

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Specializuotosios kalbos ir jų taikymas edukacinių žaidimų projektavime

Kursinis darbas

Atliko: 3 kurso 5 grupės studentė
Gabrielė Žielytė (parašas)

Darbo vadovas: Lekt. Irus Grinis (parašas)

Vilnius – 2020

TURINYS

IVADAS	3
Darbo tikslas	3
Darbo uždaviniai	3
1. SPECIALIZUOTOSIOS KALBOS	4
1.1. Metaprogramavimas	4
1.2. Metaprogramavimo įrankiai	5
2. EDUKACINIAI ŽAIDIMAI	6
2.1. Mokytojai.....	6
3. „JETBRAINS MPS“	8
3.1. Specializuotųjų kalbų ir įprasto kodo darna	8
3.2. Specializuotųjų kalbų projektavimas.....	8
3.2.1. Abstrakti sintaksė	9
3.2.2. Konkreti sintaksė	9
3.2.3. Kodo transformacija.....	10
3.3. Bazinė kalba.....	10
4. META KALBŲ BEI „JETBRAINS MPS“ NAUDA KURIANT EDUKACINIUS ŽAIDIMUS	12
4.1. Redaktorius	12
4.2. Laiko taupymas.....	12
4.2.1. Kodo eilučių sumažinimas	13
4.3. Retaesnės klaidos.....	13
4.4. Kūrėjai ir srities specialistai	13
4.4.1. Geresnis srities supratimas	14
5. TAIKYMAS	15
5.1. Pradinis projektas	15
5.2. Žaidimo projektas	16
5.3. Galutinis projektas	17
5.4. Žaidimo meta kalbos sąvokos	18
5.4.1. „Engine“ sąvokos.....	18
5.4.2. „FeedbackLang“ sąvokos	19
5.4.3. „MathTaskLang“ sąvokos	19
5.5. Kodo generavimas ir paleidimas	19
REZULTATAI IR IŠVADOS	20
LITERATŪRA	22
SĄVOKŲ APIBRĖŽIMAI	24
SANTRUMPOS	25
PRIEDAI	25
1 priedas. MPS pradinis skaičiuotuvo projektas.....	26
2 priedas. „MPS“ edukacinio žaidimo projektas	27
3 priedas. MPS žaidimo projektas	28

Įvadas

Augant technologijų plėtrai, labiau domimasi naujoviškų technologinių priemonių, tokių kaip vaizdo žaidimų, virtualių pasaulių ir masinių kelių žaidėjų internetinių žaidimų (angl. textitMulti-Player Online Games (MMPOGs)), diegimu [BKS14]. Nuo pirmųjų kompiuterinių žaidimų kompanijų pasirodymo septintajame dešimtmetyje globali žaidimų industrija pavirto masyviu pramonės verslu, kurio bendra vertė viršija 100 milijardų dolerių. Per pastarąjį dešimtmetį įvyko daug kultūrinių pokyčių siekiant platesnio žaidimų pripažinimo kasdieninės kultūros bei švietimo kontekstuose [AHH⁺18].

Nors mokomųjų ar „rimtų“ žaidimų sąvoka yra palyginti nauja, idėja ir panaudojimas nėra. Jau šeštajame dešimtmetyje vaizdo žaidimai buvo naudojami JAV karinėms, medicinos mokykloms, taip pat ir likusiai akademinėi bendruomenei [Ber06].

Skandinavijoje yra išlikusi stipri tradicija žmogaus ir kompiuterio sąveikos srityje, kur yra remiamasi vadinamuoju „skandinavišku požiūriu“ (angl. *scandinavian approach*). Šios tradicijos pagrindas yra dalyvavimo koncepcija, t. y. būsimų vartotojų įtraukimas į visą projektavimo procesą [AHH⁺18]. Šis metodas yra naudingas taikyti pasitelkiant specializuotąsias kalbas, kurios gali būti kuriamos įvairioms sritims, taip pat ir edukaciniams žaidimams kurti bei modifikuoti. Specializuotosios kalbos bei patogus jų kūrimo IDE gali padėti vartotojams geriau išmokti dalyką, kurį mokosi, bei suartinti edukacinio žaidimo kūrėjus ir mokytojus ar dėstytojus.

Darbo tikslas

Šio darbo tikslas – išanalizuoti „JetBrains MPS“ specializuotųjų kalbų kūrimo įrankį, sukurti edukacinį žaidimą su jam skirtomis specializuotosiomis kalbomis, bei įvertinti specializuotųjų kalbų ir paties MPS taikymo naudą edukacinių žaidimų kūrime.

Darbo uždaviniai

1. Remiantis literatūra apibūdinti, kas yra specializuotosios kalbos bei edukaciniai žaidimai.
2. Aprašyti „JetBrains MPS“ specializuotųjų kalbų kūrimo įrankį.
3. Remiantis MPS sukurti edukacinį žaidimą ir specializuotąsias kalbas jam.
4. Įvertinti specializuotųjų kalbų, bei „JetBrains MPS“ sprendžiamas problemas edukacinių žaidimų kūrime.

1. Specializuotosios kalbos

Specializuotoji kalba, kitaip dar vadinama metakalba – tai specialios paskirties kalba, kurios priemonėmis aprašoma kuri nors kita kalba, dažnai vadinama objektine kalba. Metakalbos tikslas – suteikti priemones (terminus, gramatiką), skirtas objektinės kalbos aprašymui, leidžiančiam apibrėžti kalbos taisykles bei terminus, naudojantis metaterminais (metakalbos terminais ir taisyklėmis).

Tradicinė kompiuterinė programa yra instrukcijų rinkinys, daugiausiai kodas, kuris manipuliuoja duomenų struktūras ir sukuria išvestį. Kita vertus interpretatorius (angl. *compiler*) – yra programa, kuri naudoja šaltinio kodą (angl. *source code*) arba duomenų struktūras kaip įvestį, tuomet paverčia šį kodą kitomis duomenų struktūromis, kurias geriau supranta, ir sukuria išvestį, kuri gali būti pvz.: dvejetainis kodas arba tarpinė kalba.

Iš kompiuterinės programos bei interpretatoriaus apibrėžimų išplaukia, kad, jei galime parašyti kodą, kuris manipuliuoja duomenų struktūromis, ir interpretatorius, kurie žmogaus parašytą kodą traktuoja kaip duomenų struktūrą, galime parašyti kodą, kuris rašo ir manipuliuoja kitu kodu. Toks kodas vadintųsi metakalba arba specializuotąja kalba.

Kompiuteriai vykdo programas, instrukcijų rinkinius žemo lygio kalba, todėl daugumos aukštesnio lygio programavimo kalbų bei interpretatorių kūrimas apima metakalbos kūrimą ir panaudojimą.

1.1. Metaprogramavimas

Programavimas susijęs su metakalbomis vadinamas metaprogramavimu – tai programavimo technika, kai kompiuterinės programos turi galimybę traktuoti kitas programas kaip duomenis. Tai reiškia, kad programa gali būti sukurta skaityti, generuoti, analizuoti ar transformuoti kitas programas ir netgi modifikuoti save pačią vykdam. Kai kuriais atvejais tai leidžia programuotojams sumažinti kodo eilučių skaičių, savo ruožtu sutrumpinant programos kūrimo laiką [CE00].

Metaprogramavimas buvo populiarus aštuntajame ir devintajame dešimtmečiuose, naudojant sąrašų apdorojimo kalbas, tokias kaip LISP, bet viena seniausių programavimo metakalbų – „Backus-Naur form“ buvo sukurta dar seniau – 1960 metais. Šiomis dienomis metakalbos, bei metaprogramavimas gali būti sutikti vis dažniau.

Skaitytojas neturėtų būti klaidinamas priešdėlio „meta“, kuris šiame kontekste reiškia „aukšto lygio“. Iš apibrėžimo išplaukia du svarbūs dalykai:

1. Metaprogramavimas ir programavimas yra tos pačios srities dalykai; Taigi meta programavimas negali būti suprantamas nežinant programavimo pagrindų.
2. Metaprogramavimas susijęs su automatiniu programavimu, dar vadinamu programų generavimu.

Nors metakalbų projektavimas susijęs su galutinių vartotojų įtraukimu į sistemos kūrimo ir evoliucijos procesus, techniniais aspektais metakalbų projektavimas gali būti laikomas metodo-

logija, kuria siekiama sukurti „projektavimo erdves“, kad galutiniams vartotojams būtų suteikta galimybė perprojektuoti sistemą naudojimo metu, kai pasirodo nauji reikalavimai [ŠD12].

1.2. Metaprogramavimo įrankiai

Egzistuoja ne vienas metaprogramavimo ir metakalbų kūrimo įrankis. Keli pavyzdžiai yra:

- „Xtext“ – „Eclipse“ projektas, sukurtas 2006 metais. Tai programavimo kalbų ir DSL kūrimo įrankis, kuriuo kalbos kuriamos pasitelkus „Xtext“ gramatikos kalbą. Su „Xtext“ galima kurti visaverčius teksto redaktorių tiek bendrosios paskirties, tiek domeno kalboms, be to „Xtext“ redaktoriai yra paruošti JavaScript, VHDL, Xtend ir kitoms kalboms. Patogu tai, kad „Xtext“ IDE yra paremtas populiariu „Eclipse“ IDE, todėl daugeliui programuotojų jis atrodytų pažįstamas.
- „Blockly“ – programavimo kalbos JavaScript biblioteka, skirta kurti vizualines programavimo kalbas ir redaktorių blokų pagrindu (angl. *block-based*). Tai „Google“ projektas, paprastai veikiantis naršyklėje. „Blockly“ naudoja vizualinius blokus, susietus ryšiais, ir gali generuoti kodą į JavaScript, Lua, Dart, Python, PHP, ar bet kokios kitos tekstinio vaizdavimo programavimo kalbos kodą.[Are12] Šio įrankio pagalba daugiausiai mokoma paprastų programavimo algoritmų principų.
- „JetBrains MPS“ – įrankis, kuris naudoja projekcinį redagavimą, leidžiantį vartotojams peržengti kalbos analizatorių ribas ir kurti specializuotųjų kalbų redaktorių su, pvz.: lentelėmis ir diagramomis [Jet19]. Apie šį įrankį daugiau bus kalbama vėlesniuose skyreliuose.

2. Edukaciniai žaidimai

Edukaciniai žaidimai – tai žaidimai, specialiai sukurti švietimo tikslais arba turintys ugdomąją vertę. Edukacinėje aplinkoje gali būti naudojami visų rūšių žaidimai, tačiau edukaciniai žaidimai yra skirti padėti žmonėms išmokti tam tikrų dalykų, išplėsti sąvokas, sustiprinti vystymąsi, suprasti istorinį įvykį ar kultūrą arba padėti jiems išmokti įgūdžių žaidžiant.

Švietimas nebėra vien žinių perdavimas studentams, tai ir žinių projektavimo mokslas. Atsižvelgiant į tai, iškyla poreikis surasti naujų būdų, kaip mokytis prasmingai XXI amžiuje, panaudojant naujus įrankius, kuriuos suteikia technologijos.

Pastaraisiais metais daugelis tyrėjų nagrinėjo skaitmeninių technologijų veiksmingumą skatinant mokymąsi, ypač žaidimų terpėje. Žaidimų įtraukimas į ugdymą dažnai yra efektyvesnis už tradicinius mokymo metodus, tai padidina mokymosi motyvaciją, aktyvų dalyvavimą, bei studentų susikaupimą. Be to, žaidimai gali pagerinti studentų socialinius įgūdžius, taip pat ir jų supratimo ir problemų sprendimo įgūdžius [KIV10]. Marc Prensky sukūrė daugiau nei 50 edukacinių kompiuterinių žaidimų ir, pasak jo, žaidimais grįstas mokymasis (angl. *game-based learning*) daug labiau motyvuoja nei šiuometinis formalus akademinis švietimas. Dauguma jaunuolių gali suprasti ir laisvai kalbėti kompiuterių, interneto ir vaizdo žaidimų skaitmenine kalba. Prensky tai reiškia, kad ši besimokančiųjų grupė yra kitokia (jie mąsto ir apdoroja informaciją iš esmės kitaip) ir kai kurie ugdymo aspektai turi pasikeisti, pavyzdžiui, tai, kaip mokytojai bendrauja su savo mokiniais, jų mokomas turinys ir kt.

Edukaciniai žaidimai laikomi turintys didelį potencialą, ir yra manoma, kad jie veiksmingai padėtų įgyvendinti užsibrėžtus mokymo tikslus, nes jie (teoriškai) pateikia įvairius metodus pažintinio ir emocinio mokymosi (pvz., žinių, įgūdžių konstravimo, požiūrio formavimo) vystymui [VVC12].

2.1. Mokytojai

Mokomieji žaidimai skiriasi nuo įprastinių, tuo, kad, jei įprastinio žaidimo sudėtis yra sistema ir vartotojas(-ai), tai edukacinių žaidimų dalyviu yra daugiau: sistema, mokins ir mokytojas.

Mokytojai prisideda prie edukacinio žaidimo kūrimo, todėl tam, kad jiems būtų patogiau dirbti savo darbą, atsiranda keletas reikalavimų sistemai:

- Sistema arba IDE, turėtų būti paprasta naudoti ir paleisti. Ne visi mokytojai turi geras kompiuterinio raštingumo raštingumo žinias, todėl ne kiekvienas gali suprasti sudėtingas sistemas, turinčias daug funkcijų. Mokytojams taip pat reikia paaiškinti mokiniams kaip naudotis sistema, todėl kuo ji yra paprastesnė naudoti, tuo mažiau laiko yra sugaištama.
- Mokytojui turi būti suteikta galimybė stebėti mokinio pažangą ir įvertinti jo įgytas žinias. Informacija apie tai, kaip gerai mokins vykdo užduotis arba įsisavina informaciją turi būti prieinama ir pateikta mokytojui sugalvotu formatu. Vertinimo sistema taip pat gali būti įdiegta pačiame žaidime – gali būti rodomas surinktų taškų skaičius, prarastos gyvybės ar kt.

Kadangi edukacinių žaidimų procese dalyvaujantys žmonės turi savo reikalavimus, gali būti patogiau turėti atskiras specializuotas kalbas specifiniams reikalavimams įgyvendinti, pvz.: atgalingo ryšio kalbą.

3. „JetBrains MPS“

„JetBrains MPS“ (angl. „*MetaProgrammingSystem*“) yra „JetBrains“ sukurtas įrankis skirtas į kalbą orientuotam programavimui (toliau vadinama LOP) [Dmi04]. Pagrindinė į kalbą orientuoto programavimo idėja yra ta, kad kuriant programinę įrangą yra naudojama ne viena kalba, bet verčiau naudojamos kelios atskiros kalbos, kurios geriausiai tinka kiekvienai užduočiai. Priešingai nei daugiakalbis (angl. *polyglot*) programavimas, kuris iš esmės pasisako už panašų požiūrį, į kalbą orientuotas programavimas aiškiai skatina kūrėjus kurti savo specializuotąsias kalbas arba išplėsti esamas kalbas su sričiai būdingomis koncepcijomis. Kad tai būtų įmanoma, svarbi LOP požiūrio sudedamoji dalis – tinkami įrankiai.

MPS yra būtent toks įrankis– jis skirtas kurti specializuotąsias tam tikros srities kalbas(angl. *domain-specific languages* (DSL)). Jis naudoja projekcinį redagavimą, kuris leidžia vartotojams peržengti kalbos analizatorių (angl. *parser*) ribas ir kurti specializuotųjų kalbų redaktorių.

Programuojant MPS įrankiu specializuotoji kalba (toliau dar vadinama DSL) paprastai yra mažiau sudėtinga nei bendrosios paskirties kalba, tokia kaip Java, C ar Ruby. Paprastai specializuotosios kalbos kuriamos glaudžiai bendradarbiaujant su srities specialistais, kuriems ta kalba yra kuriama. Daugeliu atvejų tokios kalbos yra skirtos naudoti ne žmonėms nusimanantiems programavime, o būtent tos srities, kuriems kalba yra kuriama, atstovams. [Jet19].

3.1. Specializuotųjų kalbų ir įprasto kodo darna

Yra du iš esmės skirtingi būdai, kaip integruoti tradicinį kodą ir specializuotosios kalbos kodą:

1. Pirmasis saugo DSL kodą ir įprastą kodą atskiruose failuose. Tada automatinis kodo generatorius DSL kodą paverčia programavimo kalbos kodu, arba programa gali įkelti konkrečiai sričiai skirtą kodą ir jį vykdyti. Šis pirmasis metodas su atskirta bendrosios paskirties kalba (angl. *General Purpose Language* (GPL)) bei atskiru DSL kodu vadinamas išorine specializuotąja kalba (angl. *external DSL*). SQL yra vienas išorinės DSL pavyzdžių.
2. Alternatyvus metodas DSL kodą ir bendrosios paskirties kodą maišo tame pačiame programos faile, todėl jų integracija yra daug griežtesnė. DSL pakartotinai naudoja bendrosios paskirties kalbos gramatiką ir analizatorių bei naudoja turimas pagrindinės kalbos išplėtimo parinktis. Tokiems scenarijams apibūdinti naudojamas terminas vidinė specializuotoji kalba (angl. *internal DSL*).[Jet19]

MPS palaiko abu būdus.

3.2. Specializuotųjų kalbų projektavimas

MPS, kaip ir daugelis kitų kalbų įrankių, siūlo DSL rinkinį įvairiems kalbų aspektams apibrėžti. Tai apima struktūrą, redaktorių, tipų sistemą, generatorių, taip pat ir sudėtingo IDE funkcionalumą palaikymą, pvz., kodo užbaigimas, ketinimai (angl. *intentions*), derinimo programa(angl. *debugger*), duomenų srautų analizė ir t.t. [PSV13]

3.2.1. Abstrakti sintaksė

Pirmasis žingsnis apibrėžiant naują specializuotąją kalbą yra nurodyti struktūrą (dar vadinama abstrakčia sintakse arba meta modeliu). Tai primena objektinį programavimą, kalbos sąvokos (angl. *concepts*) panašios į klases. Sąvokos MPS atstovauja abstraktaus sintaksės medžio (angl. *Abstract syntax tree (AST)*)(toliau vadinamas AST) mazgų tipus ir apibrėžia savybes, metodus, vaikus ir ryšius, kuriuos mazgai gali turėti. MPS siūlo tris kalbos aspektus abstrakčiai sintaksei apibrėžti: *struktūros* aspektas nusako sąvokas, jų savybes ir ryšius, *apribojimai* riboja leistiną aibę savybių ir nuorodų verčių, o *elgesys* (angl. *behavior*) metodus sieja su sąvokomis [PSV13].

Taigi, naudojant naująjį DSL programai rašyti, reikia sukurti tos kalbos sąvokas, priskirti reikšmes sąvokų savybėms ir susieti programos mazgus pagal ryšius, apibrėžtus jų sąvokų. Visa tai palaiko galingi redaktoriai, kuriuos galima apibrėžti savo kalbai.

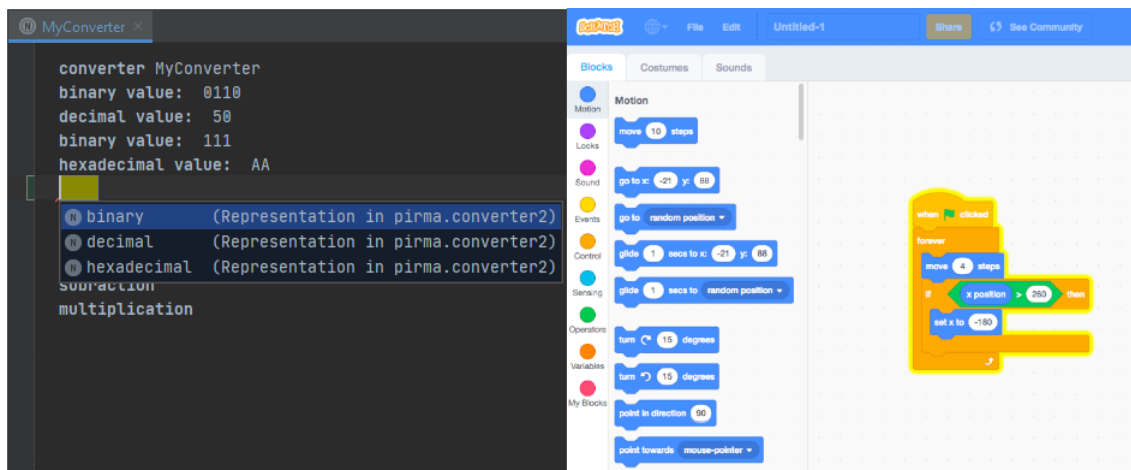
Ne visi DSL kūrimo įrankiai abstraktų sintaksės medį vaizduoja taip pat kaip MPS. „Scratch“ ir „Blockly“ sistemos, vaizdinių redaktorių pagalba skiria šakniniams ir kitiems mazgams savitą formą. Šakninių mazgų blokus galima dėti tik viršuje kitų. Tai suteikia galimybę intuityviai konstruoti AST, sutaupyti laiko kuriant.

3.2.2. Konkreti sintaksė

Antras žingsnis – apibrėžti sąvokų redaktorių. Kadangi MPS aplinkoje kodas niekada nėra vaizduojamas kaip paprastas tekstas (nei ekrane, nei diske), MPS kalbos niekada nėra formatuojamos ar analizuojamos (angl. *parsed*), taigi nereikia jokios gramatikos toms kalboms. Tai įgalina naudojimąsi neanalizuojamais žymėjimais (angl. *non-parseable notations*), pvz., lentelėmis ar matematiniais simboliais. Vietoj analizatoriaus yra apibrėžiamas redaktorius kalbų sąvokoms – vaizdinė AST mazgų reprezentacija. MPS redaktorius leidžia kalbos kūrėjui apibrėžti savo naujosios kalbos konstrukcijos redaktorių, taip pat perrašyti (angl. *override*) esamų kalbų sąvokų redaktorių. Redaktoriai gali būti skirstomi į modulius, kad būtų galima pakartotinai naudoti vaizdinės sintaksės elementus, taip pat keli redaktoriai gali apibrėžti sąvoką, o tai leidžia programuotojui pasirinkti tinkamiausią žymėjimą tam tikrai sričiai [PSV13].

Redaktorius padeda išdėstyti kiekvienos sąvokos poziciją jame. Galima apibrėžti, kurios dalys yra pastovios/konstantos, pavyzdžiui, skliausteliai ar kitos detalės, o kurios dalys yra kintamos ir jas reikia nustatyti vartotojui.

Programuotojams, naudojantiems teksto redaktorių, reikia laiko priprasti prie numatytojo MPS redaktoriaus. Kodo konstravimas iš anksto apibrėžtų kalbos sąvokų neabejotinai jaučiasi kitaip, nei kodo rašymas paprastame teksto redaktoriuje. Vartotojams taip pat nėra lengva priprasti prie tokio redaktoriaus. Akiai malonesnio ir patogesnio redaktorių pavyzdys būtų „Scratch“ arba „Blockly“ kalbų kūrimo sistemų redaktoriai. Jie abu yra skirti mokyti vaikus ar žmones, norinčius išmokti programavimo ar kitų dalykų, todėl jų grafinė vartotojo sąsaja yra pritaikyta mokytojams bei mokiniams. Palyginus su šiomis sistemomis MPS redaktorius nėra pats tinkamiausias variantas naudoti švietimo įstaigose (1 pav.).



1 pav. Kairėje: MPS redaktorius, dešinėje: „Scratch“ redaktorius

3.2.3. Kodo transformacija

Aprašytos abstrakti ir konkreti sintaksės kartu jau suteikia tam tikrą galią. Galima būtų ją panaudoti idėjų perdavimui kitiems žmonėms, pavyzdžiui, piešti UML diagramas ar rašyti kitų tipų statinius dokumentus. Tačiau dažniausiai norima, kad kodas ką nors ir padarytų.

MPS kalbos nusako transformavimo į žemesnio lygio kalbas arba paprastą tekstą taisykles. Generavimo procesą sudaro dvi fazės:

1. Pirma fazė naudoja šablono pagrindu pagrįstą „modelis modeliui“ (angl. *model-to-model*) transformacijos variklį, leidžiantis redukuoti programos kodą į tikslinę kalbą, remiantis generatoriaus nurodytomis redukcijos taisyklėmis. Tikslinė kalba gali būti dar labiau redukuota remiantis jos pačios redukcijos taisyklėmis ir pan.
2. Kai redukcija nebegali būti taikoma tolesniam modeliui, prasideda antroji fazė, kuri naudoja teksto generatorius, kad konvertuotų galutinį modelį į įprastą programos tekstą, kurį galima pateikti kompiliatoriui.

[PSV13]

Kalbos generatorių sudaro du pagrindiniai elementai. *Konfigūracijų žemėlapis* (angl. *Mapping Configurations*) apibrėžia, kurios sąvokos yra apdorojamos kokiais šablonais ir kokiame kontekste. *Šablonai* apibrėžia kodą tiksline kalba, kuri pakeis nurodytą (-ų) šaltinio sąvokos (-ų) fragmentą.

3.3. Bazinė kalba

MPS nuo seno stipriai susijęs su programavimo kalba Java. Pats MPS yra sukurtas Java pagrindu, o veikia Java dialekto, vadinamo bazine kalba (angl. *Base Language*), pagalba. Ši kalba palaiko tokias beveik universalias kalbos ypatybes kaip aritmetika, sąlygos sakiniai, ciklai, funkcijos, kintamieji ir kt.

Bazinė kalba taip pavadinta, nes ji yra geras pagrindas daugeliui kalbų, kurioms reikalingi baziniai programavimo duomenys (kintamieji, ciklai ir t.t.). Ji gali būti naudojama trimis atvejais:

galima išplėsti šią kalbą, kad ja remiantis būtų sukurta kita kalba, galima naudoti jos sąvokas savo programose, o taip pat ir sugeneruoti savo kodą į bazinę kalbą.

Nėra sunku importuoti egzistuojantį Java kodą arba bibliotekas į MPS projektus, pačias specializuotąsias kalbas galima „supakuoti“ į Java bibliotekas ir naudoti Java integruotose kūrimo aplinkose. Visa tai daro MPS jaukiu įrankiu programuotojams dirbusiems Java aplinkoje.

4. Meta kalbų bei „JetBrains MPS“ nauda kuriant edukacinius žaidimus

Šiame skyrelyje apžvelgiama MPS ir meta kalbų nauda kuriant edukacinius žaidimus.

4.1. Redaktorius

Kadangi specializuotąsias kalbas bei MPS gali naudoti ne tik programuotojai, bet ir ne tokie patyrę programavime žmonės, pavyzdžiui, mokytojai, pamačius kalbų kūrimo įrankį jie neturėtų būti nustebinti ar priblokšti jo sudėtingumo. Taigi, kur įmanoma, kalbai turėtų būti naudojamos žinomos ar lengvai išmokstamos abstrakcijos ir žymėjimai. Kodo bei sąvokų skaitomumas yra svarbus aspektas. Jei kuriant meta kalbas į šiuos reikalavimus atsižvelgiama, naudotojui turėtų būti nesunku naudotis programa.

Vienas MPS pliusų yra tas, kad kalbomis apibrėžti yra naudojami redaktoriai su griežta sintakse, o tai gali padėti pradedantiesiems programuotojams. Patogu tai, kad MPS platformoje įdiegtas kodo užbaigimas (angl. *code completion*), taigi vartotojams nereikia rūpintis specialiųjų simbolių, pavyzdžiui, kabliataškių ar skliaustelių, rašymu. Redaktorius taip pat turi iš anksto apibrėžtą išdėstymą, taigi jis pats pasirūpina tarpais ir įtraukomis. Redaktoriuje pridedamas papildomas tekstas gali keistis, vartotojui keičiant kodą, pvz. kai trūksta kažkokio mazgo, toje vietoje gali būti pridėta tiksli užuomina, kuri padės vartotojui tai ištaisyti, o kai kodas teisingas, užuomina dingsta.

Deja viena redaktoriaus problemų yra ta, jog redaktoriaus tekstas gali būti kopijuojamas, bet norint įterpti tekstą į patį redaktorių prireikia žymiai daugiau pastangų. Kita problema yra ta, kad, nors MPS redaktorius turi daug naudingų funkcijų, jis vis tiek gali būti neaiškus pradedančiajam.

Edukacinių žaidimų kontekste, MPS yra labiau skirtas vyresniems mokiniams ar studentams ir dėstytojams, nes jie gali mokėti naudotis sudėtingesniu IDE ir gali labiau norėti naudoti tekstinę kalbą, o ne vaizdinius kodų blokus. Kita vertus, jaunesniems žmonėms ir jų mokytojams labiau tiktų įrankis, naudojantis vizualinių programavimo kalbų priemones, pavyzdžiui, „Scratch“ arba „Blockly“.

4.2. Laiko taupymas

Rimta problema įprastiniame programavime yra ta, jog praeina daug laiko tarp žinojimo, kaip išspręsti problemą, ir sėkmingo sprendimo perdavimo kompiuteriui programos pavidalu. Kartais įmanoma paaiškinti problemą ir sprendimą jai per kelias valandas, bet to sprendimo kodavimas užtrunka žymiai ilgiau. Taip yra todėl, kad su kitais žmonėmis kalbama natūralia kalba, kuri yra labai turtinga, o komunikuojant kodu reikia naudoti bendrosios paskirties programavimo kalbą, tokia kalba žymiai mažiau išraiškinga. Taigi, tam, kad žmogus paaiškintų programą kitam, įmanoma išreikšti vien aukšto lygio (angl. „*high-level*“) idėjas, kai kompiuteriui reikia paaiškinti kiekvieną žingsnį ir detales. Šiais laikais daug laiko skiriama objektiškai orientuotam projektavimui, kur programuotojas apibrėžia klases, jų santykius, hierarchijas ir t.t. OOD (angl. *Object-oriented design*) procesas yra būtinas, nes klasės ir metodai yra vienintelės abstrakcijos, kurias supranta ob-

jektinės kalbos [Dmi04]. Atrodo, kad tai kūrybinga ir būtina, tačiau naudojant į kalbas orientuotą programavimą, OOD visai nereikia, meta kalbos pašalina šį trūkumą.

MPS ir meta kalbos gali padėti sutrumpinti laiko tarpą nuo žaidimo idėjos sugalvojimo iki jos įgyvendinimo tuomet, kai to žaidimo kūrėjas yra susipažinęs su MPS įrankiu ir DSL kūrimu. Kitu atveju gali tekti sugaišti daug laiko aiškinantis DSL detales ir tuo, kaip veikia MPS. Tai nėra labai patogus laiko atžvilgiu, atsižvelgiant į tai, kad edukacinius žaidimus gali norėti kurti mokytojai ar patys mokiniai.

4.2.1. Kodo eilučių sumažinimas

Vienas DSL naudojimo pranašumų yra tas, kad gavus kalbą ir generatorių, darbas DSL apimamoje programinės įrangos kūrimo srityje tampa daug efektyvesnis vien todėl, kad nereikia daryti dalies monotoninio darbo rankiniu būdu. Kai DSL programos šaltinio kodas yra generuojamas (o ne interpretuojamas), galima naudoti gražias, domenui būdingas abstrakcijas, kadangi generatorius, kaip ir kompiliatorius, gali pašalinti abstrakcijas ir generuoti efektyvų kodą, taigi galima žymiai sumažinti rašomų kodo eilučių skaičių.

4.3. Retesnės klaidos

Naudojant DSL ir vykdymo variklį (angl. *execution engine*), taikymo logika, išreikšta DSL kode, tampa nepriklausoma nuo tikslinės platformos. DSL naudojimas gali pagerinti sukurto produkto kokybę: mažiau klaidų, geresnis architektūrinis atitikimas, padidėjęs techninės priežiūros efektyvumas. Taip įvyksta pašalinus (nereikalingus) laisvės laipsnius, kodo dubliavimąsi ir automatizavus pasikartojančius darbus.

Tai svarbu atsižvelgiant į tai, jog į kalbos bei programos kūrimą gali būti įtraukti ir ne programuotojai, o pavyzdžiui – mokytojai ar mokiniai. DSL naudojimas sumažina tikimybę, kad kode atsirastų naujų klaidų.

4.4. Kūrėjai ir srities specialistai

Jei yra būdas išreikšti srities problematiką kalba, kuri yra suderinama su ta sritimi, vartotojo mąstymas tampa aiškesnis, nes parašytas kodas nėra užgriozdintas įgyvendinimo detalėmis. Kitaip tariant, DSL naudojimas leidžia atskirti esminį nuo atsitiktinio sudėtingumo. DSL, kurių sritis, abstrakcijos ir žymėjimai yra glaudžiai suderinti su tuo, kaip srities ekspertai (t.y. ne programuotojai) reiškia mintis, leidžia sukurti gerą integraciją tarp programuotojų ir tos srities žmonių, tai padeda gerinti komandos bendradarbiavimą.

Kai DSL sritis yra švietimas, bei edukaciniai žaidimai, tai suteikia galimybę mokytojams, studentams ar tiems, kurie nori kurti tokius žaidimus įsitraukti į žaidimo kūrimą, pvz.: kurti matematinius klausimus ir t.t.

4.4.1. Geresnis srities supratimas

Be geresnio komandos bendradarbiavimo ir vartotojų įsitraukimo, kalbos kūrimas gali padėti geriau suprasti sritį, kuriai kuriamas DSL. Tai taip pat padeda išsklaidyti nesutarimus susijusius su ta sritimi, kylančius iš skirtingų žmonių, skirtingai sprendžiančių tą pačią problemą. Kai kuriomis prasmėmis kalbos apibrėžimas yra „vykdomosios analizės modelis“. Markus Völter teigia ne kartą girdėjęs, kad po trijų dienų DSL prototipų kūrimo seminario klientai sako, jog jie daug sužinojo apie savo sritį ir kad net jei niekada nebesinaudotų DSL, vien tik tai būtų verta pastangų, išleistų kuriant specializuotąją kalbą [VBD⁺13].

Tai ypač naudinga kuriant edukacinius žaidimus. Jei mokinys ne vien žaidžia paruoštą žaidimą, bet ir yra įtraukiamas į kūrimo procesą, šis gali išmokti apie tą sritį kur kas daugiau, nei tiesiog žaidžiant.

5. Taikymas

Šiame skyrelyje aprašomas šiam darbui sukurtas pavyzdinis edukacinis žaidimas bei specializuotos kalbos jam, taip pat ir keli kiti MPS projektai, kurie padėjo pasiekti galutinį rezultatą ir gali būti naudingi skaitytojui.

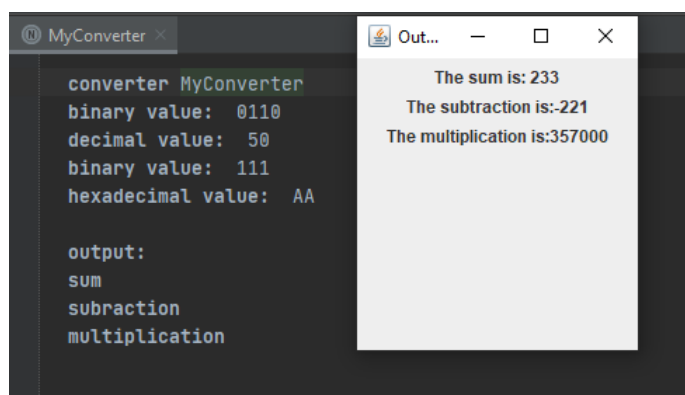
5.1. Pradinis projektas

Pirmasis sukurtas MPS projektas (žr. Priedas nr. 1), skirtas susipažinti su įrankiu, yra skaičiuotuvo tipo sistema, kur vartotojas gali įvesti binarinį, dešimtainį arba šešioliktainį skaičių ir su visais įvestais skaičiais atlikti sumos atimties arba daugybos veiksmus. Ši programa visas įvestas vertes paverčia į dešimtainius skaičius, o tuomet vykdomi skaičiavimai su jais.

Šiai programai sukurta viena skaičiuotuvo meta kalba, susidedanti iš sąvokų:

- „Converter“ arba konvertavimo skaičiuoklė. Tai šakninis mazgas, turintis vaikus: „Representation“ ir „Output“ sąrašus.
- „Representation“ – tai abstrakti sąvoka, kurią išplečia tokios sąvokos kaip „Binary“, „Decimal“, „Hexadecimal“.
- „Output“ sąvoka taip pat yra abstrakti, o ją išplečia sąvokos: „Sum“, „Subtraction“, „Multiplication“
- „Binary“, „Decimal“, „Hexadecimal“ konceptai turi reikšmės savybę. Šių sąvokų redukcijos taisyklės konvertuoja jų reikšmes į dešimtainę sistemą.
- „Sum“, „Subtraction“, „Multiplication“ sąvokos reiškia pasirinktą skaičiavimo veiksmą.

Paleidus programą generatorius iš redukcijos šablonų gauna įvestus skaičius dešimtaine sistema, taip pat ir parinktus skaičiavimo veiksmus. Šiems veiksmams kviečiamos atitinkamos sukurtos funkcijos ir rezultatas parodomas „JFrame“ lange (2 pav.).



2 pav. Skaičiuotuvo projekto redaktorius ir gauti rezultatai

5.2. Žaidimo projektas

Antrasis MPS projektas (žr. Priedas nr. 2) yra skirtas išsiaiškinti kaip galima pritaikyti MPS įrankį žaidimų kūrimui. Šis projektas yra konsolinis žaidimas „Kryžiukai-nuliukai“, kuris leidžia vartotojui sukurti meta kalbą rašyti kodą, kuris nusakytų kokius ėjimus žaidėjas turėtų daryti.

Šio žaidimo meta kalba yra suskirstyta į komandų ir loginių sakinių sąvokų paketus. Komandų paketo sąvokos:

- „AbstractCommand“ – abstrakti sąvoka apibūdinanti komandą. Ją praplečia visos vėliau aprašytos komandos.
- „CommandList“ yra sąvoka, turinti vaiką – abstrakčių sąvokų sąrašą.
- „EmptyLine“ leidžia vartotojui sukurti tuščią eilutę tarp kitų kodo eilučių. Ši sąvoka turi celių veiksmų žemėlapi, kuriame aprašoma kas turi įvykti, jei vartotojas nori tuščią eilutę ištrinti.
- „IfStatement“ arba sąlygos sakinyss „if“. Tai sąvoka turinti vaikus: „trueBranch“, „falseBranch“ ir „condition“. „True“ ir „False“ šakos apibūdina kokios komandos turi būti vykdomos, jei sąlyga tenkinama/netenkinama.
- „While“ taip pat yra sąlygos sakinyss, naudojamas daugelyje programavimo kalbų. Ši sąvoka turi vaikus „condition“ ir „body“ – tai yra komandų sąrašą, kuris vykdomas, jei sąlyga tenkinama.
- „Move“ sąvoka nusako žaidėjo judėjimą lentoje ir ji turi vieną savybę – lentos vietą, į kurią norima padaryti ėjimą

Loginių sakinių paketo sąvokos:

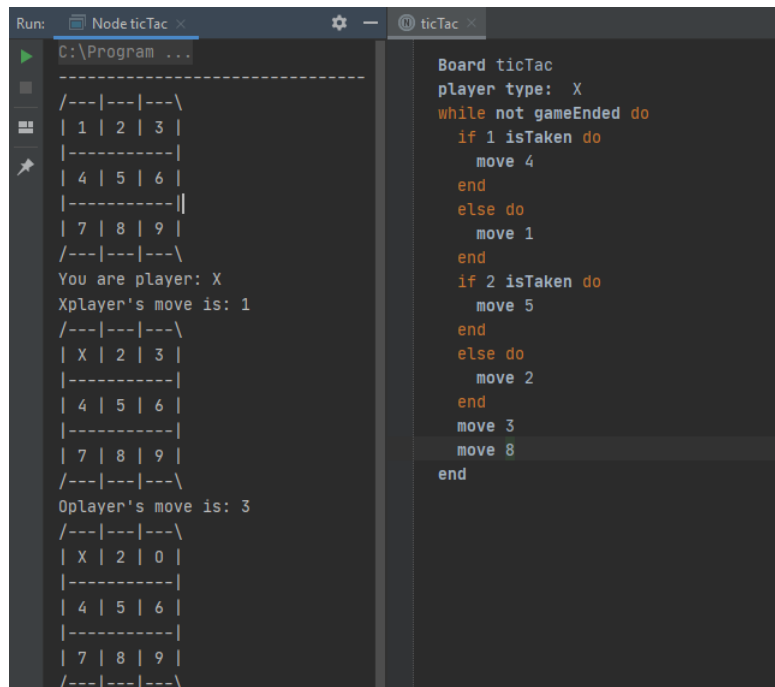
- „LogicalExpression“ yra abstrakti koncepcija, kurią praplečia „GameEnded“, „Not“ ir „isTaken“ sąvokos.
- „GameEnded“ redukcijos šablone kviečiama funkcija, nustatanti, ar žaidimas yra pasibaigęs ir yra žinomas laimėtojas. Šią koncepciją galima naudoti, pavyzdžiui, „while“ arba „if“ sakinyje.
- „isTaken“ turi savybę „vieta“, o redukcijos šablone kviečiama funkcija, nustatanti, ar ta vieta jau yra užimta. Ją taip pat galima naudoti, „while“ arba „if“ sakiniuose.
- „Not“ turi vaiką kitą loginį sakinį, prie kurio galima pridėti sąvoką „not“. Ji pakeičia sakinio būseną iš „true“ į „false“ ir atvirkščiai.

Apibendrinančios sąvokos:

- „Player“ apibūdina tai, koks veikėjas yra pasirinktas: „x“ ar „o“ tipo.

- „Board“ yra šakninis mazgas apibūdinantis visą kalbą. Ši sąvoka turi vaikus: „Player“ ir komandų sąrašą.

Paleidus programą generatorius transformuoja sąvokas į Java kalbos kodą ir vartotojas gali pamatyti visą žaidimo eigą konsolėje (3 pav).



```

Board ticTac
player type: X
while not gameEnded do
  if 1 isTaken do
    move 4
  end
  else do
    move 1
  end
  if 2 isTaken do
    move 5
  end
  else do
    move 2
  end
  move 3
  move 8
end
  
```

```

-----
/---|---|---\
| 1 | 2 | 3 |
|---|---|---|
| 4 | 5 | 6 |
|---|---|---|
| 7 | 8 | 9 |
/---|---|---\
You are player: X
Xplayer's move is: 1
/---|---|---\
| X | 2 | 3 |
|---|---|---|
| 4 | 5 | 6 |
|---|---|---|
| 7 | 8 | 9 |
/---|---|---\
Oplayer's move is: 3
/---|---|---\
| X | 2 | 0 |
|---|---|---|
| 4 | 5 | 6 |
|---|---|---|
| 7 | 8 | 9 |
/---|---|---\
  
```

3 pav. „Kryžiukų-nuliukų“ projekto redaktorius ir žaidimo eiga konsolėje

5.3. Galutinis projektas

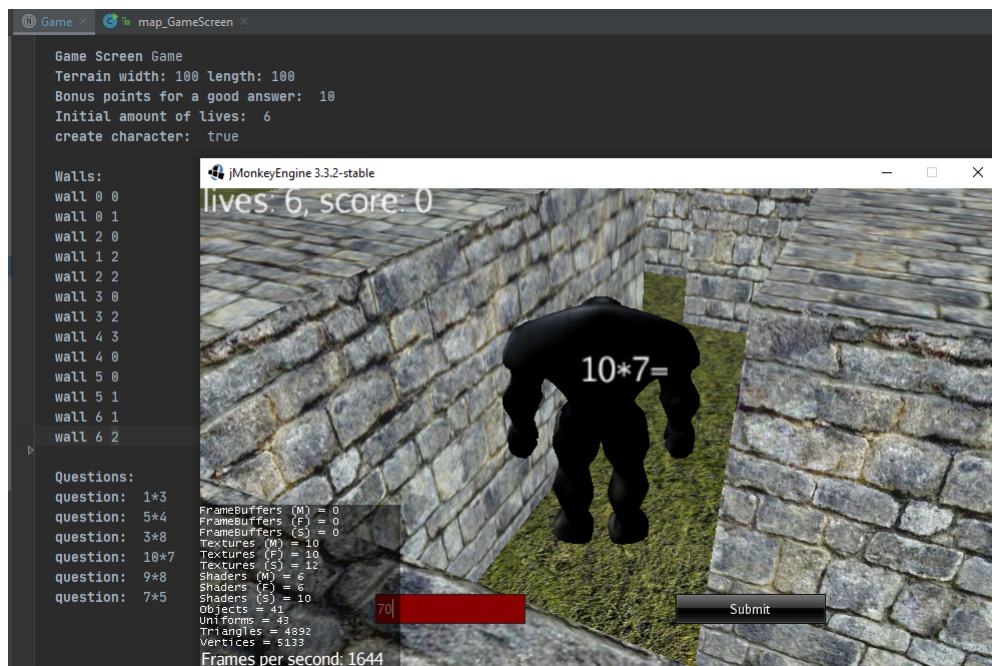
Galutinio projekto tikslas yra sukurti edukacinį žaidimą, kuris padėtų mokytis matematikos, ir specializuotąsias kalbas jam. Žaidimo ir meta kalbų kūrimui buvo naudotas „JetBrains MPS“ įrankis bei „jMonkeyEngine“ atviro kodo žaidimų variklis (angl. *game engine*). „jMonkeyEngine“ sukurtas Java kalbos pagrindu, todėl yra nesunkiai prieinamas MPS aplinkoje, kuri taip pat paremta Java kalba.

Šis projektas yra edukacinis žaidimas (žr. Priedas nr. 3), kuris padeda mokytis daugybės lentelės ar kitokių matematinių skaičiavimų. Žaidimo veikėjas yra vaikščiojantis „trolis“, kuris eina per labirintą ir kas kelis metrus jam yra užduodami matematiniai klausimai (4 pav.)

Programos naudotojas redaktoriuje nurodo:

1. žaidimo pasaulio ilgį ir plotį,
2. taškus, kurie suteikiami už gerai atsakytą klausimą,
3. pradinį gyvybių skaičių,
4. „true“ prie veikėjo sukūrimo sakinio, jei norima sukurti veikėją
5. sąrašą labirinto sienų, kurioms nurodomos x ir z vertės

6. daugybės lentelės ar kitokių matematinių klausimų sąrašą, pvz.: $5*7$.



4 pav. Edukacinis žaidimas ir jo redaktorius(kairėje)

Projektas yra sudarytas iš trijų specializuotųjų kalbų: „Engine“, „FeedbackLang“, „Math-TaskLang“. MPS kalbos gali lengvai naudoti, įsiterpti ar kvieisti viena kitą. Kadangi projekcinis redagavimas nepalaiko dviprasmiškų konstrukcijų, kilusių iš skirtingų kalbų, MPS nereikia vienijančio analizatoriaus konstravimo ar sudėtinių gramatikų apibrėžimo. Kalbas galima praplėsti naujomis konstrukcijomis ar naujais būdais jas peržiūrėti ir redaguoti.

5.4. Žaidimo meta kalbos sąvokos

MPS sąvokos/koncepcijos yra apibrėžimai, apibūdinantys abstrakčią DSL sintaksės elemento struktūrą. Todėl DSL kūrimas prasideda nuo jo sąvokų apibrėžimo.

5.4.1. „Engine“ sąvokos

Šiam žaidimui sukurti prireikė tokių esminių sąvokų, apibrėžtų „Engine“ kalboje:

- „MainCharacter“ arba „pagrindinis veikėjas“: ši sąvoka apibrėžia žaidimo veikėjo sukūrimą ir ji turi vieną savybę – „created“, kuri yra loginio tipo ir reiškia, jog veikėjas yra sukuriamas, jei savybei priskirta būsena „true“.
- „TerrainSize“ arba „vietovės dydis“ su savybėmis ilgis ir plotis.
- „Wall“ arba „sienos“ koncepcija apibūdina statomą labirinto sieną, kuri turi savybes „x-value“ bei „z-value“.

- „GameScreen“ arba „žaidimo ekranas“: ši sąvoka padeda apibūdinti žaidimo struktūrą, ji turi tokius vaikų mazgus kaip: „TerrainSize“, „Wall“, „MainCharacter“, „Lives“, „Score“, bei „Questions“. Ši sąvoka yra šakninis mazgas, tai reiškia, jog sukurtas generatorius tai kalbai bus veikiamas būtent tų savokų, kurios yra nurodytos šakniniame mazge. Tai leidžia kurti atskirus „GameScreen“ atvejus.

5.4.2. „FeedbackLang“ sąvokos

„FeedbackLang“ kalboje buvo sukurtos sąvokos, padedančios žaidėjui gauti grįžtamąjį ryšį:

- „Lives“ arba „gyvybės“ – žaidėjas gali aiškiai matyti kiek yra likę gyvybių. Ši sąvoka turi savybę „initialLives“ arba „pradinės gyvybės“.
- „Score“ arba „taškai“ su savybe „bonus“, kuri nurodo kiek taškų bus pridėdama už gerai atsakytą klausimą.

5.4.3. „MathTaskLang“ sąvokos

„MathTaskLang“ kalboje apibrėžtos matematinių klausimų sąvokos:

- „Question“ sąvoka su savybe „task“, kuri apibūdina atskirą matematinį uždavinį
- „Questions“ sąvoka turi vaiko mazgą „Question“, todėl ši sąvoka laiko savyje matematinių uždavinių sąrašą.

5.5. Kodo generavimas ir paleidimas

Norint paleisti žaidimą, būtinas vykdomasis kodas, kuris sukuriamas kodo generatorių. Kiekvienai kalbų sąvokai, kuri nėra šakninė, yra sukuriama redukcijos taisyklė, o jos panaudojamos generuojant Java kalbos kodą. Kadangi „jMonkeyEngine“ taip pat naudoja Java kalbą nesunku susieti šio „variklio“ bibliotekas su kuriu projektu.

Rezultatai ir išvados

Šiame darbe buvo aprašyta, kas yra meta kalbos bei meta programavimas. Apžvelgta edukacinių žaidimų sąvoka ir jų nauda mokymosi tikslų siekimui.

Buvo išanalizuotas „JetBrains MPS“ specializuotųjų kalbų kūrimo įrankis, apibūdinti tradicinio ir meta kalbos kodų darna, išvardinti žingsniai, vedantys į specializuotosios kalbos kūrimą MPS, taip pat aptarta bazinė MPS kalba.

Pasinaudojus pateikta informacija sukurti du projektai, skirti susipažinimui su žaidimų kūrimu MPS aplinkoje, ir trečiasis – edukacinis žaidimas ir meta kalbos jam, pasinaudojus MPS įrankiu. Šios kalbos aprašytos ir apibūdintos jų sąvokos, bei sąvokų svarba. Nurodyta, kaip yra generuojamas baigtinis žaidimo kodas.

Pasirėmus patirtimi kuriant projektus ir papildoma informacija, suformuoti punktai, aprašantys meta kalbą ir „JetBrains MPS“ naudą kuriant edukacinius žaidimus:

- Abstrakcijos ir geras kodo bei sąvokų skaitomumas daro MPS paprastu naudoti, žmonėms patyrusiems MPS programavime.
- Meta kalbų abstrakcijos padeda taupyti laiką, o taip pat DSL padeda atsikratyti dalies monotoniško darbo, kas padeda sumažinti kodo eilučių skaičių.
- DSL naudojimas gali pagerinti produkto kokybę, daroma mažiau klaidų.
- DSL leidžia srities specialistams ir vartotojams labiau įsiliesti į žaidimo kūrimą, o tai veda link geresnio tos srities supratimo, bei efektyvesnio idėjų generavimo ir projekto kūrimo.

Apibendrinus darbo rezultatus: surinktą informaciją, bei sukurtą edukacinio žaidimo projektą, galima suformuoti tokias išvadas:

- Kadangi pasaulis tampa vis modernesnis, technologijos išlieka svarbiu kasdienio žmogaus gyvenimo aspektu, todėl ateities švietimo sistema taip pat turės keistis. Edukaciniai žaidimai yra ateities vaikų švietimo proceso dalis, kadangi manoma, jog mokomieji žaidimai gali pagerinti moksleivių socialinius įgūdžius, padidinti motyvaciją mokytis ir kt.
- Tam, kad edukaciniai žaidimai taptų populiariesni švietimo įstaigose, norisi turėti įvairesnius instrumentus jiems kurti. Manoma, kad DSL panaudojimas ir atitinkami instrumentai gali išplėsti edukacinių žaidimų sritį, praturtinti praktikas ir labiau įtraukti moksleivius į mokymosi procesą.
- MPS gali būti tinkamas įrankis kurti edukacinius žaidimus vyresniems vaikams ar paaugliams ir labiau programavime patyrusiems mokytojams/dėstytojams, nes nėra labai lengva priprasti prie MPS redaktoriaus, o laikas yra sutaupomas tik tuomet, kai žmogus yra susipažinęs su sistema.

Tiesa, jei žaidimą kuria programavime patyręs ir su MPS susipažinęs žmogus, jis gali sutrumpinti laiko tarpą nuo žaidimo idėjos sugalvojimo iki jos įgyvendinimo, bei sumažinti

kodo eilučių skaičių. Taip pat MPS gali padėti rečiau daryti klaidas kode, bei padėti vartotojui labiau įsigilinti į sritį, kuriai yra kuriamos meta kalbos. Studentai arba moksleiviai prisidėję prie edukacinio žaidimo kūrimo gali išmokti dar daugiau apie tą sritį (ir potencialiai programavimą) nei būtų išmokę vien tik žaidžiant žaidimą.

- Ateityje galima būtų labiau išplėsti sukurtą projektą: sukurti atskiras kalbas mokytojams ir mokiniams. Mokytojai turėtų galėti patogiai naudotis jiems sukurta kalba, kuri būtų sudaryta iš vertinimo sistemos, labirinto sudarymo, nuobaudų skyrimo ir kitokių sąvokų. Mokinai galėtų turėti jiems skirtą kalbą, kurioje galima būtų nustatyti tam tikrą žaidimo sudėtingumą. Kurti scenas, labirintus dabartiniame projekte vartotojui nėra labai patogiu, nes jis nežino, kaip atrodys žaidimo pasaulio vaizdas paleidus programą. Galima būtų panaudoti daugiau MPS redaktoriaus privalumų ir sukurti lentelę, vaizduojančią žaidimo žemėlapių vaizdą iš viršaus.

Literatūra

- [AHH⁺18] Hans Christian Arnseth, Thorkild Hanghøj, Thomas Duus Henriksen, Morten Misdeldt, Robert Ramberg ir Staffan Selander. *Games and Education: Designs in and for Learning*. BRILL, 2018.
- [Are12] NBC Bay Area. Google's blockly teaches you to create apps. 2012. URL: <https://www.nbcbayarea.com/news/national-international/googles-blockly-teaches-you-to-create-apps/1918242/> (tikrinta 2012-06-13).
- [Ber06] Bryan P. Bergeron. *Developing serious games*. eng. Hingham, 2006. ISBN: 1584504447. URL: <http://lib.ugent.be/catalog/rug01:001364766>.
- [BKS14] Frank A Buckless, Kathy Krawczyk ir D Scott Showalter. Using virtual worlds to simulate real-world audit procedures. *Issues in Accounting Education*, 29(3):389–417, 2014.
- [CE00] K. Czarnecki ir U. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison Wesley, 2000. ISBN: 9780210309773. URL: <https://books.google.lt/books?id=4erCMgEACAAJ>.
- [Dmi04] Sergey Dmitriev. Language oriented programming: the next programming paradigm. *JetBrains onBoard*, 1(2):1–13, 2004.
- [Jet19] JetBrains. Domain-specific languages. 2019. URL: <https://www.jetbrains.com/mps/concepts/domain-specific-languages> (tikrinta 2019-01-01).
- [KIV10] Esma Buluş Kırıkkaya, Sebnem Iseri ir Gurbet Vurkaya. A board game about space and solar system for primary school students. *Turkish Online Journal of Educational Technology*, 9:1–13, 2010.
- [PSV13] Vaclav Pech, Alex Shatalin ir Markus Voelter. JetBrains mps as a tool for extending java. *Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools, PPPJ '13*, p. 165–168, Stuttgart, Germany. Association for Computing Machinery, 2013. ISBN: 9781450321112. DOI: 10.1145/2500828.2500846. URL: <https://doi.org/10.1145/2500828.2500846>.
- [ŠD12] V. Štuikys ir R. Damaševičius. *Meta-Programming and Model-Driven Meta-Program Development: Principles, Processes and Techniques*. Advanced Information and Knowledge Processing. Springer London, 2012. ISBN: 9781447141266. URL: <https://books.google.lt/books?id=B85FcaE9YdgC>.
- [VBD⁺13] Markus Völter, Sebastian Benz, Christian Dietrich, Birgit Engelmann, Mats Helander, Lennart C. L. Kats, Eelco Visser ir Guido Wachsmuth. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013. ISBN: 978-1-4812-1858-0. URL: <http://www.dslbook.org>.

- [VVC12] Sylke Vandercruysse, Mieke Vandewaetere ir Geraldine Clarebout. *Game-based learning: a review on the effectiveness of educational games*. Tom. 1. 2012-02, p. 628–647. ISBN: ISBN10: 1466601493. DOI: 10.4018/978-1-4666-0149-9.ch032.

Sąvokų apibrėžimai

Specializuota kalba / meta kalba – specialios paskirties kalba, kurios priemonėmis aprašoma kuri nors kita kalba.

Metaprogramavimas – programavimas susijęs su metakalbomis.

Edukaciniai žaidimai – žaidimai, specialiai sukurti švietimo tikslais arba turintys ugdomąją vertę.

MPS – „JetBrains“ sukurtas įrankis skirtas į kalbą orientuotam programavimui (angl. „*MetaProgrammingSystem*“).

Žaidimų variklis – programinės įrangos sistema, skirta kompiuterinių žaidimų kūrimui ir plėtojimui.

„jMonkeyEngine“ – žaidimų variklis, specialiai sukurtas moderniam 3D kūrimui, nes jame plačiai naudojama šešėlių technologija. Naudojant šį variklį 3D žaidimus galima kurti tiek „Android“ įrenginiams, tiek kompiuteriams.

Santrumpos

DSL – kalba, specializuota tam tikroje taikymo srityje (angl. *domain-specific language*).

LOP – programinės įrangos kūrimo paradigma, kai „kalba“ yra programinės įrangos kūrimo pagrindas, turintis tokį patį statusą kaip objektai, moduliai ir komponentai, o užuot sprendę bendrojo naudojimo programavimo kalbų problemas, programuotojai sukuria vieną ar daugiau DSL problemai spręsti, ir tada išsprendžia problemą tomis kalbomis. (angl. *Language-oriented programming*)

IDE – integruota kūrimo aplinka, programų kūrimo aplinka.

AST – abstraktus sintaksės medis (angl. *Abstract syntax tree*).

OOD – objektiškai orientuotas projektavimas (angl. *Object-oriented design*).

Priedas nr. 1

MPS pradinis skaičiuotuvo projektas

Šiam darbui sukurtą MPS projektą, kuriame atliekami matematiniai veiksmai su įvairių skaičiavimo sistemų skaičiais, galima rasti adresu:

<https://github.com/gabrielezi/ConverterMPS>

Reikalavimai, prieš paleidžiant projektą:

- Reikia įdiegti MPS įrankį. Atsisiuntimo nuoroda: <https://www.jetbrains.com/mps/download>
- Reikia kompiuteryje turėti Java JDK 1.8 ar naujesnę versiją.

Kaip paleisti projektą:

1. Atsisiųsti projekto zip aplanką arba klonuoti Git projektą.
2. Atsidaryti projektą MPS IDE.
3. Išplėtus kairėje esantį failų medį, eiti į katalogą „`pirma.converter2.sandbox/pirma.converter2/sandbox`“.
4. Nuvedus pelę ties „MyConverter“ failu paspausti dešinę pelės mygtuką,
5. Paspausti „Run „Node MyConverter““

Priedas nr. 2

„MPS“ edukacinio žaidimo projektas

Šiame darbe aprašytą MPS „Kryžiukų-nuliukų“ žaidimą galima rasti adresu:

<https://github.com/gabrielezi/MPSTictactoe>

Reikalavimai, prieš paleidžiant projektą:

- Reikia įdiegti MPS įrankį. Atsisiuntimo nuoroda: <https://www.jetbrains.com/mps/download>
- Reikia kompiuteryje turėti Java JDK 1.8 ar naujesnę versiją.

Kaip paleisti projektą:

1. Atsisiųsti projekto zip aplanką arba klonuoti Git projektą.
2. Atsidaryti projektą MPS IDE.
3. Išplėtus kairėje esantį failų medį, eiti į katalogą „antra.Tictactoe.sandbox/antra.Tictactoe/sandbox“ .
4. Nuvedus pelę ties „ticTac“ failu paspausti dešinę pelės mygtuką.
5. Paspausti „Run „Node ticTac““

Priedas nr. 3

MPS žaidimo projektas

Šiame darbe aprašytą MPS edukacinio žaidimo projektą (galutinį projektą) galima rasti adresu:
`github.com/gabrielezi/MPS_JMonkeyEngine`.

Reikalavimai, prieš paleidžiant projektą:

- Reikia įdiegti MPS įrankį. Atsisiuntimo nuoroda: `https://www.jetbrains.com/mps/download`
- Reikia kompiuteryje turėti Java JDK 1.8 ar naujesnę versiją.

Kaip paleisti projektą:

1. Atsisiųsti projekto zip aplanką arba klonuoti Git projektą.
2. Atsidaryti projektą MPS IDE.
3. Išplėtus kairėje esantį failų medį, eiti į katalogą „Engine.sandbox/Engine/sandbox“.
4. Nuvedus pelę ties „Game“ failu paspausti dešinę pelės mygtuką,
5. Paspausti „Run „Node Game““