LATVIJAS UNIVERSITĀTE
FIZIKAS UN MATEMĀTIKAS FAKULTĀTE
DATORIKAS NODAĻA

IZGLĪTOJOŠO PROGRAMMĒŠANAS VALODU IESPĒJAS UN PIEMĒROTĪBA SĀKUMSKOLAS VECUMA BĒRNIEM

BAKALAURA DARBS

Autors: **Rasma Lejniece**
Stud. apl. Nr. DatZ040093
Darba vadītājs:
asoc.prof. Dr.sc.comp.
Guntis Arnicāns

RĪGA 2008

ANOTĀCIJA

Darbs apraksta izglītojošo programmēšanas valodu iespējas un piemērotību sākumskolas vecuma bērniem. Tajā tiek aprakstīta datora iespējamā ietekme uz bērna veselību, sākumskolas mācību metodes, vispārīgie datorspēļu izstrādes principi. Veikta analīze par šīs informācijas iespējamo pielietojumu sākumskolas vecuma bērniem domātas programmatūras raksturiezīmju noteikšanā. Novērtētas izglītojošo programmēšanas valodu iespējas. Piedāvāts autores risinājums izglītojošo programmēšanas valodu pielāgošanai sākumskolas vecuma bērniem.

Atslēgas vārdi: sākumskolas vecuma bērni, programmatūras raksturiezīmes, datorspēles, izglītojošās programmēšanas valodas.

ABSTRACT

The research describes educational programming language options and suitability for children of primary school age. The paper describes the possible influence of computer to the health of children, preschool education methods, and general computer games development principles. The analysis has been done to identify the main characteristics of software for children of primary school age. Educational programming language options have been evalued. Solution by author of the paper has been offered to customize educational programming languages for primary school age children.

Key words: primary school age children, characteristics of software, computer games, educational programming languages.

SATURS

IEVADS

Var novērot, ka datora lietotāja minimālajam vecumam ir tendence samazināties. Līdz ar to aktuālāks kļūst jautājums par lietderīgu datora izmantošanu atšķirīgos bērna vecuma posmos. Kā viens no izglītojošās izklaides veidiem bērniem tiek piedāvāta programmēšana. Darba mērķis ir noskaidrot, vai izglītojošā programmēšana ir piemērota sākumskolas vecuma bērniem un kādas ir tās iespējas.

Darbā aprakstīta datora ietekme uz bērna veselību un komunikācijas spējām, mācību metodes sākumskolas vecuma bērniem, vispārīgie datorspēļu izstrādes principi. Apkopojot šo informāciju, izstrādātas sākumskolas vecuma bērniem domātas programmatūras raksturiezīmes. Ņemot vērā šīs raksturiezīmes un programmēšanas pamatprincipu pārklājumu novērtētas izglītojošās programmēšanas valodas LOGO un RoboMind. Piedāvāti risinājumi, kā novērst izglītojošo programmēšanas valodu neatbilstību sākumskolas vecuma bērniem domātās programmatūras raksturiezīmēm. Analizēti turpmākie pētījuma virzieni.

Pirmajā nodaļā teorētiski aprakstīti sākumskolas vecuma bērnu programmatūras raksturiezīmes ietekmējošie faktori.

Otrajā nodaļā veikta šo faktoru analīze, aprakstot to pielietojumu un nozīmi programmatūras izstrādē.

Trešajā nodaļā novērtētas izglītojošo programmēšanas valodu LOGO un RoboMind piedāvātās iespējas.

Ceturtajā nodaļā novērtēta izglītojošo programmēšanas valodu piemērotība sākumskolas vecuma bērniem, balstoties uz iepriekš izstrādātajām raksturiezīmēm. LOGO un RoboMind vide šajā aspektā apskatīta sīkāk.

Piektajā nodaļā sniegti iespējamie risinājumi izglītojošo programmēšanas valodu pielāgošanai sākumskolas vecuma bērniem.

1. SĀKUMSKOLAS VECUMA BĒRNIEM DOMĀTĀS PROGRAMMATŪRAS RAKSTURIEZĪMES IETEKMĒJOŠIE FAKTORI

1.1. Datora ietekme uz bērna veselību un komunikācijas spējām

Mūsdienās pasaulē ir veikts ļoti liels skaits dažādu pētījumu ar mērķi noskaidrot darba ar datoru ietekmi uz bērnu fizisko un garīgo veselību. Ir mēģināts identificēt problēmas, kas varētu rasties gan tuvākajā laikā, gan arī nākotnē. Tomēr, ņemot vērā relatīvi neseno datoru parādīšanos bērnu ikdienas dzīvē un diametrāli pretējos pētījumu rezultātus, viennozīmīgus secinājumus par atstāto iespaidu izdarīt nevar.

Pētījumi liecina, ka vienatnē pavadītais laiks pie datora, nomāc citu aktivitāšu un prasmju pilnveidošanos un tādējādi potenciāli rada problēmas nākotnē socializēties darba vidē, kas rada mazākas grūtības, ja bērns šo laiku ir pavadījis kontaktējoties ar ģimeni vai citiem bērniem.

Tajā pašā laikā, apkopojot informāciju par kāda cita pētījuma rezultātiem, iegūtie secinājumi ir sekojoši: Neskatoties uz daudzajiem brīdinājumiem par datoru negatīvo ietekmi, uz lielāko daļu bērnu tie īpaši negatīvu ietekmi neatstāj. Datori šobrīd ir aizraujoši un saistoši, un internets bez šaubām valdzina ar savām meklēšanas un komunikācijas iespējām. Bet kopumā tehnoloģijas var novērtēt kā pozitīvu pastiprinājumu attīstībai. Bērni, lietojot datoru, lasa, domā, analizē, kritizē un novērtē – sakārto savas domas. Viņi pielieto datorus tādām aktivitātēm, kas pēc mūsu saprašanas iet roku rokā ar pilnvērtīgu bērnību no tradicionālā redzes viedokļa. Viņi izmanto tehnoloģijas, lai spēlētos, mācītos, komunicētu un veidotu attiecības, kā bērni to ir darījuši vienmēr [1].

Mazāk sarežģīta situācija ir novērtējot atstāto ietekmi uz fizisko veselību, jo nav jāņem vērā tādas diezgan individuālas iezīmes kā raksturs un temperamenta tips un fakti tiek konstatēti veicot konkrētus mērījumus. Ir noskaidrots, ka, spēlējot datorspēles, bērns izdara vairāk kā 4000 acu kustības vienas stundas laikā. Bieži bērniem rodas problēmas izsekot horizontālām, vertikālām un slīpām kustībām. Datorspēļu spēlēšana un cieša datora ekrāna vērošana kaut vai tikai 20 minūtes var izraisīt skatiena aizmiglošanos pēc novēršanās no ekrāna, kas saistīta ar bloķētu skatiena fokusēšanos noteiktā attālumā. Normāli ikdienā cilvēks mirkšķina 15 reizes minūtē, un tas ir nepieciešamais mirkšķināšanas skaits, lai dabīgi mitrinātu acis. Lietojot datoru, šo mirkšķināšanu skaits ir samazināts, un tas veicina acu izžūšanu. Un, lai gan vislielāko ietekmi datoru lietošana atstāj tieši uz redzi, tā ietekmē arī delnas locītavas, kas var tikt bojātas no atkārtotajām kustībām, kuras rada tastatūras un peles lietošana; kaklu un muguru, kas attiecīgi var

traucēt bērna miegu, kvalitatīvu mācību procesu, un diskomforts var radīt negatīvu noskaņojumu un attieksmi kopumā [2].

1.2.Mācību metodes sākumskolā

V. Zelmenis definē: "Mācību metode ir skolotāja un skolēnu didaktiskās sadarbības paņēmienu sistēma skolēnu zināšanu un prasmju veidošanai un izziņas spēju attīstīšanai." Paņēmiens ir metodes sastāvdaļa (elements). Metodes izvēli nosaka mācību priekšmeta un temata specifika, kā arī skolēnu vecuma posma īpatnības un attīstības līmenis. Tātad, mazākiem bērniem nepieciešama lielāka metožu dažādība, kā arī vienā metodē iespējams iestarpināt citu metožu elementus [3].

Didaktikā mācību metodes klasificē pēc dažādiem kritērijiem. Populāra ir klasifikācija pēc izziņas veidiem (avotiem, ceļiem). Te runā par runas, rakstu, uzskates un praktiskajām metodēm. Tās sevī ietver [3]:

1. Sistemātiskais izklāsts- tas ir plānveidīgs, saturīgs, vecuma posmam atbilstošs, emocionāls, balstīts uz priekšzināšanām;
2. Pārruna;
3. Vingrinājumi;
4. Darbs ar grāmatu;
5. Ekskursijas, pastaigas, novērojumi;
6. Radoši uzdevumi;
7. Praktiska darbība;
8. Rotaļas, spēles.

Tradicionālā metožu klasifikācija pēc izziņu veidiem (ceļiem, avotiem) atbilst kā cilvēces izziņas attīstības vēsturei (filoģenēzei), tā arī bērna individuālās izziņas attīstības gaitai (ontoģenēzei). Tātad sākums ir vairāk vai mazāk apzināta apkārtnes vērošana, tad seko darbs (bērnam - rotaļa), kas, savukārt, paplašina un padziļina novērojumos iegūtās atziņas. Līdz ar valodas attīstību informācijas uzkrāšana un transformācija paātrinās [3].

Bērnības pavadonis ir rotaļa. Bērns un rotaļa ir nesaraujami saistīti, jo tā bērnā rada prieku, enerģiju, kā arī attīsta viņu. Rotaļas attīstība cieši saistīta ar viņa vispārējo attīstību [4].

Par rotaļspēlēm jau uzskatām bērnu rotaļas ar jau izstrādātu saturu, noteiktu struktūru, kurā bērns aktīvi darbojas un variē tās, izdomā jaunas darbības, noteikumus u.c. Rotaļspēlēs bērns mācās rotaļājoties, tā bērnu māca un attīsta, jo tai ir daudzveidīgas funkcijas: kognitīvās (izziņas), sociālās, darbības, rekreatīvās (izklaidējošās) [4].

1.3.Datorspēļu izstrādes principi

Šobrīd datorspēlēm ir aptuveni 50 gadus sena vēsture, bet to izstrādes process un pamatprincipi vēl joprojām nav pilnīgi aprakstīti. Var pat teikt, ka datorspēles ir vismazāk aprakstītā mūsu kultūras sastāvdaļa. Šobrīd vēl joprojām nav pilnīgas vienotas teorijas par spēļu izstrādi. Šādu situāciju iespējams ir radījis uzskats, ka datorspēles pārstāv zemākos kultūras slāņus un ir mazsvarīgas, un paredzētas tikai izklaidei. Trūkst teorētiskas izpratnes par spēlēm – kādām tām vajadzētu būt un kāda ir to saistība ar stāstošajiem paņēmieniem – romāniem un filmām, to novietojums vēsturiski un attiecībās ar citām spēlēm [5].

Datorspēles nerodas pašas no sevis, pamatā tā ir forma, kas sevī ietver kaut kā cita, citas spēlēs ideju. Pēdējo 25 gadu laikā pasaulē periodiski parādās iecere radīt datorspēles par dziļākām tēmām – stāstus, kas ietver sevī mīlestību, intrigas, ambīcijas. Problēma rodas faktā, ka spēles pieder formālai, algoritmiskai sfērai, savukārt stāsti vairāk saistīti ar interpretāciju, tāpēc spēles pretojas stāstu radošajai dabai, jo tie nevar tikt formalizēti [5].

Pamatā tiek izmantoti divi pamatprincipi, lai izveidotu spēles ar dziļāku nozīmi – "Interaktīvas filmas", kurās pēc neliela videoklipa noskatīšanās tiek piedāvātas vienkāršas izvēles, "Piedzīvojumu spēles" stils, kurā pēc sarežģītāka uzdevuma atrisināšanas var saņemt apbalvojumus no stāstītāja (personas, kas sniedz uzdevumus un informāciju par esošo situāciju). Tomēr abas šīs iespējas datorspēļu lietpratējiem nešķiet pietiekoši labas. Nepieciešams kaut kas tāds, kas ir dinamiskāks un ko var spēlēt atkārtoti. Tāpēc tika radīti 4 nosacījumi, kas palīdzētu izveidot saturīgu un tajā pašā laikā interesantu spēli [5]:

1. Tematiski tuva stāstam vai filmai – par cilvēku attiecībām, jūtām, ambīcijām;
2. Reālā laika spēle bez stāstītāja;
3. Ar iespēju mijiedarboties ar visu uz ekrāna attēloto;
4. Atbilstošu ne tikai nosauktajiem principiem, attīstīties spējīgu .

Diemžēl stāstu un spēļu lielo atšķirību dēļ, radīt spēli pēc šāda plāna šobrīd ir diezgan utopiski. Kaut vai tikai laika un telpas ierobežojumu dēļ, kas ir spēlēs. Tāpēc liela daļa spēļu izstrādātāju un izstādes principu pētnieku vēl joprojām balstās uz Tomasa Malones (*Thomas Malone*) 1980. gadā izstrādāto teoriju, ka tieši "izaicinājums", "fantāzija", "zinātkāre" motivē cilvēkus spēlēt datorspēles. Lai gan apšaubāmi sasniedzamu, noteiktu galamērķi viņš uzskatīja svarīgāku par izaicinājumiem bagātu vidi.

Līdz ar datoru un datorspēļu parādīšanos ikdienas dzīvē arvien interesantāks sāka šķist jautājums, kas tik ļoti piesaista bērnus datorspēlēs un kādi trūkumi ir formālajiem mācīšanās

paņēmieniem. Pētījumu rezultāti parādīja, ka datorspēļu spēlēšana rada bērnos jaunu izziņas spēju grupu [6]:

1. Strauja ātruma palielināšanās pret standarta ātrumu;
2. Paralēlā apstrāde pret lineāro apstrādi;
3. Primārais – grafika pret primārais – teksts;
4. Savienots pret savrups;
5. Aktīvs pret pasīvs;
6. Spēlēt pret strādāt;
7. Atriebība pret pateicību;
8. Fantāzija pret realitāti;
9. Tehnoloģija kā draugs pret tehnoloģija kā ienaidnieks .

Arī datorspēles struktūra kaut arī nepilnīgi ir definēta [6]:

1. Noteikumi;
2. Uzdevumi un mērķi;
3. Iznākums un atbildes reakcija;
4. Konflikts/ sacensības/ izaicinājums/ opozīcija;
5. Mijiedarbība.

Vispārīgi var konstatēt, ka šobrīd ir tikai aptuveni apjausti datorspēļu izstrādes principi. Tomēr pie viena kopīga secinājuma ir nonākuši visi šo jomu aprakstījušie autori – spēlei ir jārada prieku. Tiek uzskatīts, ka jautrai spēlei un tās mērķim jāatbilst sekojošiem nosacījumiem [7]:

1. Vienkāršām spēlēm jābūt acīmredzamiem mērķiem. Parasti, jo acīmredzamāks un nepārvaramāks mērķis ir, jo labāk;
2. Sarežģītai spēlei bez iestrādātiem mērķiem jābūt ar tādu struktūru, lai būtu viegli ģenerēt mērķus ar piemērotu sarežģītības līmeni;
3. Labākie mērķi parasti ir praktiski vai fantastiski;
4. Spēlētājam jāvar noteikt, vai viņš atrodas tuvāk mērķim.

Nevienam cilvēkam nebūtu interesanti spēlēt spēli, ja jau iepriekš būtu zināms tās iznākums. Tāpēc svarīgi sekot ieteikumiem, kas palīdz nodrošināt neparedzamu iznākumu, spēlētājiem ar dažādiem prasmju līmeņiem [7]:

5. Dažādi sarežģītības līmeņi:
    a. Noteikti automātiski, ņemot vērā spēlētāja sekmes spēles gaitā;
    b. Sākotnēji piedāvājot izvēli spēlētājam;
    c. Noteikti balstoties uz pretinieka spējām;

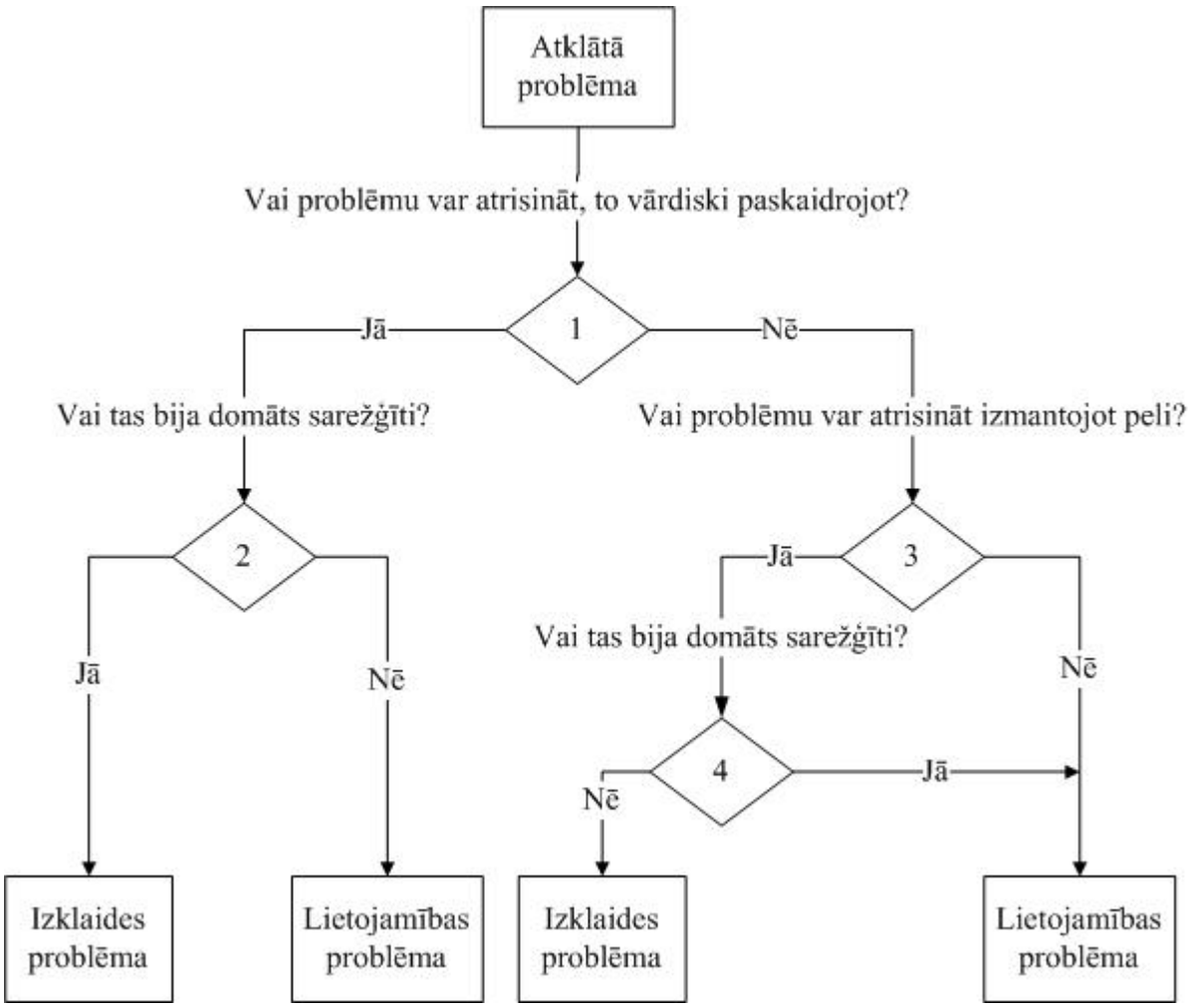
6. Vairāki spēles līmeņi katrs ar savu mērķi vai arī iespēja spēles mērķi uzlabot:
    a. Saglabājot rezultātu;
    b. Palielinot ātrumu;
7. Informācijas slēpšana;
8. Nejaušības princips.

Ļoti svarīgs datorspēles uzdevums ir zinātkāres stimulēšana. Spēles vide nevar būt nedz pārāk sarežģīta, nedz pārāk vienkārša, lai respektētu tās zināšanas, kas spēlētājam jau ir. Tomēr tai jāparāda, ka spēlētāja rīcībā esošā informācija ir nepilnīga vai pretrunīga. Spēlei jāpārsteidz ar kaut ko jaunu un jāpastāsta par to, bet tajā pašā laikā, tā nedrīkst būt nesaprotama, kas arī ir galvenais iemesls, kāpēc spēles ar dažādiem sarežģītības līmeņiem ir tik veiksmīgas. Tās piemērotas daudz plašākai auditorijai.

Arī veiksmīgi izmantojot vizuālos un skaņu efektus, iespējams iespaidot cilvēka zinātkāri. Ir vairāki paņēmieni, kā izmantot šos efektus [7]:

1. Dekorācija (palielina sākotnējo interesi, bet var kļūt garlaicīgi);
2. Nebūtiskās fantāzijas (fantāzijas, kas atkarīgas tikai no spēlētāja prasmēm);
3. Apbalvojums;
4. Attēlošanas sistēma (alternatīvs paziņojumu attēlošanas veids).

Svarīga, protams, ir ne tikai nepieciešamo pazīmju noskaidrošana, bet arī jau gatavu datorspēļu problēmu identificēšana. Ņemot vērā izglītojošo spēļu kvalitātes balstus – lietojamību, kas sevī ietver lietderību, efektivitāti, vajadzību apmierināšanu, un izklaidi ir izstrādāta metode, ar kuras palīdzību var identificēt problēmas apkārtni (sk. 1. att.).



1.1. att.

Lietojamības un izklaides problēmu atšķiršanas algoritms [8]

Diezgan labi ir izdalāmas atšķirības starp formālām – uz noteikumiem balstītām spēlēm un bērnu spēlēm, kas vairāk orientētas uz kaut kā cita simulēšanu. Tāpēc, izstrādājot bērniem domāto spēļu raksturiezīmes, ir svarīgi ņemt vērā gan jau šobrīd pasaulē izveidoto spēļu izstrādes pamatprincipu ievirzes, gan arī mazu bērnu specifiskās vajadzības attīstības veicināšanai.

2. SĀKUMSKOLAS VECUMA BĒRNIEM DOMĀTĀS PROGRAMMATŪRAS RAKSTURIEZĪMES

2.1. Datora ietekmes uz bērna veselību analīzes rezultātā iegūtās bērnu programmatūras raksturiezīmes

Ir noskaidrots, ka datoru lietošana atstāj gan pozitīvu, gan negatīvu ietekmi uz bērnu veselību un komunikācijas spējām. Ļoti svarīgi bērniem izstrādāt tādu programmatūru, kas rada pēc iespējas mazāku kaitējumu veselībai un pēc iespējas lielāku ieguvumu bērna attīstībai.

Izpētot iespējamo negatīvo ietekmi, atlasīju 4 nozīmīgākās bērnu programmatūras raksturiezīmes, kurām vajadzētu pievērst uzmanību:

1. Daudzveidība – svarīgi, attīstot vienu prasmi, nenomākt citu prasmju attīstību;
2. Vizuālais izskats, kustība – sakārtota lietotāja saskarne un vienmērīga kustība samazina redzes noslogojumu;
3. Reālisms – iespēja saistīt programmu ar reālo dzīvi ļauj programmā iegūtās zināšanas izmantot arī ikdienā, komunicējot ar citiem cilvēkiem;
4. Ilgums – uzdevuma izpildīšanai programmā nevajadzētu ilgt vairāk kā 10 - 20 minūtes.

2.2. Mācību metožu izmantojums bērnu programmatūrā

Sistemātisks izklāsts principā jāņem vērā programmatūrā ar vairākiem sarežģītības līmeņiem, lai noteikta līmeņa uzdevumus varētu izpildīt ar iepriekšējā līmeņa zināšanām un noteiktā līmeņa jauno sniegto informāciju. Iespējams vairāk pielietojams izziņas tipa programmās un lasīt protošiem bērniem. Izstrādājot bērnu programmatūru, svarīgi izstrādāt ne tikai pašu programmu, bet arī palīglīdzekļus, kurus bērns var pielietot zināšanu apguvei laikā, kad neizmanto datoru [9]. Šādi palīglīdzekļi ir vēl viens ne mazāk svarīgs sistemātiska izklāsta pielietojums.

Vingrinājumi ir viena no visvieglāk realizējamām mācību metodēm programmatūrā. Konkrēta tipa uzdevuma dažādas variācijas. Tipiskākie piemēri – programmatūra, kas balstīta uz aritmētikas uzdevumu atrisināšanu vai svešvalodas vārdu tulkošanu.

Tomēr izmantojot vingrinājumus, ka mācību metodi programmatūrā, svarīgi atcerēties un nepieļaut kļūdu, kuru pieļāva daudzi sākotnējo izglītojošo programmu izstrādātāji – tās ne ar ko neatšķīrās no uzdevumu risināšanas izmantojot papīru un zīmuli. Kā atbildes reakcija uz uzdevuma risinājumu tika sniegts lēmums – pareizs vai nepareizs. Būtiska ir ne tikai programmas

struktūra, bet arī atgriezeniskā saite, ko programma sniedz nepareizas atbildes gadījumā. Lai nezaudētu bērna interesi, jānorāda ne tikai sniegtās atbildes stāvoklis, bet jānorāda arī pareizais virziens nepareizas atbildes gadījumā [9].

Ekskursijas, pastaigas, novērojumi – šo mācību metodi iespējams realizēt programmatūrā, kas balstīta uz tā saucamajām „Interaktīvo filmu" datorspēlēm, kurās pēc neliela videomateriāla jāizdara vieglas izvēles, vai jāveic noteikti uzdevumi. Šajā gadījumā to varētu nosaukt par pētniecības metodi.

Radoši uzdevumi – izmantojami programmās, kas radītas kā simulācijas, kas saistītas ar zīmēšanu, kombinācijām, veidošanu (ar piedāvāto rīku palīdzību).

Darbs ar grāmatu – metodes pielietojums diezgan nosacīts, izmantojami dažādi grāmatās atrodami elementi – animācijas, stāsti, pantiņi, mīklas u.c. Šajā gadījumā to varētu nosaukt teksta – attēla apstrādes metode.

Diemžēl tādas noderīgas metodes kā pārruna un praktiska darbība šobrīd datorspēlēs nav kvalitatīvi pielietojamas.

2.3. Datorspēļu izstrādes principu pielāgojums sākumskolas vecuma bērnu programmatūrai

Fakts, ka rotaļa un spēle ir minēta, kā svarīgākā mācību metode bērniem, apliecina iespēju veiksmīgi izmantot datorspēļu izstrādes principus bērnu programmatūras izstrādē. Tātad varam pieņemt, ka katra veiksmīga bērnu programma būtībā ir datorspēle, tāpēc turpmāk šajā nodaļā kā piemēru apskatīsim vispārīgo datorspēļu izstrādes principu pielietojumu bērnu datorspēļu izstrādē.

Ņemot vērā faktu, ka jau vairāk kā 25 gadus neapšaubīta ir Tomasa Malones izstrādātā teorija par "izaicinājuma", "fantāzijas", "zinātkāres" nozīmību saistošas datorspēles izveidē un pasaulē veiktos pētījumus, jāsecina, ka arī bērniem domātā programmatūrā šīm lietām ir sava nozīme. Iepazīstoties ar šobrīd lietotajām metodēm datorspēļu izstrādē, esmu izveidojusi paņēmienu tabulu, kas saista datorspēļu struktūras sastāvdaļas ar T. Malones minētajām veiksmīgas datorspēles īpašībām (sk. 1. tabulu).

2.1. tabula.

Izstrādes paņēmieni [10]

Īpašība	*Uzdevumi, mērķi*	*Iznākums, atbildes reakcija*	*Mijiedarbība*
Izaicinājums	*Skaidrs mērķis, vairāku līmeņu mērķi, dažādi sarežģītības līmeņi*	*Neparedzams iznākums*	*Apjaušama esošā pozīcija*
Fantāzija	*Metaforas, emocionāli saistošas*	*Metaforas, emocionāli saistošas*	*Metaforas, emocionāli saistošas*
Zinātkāre	*Uz zināšanu nepilnībām norādošs*	*Nejaušības princips, vizuālie un skaņas efekti, atbilstošs humora līmenis, jaunas informācijas sniegšana*	*Nejaušības princips, vizuālie un skaņas efekti, atbilstošs humora līmenis, jaunas informācijas sniegšana*

Izaicinājums šajā situācijā apzīmē uzdevuma izpildi. Tiesa gan tas var nebūt raksturīgs spēlēm, kas veidotas kā kaut kā simulācija. Uzdevumam, protams, jābūt skaidram, vēlams praktiski izpildāmam, jo, kā zināms, bērni labāk atceras un apgūst redzētas, fiziski izdarītas lietas. Nozīmīgi bērnam ir apjaust, cik tuvu viņš ir uzdevuma pareizajam atrisinājumam, jo, neiegūstot informāciju par izdarītajām kļūdām, spēle bērnam var kļūt nesaprotama un viņa uzmanību nesaistoša. Tā kā bērnu attīstība pirmsskolas vecuma grupā ir straujākā visā cilvēka mūžā, tad datorspēlei būtu jābūt nodrošinātai ar diezgan daudz sarežģītības līmeņiem, lai apmierinātu spēlētājus ar atšķirīgām spējām. Svarīgi būtu, lai nākošais līmenis balstītos uz iepriekšējā līmenī iegūtajām zināšanām, neradot nesaprotamas situācijas. Nodrošinot neparedzamu iznākumu, tiek radīta interese spēli nospēlēt līdz galam, kas ļauj realizēt pilnīgas informācijas sniegšanu, piemēram, spēle, kuras galvenais mērķis ir izskaidrot bērnam dažādu putnu īpašības un atšķirības. Nenospēlējot spēli līdz galam, bērns var uzzināt putniem raksturīgāko īpašību – spēju lidot, bet neuzzināt par to, ka ir arī tādi putni, kas nelido.

Fantāzija saistīta ar patiesībā neesošu fizisku objektu vai sabiedrisku situāciju iztēlošanos. To varētu arī nosaukt par uzdevuma vizuālo attēlojumu. Cilvēku dažādības dēļ, būtu svarīgi izvēlēties atbilstošās, viņus saistošās fantāzijas. Vislabākajā gadījumā vienai spēlei būtu jāsatur vairākas fantāziju iespējas, lai lietotāji varētu izvēlēties. Kaut vai par piemēru ņemot elementāru

spēli, kas saistīta ar kāda dzīvnieka apkopšanu, kāds šī zvēra lomā labprātāk redzētu kaķi, nevis suni, kāds tieši pretēji. Ja fantāziju saista ar dažādo atbildes reakciju iespējamību un mijiedarbību starp spēli un spēlētāja rīcību, tad tā ir īpaši nozīmīga maziem bērniem, kas tikai nesen sākuši apjaust cēloņu un seku sakarības, kā arī lielākoties mācās tieši no sniegtajiem paraugiem. Tāpēc labs fantāzijas realizācijas veids ir metafora – kādas zināmas situācijas vai darbības pielietošana citas situācijas vai darbības pasniegšanai.

Zinātkāre patiesībā jau ir lielāko daļu bērnu raksturojoša īpašība. Bērnus interesē viss jaunais, neredzētais un nedzirdētais. Tieši tāpēc šeit lielu nozīmi spēlē arī vizuālie un skaņas efekti. Visplašākais to izmantošanas veids var būt dažādos apbalvojumu un alternatīvās attēlošanas paņēmienos, jo bērni vēl joprojām ļoti labi uztver gan krāsu, gan intonāciju nozīmi, daudzreiz pat labāk par tekstā pasniegtu informāciju. Spēlei svarīgi norādīt uz to, ko bērns vēl nav apguvis, kā arī, lai neradītu nesaprotamas situācijas un nezaudētu bērna uzmanību, piedāvāt apgūt jaunās, neskaidrās lietas. Liela nozīmes zinātkāres veicināšanai ir arī nejaušības principam visā spēles gaitā, kas katru spēlēšanas reizi padara gandrīz unikālu, tādējādi izslēdzot rutīnu, kas parasti ātri vien kļūst garlaicīga.

3. IZGLĪTOJOŠO PROGRAMMĒŠANAS VALODU IESPĒJAS

Šobrīd ir izstrādātas vairākas programmēšanas valodas, kuru mērķis ir nodrošināt lietotājam draudzīgu vidi, ar kuras palīdzību iespējams apgūt programmēšanas pamatus. Lai novērtētu šādu valodu sniegto programmēšanas pamatzināšanu pārklājumu un citas to piedāvātās iespējas, tika apskatītas divas no šāda veida programmēšanas valodām – LOGO un RoboMind. Iemesls, kāpēc tika izvēlētas tieši šīs valodas, balstīts uz faktu, ka LOGO ir viena no pirmajām izglītojošajām programmēšanas valodām, kas radīta kā izglītojošs rīks, un vairāk kā 40 gados vēl joprojām nav zaudējusi savu aktualitāti [12]. Uz LOGO idejām balstīta arī lielākā daļa vēlāk radīto izglītojošo programmēšanas valodu. Kā otrā apskatāmā valoda izvēlēta RoboMind, jo tās izstrādātāji apgalvo, ka šo valodu radījuši, lai novērstu LOGO vidē esošos trūkumus vai nepilnības [13].

3.1. Programmēšanas pamati

Ļoti bieži programmēšana tiek uztverta tikai kā programmas koda uzrakstīšana. Tomēr var izdalīt vēl vairākus programmēšanas procesa posmus, kas ir vienlīdz svarīgi [11]:

1. Uzdevuma identificēšana;
2. Risinājuma plāns – svarīgi pārzināt dažādu programmēšanas valodu stiprās un vājās puses, lai izvēlētos konkrētajam uzdevumam piemērotāko programmēšanas valodu;
3. Kodēšana – saistīta ne tikai ar programmas pierakstīšanu, bet arī ar uzdevuma sadali apakš uzdevumos, tādējādi atvieglojot vai padarot iespējamu tā izpildi;
4. Testēšana un palaišana – iekļauj sevī tādu nozīmīgu procesu kā atkļūdošana.

Lai iegūtu pietiekamu priekšstatu par programmēšanu, nepieciešams iepazīties ar sekojošiem programmēšanas elementiem un to nozīmi [11]:

1. Vārdi, mainīgie, konstantes:

   Vārds parasti ir simbolu virkne, kuru izmanto, lai identificētu kādu programmas elementu. Vārdi nevar pieņemt kādu iepriekšdefinētu programmēšanas valodas vērtību. Mainīgie ir programmas elementi, kas var manīt savu vērtību, turpretī konstantes ir programmas elementi, kuriem ir nemainīga vērtība visā to dzīves ciklā. Mainīgos var iedalīt atkarībā no tā, kur tie tiek deklarēti – lokālajos, globālajos un ārējos. Lokālie mainīgie ir tādi, kas deklarēti funkcijas iekšienē. Globālie mainīgie deklarēti ārpus funkcijas un pieejami visās funkcijās faila ietvaros. Ārējie mainīgie principā ir globālie mainīgie, kas lietojami vairākos failos.

2. Deklarācija:

   Deklarācija nosaka, kā konkrēts mainīgais tiks lietots programmā.

3. Tipu pārbaude, konversija, savienojamība:

   Tipu pārbaude nepieciešama, lai pārliecinātos, ka katra izpildāmā programmas operācijas saņem piemērotu argumentu skaitu ar piemērotiem datu tipiem. Situācijā, kad padotais arguments neatbilst sagaidāmajam datu tipam, var tik izdots kļūdas paziņojums vai veikta nepieciešamā argumenta konversija.

4. Datu tipi, atmiņas pārvaldība:

   Datu tipus iespējams iedalīt – elementāros, strukturētos un abstraktos datu tipos. Elementārie datu tipi ir pamata datu tipi. Strukturēti datu tipi ir vairāku elementāru datu tipu vai vairāku strukturētu datu tipu kompozīcija. Abstrakti datu tipi ir tādi datu tipi, kuriem piemīt gan elementāru, gan strukturētu datu tipu īpašības.

   Svarīgi nelietot pārāk daudz atmiņas vai vismaz lietot to, cik efektīvi vien iespējams.

5. Izteiksmes un piešķiršanas priekšraksti:

   a. Aritmētiskās izteiksmes:

      Aritmētiskās izteiksmes daudz neatšķiras no parastām matemātiskām izteiksmēm. Tiek izmantotas tādas zīmes kā: +, - , *, /.

   b. Pārslogoti operatori:

      Pārslogoti ir operatori ir zīmes, kas var tikt lietotas ar vairākām nozīmēm, piemēram, zīme „+” vairākās valodās tiek izmantota gan palielināšanai, gan saskaitīšanai, gan savienošanai;

   c. Attiecību izteiksmes;

      Attiecību izteiksmes nosaka, kādas ir operandu attiecības. Tiek izmantotas tādas zīmes kā: >, <, =, <>, <=, >=;

   d. Loģiskās un nosacījuma izteiksmes:

      Nosacījuma izteiksmes pieļauj darbību tikai noteiktās situācijās vai apstākļos;

   e. Īsslēguma (īssavienojuma) vērtējums:

      Ir gadījumi, kas izteiksme nemaz netiek novērtēta, piemēram, Būla izteiksme, kas sastāv no diviem operandiem, kas saistīti ar „and”. Ja pirmais izteiksmes operands būs aplams, tad ļoti iespējams, ka otrais operands nemaz netiks novērtēts;

   f. Piešķiršanas priekšraksti:

      Piešķiršanas priekšraksti darbojas no labās puses uz kreiso, piemēram, situācijā A=B=C, C vērtība tiks piešķirta B un tad B vērtība tiks piešķirta A.

6. Priekšrakstu izpildi kontrolējošās struktūras:
    a. Secība:

       Struktūras, kas nosaka operāciju izpildes secību. Var tikt izmantotas gan izteiksmēs, gan starp priekšrakstiem un priekšrakstu grupām.
    b. Izvēle:

       Izvēle, atbilstoši nosacījumam, piedāvā izvēlēties vienu no vairākiem priekšrakstiem, kuru izpildīt.
    c. Atkārtošana:

       Atkārtošana atļauj izpildīt konkrētus priekšrakstus noteiktu skaitu reižu, vai nu tik ilgi, kamēr nosacījums izpildās.
7. Apakšprogrammas:
    a. Parametri, parametru nodošanas metodes:

       Apakšprogrammas parametriem jāatbilst parametriem apakšprogrammas definīcijā. Ir vairākas parametru nodošanas metodes – pēc vērtības, pēc adreses, pēc vērtības un rezultāta un pēc vārda.
    b. Rekursīvas apakšprogrammas:

       Rekursīva apakšprogramma būtībā ir programma, kas izsauc pati sevi.

3.2.Programmēšanas valoda LOGO

Programmēšanas valoda LOGO tika radīta kā izglītojošs rīks. No šī mērķa izriet arī LOGO īpašības - mijiedarbība, modularitāte, datu tipu pielāgojamība. Ja sākotnēji LOGO tika izveidots kā bruņurupuča izskata robots, kas atbilstoši komandām zīmēja zīmējumu uz virsmas, tad šobrīd tas ir bultiņas tipa kursors, kas kustās pa datora ekrāna virsmu [12].

Izplatītākās LOGO komandas [12]:

1. Zīmuļa un krāsas komandas:
    a. Pacelt zīmuli: PU;
    b. Nolaist zīmuli: PD;
    c. Norādīt zīmuļa izmēru: SetPenSize[n n];
    d. Norādīt zīmuļa krāsu: SetPC [r g b];
    e. Izmantot zīmuli kā dzēšgumiju: Penerase, pe;
    f. Izmantot zīmuli atkal kā zīmuli: Pennormal;
    g. Norādīt apgabala krāsu: setfloodcolor [r g b];
    h. Izkrāso apgabalu, kas ierobežots ar līnijām, ar iepriekš norādīto krāsu: fill.

2. Kustības, zīmēšanas komandas:
    a. Pārvietoties uz priekšu par n pikseļiem: FD n;
    b. Pārvietoties atpakaļ par n pikseļiem: BK n;
    c. Pagriezties pa labi par n grādiem: RT n;
    d. Pagriezties pa kreisi par n grādiem: LT n;
    e. Uzzīmēt loku ar ietverto leņķi a un rādiusu r (pabeidzot komandu, bruņurupucis atrodas loka vidū): ARC a r;
    f. Uzzīmēt loku ar ietverto leņķi a un rādiusu r (pabeidzot komandu, bruņurupucis atrodas loka beigās): ARC2 a r;
3. Bruņurupuča un pozīcijas komandas:
    a. Parādīt pašreizējo bruņurupuci: ST;
    b. Paslēpt pašreizējo bruņurupuci: HT;
    c. Atgriež bruņurupuča orientāciju: Orientation;
    d. Atgriež bruņurupuča pozīciju: POS;
    e. Norāda bruņurupuča orientāciju: setorientation[roll pitch heading]
    f. Norāda bruņurupuča pozīciju: setpos[x y]
    g. Izveido n bruņurupučus, ja tie jau nav izveidoti, pirmais ir 0. bruņurupucis, maksimālais bruņurupuču skaits 1023: SetTurtle n.
4. Programmas komandas:
    a. Procedūra:

        To name arg1 arg2 ... argN

            ... instructions...

        End
    b. Dzēš ekrānu un novieto bruņurupuci tā sākotnējā pozīcijā: CS;
    c. Atkārto norādījumus kvadrātiekavās n reizes:

        Repeat n

        [

            ... instructions...

        ]
    d. Parāda komandieri: Show;
    e. Apstājas uz n/60 sekundēm: Wait n;
5. Matemātiskās un mainīgo komandas:
    a. Atgriež patvaļīgu skaitli, kas mazāks par n: Random n;

b. Izveido mainīgo name un piešķir vērtību x: Make "name x;

6. Programmas plūsmas komandas:
    a. Klasiskais for cikls, i ir pašreizējā skaitītāja mainīgais, start - vērtība, no kuras jāsāk, stop – vērtība, līdz kurai jāizpilda, step – solis (ja nav norādīts, tad tiek uzstādīts kā 1 vai -1): FOR [i start stop step][... instructions...]
    b. If nosacījums, test – izteiksme, kurai nosaka patiesa vai aplama: IF test ...instructions...
    c. While nosacījums, list – izteiksmju saraksts, kuram nosaka patiess vai aplams: WHILE list ...instructions...

3.3.Programmēšanas valoda RoboMind

Programmēšanas valoda RoboMind ir jauna un ļoti vienkārša izglītojoša programmēšanas valoda, kas ir radīta kā alternatīva LOGO vides bruņurupucim. Ar tās palīdzību iespējams kontrolēt robota uzvedību RoboMind izstrādes vidē, tajā pašā laikā apgūstot datoru zinātnes un programmēšanas pamatus. RoboMind izstrādātāji izdala vairākus nozīmīgus iemeslus, kāpēc bija nepieciešama jauna modificēta vides versija [13]:

1. Lielākā daļa LOGO implementāciju ir pārsniegušas sākotnēji izvirzīto mērķi: ievads programmēšanā;
2. Neestētiska sintakse;
3. Vides mijiedarbības trūkums.

RoboMind izstrādes mērķis bija iegūt pēc iespējas lietotājam draudzīgāku izstrādes vidi, izmantojot patīkamus vizuālus efektus un vienkāršu, ne bezjēdzīgu saskarni. RoboMind vidē izvēlēts imperatīvai programmēšanas valodai raksturīgs, tradicionālāks sintakses veids. RoboMind valodai raksturīgs minimālisms, tajā nav pat operatoru un mainīgo. Tādā veidā robots teorētiski līdzinās Tjūringa mašīnai. Ja LOGO bruņurupucis vairāk koncentrējas uz zīmēšanu, tad RoboMind robots spēj uztvert apkārtējo vidi, tādā viedā paverot ceļu daudz un dažādām interesantām programmām (piemēram, līniju sekošana)[13].

Izplatītākās RoboMind komandas [13]:

1. Nosacījumu un ciklu konstrukcijas:
    a. Atkārto norādījumus figūriekavās tieši n reizes:

```
repeat(n)
{
        …instructions…
```

}

b. Bezgalīgi turpina atkārtot norādījumus figūriekavās:

```
repeat()
{
        …instructions…
}
```

c. Turpina izpildīt norādījumus figūriekavās tik ilgi, kamēr izpildās nosacījums:

```
repeatWhile(condition)
{
        ...instructions...
}
```

d. Ja izpildās nosacījums, izpilda norādījumus figūriekavās:

```
if(condition)
{
        ...instructions...
}
```

e. Ja izpildās nosacījums, izpilda norādījumus pirmajās figūriekavās, pretējā gadījumā izpilda norādījumus otrajās figūriekavās:

```
if(condition)
{
        ...instructions...
}
else
{
        …instructions…
}
```

2. Loģiskās operācijas (noliegums ar augstāko prioritāti):
    a. Noliegums: not, ~ ;
    b. Un: and, & ;
    c. Vai: or, | .
3. Iebūvētās funkcijas:
    a. Paiet n soļus uz priekšu: forward(n);
    b. Paiet n soļus atpakaļ: Backward(n);

c. Pagriezties pa kreisi: Left();
d. Pagriezties pa labi: Right();
e. Paiet n soļus uz ziemeļiem, dienvidiem, austrumiem vai rietumiem: North(n), South(n), East(n) vai West(n);
f. Krāsot baltu vai melnu: paintWhite(), paintBlack();
g. Beigt krāsot: stopPainting();
h. Pacelt, nolikt bāku: pickUp(), putDown();
i. Mest monētu (atgriež vērtību – patiess vai aplams): coinFlip();
j. Apkārtnes novērtēšana (pa kreisi, uz priekšu, pa labi):
    (i) Šķērslis: leftIsObstacle(), frontIsObstacle(), rightIsObstacle();
    (ii) Brīvs: leftIsClear(), frontIsClear(), rightIsClear();
    (iii) Bāka: leftIsBeacon(), frontIsBeacon(), rightIsBeacon();
    (iv) Balts: leftIsWhite(), frontIsWhite(), rightIsWhite();
    (v) Melns: leftIsBlack(), frontIsBlack(), rightIsBlack().

4. Citi norādījumi:
    a. Komentāri tiek norādīti izmantojot # simbolu:
    # Komentārs
    b. Lai pārtrauktu cikla norādījumu izpildi, jāizmanto komanda break.
    c. Lai pārtrauktu programmas izpildi, jāizmanto komanda end.
    d. Iespējams izvedot savas procedūras ar nevienu vai vairākiem parametriem:

```
procedure name (par1, par2, ..., parN)
{
        ...instructions...
}
```

3.4.Izglītojošo valodu LOGO un RoboMind iespēju novērtējums

Lai gan abas programmēšanas valodas ir izglītojošās programmēšanas valodas, to iespējas un pielietojums atšķiras. Tāpēc arī izvēloties mācību līdzekli svarīgi identificēt uzdevumu un izvēlēties piemērotu risinājuma plānu. Kā piemērus apskatīsim kvadrāta un trīsstūra grafisku attēlošanu:

3.1. tabula.

Kvadrāta, trīsstūra attēlošana

Programmēšanas valoda	*Kvadrāts*	*Trīsstūris*
LOGO	*Repeat 4[forward 40 left 90]*	*Repeat 3 [forward 40 left 120]*
RoboMind	*paintBlack()* *repeat(4)* *{* *forward(4)* *left()* *}*	

Nenovērtējot programmēšanas valodas iespēju atbilstību veicamajam uzdevumam, rezultāts ir neparedzams. Piemērā minētajam uzdevumam izvēloties programmēšanas valodu RoboMind risinājuma otrajā fāzē – trīsstūra grafiska attēlošana, programmētājs nonāk situācijā, kad uzdevuma realizācija nav iespējama, jo programmēšanas valoda RoboMind piedāvā iespēju pagriezties tikai par 90 grādiem. Lai arī pēc uzdevuma pirmās fāzes – kvadrāta grafiska attēlošana, izpildes, šīs valodas programmas kods šķiet uzskatāmāks.

Varētu iebilst, ka tā kā izglītojošajai programmēšanas valodai ir jānodrošina programmēšanas pamatprincipu apguve, tad svarīgāko lomu spēlē programmēšanas pamatu pārklājums izvēlētajā valodā, nevis spēja realizēt kādu konkrētu uzdevumu. Turpretī, apskatot šo jautājumu bērnu programmatūras aspektā, jāsecina, ka ierobežots realizējamo uzdevumu daudzums cieši saistīts arī ar fantāzijas ierobežojumu.

Apskatot elementus, ar kuriem parasti iepazīstas cilvēki, lai iegūtu sākotnējo priekšstatu par programmēšanu, varam novērtēt izglītojošo programmēšanas valodu sniegto zināšanu līmeni.

Programmēšanas valoda LOGO dod iespēju iepazīties ar vārdu un mainīgo nozīmi programmēšanā. Piemēram, iespējams izveidot mainīgo „step” ar vērtību 1:

make "step 1

Tā kā šajā programmēšanas valodā datu tipi netiek atdalīti, tad tādā pašā veidā iespējams mainīgajam „step” piešķirt vārdisku vērtību „name”:

make "step [name]

Tāpat dotais mainīgais var pieņemt vērtību, kas atbilst strukturēties datu tipiem:

```
make "step [one two three]
```

Tā kā mainīgie netiek deklarēti, tie programmas izpildes laikā var pieņemt gan skaitlisku, gan vārdisku, gan strukturētu vērtību. Programmējot valodā LOGO, var iepazīties ar aritmētiskajām, attiecību, loģiskajām un nosacījuma izteiksmēm, kā arī apgūt populārākās priekšrakstu izpildi kontrolējošās struktūras. LOGO apakšprogrammām piedāvā parametru nodošanu pēc vērtības.

Valodā RoboMind piedāvātās iespējas ir ierobežotākas. Ne tikai netiek izdalīti konkrēti datu tipi, bet netiek piedāvāts arī darbs ar mainīgajiem. Iepazīšanai ar vārdu nozīmi programmēšanā tiek izmantotas procedūras:

```
procedure square (side)
{
    repeat(4)
    {
        forward(side)
        left()
    }
}
```

Arī no izteiksmēm un piešķiršanas priekšrakstiem atbalstītas tiek tikai loģiskās operācijas. Bet populārākās priekšrakstu izpildi kontrolējošās struktūras, kaut gan mazliet modificētā veidā, tiek aprakstītas.

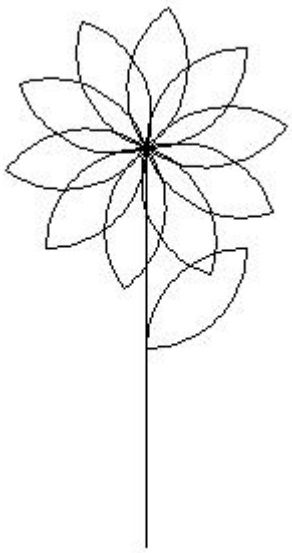
Salīdzinot programmēšanas valodu vārdnīcas, sākumā var likties, ka valodu ar mazāku vārdu krājumu ir vieglāk apgūt. Tomēr arī šeit jāņem vērā bērnu specifika. Piemēram, ja bērniem dzimtās valodas apguvei tiktu radīta valoda, kura sastāvētu tikai no 50 vārdiem. Šie vārdi būtu izvēlēti tik labi, lai ar to palīdzību būtu iespējams pateikt visu, ko vien vēlas. Iespējams vārdu krājumu apgūt būtu vieglāk, bet tikai paši apdāvinātākie bērni iemācītos pielietot šos vārdus. Neatkarīgi no vārdnīcas lieluma, programmēšanas valoda principā ir valoda. Un valodas ir tas, ko bērni apgūst visvieglāk, tāpēc programmēšanas valodas ir veiksmīgi pielietojamas agrā vecumā [14]. Protams, angliski runājošiem bērniem, izglītojošās programmēšanas valodas apgūt ir vieglāk, jo komandu nosaukumi ir ar savu nozīmi viņu dzimtajā valodā - „if ... then" kā „ja ... tad". Bet programmēšanā tas nerada valodas barjeru, jo principā bērns mācās valodu, ar kuras palīdzību cilvēks var sazināties ar datoru.

Izglītojošo programmēšanas valodu apguve cieši saistīta arī ar citu jomu zināšanu apguvi, jo dators var runāt gan matemātiskā, gan alfabētiskā valodā. Piemēram, valodā RoboMind

izmantojot komandas North(n), South(n), East(n) vai West(n) papildus programmēšanai iespējams apgūt arī ģeogrāfiskos virzienus.

Skolās bieži tiek mācīts, ka kļūdīties ir slikti. Tāpēc reti kurš vēlas pētīt kļūdas, kavēties pie tām vai domāt par tām. Bērni ļoti priecājas par datora spēju izdzēst kļūdas, lai neviens tās neredzētu. Tomēr „lāgošanas" filozofija iesaka pretēju attieksmi. Tā vietā lai aizmirstu kļūdu, programmētājs tiek mudināts izpētīt to, lai izprastu, kas notika nepareizi, lai varētu to izlabot. Jautājums, kas jāuzdod ir nevis vai programma ir pareiza vai nepareiza, bet vai tā ir izlabojama. Programmēšana cīnās ar bailēm kļūdīties, kas valda lielākajā daļā kultūru [14]. Tomēr bailes pastāv, un, lai ar tām cīnītos, nepieciešama vide ar mazāku pareizs, nepareizs dominanti. Programmēšanas valoda LOGO to piedāvā intuitīvi. Kā piemēru var minēt situāciju, kad bērni, kas jau kādu laiku strādājuši ar LOGO programmu, izmantojot divas apļa ceturtdaļas, grib uzzīmēt ziedlapiņu [14]:

1. Mērķis uzzīmēt ziedu:



3.1. att.

Uzdevums

2. Lai varētu uzzīmēt ziedu, svarīgi no sākuma iemācīties uzzīmēt ziedlapiņu. Tā kā ziedlapiņa sastāv no divām apļa ceturtdaļām, bērni secina, ka varētu izmantot jau iepriekš uzrakstītu un izmantotu funkciju QCIRCLE, kas, saņemot vienu argumentu, uzzīmē norādītā izmēra apļa ceturtdaļu:

   QCIRCLE 50

   QCIRCLE 50

3.2.att.

1. mēģinājums

3. Pēc programmas izpildes iznākums liekas loģisks – divas apļa ceturtdaļas veido pusapli. Var izsecināt – lai sasniegtu rezultātu, pēc pirmās ceturtdaļas uzzīmēšanas „bruņurupucim" jāveic pagrieziens. Tiek veikts eksperiments ar pagriezienu pa kreisi:

   QCIRCLE 50

   LEFT 120

   QCIRCLE 50

3.3.att..

2.mēģinājums

4. Lai gan rezultāts nesakrīt ar iecerēto un teorētiski to var uztvert kā kļūdu, iznākums ir pozitīvs – ir iegūts programmas teksts, ar kura palīdzību iespējams uzzīmēt putnu. Nākošais secinājums - pagrieziens jāveic uz labo pusi:

   QCIRCLE 50

   RIGHT 120

   QCIRCLE 50

3.4.att.

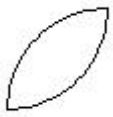
3.mēģinājums

5. Lai gan iegūtais rezultāts vēl joprojām nav tāds, kāds bija gaidīts, arī šis zivs piemērs ir vēlāk izmantojams. Tālāk, atkarībā no bērna zināšanām, nedaudz padomājot vai paeksperimentējot ar skaitļiem tiek iegūts vēlamais rezultāts:

   QCIRCLE 50

   RIGHT 90

   QCIRCLE 50



3.5.att.

Apakš uzdevuma risinājums

Programmēšana ir process, kurā bērns var brīvi izpausties – tajā nav konkrētu uzdevumu un arī noteiktu risinājumu, tātad nav arī stingri atdalīts – pareizs un nepareizs. Programmēšana diezgan brīvi ļauj bērnam pašam izdomāt kādu uzdevumu viņš veiks. Iepriekšminētajā piemērā bērns, mēģinot uzzīmēt ziedlapiņu un uzzīmējot zivi, netiek ierobežots ar konkrēta uzdevuma nostādnēm. Viņš var turpināt mēģināt uzzīmēt ziedlapiņu vai arī izmanīt uzdevuma noteikumus un zīmēt zemūdens pasauli. Bērna fantāzija netiek ierobežota.

Tomēr mēģinot atrisināt problēmu būtiski uzdot divus jautājumus: Vai šī problēma var tikt sadalīta mazākās problēmās? Vai šī problēma var tikt attiecināta uz kādu problēmu, kuru es jau protu atrisināt [14]? Šīs metodes pielietojums acīmredzams arī iepriekšminētajā piemērā. Problēma – uzzīmēt ziedu. Apakš problēma – uzzīmēt ziedlapiņu. Vai ziedlapiņas uzzīmēšanas problēma ir attiecināma uz kādu problēmu, kuru es jau protu atrisināt? Ziedlapiņa sastāv no 2 apļa ceturtdaļām, kuras iespējams uzzīmēt izmantojot funkciju QCIRCLE.

Varam secināt, ka RoboMind izstrādātāju izvirzītā problēma - lielākā daļa LOGO implementāciju ir pārsniegušas sākotnēji izvirzīto mērķi, patiesībā nav uztverama kā problēma. Piedāvātās iespējas ļauj viegli nodrošināt vienu no bērnu programmatūras svarīgākajām raksturiezīmēm – vairāki sarežģītības līmeņi.

Protams, RoboMind sintakse šķiet intuitīvāk saprotama un pārskatāmāka cilvēkiem, kas jau kādreiz saskārušies ar programmēšanu, bet cilvēkiem, kas nekad dzīvē nav programmējuši vienlīdz laba šķiet abu piedāvāto programmu sintakse.

4. IZGLĪTOJOŠO PROGRAMMĒŠANAS VALODU PIEMĒROTĪBA SĀKUMSKOLAS VECUMA BĒRNIEM

Bērnam domātai programmai principā vienlaicīgi jābūt arī spēlei. Vai izglītojošās programmēšanas valodas ir piemērotas sākumskolas vecuma bērniem un vai tās var būt kā spēles? Apskatīsim programmēšanas atbilstību datorspēļu izstrādes principiem, kuri izmantojami sākumskolas vecuma bērnu programmatūrā:

1. Skaidrs, praktiski izpildāms uzdevums – programmēšanā viegli realizējam prasība. Tomēr, lai izglītojošo programmēšanas valodu varētu izmantot, kā spēli, nepieciešams izstrādāt vairāku uzdevumu komplektu, tādējādi radot mērķi, kuru bērnam jāsasniedz;
2. Esošā situācija – tā kā programmēšana ir radošs process, diemžēl nevar tikt sniegta informācija par to, cik drīz mērķis tiks sasniegts;
3. Vairāki sarežģītības līmeņi – arī šo prasību iespējams nodrošināt veiksmīgi izstrādājot iepriekšminēto uzdevumu komplektu;
4. Informācijas nodrošināšana – lai arī lielākoties visās programmēšanas vidēs tiek nodrošināta palīdzība, informācija netiek sniegta tiešā veidā un bieži ir bērnam nepiemērotā formātā vai ne pietiekoši detalizēta;
5. Neparedzams iznākums – tiek nodrošināts pašā programmēšanas procesā;
6. Dažādas realizējamās fantāzijas – izvēloties atbilstošu uzdevumu tematiku, var saistīt gan dažāda dzimuma, gan dažāda vecuma lietotājus;
7. Mijiedarbība – tiek nodrošināta tiešā veidā, uzrakstot noteiktas komandas, programma dod noteiktu rezultātu;
8. Metafora – saistīta ar informācijas pasniegšanas veidu. Tā kā programmēšanas valodas nodrošina vidi, kurā darboties, bet jaunu informāciju nesniedz, metaforas izmantojums minimāls. Daudzās programmēšanas valodās tiek izmantota metaforai tuvā personifikācija - „dators”, ar kuru bērns „sarunājas”, identificēts ar kādu personu (bruņurupuci, robotu u.c.);
9. Vizuālie un skaņas efekti – atkarīgi no programmēšanas vides. Bet tā kā lielākā daļa izglītojošo programmēšanas valodu saistītas ar attēlu un animāciju veidošanu, tad galarezultātā tiek nodrošināti gan vizuālie, gan skaņas efekti;
10. Nejaušības princips – programmējot notikumu gaita ir neparedzama, jo atkarīga no paša bērna. Tādējādi, strādājot ar izglītojošajām programmēšanas valodām, spēles vai darbības scenārijs katru reizi ir unikāls. Pieļautās kļūdas rada neparedzamu rezultātu, kas liek

bērnam aizdomāties par to, kas, kā un kāpēc notika. Katra jauna ideja, ko bērns vēlas realizēt, visbiežāk norāda uz nepilnībām viņa zināšanās, tādējādi motivējot pilnveidot tās.

Apskatot pārējos faktorus, kas var ietekmēt programmatūras atbilstību sākumskolas vecuma bērniem, iespējams vēl izdalīt sekojošus izglītojošo programmēšanas valodu novērtēšanas kritērijus:

1. Daudzveidība – izmantojot programmēšanas valodas, iespējams apgūt zināšanas arī citās jomās, piemēram, ģeometrijā;
2. Ilgums – laiks, ko bērns pavada programmējot principā ir atkarīgs no veicamā uzdevuma. Tātad atbilstoši piemeklējot uzdevumu, izglītojošā programmēšana var tikt izmantota sākumskolas vecuma bērniem;
3. Mācību metožu pielietojums – programmēšanā plaši var tikt izmantotas gandrīz visas mācību metodes. Vingrinājumi – dažāda veida uzdevumi, kas jārealizē, vai arī vairāki uzdevumi, kuri paredzēti noteiktas prasmes apguvei. Radoši uzdevumi un teksta un attēla apstrāde ir pamatuzdevumi, uz kuriem balstīta izglītojošā programmēšana. Sistemātiska izklāsta pielietojums izglītojošajās programmēšanas valodās ir minimāls. Tomēr būtu jāpievērš būtiska uzmanība tā potenciālajām izmantošanas iespējām. Tāpat kā lielākajai daļai bērnu programmatūras arī izglītojošajām programmēšanas valodām trūkst palīglīdzekļu, kuri būtu piemēroti tieši bērniem.

4.1. Programmēšanas valodu LOGO un RoboMind atbilstība sākumskolas vecuma bērniem domātas programmatūras raksturiezīmēm

Arī LOGO un RoboMind tāpat kā citās izglītojošajās programmēšanas valodās iespējams realizēt dažāda veida uzdevumus. Piemēram, jau iepriekšminētā figūru zīmēšana. Mēģinot attēlot trīsstūri un kvadrātu LOGO vidē, bērns var apgūt:

Ģeometrijas elementus – punkts, attālums, leņķis;

Programmēšanas elementus – ciklu konstrukcijas, atsevišķas programmēšanas valodas komandas.

Apskatīsim šī visnotaļ elementārā piemēra iespējamo attīstību. Nākošais uzdevuma līmenis var tikt saistīts ar procedūru, vārdu un mainīgo apguvi. Nosacījums: uzrakstīt programmu, kas attēlo norādītā izmēra kvadrātu:

```
to square :size
        Repeat 4[forward :size left 90]
end
```

Turpinājums šim piemēram var būt papildus nosacījums par krāsu:

```
to colorSquare :size :color
        SetPC  :color
        Repeat 4[forward :size left 90]
end
```

Tāpat nākošais uzdevums var būt uzrakstīt programmu, kas uzzīmē regulāru figūru ar noteiktu malu skaitu:

```
to figure :count
        Repeat :count [forward 40 left 360/:count]
end
```

Apskatot figūru attēlošanas piemēru, acīmredzams, ka programmēšana ir spējīga nodrošināt ļoti daudzus sarežģītības līmeņus, kas ir viens no izaicinājuma nodrošināšanas balstiem.

Diemžēl programmēšanas valodas nenodrošina jaunas informācijas sniegšanu bērniem piemērotā veidā, kā arī nenorāda pareizo virzienu kļūdas gadījumā. Arī šajā jau minētajā piemērā, ja pieaugušajam pietiks ar komandas „left n" paskaidrojumu - pagriezties pa kreisi par n grādiem, tad bērnam šāds informācijas daudzums ir nepietiekams.

Tā kā programmēšana ir radošs process, gan LOGO, gan RoboMind vidē bērnam tiek dota izvēles brīvība, kāda veida uzdevumu viņš veiks, tādējādi nodrošinot fantāzijas neierobežotību.

Lai bērns varētu kaut ko viegli iemācīties, viņam nepieciešams modelis uz ko balstīt jaunās zināšanas. Ja viņam šāda modeļa nav, jaunās vielas apguve sagādās lielas grūtības [14]. Abās programmēšanas valodās tiek piedāvāts personificēt sevi ar darbojošos objektu – „bruņurupuci" vai robotu. Bez tam sarežģītāku uzdevumu veikšanā - datoru grafikas, mūzikas vai simulēta gaisa kuģa lidojuma programmēšanā, LOGO dod iespēju bērnam asociēt sevi ar kādu no pieaugušajiem, ko bieži nepiedāvā mācību metodes, kuras tiek izmantotas skolās.

LOGO lietotāja saskarne ir sakārtota un nesatur liekus elementus, bet diemžēl bērniem tā var šķist pārāk vienmuļa krāsu trūkuma dēļ. RoboMind lietotāja saskarne ir bērniem draudzīga. Tā nav ar elementiem pārsātināta, tomēr imitē reālo vidi, tādējādi saistot bērna uzmanību. Kļūdu identificēšana realizēta gan ar teksta, gan ar krāsas efektu palīdzību. Izpildot programmu vizuāli iespējams sekot līdzi, kura programmas daļa tiek izpildīta. Programmas vides vizuālo izskatu iespējams individuāli pielāgot. Atšķirībā no LOGO, RoboMind vidē viegli iespējams norādīt kustības ātrumu.

Programmēšanas valoda LOGO nodrošina ne tikai programmēšanas pamatu apguvi, tā sniedz praktisku pieredzi ģeometrijā – bērniem, kas nav saskārušies ar ģeometriju, rada intuitīvu priekšstatu par leņķu lielumu. Iespējams izmantot arī krāsas un skaņas efektus, tādējādi ievērojami palielinot realizējamo uzdevumu skaitu. Savukārt, programmēšanas valoda RoboMind paralēli programmēšanas pamatu apgūšanai piedāvā pavisam nelielu ieskatu mākslīgā intelekta problēmu jautājumos. Bērns var iemācīties asociēt sevi ar citu personu, kam, ņemot vērā Šveices psihologa un biologa Žana Piažē (*Jean Piaget*) uzskatus, ir nozīmīga loma tieši sākumskolas vecuma bērniem [15].

Protams, var šķist, ka LOGO izmantošana Latvijā varēja sākties līdz ar plašāku datoru parādīšanos skolās, bet patiesībā, tas viss sākās vēl agrāk. Vairākās Latvijas skolās ārpusklases nodarbībās tika izmantoti LOGO principi matemātikas pamatu apguvei. Toreiz atbilstoši priekšrakstiem – LOGO komandām, bērni zīmēja grafiskos attēlu ar roku. Šobrīd, kad ir izstrādāta primitīva LOGO programmēšanas valodas latviskā versija, iegūta vairāku gadu pozitīva pieredze darbā ar LOGO vidi vairākās Latvijas skolās un informāciju tehnoloģijas ir pieejamas gandrīz ikvienam, izglītojošā programmēšanas valoda LOGO tomēr nav ieguvusi lielu popularitāti [16] [17]. Kā vienu no iemesliem var minēt tieši palīgmateriālu trūkumu. Lai gan atrodama programmas palīdzība un neskaitāmas grāmatas, kas apraksta LOGO izmantošanas metodes, tomēr lielākā daļa no tām ir piemērotas pieaugušajiem.

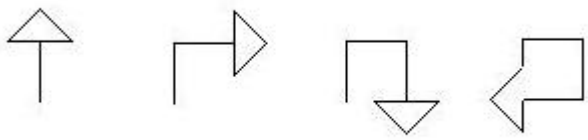
5. IESPĒJAMIE RISINĀJUMI IZGLĪTOJOŠO PROGRAMMĒŠANAS VALODU PIELĀGOŠANAI SĀKUMSKOLAS VECUMA BĒRNIEM

Apskatot izglītojošās programmēšanas valodas LOGO un RoboMind, to iespējas un piemērotību sākumskolas vecuma bērniem, var izdalīt divas kopējās problēmas:

1. Informācijas trūkums;
2. Atgriezeniskās saites problēma.

Programmas nesniedz bērnam jaunu informāciju, viņam piemērotā veidā. Tādējādi darbs ar izglītojošajām programmēšanas valodām iespējams tikai ar pieaugušo palīdzību. Bērnam netiek piedāvāti informatīvi materiāli, ar kuru palīdzību viņam būtu iespēja pašam atrisināt radušos problēmu vai uzzināt nepieciešamo informāciju.

Tā kā bērniem ir vizuālā uztvere un, ņemot vērā Ž. Piažē uzskatus, bērniem vecumā no 7 līdz 11 gadiem ir problēmas saprast tēmas, kuras viņi neredz vai nesajūt [15], svarīgi lai informatīvie materiāli saturētu ne tikai programmēšanas valodas teorētisko bāzi, bet reālus, vizuāli attēlotus uzdevumus. Piemēram, uzsākot darbu ar LOGO bruņurupuci, svarīgi bērnus ne tikai iepazīstināt ar to kā bruņurupucis var paspert vairākus soļus uz priekšu (FD n), paspert vairākus soļus atpakaļ (BK n), pagriezties vairāk vai mazāk pa labi (RT n) vai pa kreisi (LT n), bet arī kā un kāpēc viņam būtu nepieciešams to darīt. Tipisks piemērs - bruņurupucis nolēmis uzzīmēt kvadrātu izmantojot viselementārākās komandas – FD 30, RT 90, FD 30, RT 90, FD 30, RT 90, FD 30. Bet tā kā šīs ir vienas no pirmajām komandām, kuras lietotājs apgūst, būtiski arī grafiski attēlot, katru veikto soli (skat. 5.1.att.).



5.1. att.

Kvadrātā attēlošanas process

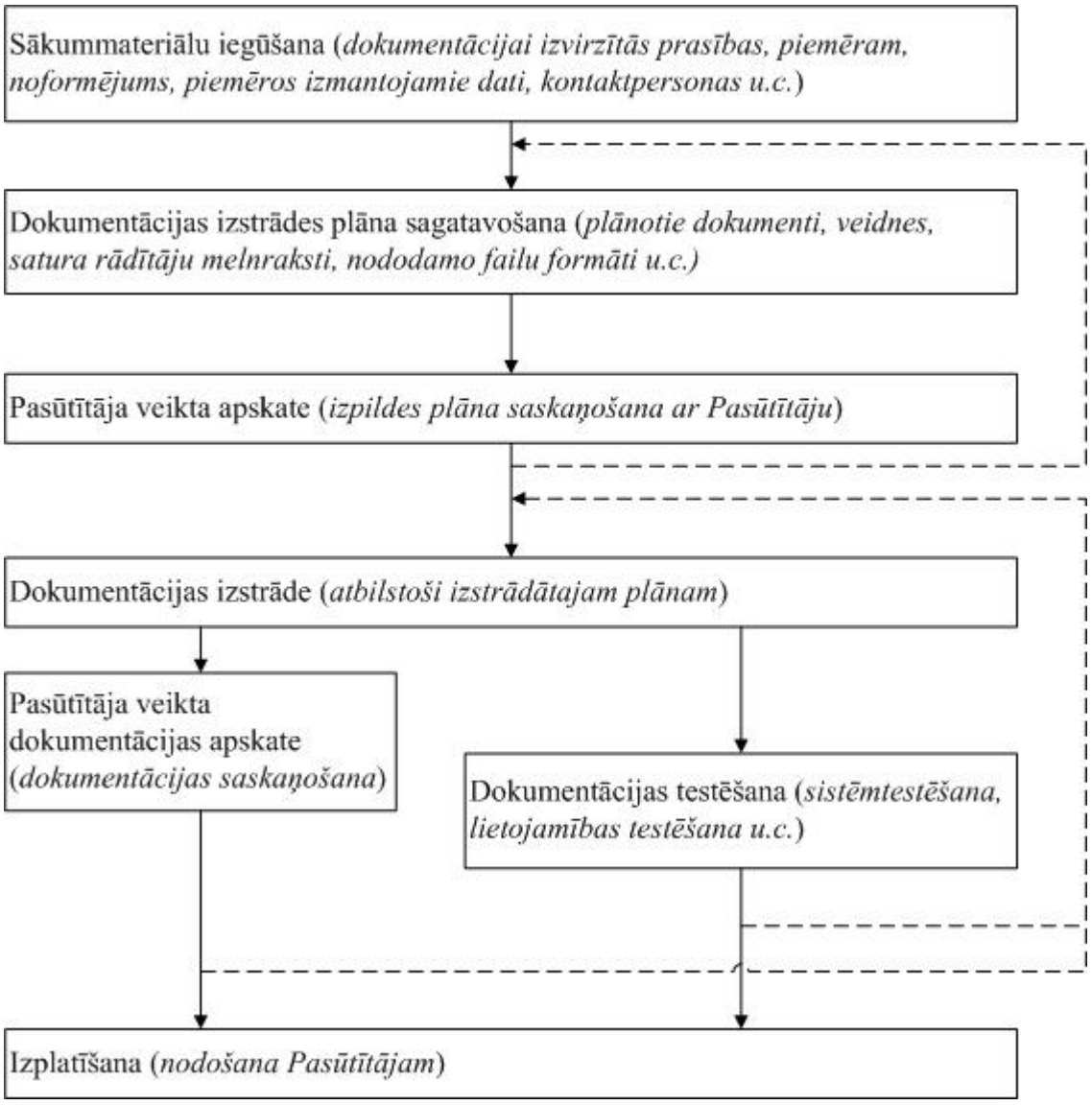
Tā kā programmēšana ir diezgan darbietilpīgs process, tad izveidojot lietotāja ceļvedi brošūras vai grāmatas formātā, var ievērojami samazināt laiku, ko bērns pavada pie datora, tādējādi samazinot kaitējumu viņa veselībai. Veiksmīgi tiek pielietoti divi uz izvietojumu bāzēti izglītojošās izklaides veidi – skolnieks kā līdzdalībnieks un skolnieks kā skatītājs [18]. Paralēli tam bērns apgūst darbu ar vairākiem informācijas avotu veidiem.

Arī atgriezeniskās saites problēmas daļējai atrisināšanai iespējams izmantot programmai piesaistītus palīglīdzekļus, izveidojot interaktīvu ceļvedi, kas satur ne tikai piemērus, bet arī veicamus uzdevums, ar norādēm, ko darīt neveiksmes gadījumā.

Novērtējot izglītojošās programmēšanas valodas LOGO un RoboMind atsevišķi, jāsecina, ka arī LOGO vizuālā izskata neatbilstību sākumskolas vecuma bērniem arī iespējams novērst izmantojot palīglīdzekļus. Ja reālā apmācība darbam ar vidi tiek pasniegta bērnam saistošā veidā un pati programmēšanas vide tiek izmantota tikai uzdevumu realizācijai, tad speciāli elementi bērna uzmanības noturēšanai nemaz nav nepieciešami.

Diemžēl RoboMind vidē konstatēto ierobežoto realizējamo uzdevumu skaitu, kas izriet no salīdzinoši nelielā programmēšanas pamatu pārklājuma, nav iespējams novērst izmantojot palīglīdzekļus.

Var secināt, ka veiksmīgi izstrādājot izglītojošo programmēšanas valodu lietotāju dokumentācijas, kas būtu piemērotas bērniem, iespējams nodrošināt to atbilstību sākumskolas vecuma bērniem domātās programmatūras raksturiezīmēm. Apskatīsim vispārīgu lietotāja dokumentācijas izstrādes procesa dzīves ciklu:



5.2. att.

Lietotāja dokumentācijas izstrādes procesa dzīves cikls [19]

Pirmā fāze – sākummateriālu iegūšana. Svarīgi noskaidrot, kāds ir lietotāja dokumentācijas izstrādes mērķis. Izglītojošās programmēšanas valodas LOGO piemērā ar lietotāja dokumentācijas palīdzību jānodrošina:

1. Jaunas informācijas nodrošināšana;
2. Uzmanības piesaistīšana;
3. Atgriezeniskā saite.

Balstoties uz Ž. Piažē izstrādātajām intelekta attīstības stadijām, sākumskolas vecuma bērniem ir raksturīgs konkrēto operāciju periods. Šeit ar vārdu „operācijas" jāsaprot principi, kuri tiek izmantoti problēmu atrisināšanai. Bērns loģiskās operācijas veic balstoties uz situāciju imitāciju [15]. Vēl joprojām svarīga nozīme ir paraugam, tāpēc saistošas liekas pasakas, stāsti. To

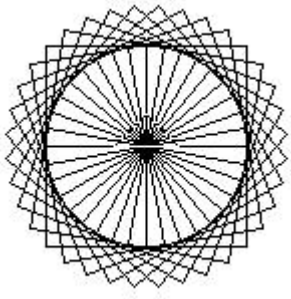
elementus veiksmīgi var pielietot bērna uzmanības saistīšanai, piemēram, bruņurupucis, kas, saskaroties ar dažādiem uzdevumiem, iepazīstina bērnu ar LOGO programmas komandām. Bruņurupuča izmantošana ļautu bērnam vieglāk saistīt divās dažādās vidēs apgūto informāciju. Tātad ierobežotu resursu gadījumā bērniem nozīmīgāk izveidot ceļvedi, kas nodrošina mazāku skaitu plašāk un piemērotāk aprakstītas funkcijas.

Iekļaujot lietotāja ceļvedī ne tikai reāli strādājošu programmu piemērus, bet arī kļūdainas programmas, var iepazīstināt bērnu ar dažādiem kļūdu labošanas modeļiem, tādējādi samazinot atgriezeniskās saites nepieciešamību. Piesaistot šo ideju jau minētajam piemēram ar bruņurupuci, varam iztēloties situāciju, kad negaidīta programmas galarezultāta gadījumā, bruņurupucis veic lāgošanu, tādējādi norādot uz iespēju apskatīt programmas darbību pa soļiem. Piemēram, bruņurupucis ir saņēmis ziņojumu no savas labākas draudzenes pūces:

```
to square
    repeat 4 [frward 50 right 90]
end
to flower
     repeat 36 [right 10 sqare]
end
flower
```

Bet diemžēl viņš to nespēj apskatīt, jo tajā ieviesusies kļūda. Rūpīgi apskatot ziņojumā ietverto programmas kodu un atgrieztos paziņojumus, bruņurupucis ievēro, ka pūce dažos vārdos nejauši izlaidusi burtus, izlabojot kļūdas, programmas radīto attēlu beidzot var ieraudzīt (sk. 5.3.att.):

```
to square
    repeat 4 [forward 50 right 90]
end
to flower
     repeat 36 [right 10 square]
end
flower
```



5.3.att.

Ziņojuma rezultāts

Šajā situācijā būtiski runāt par programmas sniegtajiem kļūdu paziņojumiem. Izglītojošajās programmēšanas valodās tie parasti nav sarežģīti – „I don't know how to sqare in flower". Bērnam svarīgi saprast, ka kļūda meklējama procesā, kurš apzīmēts ar pēdējiem 3 vārdiem.

Atgriežoties pie iepriekšminētā piemēra, saistot lietotāja ceļvedi ar programmu, var izaicināt bērna zinātkāri un neparādīt pūces sūtītās programmas attēlu, liekot bērnam pašam to uzrakstīt LOGO vidē, lai iegūtu rezultātu.

Svarīgi ņemt vērā lietotāju grupas priekšzināšanas. Piemērā par funkcijas „left n" skaidrojumu, redzējām, ka pieaugušajam, kuram ir ģeometrijas zināšanas, skaidrojums „pagriezties pa kreisi par n grādiem" liekas saprotams, bet bērnu gadījumā jāizvēlas – vai nu jānodrošina bērnu ar papildinformāciju, kas ir pagrieziens par noteiktu grādu skaitu, vai arī jāpielieto savādāks komandas raksturojums. Piemēram, no sākuma jāiepazīstina lietotāju ar komandām „left 90" un „right 90", un tikai pēc tam jāinformē par iespējamību mainot cipara lielumu pagriezties jebkurā virzienā.

Rodas jautājums, vai bērniem vispār ir piemērotas programmēšanas valodas, kurās jāraksta kods? Programmēšanas valodas Scratch un StarLogo TNG izstrādātāji atvieglojuši šo procesu, ļaujot programmas komandas izvēlēties ar peles palīdzību. Diemžēl arī šajā valodā komandu apzīmēšanai izmantots teksts. Protams, ja programma pieejama bērna dzimtajā valodā, tas rada tikai priekšrocības, bet ja ne, tad bērnam tāpat ir jāapgūst komandu nozīme. Šādā situācijā, tā kā bērni vislabāk atceras veiktās darbības, rakstot kodu, bērns šo procesu veiks daudz ātrāk.

Tad vai labāk nebūtu izstrādāt grafisku programmēšanas valodu, kurā funkcijas vārds būtu piktogramma, kas attēlo funkcijas veidoto zīmējumu? Domājot par šādu iespēju, svarīgi noskaidrot vairākus jautājumus:

1. Vai tādā veidā netiks pazaudēta pamatdoma – valoda, kurā sazināties ar datoru?
2. Vai atvieglojot programmēšanas procesu, tiks nodrošināta kvalitatīvāka zināšanu apguve?
3. Vai iespējams visām nepieciešamajām funkcijām piemeklēt intuitīvi saprotamas piktogrammas?

SECINĀJUMI

Iepazīstoties ar sākumskolas vecuma bērniem domātās programmatūras raksturiezīmju veidojošajiem faktoriem, var secināt, ka norādītajā vecumā vēl joprojām svarīgākā mācību metode ir spēle. Tātad jebkurai sākumskolas vecuma bērniem domātai programmatūrai jāsatur arī datorspēlēm raksturīgās iezīmes.

Balstoties uz līdz šim neapšaubīto T. Melones teoriju par "izaicinājuma", "fantāzijas" un "zinātkāres" nozīmību saistošas datorspēles izveidē, varam secināt, ka izglītojošās programmēšanas valodas atbilst nozīmīgākajiem sākumskolas vecuma bērnu programmatūras kritērijiem. Neparedzams iznākums kā zinātkāres nodrošinātājs, vairāki sarežģītības līmeņi kā izaicinājuma nodrošinātāji, neierobežotas iespējas kā fantāzijas nodrošinātājas.

Tomēr tikpat nozīmīgu lomu fantāzijas nodrošināšanā spēlē arī piemērota izglītojošā rīka izvēle. Svarīgi novērtēt realizējamos mērķus un programmēšanas valodas iespējas, lai nezaudētu interesi ierobežoto iespēju dēļ.

Balstoties uz sākumskolas vecuma bērniem domātās programmatūras raksturiezīmēm un izglītojošo programmēšanas valodu analīzi, var izdalīt vēl dažas nozīmīgas izglītojošo programmēšanas valodu problēmas:

1. Informācijas nodrošinājuma trūkums;
2. Atgriezeniskās saites problēma.

Kā iespējamais risinājums informācijas nodrošināšanai izvēlēta sākumskolas vecuma bērniem piemērota lietotāja ceļveža izstrāde. Atgriezeniskās saites problēmas atrisināšanai iespējams izmantot kļūdu atrisināšanas modeļus, kurus iekļaut ceļvedī.

Apkopojot visu iegūto informāciju, var secināt, ka izglītojošās programmēšanas valodas iespējams lietot neizjūtot valodas barjeru, jo bērns principā mācās svešvalodu, kurā sarunāties ar datoru, un viņam nav būtiski, ko šie vārdi nozīmē kādā svešvalodā. Tāpēc, kaut arī angļu bērniem daudzu programmēšanas valodu apgūšana ir elementārāka, lokalizācijas trūkumu nevar uzskatīt kā būtiskāko iemeslu kādas izglītojošās programmēšanas valodas ierobežotam lietotāju skaitam.

Apskatot vairākas izglītojošās programmēšanas valodas, jāsecina, ka liela daļa no tām neapskata tādu būtiskas programmēšanas pamatu sastāvdaļas kā datu tipi un to konversija. Nākotnē būtu svarīgi izvērtēt, kādas pasniegšanas metodes būtu izmantojamas šīs vielas apguves nodrošināšanai.

Turpmākai izpētei – grafiskas programmēšanas valodas izstrādes nepieciešamība un iespējas.

IZMANTOTĀ LITERATŪRA UN AVOTI

1. **Affoso Bob.** *Is the Internet Affecting the Social Skills of Our Children?* [tiešsaiste]. Reno: University of Nevada, 1999 - [atsauce 12.04.2007.]. Pieejams: http://sierrasource.com/cep612/internet.html
2. **Anan Kogan Barbara.** *How Computers Affect Your Child's Health* [tiešsaiste] Hagerstown: Vibrant life, September 2001 - [atsauce 12.04.2007.]. Pieejams: http://www.vibrantlife.com/vl/article-7.html
3. **Zelmenis V.** *Pedagoģijas pamati.* – Rīga: RaKa, 2000. 291 lpp.
4. **Dzintere D. Boša R.** *Rotaļspēles.* – Rīga: Mācību apgāds NT, 1997. 82 lpp.
5. **Juul Jesper.** *What computer games can and can't do* [tiešsaiste]. Copenhagen: IT University of Copenhangen, 2000 - [atsauce 21.05.2007]. Pieejams: http://www.jesperjuul.net/text/wcgcacd.html
6. **Facer Keri.** *Computer games and learning* [tiešsaiste]. Harbourside: Futurelab, September 2003 - [atsauce 4.06.2007]. Pieejams: http://www.futurelab.org.uk/resources/documents/discussion_papers/Computer_Games_and_Learning_discpaper.pdf
7. **Malone Thomas W.** *What makes things fun to learn? Heuristics for designing instructional computer games* [tiešsaiste]. New York: ACM Press, 1980 [atsauce 4.06.2007.]. Pieejams: http://portal.acm.org/portal.cfm ISBN:0-89791-024-9
8. **Barendregt Wolmet, Bekker Mathilde M., Speerstra Mathilde.** *Empirical evaluation of usability and fun in computer games for children* [tiešsaiste] Eindhoven: Faculty of Industrial Design, 2003 - [atsauce 4.06.2007.]. Pieejams: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/conferences/INTERACT2003/INTERACT2003-p705.pdf
9. **Shalom M. Fisch**. *Making Educational Computer Games „Educational"* [tiešsaiste] New York: ACM Press, 2005 [atsauce 18.04.2008.]. Pieejams: http://portal.acm.org/citation.cfm?id=1109548
10. **Malone Thomas W.** *Heuristics for designing enjoyable user interfaces - Lessons from computer games* [tiešsaiste]. New York: ACM Press, 1982 - [atsauce 4.06.2007.]. Pieejams: http://doi.acm.org/10.1145/800049.801756
11. *Programming basics* [tiešsaiste]. Toronto: Spark Publishing Inc., 2007 – [atsauce 20.04.2008.]. Pieejams: http://www.edumax.com/programming-basics.html

12. *LOGO Foundation* [tiešsaiste] – [atsauce 18.04.2008.]. Pieejams: http://el.media.mit.edu/Logo-foundation/index.html
13. *RoboMind* [tiešsaiste] – [atsauce 26.04.2008.]. Pieejams: http://robomind.net/en/robomind.htm
14. **Seymour Papert.** *Mindstorms Children, Computer, and Powerful ideas.* New York: Basic Books, Inc., Publishers, 1980. 230 p.
15. **Dr. C. George Boeree.** *Jean Piaget* [tiešsaiste] Shippensburg: Shippensburg University 2006 - [atsauce 10.05.2008.]. Pieejams: http://webspace.ship.edu/cgboer/piaget.html
16. *ZR_LOGO* [tiešsaiste] – [atsauce 07.05.2008.]. Pieejams: http://ifs.auce.lv/algoritmi/index.htm
17. **V. Vanaģelis.** *Informātika sākumskolā* [tiešsaiste]. Auce: LatSTE, 2000 – [atsauce 14.05.2008.]. Pieejams: http://latste2000.latste.lv/Refer_kolekc/Vanagelis_sakumskola.doc
18. **K. Rapeepisarn, Kok Wai Wong, Chun Che Fung, A. Depickere.** *Similarities and diferences between „learn through play" and „edutainment"* [tiešsaiste]. Murdoch: Murdoch University, 2006 – [atsauce 18.04.2008.]. Pieejams: http://portal.acm.org/citation.cfm?id=1231894.1231899
19. **V. Karnīte** *Lietotāja dokumentācijas kvalitātes uzlabošana ar testēšanas palīdzību* [tiešsaiste] Rīga: Exigen Services, Ltd. 2005 – [atsauce 20.05.2008.]. Pieejams: http://www.riti.lv/testkonf/konf_ttp2005_vita_karnite_1_txt.pdf

Bakalaura darbs „Izglītojošo programmēšanas valodu iespējas un piemērotība sākumskolas vecuma bērniem” izstrādāts LU Fizikas un matemātikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autore: Rasma Lejniece

Rekomendēju darbu aizstāvēšanai

Vadītājs: asoc.prof. Dr.sc.comp. Guntis Arnicāns

Recenzents: profesors Māris Vītiņš

Darbs iesniegts Datorikas nodaļā 27.05.2008.

Metodiķe: Ārija Sproģe

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

09.06.2008.

Komisijas sekretāre: lektore Laila Niedrīte