mindstorms, dátum prístupu: 12.5.2015

http://www.lego.com/en-us/mindstorms/

[6] Alza

Orbotix sphere 2.0, dátum prístupu: 15.5.2015

https://hracky.alza.sk/orbotix-sphere-2-0-d481118.htm

[7] Sphero 2.0

Sphero Store, dátum prístupu: 18.5.2015

http://store.gosphero.com/products/sphero-2-0

[8] Orbotix, 29.5.2013.

OrbBasic 1.07, dátum prístupu: 22.5.2015

https://github.com/orbotix/DeveloperResources/blob/master/docs/orbBasic_1.07.pdf

[9] Melicherčík, M.:

Výukové prostredie na programovanie mobilného guľového robota, bakalárska práca, FMFI UK, Bratislava, 2014, 48 s

[10] Orbotix, 22.11.2013.

Sphero Android SDK, dátum prístupu: 26.5.2015

https://github.com/orbotix/Sphero-Android-SDK/

Prílohy

Súčasťou práce je aj CD. Popis jeho obsahu je nasledovný:

- zdrojový kód aplikácie
- bakalárske práca v PDF formáte