ŠIAULIŲ UNIVERSITETAS
FIZIKOS IR MATEMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

ZIGMANTAS RAČKAUSKAS

JUDESIO OBJEKTIŠKAI ORIENTUOTO PROJEKTAVIMO MODELIAI

MAGISTRO DARBAS

Darbo vadovas:
prof. habil. dr. G. Kulvietis

Darbo recenzentas:
asist. V. Giedrimas

Šis darbas yra originalus ir nebuvo teikiamas kuriam nors laipsniui ar kvalifikacijai įgyti…….

ŠIAULIAI, 2005

Turinys

I. Įvadas....2

II. Kompiuteriniai judesio modeliai ir problemų analizė 4

1) Temos analizė 4
2) Kompiuteriniai judesio modeliai ir problemų analizė 4
3) Animacijos modeliavimo aplinkų analizė 9
4) Apibendrinimas.... 13

III. Judesio objektiškai orientuotas programavimas....14

1) Programavimo kalbos pasirinkimas.... 14
2) Judesio objektiškai orientuoto programavimo metodologija 14
3) Judesio objektiškai orientuoto programavimo taikymas.... 24
4) Taikymas 27

IV. Objektiškai orientuoto projektavimo technologijų analizė....28

1) Modeliavimo įrankių ir priemonių pasirinkimas 28
2) Objektiškai orientuoto projektavimo technologijos.... 28
3) Animacijos objektiškai orientuotas projektavimas 33
4) Apibendrinimas.... 40

V. Išvados 42

VI. Literatūros ir informacinių šaltinių aprašai....43

VII. Anotacija (Summary) 45

I. Įvadas

Kompiuterinei grafikai pradžią davė karinės pramonės vystymasis ir joje užimtų žmonių profesionalus rengimas. Taip pat populiarėjantys kompiuteriniai žaidimai, dabar netgi sugebantys sukurti dalyvavimo visame tame jausmą. Spartus kompiuterinės grafikos vystymasis prasidėjo 8-jame dešimtmetyje.

Aktualumas. Tema pasirinkta todėl, kad judesio objektiškai orientuotas modeliavimas Internetinei animacijai (judesiui) Lietuvoje praktiškai netaikomas arba taikomas labai mažai

Tyrimo objektas. Šio magistro darbo tyrimo objektas – tai judesio modeliavimas naudojant objektiškai orientuotą projektavimą. Darbe nagrinėjami ne tik patys judesio modeliai, bet ir judesiams modeliuoti skirtos sistemos, turinčios objektiškai orientuoto projektavimo galimybių.

Hipotezė. Nors dauguma Lietuvos programuotojų ir projektuotojų dar netaiko objektiškai orientuoto projektavimo, kurdami dinamiškas Interneto svetaines, prezentacijas, judančius logotipus ir pan., tačiau pasaulyje ši projektavimo technologija plačiai naudojama, todėl Lietuvoje greitu laiku ji bus taip pat naudojama.

Tikslas ir uždaviniai. Pagrindinis šio darbo tikslas yra apibrėžti animacijos kūrimą Internetinėje aplinkoje, taikant objektiškai orientuotą projektavimą, bei kiek leidžia magistro darbo apimtis, išnagrinėti galimus šių problemų sprendimo kryptis ir būdus. Papildomas tikslas – kompiuterinio judesio modeliavimas taikant objektiškai orientuotą projektavimą įvairiose sistemose ir aplinkose, modeliavimo sistemų palyginimas ir analizė

Siekiant nurodytų tikslų, šiam darbui iškelti tokie uždaviniai:

1) Apžvelgti objektiškai orientuoto projektavimo sistemas judesiui modeliuoti.
2) Išnagrinėti sistemas, kuriose yra galimybė kurti judesio modelius Internetinei aplinkai.
3) Aptarti, kitų sistemų, programavimo kalbų panaudojimą judesio modeliavime. Rasti tų sistemų pagrindinius privalumus ir trūkumus.
4) Parengti keletą judesio modelių.
5) Panagrinėti sukurtas objektų klases.
6) Skatinti judesio objektiškai orientuotą projektavimą Lietuvoje.

Tyrimo metodologija. Tiriant judesio objektiškai orientuoto projektavimo modeliavimą, sistemas bei formuluojant šio darbo išvadas buvo remiamasi sisteminiu, loginiu, istoriniu, lyginamuoju metodais.

Sisteminis metodas. Šis metodas padėjo analizuoti objektiškai orientuoto projektavimo problemas tiek Informatikos tiek Geometrijos mokslų kontekste. Pavyzdžiui norint sumodeliuoti besisukantį kubą neužtenka vien programavimo žinių, reikalingos ir geometrinės transformacijos žinios.

Loginis metodas. Būtinas atskleidžiant, bet kokio tiriamojo darbo tikslus, išvadas bei apibendrinimus.

Objektiškai orientuoto projektavimo sistemų analizė. Panaudota išsiaiškinti vienos ar kitos sistemos pritaikymą, kuriant objektiškai orientuotą modelį.

Istorinis metodas. Reikalingas: nagrinėjant judesio modelių atsiradimą bei tobulėjimą; objektiškai orientuoto projektavimo pritaikymo judesiui modeliuoti atsiradimą, vystymąsi. Pavyzdžiui šiame moksliniame darbe apžvelgta, judesio modelių ir sistemų tiem judesiams modeliuoti, evoliucija.

Lyginamasis metodas padeda palyginti vieną ar kitą sistemą tarpusavyje. Darbe šis metodas panaudotas lyginant judesio objektiškai orientuoto projektavimo sistemas.

Tyrimo etapai. Pirmas etapas (2003-2004). Temos formulavimas. Tikslo ir uždavinių iškėlimas. Hipotezių formulavimas. Literatūros analizavimas. Kai kurių projektų kūrimas ir įgyvendinimas.

Antras etapas (2004-2005). Literatūros analizavimas. Projektų kūrimas ir įgyvendinimas. Darbo aprašymas. Išvadų formulavimas.

Darbo naujumas. Tiek objektiškai orientuoto projektavimo tiek programavimo panaudojimas Interneto animacijai, žaidimams, reklamai kurti dar visiškai naujas dalykas Lietuvoje. Animacija dar vis kuriama kadrų principu.

Darbo praktinė vertė. Sukurti realūs projektai naudojami įvairiose Internetinėse svetainėse. Puslapių adresai pateikti kompaktiniame diske kataloge *Priedai/Projektai/Adresai.html*.

Darbo aprobacija. Beveik visi sukurti realūs modeliai su aprašymais yra pateikti ir naudojami IĮ „SSP“, IĮ „DOUBLE“ ir kt.

Darbo struktūra. Magistro darbą sudaro įvadas, trys dalys ir išvados. Kiekviena darbo dalis suskirstyta į atitinkamus skyrius pagal nagrinėjamų klausimų pobūdį. Skyriai skirstomi į poskyrius, kur atitinkamai nagrinėjami galimi to paties klausimo aspektai. Darbo sandarai įtakos turėjo tai, jog darbe nagrinėjami ne tik judesio objektiškai orientuoto projektavimo modeliai, bet ir sistemos, kuriuose tie modeliai projektuojami bei naudojami.

II. Kompiuteriniai judesio modeliai ir problemų analizė

1) Temos analizė

Pagal magistrinio darbo pavadinimą galimos dvi temos plėtojimo kryptys:

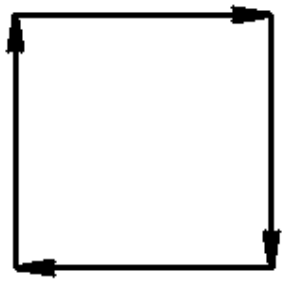
- Robotų judesių projektavimas naudojant objektiškai orientuotas sistemas;
- Judesio kaip animacijos projektavimas naudojant OOP;

Darbe pasirinkau antrąją kryptį, t.y. judesio kaip animacijos projektavimas naudojant objektiškai orientuotas projektavimo sistemas.

Tobulėjant technologijoms, kyla vis didesni reikalavimai programoms, svetainėms, žaidimams, filmams. Ypač keliami aukšti reikalavimai judesio perteikimui. Taip pat per televiziją vyksta reklamų lenktynės – kas kokią animaciją geriau pateiks. Internete pilna dinaminių reklaminių skelbimų ir kiekvienas iš jų už kitą gražesnis – patrauklesnis. Dinaminis vaizdas visada labiau traukė vartotojo dėmesį, negu statinis. Kartais vartotojas naršydamas Internete, žaisdamas kompiuterinį žaidimą, ar žiūrėdamas specialiųjų efektų prikimštą filmą, net nesusimasto kokiomis technologijomis naudojantis visa tai buvo sukurta, kiek į tai įdėta darbo ir pan.

Kompiuterinė animacija, kuo toliau tuo atrodo realistiškesnė. Naudodami įvairias simuliacijas, fizikos dėsnius, programuotojai į kompiuterį įkelia dirbtinius: bėgantį, plaukiantį, važiuojantį dviračiu ar šuoliuojantį žmogų.

2) Kompiuteriniai judesio modeliai ir problemų analizė



1.2.1 pav. Vektoriniuose paveikslėliuose, vaizdas sudaromas iš vektorių

Vektorinė ir rastrinė grafika. Pagal kodavimo būdą kompiuteriniai vaizdai skirstomi į vektorinius ir rastrinius[1].

Vektorinė grafika. Pagrindinis vektorinės grafikos elementas yra *objektas* – uždara arba atvira geometrinė figūra, turinti matematinėmis formulėmis aprašomą kontūrą. Vaizdas sudaromas tik iš objektų, kurių kontūrų koeficientai ir formulės saugomi, kompiuterio atmintinėje, kaip parametrai. Vektoriniuose grafiniuose failuose yra matematiškai aprašytos paveikslo vaizdavimo taisyklės. Visas paveikslas yra sudarytas iš vektorių, pavyzdžiui, paveiksliukas, kuriame pavaizduotas kvadratas, aprašomas keturiais vektoriais (1.2.1 pav.) [2.6].

Vektorinės grafikos privalumai:

[1] Rastriniai vaizdai kartais dar yra vadinami taškiniais vaizdais. Tai tokie grafiniai paveikslėliai sudaryti iš taškų.

- Vaizdinės bylos yra palyginti nedidelės, nes saugomas ne pats piešinys turintis konkretų aukštį ir plotį, bet tik būtiniausi jo parametrai, pagal kuriuos programa kaskart vaizdą atkuria iš naujo;
- Objektai lengvai transformuojami[2] ir dėl to nenukenčia jų kokybė;

Vektorinės grafikos trūkumai:

- Neįmanoma sukurti tikroviškų vaizdų ir automatizuotai įvesti grafinę informaciją;
- Reikia gana galingo kompiuterio, kad būtų galima atlikti matematinius skaičiavimus paveikslėlio atvaizdavimui[3].

Populiariausios vektorinės grafikos programos:

- CorelDRAW;
- Macromedia Flash;
- Macromedia FreeHand;
- 3D Max;

Populiariausias dinaminės, vektorinės grafikos, formatas[4] naudojamas Internete: SWF.

1.2.2 pav. Rastrinis vaizdas sudarytas iš taškų

Rastrinė grafika. Iš smulkių taškų arba linijų sudaryto vaizdo struktūra vadinama rastru[5]. Norint kompiuteryje šiuo principu užkoduoti piešinį, jis padalijamas į labai mažus vienodo dydžio elementus – taškus[6].

Taškinį vaizdą apibūdina parametrai:

- Aukštis;
- Plotis;
- Raiška;
- Kiekvieno rastro taško spalva;

Fizinis rastro elemento dydis įvertinamas raiška[7].

Taškinės grafikos privalumai:

- Kodavimo paprastumas;
- Galimybė automatizuotai įvesti tikroviškus vaizdus vaizdo kameromis, skaitmeniniais fotoaparatais, skaitytuvais;
- Paveiksliukai gali būti vaizduojami, turint nesudėtingą video plokštę.

Taškinės grafikos trūkumai:

[2] Transformuojami – reiškia, jog paveikslėlius galima sukinėti tam tikru kampu, keisti aukštį ir plotį.
[3] Šiais laikais dauguma asmeninių kompiuterių yra gana galingi, tačiau kai kuriuose įstaigose, ypač mokyklose vis dar naudojami seni, mažo dažnio kompiuteriai.
[4] Formatas – tai taisyklių rinkinys, bylos kodavimo sistemos apibrėžimas.
[5] Rastras – mozaika, vitražas
[6] Pixel – *picture element*, taškas.
[7] Raiška – taškų kiekis ilgio vienete, dažn. colyje, žym. **ppi** (ppi – pixels per inch)

- Mažinant vaizdą, keli gretimi taškai pakeičiami vienu, todėl dingsta smulkios piešinio detalės, o didinant – padidėja kiekvieno taško fiziniai matmenys, todėl atkurto vaizdo kontūras būna laiptuotas, dantytas.
- Didinant piešinio matmenis ir nemažinant raiškos atsiranda daugiau vaizdo elementų, todėl piešinio failas smarkiai padidėja.

Populiariausi taškinės grafikos redaktoriai:

- Adobe Photoshop;
- Microsoft Paint;

Populiariausias rastrinės grafikos formatas[8], naudojamas paveikslėlių animacijai Internete: GIF.

Animacijos atsiradimas. Animacijos gabalėlis buvo rastas ant molio taurės, kaip manoma ši taurė padaryta prieš 5000 tūkstančius metų pietryčių Irane. Šį senovinį gabalėlį galima pavadinti pirmąja animacija pasaulyje. Senovinės animacijos herojus tai ožys, kuris prišoka prie medžio ir nusiskina lapą (1.2.3 pav.) [2.8].

1.2.3 pav. Animacija ant molinės taurės

Animacija atsirado ne iš karto. Šiais laikais dauguma žmonių prisideda prie animacijos gamybos, nes kompiuterio pagalba kurti judesius labai paprasta. Kiekvienas žingsnis, modeliuojant judesį turi savo vietą, pirmiausia sukuriami paveiksliukai, vėliau taikoma tam tikra technika kaip tuos paveiksliukus priversti judėti. Yra tam tikri istorijoje etapai, dėl kurių animacija yra tokia kokia yra.

Pirmosios užuominos apie judančius objektus popieriuje atsirado 1824 m. kai Peter Roget karališkajai visuomenei pristatė laikraštį „Judančių objektų vizija“[9]. Nuo tada viskas ir prasidėjo. Filmukai būdavo trumpi, begarsiai ir žinoma nepasižymėjo kokybe. Tokių filmukų principas buvo besikeičiantys kadrai[10].

1964 metais Ken Knowlton, dirbdamas Bell‘o laboratorijose, pradėjo judančių objektų bandymus pasinaudodamas kompiuterio technologijomis [2.7].

Animacijos technologijos. Sparčiai besivystančios animacijos priežastis labai sparčiai besivystančios technologijos, vos ne kas mėnesį priimami vis nauji animacijos kūrimo standartai.

- *Tradiciniai animacijos kūrimo metodai.* Be kompiuterio animaciją galima kurti tik popieriaus lapo ir pieštuko pagalba. Visi tokios animacijos kadrai piešiami ranka. Pažymėtina jog

[8] Formatas – tai taisyklių rinkinys, bylos kodavimo sistemos apibrėžimas.
[9] Vertimas iš anglų kalbos. Originalus pavadinimas: “The persistance of vision with regard to moving objects”
[10] Čia kadras suprantmas kaip popieriaus lapas ar nuotrauka.

kiekvieną animacijos sekundę sudaro 24 kadrai[11], taigi reikia įdėti milžiniškų pastangų norint sukurti, kad ir labai trumpą, kelių minučių animaciją.

- *Kompiuterinė grafika ir animacija.* Kompiuterinė animacija kuriama kompiuterio pagalba. Kompiuteriniams judesiams modeliuoti yra daug skirtingų būdų. Vienas iš jų tai trijų matavimų animacija[12]. Vienas būdas kurti kompiuterinius judesius tai sukurti objektą ir jį perpiešti[13]. Tokia trijų matavimų animacija atrodo labai realistiškai. Kitas būdas, kurti animacijai, tai pasinaudoti standartiniais kompiuteryje esančiais piešimo įrankiais, nupiešti atskirus kadrus ir pasinaudojus tam tikromis programomis juos sujungti į vieną paleidžiamąją bylą (dažniausiai GIF) ar išsaugoti kaip video bylą. Dar kitas būdas kurti animaciją tai pasinaudojimas tam tikrais perėjimais ar kitais specialiais efektais, modifikuojant atskirus paveikslėlius ar video.

Vektorinės grafikos ir animacijos technologijos Internete. Duomenų perdavimo greičio apribojimai esant paprastam prisijungimui ne visada leidžia patogiai peržiūrinėti svetaines su statine grafika, nekalbant jau apie prisotintus dinamika Interneto puslapius. Šios problemos sprendimu prieš keletą metų užsiėmė eilė kompanijų, pradėjusių kurti naujus turinio pristatymo būdus Internetui. Ir nežinia kaip toliau būtų susiklostęs nedidelės Džonatano Gajaus programos likimas, jei nebūtų į ją dėmesio atkreipusi Macromedia. Šis pirkinys kardinaliai pakeitė Pasaulinio tinklo veidą, tapęs patrauklių, dinamiškų puslapių kūrimo instrumentu su garso ir vaizdo panaudojimu. Pirminis vektorinės animacijos redaktoriaus pavadinimas (Future Splash Animator) davė naujai technologijai vardą – Flash [3.1 p. 1; 2.12].

Be Flash dabar aktyviai vystosi ir kitos technologijos, pagrįstos vektorine objekto išraiška. Tai Adobe SVG (Scalable Vector Graphics) technologijos.

Palyginus neseniai pradėjo atsirasti įvairios technologijos trijų matavimų objektų modelių perdavimui su tolimesniu jų atvaizdavimu naršyklės lange (taip pat vektoriniu būdu). Pažymėtinas kompanijos Viewpoint - Metastream produktas.

Visų vektorinių technologijų pagrindas yra vektorinė objektų išraiška, plačios galimybės darbui su multimedija ir interaktyvumas, duodantis šioms technologijoms neginčytinus pranašumus prieš tradicinius informacijos pateikimo būdus Internete.

Vektorių panaudojimas vietoj rastrinio vaizdo duoda žymiai mažesnius bylų dydžius[14], o tai leidžia parsisiųsti bylas, turint netgi labai lėtą Internetą. Be to, failo dydis nepriklauso nuo naudojamų jame elementų geometrinių dydžių. Technologijos specifika ir tame, kad didelis vienodų

[11] Frames – kadrai. 24 kadrai per sekunde rekomenduojamas filmo kadrų keitimo dažnis
[12] Žinoma kaip 3D (3 dimensions = trys matavimai)
[13] Render – objektas sutvarkomas, perpiešiamas, uždedami šešėliai, tekstūros
[14] 50KB dydžio byloje gali tilpti gan spalvingas, ilgas ir informatyvus klipas

elementų kiekis nesąlygoja bylos padidėjimo[15]. Svarbus bruožas - vektoriškumas, - klipo mastelis keičiamas be kokybės praradimo.

Savaime suprantama, greta paprastų vektorinių objektų, Flash technologijoje galima naudoti rastrinius atvaizdus. Palaikomi rastriniai atvaizdai:

- JPEG;
- GIF;
- PNG su nevienalyčiu permatomumu;

Yra galimybė panaudoti Multimediją, kuri leidžia prijungti prie animacijos garsą (wav, mp3) ir video (quick time mov, AVI).

ActionScript palaikymas. Šios programavimo kalbos pagalba galima kurti interaktyvią animaciją su gan sudėtingais scenarijais.

Metastream pranašumas - vaizdo realistiškumas. Yra galimybė apžiūrinėti modelį iš visų pusių, taip pat iš vidaus.

Pagrindinių vektorinės grafikos panaudojimo sferų Internete yra keletas:

- Interaktyvių sąsajų kūrimas su įvairiais vizualiais ir garso efektais (mygtukai, išplaukiantys meniu-sąrašai, objektai, keičiantys savo savybes užėjus ant jų su pelyte);
- Pilnaverčių animacinių klipų kūrimas su dalyvaujančiais animaciniais personažais;
- Žaidimai on-line;

Kas liečia trimačio turinio pristatymo technologijas, taip pat išskiriamos kelios panaudojimo sferos:

- Automobilių pristatymas (Ford, Subaru, Volvo);
- Sportinių prekių pristatymas;
- Reklamos agentūros;
- Patalpų dizainas;
- Vietovės žemėlapiai su tūriniu landšafto[16] perteikimu

Vektorinės animacijos panaudojimo platumas ir apsprendė mano pasirinkimą realizuoti modelius būtent šia technologija.

Pagrindinis vaidmuo Flash technologijoje tenka animacijai. Egzistuoja keletas jos sukūrimo būdų:

- Kadrų sekos (kiekvieną kartą judančio objekto vaizdas piešiamas iš naujo);
- Pagal bazinius kadrus - (programa paprasčiausiai sukuria tolygų perėjimą (Tween) iš vieno bazinio kadro į kitą);

Vektorinės grafikos redaktoriai pasižymi:

[15] Panaudotas vienas objektas-pagrindas, likusieji pateikti kaip jo kopijos, kurioms galima taikyti įvairias transformacijas

[16] Landšaftas – žemės paviršius

- Spalvinių efektų gausa;
- Įvairių formų transformacija;
- Skaidrumo kaukių palaikymu;
- Objektų netolyginiu judėjimu;
- Judėjimu užduota trajektorija;

Bazinių kadrų animacija gali būti vykdoma įvairiai. Viena, kai ekranu tiesiog juda paprastas objektas, visai kas kita - personažų animacija. Čia judėjimas pagrįstas „kaulų" panaudojimu (bones), plačiai vartojamų klasikinėje animacijoje. Labai svarbi objektų hierarchija ir kontrolinių taškų išdėstymo teisingumas.

3) Animacijos modeliavimo aplinkų analizė

Macromedia Flash. Macromedia susitelkusi išskirtinai sprendimams Internetui, tapo viena iš įtakingiausių šio rinkos segmento veikėjų, pagal finansinius rodiklius nusileisdama tik pačiai Adobe.

Pradedant nuo ketvirtos, kai į redaktorių buvo įdiegtas ActionScript palaikymas, žymiai išplėtęs standartines galimybes, susidomėjimas šiuo redaktoriumi stipriai padidėjo. Dabar prieinama jau septinta versija – Flash MX 2004. Ji siūlo: objektų kūrimo priemones, specifinių savybių jiems suteikimo ir sudėtingos animacijos realizavimo patogumą.

Objektų kūrimas. Flash sąsaja išlaikyta visų Macromedia produktų stiliuje, kas duoda privalumą jų vartotojams. Programos galimybės traukia prie jos kaip ir specializuotų vektorinių programų šalininkus, taip ir paprastus dailininkus. Redaktoriuje yra kreivių redagavimo priemonės segmentų ir taškų lygyje. Taip pat patogus darbas su originaliu *Brush tool* instrumentu, turinčiu analogą rastriniuose redaktoriuose: piešiant vieną objektą virš kito programa automatiškai nukerpa apatinį, vienspalviai objektai sujungiami į vieną.

Galima išskirti kelis redaktorius skirtus trijų matavimų modeliams kurti su galimybe konvertuoti bylą į SWF formatą:

- Ulead EnVector;
- Vecta 3D;
- Swift3D;

Platus grafinės sąsajos kūrimo instrumentų pasirinkimas:

- Interaktyvūs mygtukai;
- Išplaukiantys meniu;
- Įvairios slinkčių juostos;
- Judantys elementai;

- Garso panaudojimas;

Tai standartas de-facto - Macromedia Flash. Šiandienai su jos potencialu sudėtingų interaktyvių klipų kūrimo srityje nei viena kita programa susilyginti negali.

Animacija. Flash animacija pagrįsta bazinių kadrų panaudojimu. Palaikomi abu tipai - motion tween[17] ir shape tween[18].

Yra galimybė naudotis biblioteka (Library) - visus objektus joje galima apjungti į grupes tam, kad bet kuriuo momentu būtu galima juo pasinaudoti. Kuriant klipą labai naudinga vieno objekto pakeitimo kitu operacija, išsaugant visą anksčiau jam priskirtą animaciją. Objektai bibliotekoje gali turėti įdėtą animaciją, kas leidžia išvengti pernelyg didelio klipo pagrindinės panelės *Timeline* užgriozdinimo.

Palyginus su analogiškais redaktoriais, gautų klipų dydis visų mažiausias - yra prasmė išsaugoti papildomai Macromedia Flash SWF failą.

Scenarijų kalba. Vienas iš pagrindinių Flash technologijos privalumų - interaktyvumas. O kadangi dabar be interaktyvumo - niekur, tai redaktorius turi žymiai išplėstą paskutinėje versijoje scenarijų kalbą *ActionScript.* Naudojant scenarijų kalbą ActionScript paprasčiau kurti interaktyvius žaidimus, apklausos formas ir netgi pokalbių sistemas, veikiančias realaus laiko režimu.

Trūkumai. Kaip ir bet kurioje programoje, Flash turi trūkumų. Kartais objektai atsitiktinai pasislenka per vieną tašką[19] - čia kalta apvalinimo klaida (atvaizduojant ekrane vektoriniai objektai pasikeičia į bitinius vaizdus).

Kitas trūkumas, kuriant naują objektą automatiškai nesukuriamas naujas sluoksnis (kai kuriuose redaktoriuose sluoksnio sukūrimas vyksta automatiškai).

Corel R.A.V.E. Kompanija Corel tikriausiai norėdama grafinį paketą padaryti kuo universalesnį, papildė jo dešimtą versiją dar vienu redaktoriumi - R.A.V.E. (Real Animated Vector Effects - „tikri animuoti vektoriniai efektai"). Šiame redaktoriuje yra galimybė kurti Flash klipus be papildomos programinės įrangos įsigijimo.

Privalumai. R.A.V.E. buvo kuriamas populiaraus grafinio redaktoriaus pagrindu – mokant dirbti grafiniu redaktoriumi CorelDRAW, nesunku bus dirbt ir Corel R.A.V.E. Redaktoriuje R.A.V.E yra visos tos pačios redagavimo galimybės kaip ir DRAW.

Animacijos kūrimas praktiškai nesiskiria nuo Flash, tačiau R.A.V.E. turi vieną privalumą: kuriant objektą, sluoksnis atsiranda automatiškai.

Trūkumai. Pagrindinių trūkumų programoje du, bet jie esminiai.

- R.A.V.E. nėra panelės, kurioje atsispindėtų objekto ir jo animacijos savybės nustatytame kadre (atvirkščiai nei Flash).

[17] Motion tween – kai objekto forma nekinta.
[18] Shape tween – objekto forma kinta
[19] Pixel – taškas, vaizdo vienetas

- Mažas pasirinkimas interaktyvios aplinkos kūrime. R.A.V.E. leidžia tik sukurti interaktyvius mygtukus (jie gali turėti tris padėtis - Normal, Over, Down[20]), dirba su garsu ir palaiko nuorodas.

Adobe LiveMotion2. Adobe produkte LiveMotion pateiktas pilnas priemonių rinkinys vektorinių objektų kūrimui. LiveMotion palaiko Photoshop filtrus.

Skirtumai tarp Flash ir LiveMotion. Pagrindiniai skirtumai animacijoje. Skiriasi Timeline panelės. LiveMotion Laiko juostoje (Timeline) skirtingai nei Flash pateikti tik baziniai kadrai. Tai didžiulis trūkumas, nes pagal jį nepatogu vizualiai sinchronizuoti įvykius. LiveMotion laiko juosta patogi, jei reikia sužinoti, kas vyksta su objektu bet kuriuo laiko momentu[21], bet dėl to nukenčia sudėtingesnės animacijos valdymo patogumui. Filtrų, išrūšiuojančių transformacijas pagal kokį nors požymį, nebuvimas (kaip 3D paketuose), sąlygoja pernelyg didelį ekraninės vietos užgriozdinimą.

Privalumai:

- Naujoje LiveMotion versijoje pagerintos vartotojiškų scenarijų kūrimo ir derinimo funkcijos, kas pagerina on-line projektų kūrimą.
- Neturintiems pakankamos patirties programuojant JavaScript arba ActionScript Adobe pasiūlytas instrumentas LiveTabs, kurio pagalba patogia forma, neliečiant betarpiškai kodo, galima kurti savus scenarijus.
- Yra glaudi integracija tarp kitų Adobe produktų. Photoshop sluoksnių palaikymas, kurie po konvertavimo tampa paprastais LiveMotion objektais.
- Palaiko video QuickTime formatą.
- Sukuriant objektą, automatiškai jam generuojamas sluoksnis;

Trūkumai:

- Nėra objektų transformavimo galimybės (tam siūloma naudoti Illustrator).
- Neaiški Adobe pozicija SVG palaikymo klausimu - būdama už jį, vadovybė, nepaisant to, neįtraukė jo palaikymo į redaktorių.
- Sudėtinga animacijos proceso realizacija. Vietoj to, kad paprastai vykdyti kokius tai veiksmus su objektu, tenka pradžioje nurodyti operacijų charakterį, pažymint atitinkamą lauką scenarijaus panelėje.
- Sunku dirbti su klipu turinčiu didelį objektų skaičių.

Kiti redaktoriai. Be tokių redaktorių kaip Corel R.A.V.E. Macromedia Flash ar Adobe LiveMotion žinomi ir kiti redaktoriai kaip:

- *DJJHoldingsSwish 2;*
- *Insane 3D Flash Animator;*

[20] Normal, Over, Down – atitinkamai normali, užvedus pelyte, paspaudus
[21] Visos transformacijos šioje paletėje atvaizduojamos kaip atskiri sluoksniai

DJJHoldingsSwish 2. Tai viena iš pirmųjų redaktorių, leidžiančių panaudoti tekstui dinaminius efektus. Antroje versijoje Swish tapo galingu redaktoriumi sudėtingų klipų kūrimui, daugeliu atžvilgių susilyginusi su Macromedia produktu, o pagal kai kuriuos parametrus netgi aplenkusi.

Privalumai:

- Palaiko skaidrumą;
- Galimybė objektams suteikti spalvų pereinamumą;
- Interaktyvių mygtukų kūrimas;
- Pridėtinę animaciją (sprite)[22];
- Netolygus judėjimas;
- Galimybė dirbti su keliais scenarijais vienu metu;
- Garso ir video palaikymas;
- Transformacijų charakteris ir jų trukmė aiškiai rodoma scenarijaus panelėje, kas labai patogu analizuojant animaciją.

Trūkumai:

- Sluoksnių valdymas (negalima užblokuoti tam tikrą sluoksnį nuo pakeitimų).
- Efektų kiekis naujoje versijoje nepadidėjo ir dirba jie tik su tekstu.

Interaktyviosios Swish galimybės sutelktos užsklandoje Actions. Programoje galima nustatyti reakciją į standartinius įvykius, tokius kaip pelytės judėjimą. Action rinkinys panašus kaip ir Macromedia Flash redaktoriuje.

Insane 3D Flash Animator. Redaktoriuje nemažas pasirinkimas intrumentų sudėtingų klipų gamybai.

Privalumai:

- Palaiko vartotojiškus scenarijus;
- Garsas;
- Mygtukų kūrimo galimybė;
- Didelis pasirinkimas efektų;
- Didžiulė savybių panelė, kurioje surinkti visi būtini valdymo elementai.

Trūkumai:

- Trumpa laiko juosta (Timeline);
- Objektų elgesio ir savybių valdymas – tik per dialogo langus;
- Visos operacijos realizuotos mygtukų pavidalu, kurių išorė turi mažai ką bendro su vykdomais veiksmais.

[22] Sprite – tas pats kaip Macromedia Flash redaktoriuje Symbol

Naujoje versijoje galimybių sąrašas pasipildė dar ir korektišku darbu su trimačiais modeliais. Importuotas modelis gali būti modifikuotas, po ko animuojasi kaip ir įprastas plokščias objektas. Eksportuojant jis virsta savo dvimate SWF kopija.

4) Apibendrinimas

Kompiuterinės animacijos modeliams kurti naudojamos tam pritaikytos aplinkos, kurių yra daugybė ir įvairių. Kompiuterinės animacijos modeliai nedaug skiriasi nuo realių mechanizmų ar gyvų organizmų judesių. Šiandien, kompiuterinė grafika yra svarbi sritis kompiuterių mokslo pramonėje, ji plečiasi ir yra prieinama plačiai visuomenei. Pramonės profesionalai ir išradėjai praleido ištisus metus kurdami dabartinę kompiuterinės grafikos situaciją.

Daugiaspalvis, dinamiškas ir interaktyvus - būtent tokiais žodžiais dažnai apibūdinamas šiuolaikinis Internetas. Pateikiamos informacijos forma yra tokia pat svarbi kaip ir turinys. Šiuo metu įspūdingiausių ir gražiausių tinklapių kūrimo įrankis yra Macromedia Flash technologija. Flash dėka Internetas tapo gyvesnis. Labai svarbu, kad tai gali pajusti ir silpnesnių kompiuterių savininkai. Vienas didžiausių Flash privalumų – mažas sukurtų bylų dydis. Vektorinės grafikos dėka Interneto kūrėjų fantazija ir kūrybingumas jau gali pasireikšti ir trimatėje erdvėje.

Animacijos projektams realizuoti pasirinkau Macromedia Flash redaktorių. Elementų valdymui naudoju objektiškai orientuotą scenarijų kalbą ActionScript.

III. Judesio objektiškai orientuotas programavimas

1) Programavimo kalbos pasirinkimas

Praktiškai visomis programavimo kalbomis galima programuoti judesį. Tačiau dauguma jų nėra visiškai tam pritaikytos. Judesio programavimui galima naudoti šias programavimo kalbas:

- C++;
- Java;
- ActionScript;
- Logo;
- Turbo Pascal;
- Ir kt.

Internetinei animacijai (judesiui) programuoti tinka objektiškai orientuota Java programavimo kalba ir paskutiniu metu vis labiau populiarėjanti objektiškai orientuota ActionScript programavimo kalba.

2) Judesio objektiškai orientuoto programavimo metodologija

Objektiškai orientuotas ActionScript. Tradicinis programavimas susideda iš įvairių taisyklių, taisyklės grupuojamos ir taip susidaro procedūros. Procedūra vykdo tam tikrą uždavinį neturėdama jokių žinių apie didesnę programą. Pavyzdžiui, procedūra gali apskaičiuoti ir gražinti rezultatą. Procedūrinio stiliaus „Flash“ programoje (projekte), pasikartojamiems veiksmams yra rašomos funkcijos, o tam tikri duomenys priskiriami kintamiesiems. Programos eigoje vykdomos funkcijos ir keičiamos kintamųjų reikšmės, tipiškas duomenų apdorojimas – įvestų duomenų apdorojimas ir išvedamų duomenų generavimas. Procedūrinis programavimas praktiškai būtinas programose; Bet kokiu atveju kai programos pasidaro didesnės ar sudėtingesnės ir susidaro didelės iteracijos tarp procedūrų, tada procedūrinis programavimas gali tapti griozdiškas. Jas sunku palaikyti, sunku redaguoti, sunku kažką keisti.

Objektiškai orientuotas programavimas - tai visiškai kitoks požiūris į programavimą, numatantis projekto vystymą, palaikymą su procedūriniu programavimu. OOP yra skirtas kurti sudėtingas programas – projektas lengviau valdomas skaidant į savarankiškus, tarpusavyje sąveikaujančius modulius. OOP leidžia perkelti abstrakčias idėjas ir tikrus realaus pasaulio daiktus į atitinkamas programos dalis – objektus.

Objektiškai orientuoto projektavimo sąvoka. Objektas yra savarankiškas modulis susidedantis iš tarpusavyje glaudžiai susijusių funkcijų (vadinamų metodais) ir kintamųjų

(suprantamų kaip savybės). Individualūs objektai sukuriami iš klasių, kurios priskiria pradines reikšmes objekto metodams ir savybėms. Klasė yra kaip šablonas iš kurios sukuriamas objektas. Klasės gali atstovauti tik teorinei sąvokai, kaip laikmatis ar fizinė būtis, kaip iššokantis meniu ar kosminis laivas. Viena klasė gali generuoti bet kokį skaičių objektų su ta pačia pagrindine struktūra (tas pats receptas tinka kepti bet kokiam skaičiui bandelių). Pavyzdžiui, sukurtas kosminių karų žaidimas taikant OOP, čia gali būti 20 skirtingų, individualių *KosminisLaivas* objektų ekrane vienu metu ir visi sukurti iš vienos vienintelės klasės *KosminisLaivas*. Panašu, kad žaidimas gali turėti vieną *2dVector* klasę, aprašančią matematinius vektorius tūkstančiams objektų žaidime.

Terminas *egzempliorius (instance)* dažnai naudojamas kaip *objekto (object)* sinonimas. Pavyzdžiui, frazė „Sukurti naują *KosminisLaivas* egzempliorių“ ir „Sukurti naują *KosminisLaivas* objektą“ reiškia tą patį. Naujo objekto kūrimas iš klasės kartais dar vadinamas *instantiation,* kuris tiesiog ir reiškia veiksmą - objekto sukūrimą.

Kad sukurti objektiškai orientuotą judesį reikia:

- Sukurti vieną ar daugiau klasių.
- Sukurti (t.y. instantiate) objektus iš klasių.
- Nurodyti, ką objektai turi daryti.

Kam objektai skirti, t.y. kokius veiksmus jie atlieka ir lemia programos funkcionavimą.

Be to, kai naudojamos sukurtos klasės, programa gali pasiimti bet kurią iš sukurtų klasių į Flash grotuvą. Pavyzdžiui, programa gali naudoti standartinę *Sound* klasę, naujiems *Sound* objektams kurti. Atskiras *Sound* objektas atstovauja ir kontroliuoja vieną ar grupę garsų. *setVolume()* metodas gali padidinti arba sumažinti garso stiprumą. *loadSound()* metodas gali išrinkti ir groti MP3 tipo bylas. Savybė *duration* milisekundžių tikslumu nusako apie užkrauto garso trukmę. Kartu standartinės ir nestandartinės klasės iš visų OOP programų formuoja pagrindinius blokus *Flash* aplinkoje.

Klasės sintaksė. Pavyzdys, iš kurio bus aišku kokia klasės sintaksė *ActionScript* kalboje. Jau buvo minėta, kad kosminių kovų žaidimas turi *KosminisLaivas* klasę. ActionScript kalboje klasė aprašomas:

```
Class SpaceShip{
        //aprašomas viešas (public) kintamasis speed
        public var speed:Number;

//aprašomas privatus (private) kintamasis damage
        private var damage:Number;

    //aprašomas klasės konstruktorius, kuris priskiria pradines reikšmes kiekvienam KosminisLaivas
    egzemplioriui (instance).
    public function SpaceShip (){
```

```
                speed = 100;
                damage = 0;
        }
        // aprašomas viešas metodas fireMissile().
        public function fireMissile():Void{
                // rašomas, kodas susijęs su kosminio laivo kulkomis
        }
        // aprašomas viešas metodas thrust().
        public function thrust():Void{
                // rašomas kodas susijęs su laivo judėjimu
        }
}
```

Pavyzdyje matyti, kaip klasė ***KosminisLaivas*** apjungia tarpusavyje susijusius aspektus į vieną grupę (tai būdinga visoms klasėms). Kintamieji (savybės), kaip ***speed*** ar ***damage***, susiję su kosminiais laivais yra grupuojami kartu su funkcijomis (metodais), visa tai skirta laivo judėjimui ir atakai. Visi kiti programos (žaidimo) aspektai, kaip surinkti taškai ir aplinkinės grafikos palaikymas, aprašomi atskirose klasėse.

Objektų sukūrimas

Objektai kuriami (*instantiated*) su operatoriumi ***new***, kaip pavyzdžiui:

```
new ClassName()
```

kur ***ClassName*** yra klasės vardas iš kurios ir sukuriamas pats objektas.

Pavyzdžiui, norėdami sukurti naują ***SpaceShip*** objektą jau minėtame kosminių karų žaidime, rašomas toks programos kodas:

```
new SpaceShip();
```

Objektų kūrimo sintaksė (t.y. new SpaceShip()) yra tokia pati tiek vidinėje programavimo kalboje ***ActionScript 2.0*** tiek ir ***ActionScript*** 1.0. Kaip bebūtų, klasių aprašymo sintaksė ***ActionScript 2.0*** skirtinga nei ***ActionScript 1.0*** kalboje.

Daugumas objektų yra sukurti ir kažkur laikomi, taigi ir panaudojami jie gali būti vėliau programoje (žaidime). Pavyzdžiui, galima ***KosminisLaivas*** egzempliorių priskirti kokiam nors kintamajam ***laivas***, kaip:

```
var laivas:KosminisLaivas = new KasminisLaivas();
```

Kiekvienas objektas yra atskiras duomenų tipas, jis gali būti priskiriamas kintamajam, masyvo elementui ar kito objekto savybei (property). Pavyzdžiui, jeigu sukuriama 20 priešo laivų, automatiškai bus sukurtas 20 KosminisLaivas objektų masyvas. Tai leidžia lengvai manipuliuoti vienodais objektais einant nuo vieno masyvo elemento prie kito, ir sakoma kad kiekvienas iš tų dvidešimties objektų yra apibrėžtas ***KosminisLaivas*** klasėje.

Objektų naudojimas. Metodai objektui suteikia tam tikrus sugebėjimus (t.y. elgesį) – kaip „ugnies ataka“, „judėjimą“. Objekto savybėse saugomi jo duomenys, kurie apibūdina objekto padėtį

duotu laiko momentu. Pavyzdžiui, tam tikrame žaidimo taške, *laivas* (kintamasis kuriam priskirtas egzempliorius (object) KosminisLaivas) gali turėti *speed* reikšmę 300, o *demage* – 14.

Metodai ir savybės objekto klasėje apibrėžti kaip vieši (public) gali būti prieinami, bet kur programoje. Priešingai, jei metodai ir savybės apibrėžti kaip privatūs (private) gali būti naudojami tik toje klasėje ar poklasyje. Geriausia jei metodai ir savybės apibrėžti kaip vieši (public) jei jie būtinai turi būti prieinami iš išorės.

Kad iškviesti metodą, naudojamas taško operatorius ir funkcijos iškvietimo operatorius. Pavyzdžiui:

```
// Iškviečiamas ship objekto fireMissile() metodas
ship.fireMissile();
```

Kad pakeisti savybes, taip pat naudojamas taško operatorius. Pavyzdžiui:

```
// nustato laivo greitį (speed) – 120
ship.speed = 120;
```

Kad susigražinti savybės reikšmę, vėl naudojami tą patį taško operatorių. Pavyzdžiui:

```
// gražina savybės speed reikšmę išvesties lange
trace(ship.speed);
```

Inkapsuliacija. Tai procesas jungimo savybių (duomenų) ir elgesio (operavimo metodų) į vieną visumą, vadinamą objektu.

- Kad valdyti mašiną, jai duodamos komandos, siunčiant jai pranešimus (*messages*). Šie pranešimai generuojami, manipuliuojant mašinos valdymo prietaisais, tada leidžiama pačiai mašinai (objektui) juos interpretuoti ir atlikti reikiamus veiksmus. Iš tikrųjų nesirūpinama, kaip mašina gali atsakyti į pranešimus ar kaip ji keičia savo charakteristikas, t.y. galima naudoti objektą, nežinant kaip jis dirba, jei tik žinoma, kaip pasiųsti reikiamus pradinius duomenis, kad atlikti norimą veiksmą. OOP ši koncepcija vadinama **duomenų abstrakcija** (*data abstraction*) [2.2].

Duomenų tipai. Klasė objektiškai orientuotoje programoje gali apibrėžti vienos rūšies duomenis, kurie oficialiai reiškia duomenų tipus programoje.

Klasė efektyviai apibrėžia įprastus duomenų tipus.

Daugumas jei yra nors kiek dirbęs su vidine programavimo kalba *ActionScript* yra susipažinęs su klasėmis, pavyzdžiui *Date* class. Kai kuriamas *Date* objektas naudojant *new Date()*, gražinama reikšmė ne tekstinė eilutė ar skaičius, bet tam tikras kompleksas duomenų tipų, kuris apibrėžia tam tikrą dieną ar tam tikrus metus. *Date* duomenų tipas palaiko įvairias savybes ir metodus susijusias su datomis.

Duomenų tipai nustato ribas kur kintamasis gali būti talpinamas, naudojamas kaip parametras ar perduodamas kaip gražinama reikšmė. Pavyzdžiui, kaip anksčiau apibrėžta ***speed*** savybė, jo tipas buvo ***Number***:

```
// išraiška „:Number“ apibrėžia kintamajam speed tipą.
public var speed:Number;
```

Stengdamiesi išsaugoti kintamajam ***speed*** ne skaičių, o kitą reikšmę kompiliatorius generuos klaidą.

Jei testuojant filmuką Flash‘o išvesties langas rodys frazę „Type mismatch“, reiškia jog kažkur programoje nurodytas neteisingas duomenų tipas (kompiliatorius parodys tiksliai kurioje vietoje). Duomenų tipai padeda garantuoti kad programa atliktų būtent tai kam ji yra skirta. Pavyzdžiui, apibrėžiant ***speed*** duomenų tipą kaip skaičių, užkertamas kelias nenumatytiems atvejams, kaip eilutės „laibai greitas“ priskyrimas. Žemiau pateiktas kodas generuoja kompiliavimo klaidą, nes nesutampa duomenų tipas:

```
public  var speed:Number = “labai greitai”;
```

Paveldėjimas. Kuriant objektiškai orientuotą programą, galima naudoti tokį dalyką kaip paveldėjimą, kuris leidžia vienai klasei pasiskolinti kitos klasės metodus ar savybes, kad apibrėžti kitus. Naudojant paveldėjimą galima formuoti programos hierarchiją, taigi daugumoje klasių gali pasikartoti kai kurios tos pačios ypatybės vienos kokios nors klasės. Pavyzdžiui, tokiose klasėse kaip ***Masina***, ***Valtis*** ar ***Lektuvas*** gali kartotis kai kurie bruožai pagrindinės klasės TransportoPriemones, tokiu būdu sumažinamas dubliavimasis programoje. Kuo mažiau dubliavimųsi tuo mažiau programos kodo ir testavimo. Be to tai leidžia nesunkiai redaguoti programos kodą – pavyzdžiui, atnaujinus judėjimo algoritmą vienoje klasėje išvengsime klaidų arba jų bus daug mažiau, negu keičiant visą programos kodą kiekvienoje klasėje atskirai.

Klasė, kuri paveldi metodus ir savybes iš kitos klasės vadinama poklasiu (subclass). O klasė iš kurios poklasis paveldi savybes ir metodus vadinama bazine klase (superclass). Savaime suprantama, kad poklasyje gali apibrėžti tik jos savybės ir metodai, priedo tai ką paveldi iš bazinės klasės. Viena bazinė klasė gali turėti daugiau nei vieną poklasį, bet vienas poklasis gali turėti tik vieną bazinę klasę.

Paketai. Didelėse programose, galima kurti tam tikrus paketus, kurie susideda iš klasių grupių. Paketai leidžia jungti klases į logines grupes ir užkerta kelią vardų konfliktui tarp klasių. Ypač tai naudinga kai įsipainioja komponentai ir trečios šalies klasių bibliotekos. Pavyzdžiui, Flash MX 2004 GUI[23] komponentai, vienas iš jų pavadinimu ***Button,*** priklauso ***mx.controls*** paketui. GUI

[23] GUI (graphic user interface) – grafinė vartotojo sąsaja

komponentų klasė *Button* gali painiotis su Flash'o standartine *Button* klase jei nebūtų nurodyta, kad mygtukas yra iš *mx.controls* paketo. Fiziškai paketai tai katalogai kuriuose saugomos klasių bylos[24].

Programavimo kalba Java. „Java" sukurta „C++" programavimo kalbos pagrindu. 1991 m. „Sun Microsystems" supaprastino „C++" kalbos sintaksę, patobulino stabilumą ir saugumą, išplėtė jos suderinamumą su įvairiomis operacinėmis sistemomis. „Java" programavimo kalba itin funkcionali, be to, atitinka saugumo reikalavimus, reikšmingus elektroniniam verslui.

Java taikymas:

- Java apletai[25]. Apletai puikiai tinka pristatyti nuotraukų galeriją, sukurti judrų firmos logotipą, bėgančią tekstinę eilutę ir pan.
- Java žaidimai. Panaudojant šią kalbą, sukurta ir virtualiųjų šachmatų turnyrų sistema www.chess.lt, leidžianti per Internetą realiuoju laiku žaisti šachmatais su kitame mieste ar šalyje prie Interneto prisijungusiais žaidėjais.
- Pokalbiai Internete. Populiarūs pokalbių kambariai www.chat.lt.
- Virtualūs modeliai Internete, pvz., ja sukurta virtuali patalpų projektavimo sistema, Vilniaus miesto žemėlapis.

Java – ne tik programavimo kalba, bet ir standartizuota programų vykdymo terpė, sugebanti jas vykdyti įvairiose operacinėse sistemose. Java kaip programų vykdymo terpė naudojama:

- Kompiuteriuose;
- Mobiliuosiuose telefonuose;
- Buitinėje technikoje;
- Intelektinėse kortelėse;

Java objektinio programavimo privalumai. Klasės yra Java objektinio programavimo pagrindas. Kiekviena klasė apibrėžia naują sukurtą tipą, kuriame yra naudojami kintamieji ir apibrėžiami metodai. Klasė apibrėžia tipą, o to tipo (klasės) kintamieji vadinami objektais. Objektai skiriasi nuo įprastų kintamųjų tuo, kad juose apibrėžiami metodai aprašantys kintamųjų keitimo taisykles.

Duomenų apsauga. ***Private*** modifikatorius apsaugoja kintamąjį nuo kitų klasių. Jei kintamasis yra *private*, jį gali pakeisti tik tos pačios klasės metodai. Jei kintamasis paskelbtas *global*[26] yra tikėtina, kad jis bus šaltinis sunkiai aptinkamos programos vykdymo klaidos, nes sunku atsekti kuriuo momentu ir kas pakeitė *global* tipo kintamojo reikšmę netinkamu būdu. Tokią klaidą dar sunkiau aptikti, jei programinis produktas kuriamas ne vieno programuotojo.

[24] Klasių bylos – tai bylos su prievardžiu *.as
[25] Aplet (apletas) – tam tikros formos, tam tikra forma naudojama judantiems piešiniams Internete kurti.
[26] Global – globalus kintamasis

Geriausia kintamąjį talpinti į klasę ir paskelbti jį *private*[27] ir toje pačioje klasėje aprašyti metodus, kurie gali naudotis šiuo kintamuoju.

Duomenų apgauba. Duomenų apsaugojimo mechanizmas remiasi hierarchija tarp kintamųjų panaudojimo. Keičiant apsaugotus kintamuosius kreipiamasi į specialiai tuo tikslu parašytą klasės metodą. Tokia kintamųjų apsauga vadinama duomenų apgauba (angl. encapsulation).

Pakartotinis panaudojamumas taikant paveldėjimą. Kintamųjų ir metodų apgaubą (encapsulation) galima kartoti kiek norima kartų. Tačiau kartais susiduriama su panašiomis užduotimis, programų kodai panašūs. Objektinio programavimo atveju naudojamas paveldimumas (inheritance), kuris ir praverčia siekiant efektyviai kurti programas skirtas panašiems uždaviniams spręsti.

Programų lankstumas (maintainability). Objektinis programavimas suteikia papildomas galimybes pasinaudoti anksčiau parašytomis programomis (Java atveju - klasėmis). Kompiuteriai ir jų architektūra labai greitai keičiasi. Vieną kartą padarytas kompiuteris yra nelankstus prisitaikymo prie naujų sąlygų prasme. Programinė įranga turi būti parašyta taip ir su tokiomis priemonėmis, kad ją nesunkiai būtų galima perkelti prie besikeičiančių kompiuterių.

Taip pat ir gretimoms probleminėms sritims rašomas programas turėtų būti nesunku modifikuoti. Pavyzdžiui algalapiams ir Sodros dokumentams rašomos programos galėtų pasinaudoti viena kitos metodais. Kartais iš anksto sunku numatyti kuria kryptimi vyks pokyčiai.

Visos išvardintos objektinio programavimo savybės vienaip ar kitaip naudingos programų lankstumui padidinti. Tiesiogiai tas pasakytina apie paveldimumo savybę. Ši savybė leidžia programas auginti panašiai kaip sniego rutulį: pradedamas nuo mažo, suridenamas didesnis, dar didesnis ir taip toliau. Kintamųjų apsaugos mechanizmas daro kodą labiau suprantamą; skaitydamas programos kodą kitas programuotojas supras, kad *private* kintamieji skirti ne jam ir kad turi koncentruoti dėmesį į atvirus kintamuosius. Metodų ir kintamųjų apgauba taip pat skaidrina programos suprantamumą. Apgauba iš esmės supaprastina programos kodo sudėtingumą - trumpos programos lengviau skaitomos.

Java klasių hierarchija. Klasių hierarchijos sąvoka svarbi kalbant apie paveldimumą. tarkime yra trys klasės: Mom, Son ir Daughter. Son ir Daughter klasės paveldi Mom klasės kintamuosius ir metodus. Kodas atrodys maždaug taip:

```
class Mom {
//declarations, definitions
}

class Son extends Mom (
```

[27] Private – privatus kintamasis

```
//declarations, definitions
}

class Daughter extends Mom {
//declarations, definitions
}
```

Sukurta klasių hierarchiją. Panašiai viskas vyksta ir realiame pasaulyje, tik čia paveldima iš tėvo ir motinos.

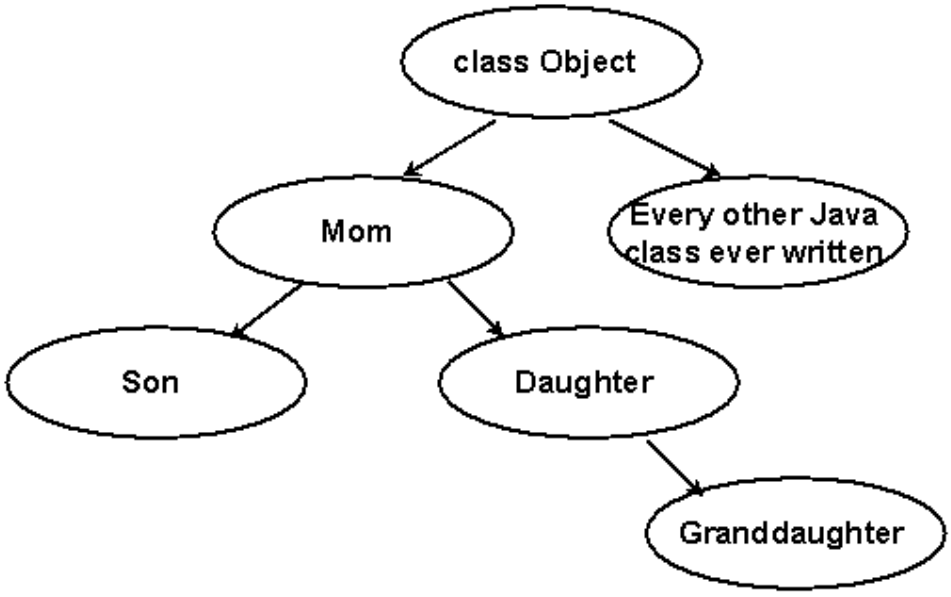
3.3.1 lentelėje išvardyti terminai, naudojami hierarchijai apibūdinti. *Mom* yra pagrindinė klasė, į kuria remiasi kitos dvi klasės. *Son* ir *Daughter* yra *Mom* klasės poklasės (subclasses) ir *Mom* yra *Son* ir *Daughter*. superklasė (superclass).

Terminas	Apibrėžimas
Klasių hierarchija	Grupė klasių apjungta paveldimumo saitais
Superklasė	Klasė, kurią praplečia kokia nors kita klasė.
Poklasė	Klasė, kuri praplečia kitą klasę
Pagrindinė (bazinė) klasė	Kokios nors klasių hierarchijos viršūnėje (neturinti sau superklasių) esanti superklasė

3.2.1 lentelė. Klasių hierarchijos terminija.

Java kalboje kiekviena klasė turi tik vieną tiesioginę *superklasę,* tai apibūdinama vienapaveldimumu. Klasė gali turėti daug *superklasių,* tačiau tarp jų tik viena bus tiesioginė *superklasė*.

Visų klasių hierarchijos viršūnėje yra Object klasė (3.2.3 pav.) ir jei nenurodytas paveldimumas, tai nutylimasis paveldimumas yra extends Object.



3.2.2 pav. Sukurta klasių hierarchija įsikomponuoja į bendrą Java klasių hierarchiją.

Sukurtos naujos klasės hierarchijos pavyzdys ekvivalentus tokiam kodui:

```
class Mom extends Object {
//declarations,definitions
}
```

Specialūs kintamieji. Kiekviena Java klasė turi tris iš anksto apibrėžtus kintamuosius: *null*, *this* ir *super*. Pirmieji du yra *Object* tipo. *null* žymi neegzistuojantį objektą, o *this* nurodo tą patį objekto egzempliorių. *super* nurodo į tiesioginę *superklasę*.

Jei kintamasis nėra inicijuotas, jo reikšmė lygi *null* specialiam kintamajam. *null* objektas neturi jokių kintamųjų ir metodų, todėl su juo negalima atlikti jokių manipuliacijų.

Klasės inicijavimas. *New* operatorius sukuria klasės kintamąjį ir tuomet klasės kintamasis vadinamas objektu arba egzemplioriu. Konstruktorius gali turėti parametrus ir gali jų neturėti:

```
someClass A=new someClass();
```

Konstruktorius yra tiesiog specifinis metodas, todėl yra leidžiama jo perkrova. Perkrovos pavyzdys:

```
public class Box {
int boxWidth;
int boxLength;
int boxHeight;

public Box(int i) {
boxWidth=i;
boxLength=i;
boxHeight=i;}
```

```
public Box(int i, int j) {
boxWidth=i;
boxLength=i;
boxHeight=j;}

public Box(int i,int j, int k) {
boxWidth=i;
boxLength=j;
boxHeight=k;}

//other methods
}
```

Jei konstruojant Box klasės objektą pateikiamas tik vienas parametras, tada tai yra kubas; jei du parametrai - pagrindas yra kvadratas, o antras parametras nurodo aukštį, kai į konstruktorių bus kreipiamasi pateikiant tris parametrus, jie bus panaudoti aprašant stačiakampio gretasienio pagrindą ir aukštį. Konstruktoriaus perkrova leidžia lanksčiai naudotis *Box* konstruktoriumi.

Java konstruktoriai kiek skiriasi nuo metodų kintamųjų pradinių reikšmių priskyrimo būdu. Jei Java klasė turi bent vieną tiesiogiai realizuotą konstruktorių, tai visi klasės konstruktoriai nepaveldi *superklasės* konstruktorių. Į tai atsižvelgiama numatant kokios bus pradinės klasės kintamųjų reikšmės.

Protected modifikatorius. *Protected* modifikatorius suteikia teisę naudotis kintamuoju arba metodu tik klasėms, kurios yra tame pačiame pakete. Kokiam paketui yra priskiriama klasė yra paskelbiama jos aprašymo pradžioje:

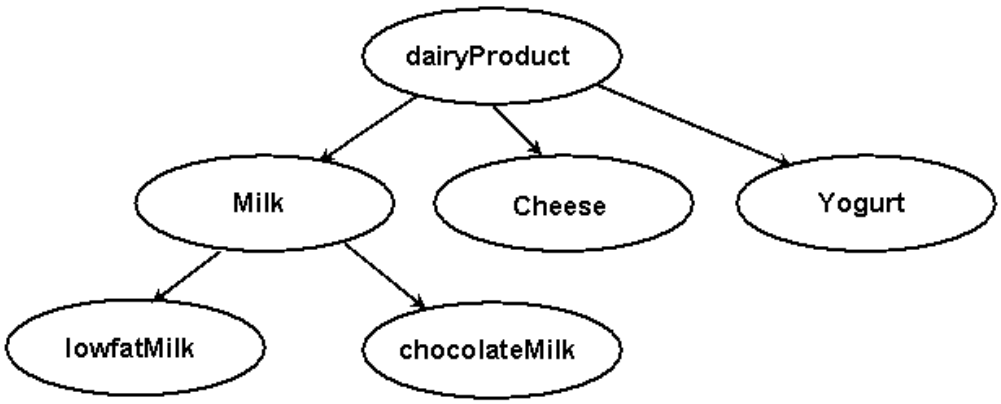
```
package somePackage;
```

Jei tiesiogiai nenurodytas klasės pavadinimas, ji patalpinama į nutylimąjį einamojo katalogo Java klasių paketą.

Jei nenurodomas tiesiogiai metodo ar kintamojo modifikatorius, tai kompiliatorius leidžia juos naudoti einamojo katalogo klasių nariams. Tačiau jei vėliau klasė įtraukiama į paketą, nutrūksta ryšys tarp einamojo katalogo klasių narių ir atsiras programos derinimo sunkumų.

Klasių hierarchijos struktūrizavimas. Paveldėjimas leidžia neperrašinėti iš naujo klasės kodo ir visą jį paveldėti poklasėms. Tačiau tai tik viena naudinga paveldimumo savybė. Kitas paveldimumo panaudojimas yra susietas su klasių hierarchijos konstravimu.

Tarkime reikia sukurti visą prekybos sistemą. Galima parašyti klasę, skirtą pienui. Bet ir pieno yra skirtingų rūšių, tarkime rūgpienis. Visas pieno rūšis apjungus į vieną klasę, gausis tik vienas prekyboje naudojamų narių pavyzdys. Todėl galima sukurti kiek platesnę klasių hierarchiją:



3.2.3 brėžinys. Pieno produktų klasės hierarchija.

Schemoje parodyta kaip galima inicijuoti poklasę ir ją naudoti kaip superklasės kintamąjį. Pieno produktų atveju bus būtina pasirūpinti metodu, kuris praneš ar produktas tinkamas naudoti.

Abstrakčios klasės ir metodai. Naudojant abstract modifikatorių, visos poklasės yra priverčiamos realizuoti abstrakčius metodus. dairyProduct klasė vis dar gali būti inicijuojama kitose klasėse. Tačiau ji gali būti paskelbta abstrakčia, kad uždrausti jos tiesioginį inicijavimą:

```
public abstract myAbstractClass {
//code
}
```

Nors klasė ir abstrakti, jos viduje galima aprašyti metodus ir kintamuosius. Šie metodai bus naudingi poklasėms, nes jos juos paveldės.

Polimorfizmas ir Java vartotojo sąsaja. Polimorfizmu vadinama Java savybė, kuri suteikia galimybę grupei metodų naudoti tą patį metodą, tačiau kiekvienas metodo panaudojimas gali duoti skirtingus rezultatus. Tokia Java savybė yra grindžiama sąsajomis.

Polimorfizmas yra kiek ribotas. Yra tik garantija, kad visos hierarchijos klasės turės metodus paskelbtus hierarchijos viršūnėje. Daugeliu atvejų poklasėms bus reikalingi metodai, kurie nereikalingi visai klasių hierarchijai. Pavyzdžiui, kadangi pienas ir jogurtas yra skysti, jiems reikės cleanUpSpill metodo. Tačiau tokio metodo paskelbimas sūriui būtų beprasmis.

3) Judesio objektiškai orientuoto programavimo taikymas

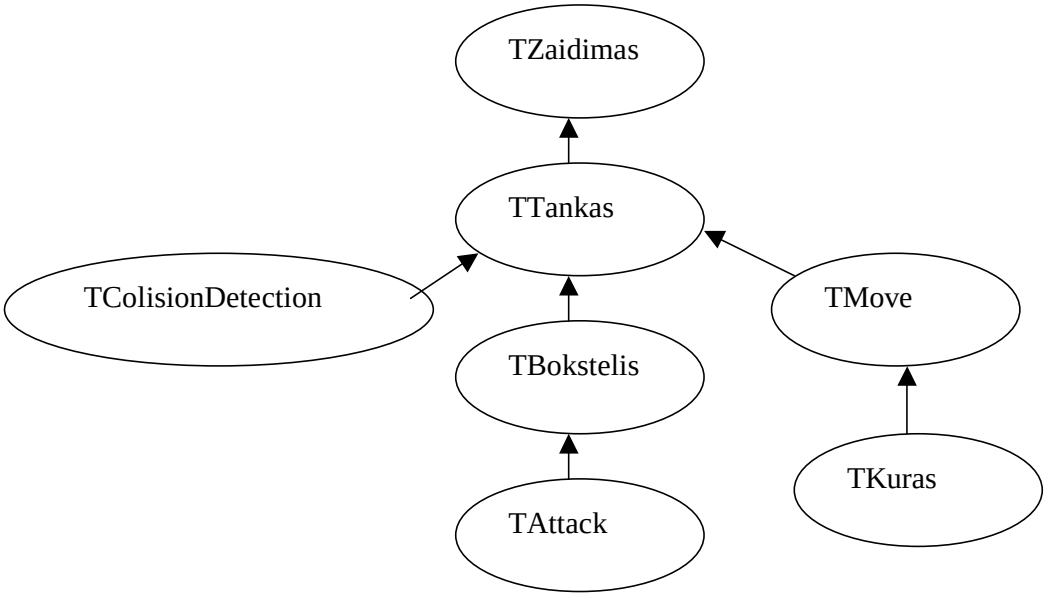
Judesio objektiškai orientuotas programavimas gali būti taikomas daugelyje sričių. Tiek Internetinės reklamos, tiek animacijos, tiek filmo efektų, tiek žaidimų kūrime.

Kolizijų[28] aptikimas animacijoje. Kolizijų (susidūrimų) aptikimas tarp judančių objektų yra vienas iš svarbiausių uždavinių šiuolaikinėje kompiuterinėje animacijoje. Objektiškai orientuotos technikos panaudojimas, ir leidžia aptikti kolizijas. Realios sistemos mažina kolizijos

[28] Collision (kolizija) – suprantama kaip susidūrimas

aptikimo laiką naudojant trijų pakopų procesą. Pirmoje pakopoje nustatomi objektai esantys vienoje tam tikroje vietoje (aplinkoje), naudojant globalių ribinių tūrių lentelę. Antra pakopa nustato visus įmanomus kolizijos zonas naudojant sferos-medžio [29] duomenų struktūrą. Kitoje paskutinėje pakopoje randami taškai (susikirtimai) tarp objekto daugiakampių paviršių, kuris savyje turi susikertančias poras mazgų. Vadinasi algoritmas naudoja sferinę geometriją tik apytiksliai, kad greitai aptiktų galimas kolizijų vietas. Tokiu būdu sistema greita ir tiksli. Įvairių geometrinių testų naudojimas palaiko tai ir pagerina sistemos sąveiką kelis kartus geriau nei daugiasienių susikirtimo testai.

Judesio OOP taikymas žaidimuose. Žaidimai vis labiau ir labiau skverbiasi į kiekvieno gyvenimą. Žaidžia vaikai, žaidžia paaugliai, žaidžia suaugusieji. Kuo realesnė žaidimo aplinka, kuo realesni veikėjai, kuo realesnis valdymas tuo labiau toks žaidimas traukia vartotoją. Pačiam paprasčiausiam žaidimui sukurti galbūt užtektų ir vienos klasės. Tačiau jei tas žaidimas yra realistiškas o jame daug įvairių funkcijų su viena klase neapsieisi. Tada kuriamos sudėtingos sistemos su keliomis ar keliomis dešimtimis klasių. 3.3.1 paveiksle pavaizduota žaidimo *Tankiukai* klasių hierarchija. Šį žaidimą galite rasti prieduose kataloge *Zaidimai*.



3.3.1. Žaidimo Tankiukai klasių hierarchija

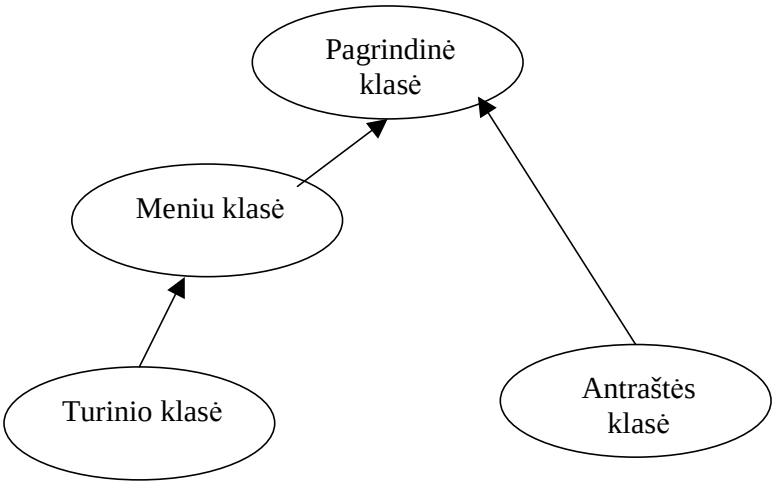
Žaidimo pagrindinė klasė yra *TZaidimas*. Ši klasė savyje turi metodus ir savybes atsakingus už galutinių duomenų atvaizdavimą ir pateikimą vartotojui. Klasė *TTankas* savyje laiko savybes ir metodus atsakingus tanko būklę, taip pat paveldi visas savybes ir metodus iš TZaidimas klasės. *TColosionDetection* atsakinga už tanko susidūrimų aptikimą su pašaliniais objektais, paveldi

[29] Sphere-tree (sferos medžio) – hierarchinis sferų medis paremtas erdviniu skaidymu

visas savybes ir metodus iš *TTankas* klasės. *TMove* yra *TTankas* klasės poklasis, savyje saugantis metodus ir savybes susijusius su tanko judėjimu. *TAttack* klasė yra TBokstelis klasės poklasis atsakingas už šaudymą. *TKuras* klasė paveldi savybes ir metodus iš klasės *TMove* ir yra atsakinga už tanko kurą (t.y. tankui važiuojant kuras mažėja). Kadangi tai tik žaidimo demonstracinė versija, tai dar nėra sukurtas priešas. Atitinkamai nėra ir klasių, kuriose būtų saugomos savybės ir metodai susiję su priešo veiksmų generavimu.

Judesio OOP taikymas Interneto reklamoje. Šiandien neįsivaizduojame svetainės, nesvarbu ar tai būtų asmeninė svetainė, firmos prezentacinis tinklapis ar portalas, be dinaminės grafikos. Atsivertus bet kokį puslapį galima išvysti įvairiomis spalvomis mirguliuojančius ir visaip besikeičiančius banerius, intro, animuotus logotipus ar net visą svetainę. Tokiai animacijai kurti paskutiniu metu pradedama plačiai naudoti Flash technologija. Ši technologija palaiko vidinę programavimo kalbą ActionScript. Tai yra objektiškai orientuota programavimo kalba. Kaip šitoje kalboje kuriami objektai (klasės) ir koks klasių ryšys jau buvo minėta ankstesniame skyriuje.

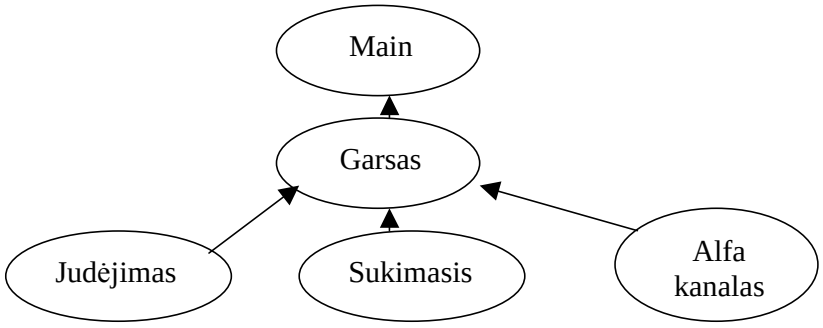
Pavyzdžiui, norint sukurti dinaminę svetainę, ją galima kurti tiesiog paprastai struktūriškai arba galima tai atlikti objektiškai. T.y. suskaidoma svetainė į atskirus modulius (objektus), nustatome jų tarpusavio ryšius (3.3.1 paveiksle).



3.3.2 pav. Supaprastintas Internetinės svetainės klasių sąryšis.

3.3.1 paveiksle pavaizduotas klasių sąryšis gali būti nebūtinai toks. Jis gali priklausyti nuo kiekvieno programuotojo individualiai. Beje schemoje kiekviena pavaizduota klasė gali turėti keletą ar net kelias dešimtis poklasių.

Lygiai taip pat kaip sudėtingą svetainę, galima modeliuoti ir paprastą banerį. Čia bus viena pagrindinė klasė (tarkime *Main*), toliau klasės (objektai) reguliuojantys klipo judėjimą, sukimąsi, alfa kanalą, garsą ir t.t. 3.3.2 paveiksle pavaizduotas banerio modelis.



3.3.3 banerio klasių hierarchija

3.3.2 paveiksle pateikta klasių hierarchija gali būti ir kitokia. Vėl viskas priklauso nuo konkretaus atvejo ir programuotojo.

4) Taikymas

Šiame skyriuje buvo apžvelgta judesio OOP metodologija bei taikymas. Išnagrinėta kokiais būdais ir priemonėmis kuriama animacija Internete pasitelkiant objektiškai orientuotus metodus judesiui modeliuoti. Palygintos objektinio programavimo galimybės:

- Java programavimo kalbos;
- Macromedia Flash vidinės programavimo kalbos ActionScript 2.0;

Taip pat skyriuje paminėta keletas judesio OOP taikymo sričių:

- Internetinės reklamos;
- Žaidimų;
- Įvairių efektų kūrime;
- Ir kt.

IV. Objektiškai orientuoto projektavimo technologijų analizė

1) Modeliavimo įrankių ir priemonių pasirinkimas

Didėjant dinaminių Internetinių aplikacijų įvairovei. Taikant dinaminiams projektams objektiškai orientuotą programavimą susiduriama su daugybe problemų jei projektą realizuoja grupė žmonių. Dėl to iškyla daugybė nesuderinamumų, dažnai nesutampa reikšmės, kintamųjų tipai ir panašiai. Tam buvo sukurta modeliavimo ir specifikacijų kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus. Tai UML[30]. Šiuo metu yra nemažai objektiškai orientuotų projektavimo sistemų palaikančių UML.

Žinomos tokios OOP sistemos kaip:

- MagicDraw;
- Rational Rose;
- ArgoUML ir Poseidon for UML;
- Borland Together;
- Enterprise architect;
- Visual Paradigm.

2) Objektiškai orientuoto projektavimo technologijos

Vaizdinis modeliavimas[31]. Modeliai padeda organizuoti, vizualizuoti, suprasti ir sukurti pačius sudėtingiausias sistemas.

Vaizdinis modeliavimas tai realių sistemos procesų vertimas į grafinį vaizdą. Modeliai naudingi tam, kad visiem[32], prisidedantiems prie kažkokio tai projekto kūrimo, būtų galima suprasti iškylančias problemas, kuriant sudėtingas sistemas, ruošiant dokumentaciją ir projektuojant programas ir duomenų bazes.

Kad efektyviai sukurti sudėtingą sistemą, kuriamos brėžinių schemos ir gilinamasi į detales. Modelis tai idealus būdas pavaizduoti sudėtingos problemos abstrakciją išmetant neesmines detales. Projektuojant išskiriama keletas sistemos vaizdų ar padaromas sistemos pradinis vaizdas, toliau plėtojant modelį būtina naudotis tam tikromis nustatytomis taisyklėmis, patikrinama, kad modelis atitiktų tam tikros sistemos reikalavimus, toliau pamažu konstruojamas, jungiant po detalę, projektas iš modelio virsta tam tikra sistema ar programa.

[30] UML - Unified Modeling Language, Vieninga modeliavimo kalba
[31] Visual modeling – vaizdinis modeliavimas
[32] Turima galvoje visus projekto dalyvius (klientai, analitikai, dizaineriai, programuotojai ir t.t.)

Bet kokios programinės įrangos sistemos modelis yra kaip pavyzdžiui, pastato schema. Architektas negali konstruoti statinio be pradinio eskizo ar schemos. Lygiai taip pat schemos yra braižomos elektrikams, vandentiekininkams, dailidėms ir t.t. Kuriant programinės įrangos sistemą, projekto dalyviai, pasiskirsto panašiai. Atitinkami modeliai nubraižomi taip, kad tiktų marketingui, programinės įrangos gamintojams, sistemos plėtotojams, inžinieriams.

Vaizdinis modeliavimas ypatingas tuo, kad modelius gali suprasti, bet kokią kalbą mokantis asmuo. Galiausiai, su vaizdiniu modeliavimu, galima atkartoti kai kurias sistemos ar programos dalis, kuriant komponentus, t.y. sukūrus viena komponentą vienai sistemai, jį galima panaudoti kitos panašios sistemos kūrime. Komponentai gali būti kuriami ir dalijami tarpusavyje atskirų komandos modeliuotojų. Tokie komponentai yra lengvai redaguojami ir įmontuojami į sistemą.

Vaizdiniame modeliavime naudojami diagramų tipai:

- Veiklos diagrama. Modeliuoja dinaminę sistemos elgseną (vaizduojami veiksmai);
- Panaudos atvejų diagrama. Apibūdina funkcinį sistemos veikimą vartotojo pažiūriu;
- Sekos diagrama. Apibūdina dinaminę veikėjų (aktorių), sistemos objektų ir sistemos sąveiką;
- Bendradarbiavimo diagrama. Apibūdina pranešimus, siunčiamus tarp komponentų;
- Klasių diagrama. Apibūdina statinę sistemos struktūrą: objektus, atributus, asociacijas;
- Būsenų diagrama. Apibūdina vieno sistemos objekto dinaminį elgesį kaip būsenų kaitą;
- Komponentų diagrama. Aprašo sistemoje naudojamus komponentus.
- Išdėstymo diagrama. Aprašo fizinį sistemos diegimą.

MagicDraw. <...> apdovanojimo šiais[33] metais sulaukė ir Lietuvos specialistai. UAB "Baltijos programinė įranga" kuriamas "MagicDraw UML" įrankis pripažintas produktyviausiu 2005 modeliavimo priemonių kategorijoje[34]. "Toks "Software Development Magazine" skaitytojų ir ekspertų įvertinimas mums suteikia garbės ir įrodo, kad nuolatiniam produkto tobulinimui skiriamos investicijos pateisina lūkesčius. Produkto vartotojų nuomone "MagicDraw" yra ne tik žymiai patogesnis įrankis palyginti su kitomis modeliavimo priemonėmis, bet ir leidžia dirbti daug efektyviau." - pareiškė "No Magic" korporacijos vadovas M. Harris.

"Software Development Product Excellence and Productivity Awards" apdovanojimai labai vertinami programinės įrangos kūrėjų ir reiškia žymiai daugiau nei skambus nominacijos pavadinimas. "Software Development Magazine" organizuojami rinkimai yra vieni profesionaliausių, o vertinimo komisiją sudaro žinomiausi programinės įrangos kūrimo industrijos atstovai. [2.15]

[33] Kalbama apie 2005 metus.
[34] 15th Annual Jolt Product Excellence and Productivity Awards in the Design Tools category

MagicDraw™ UML atitinka naujausius Java bei UML technologijų standartus, turi vieną iš patikimiausių išeities kodų inžinerijos mechanizmų programavimo kalboms kaip:

- Java;
- C#;
- C++;
- CORBA IDL;

Yra galimybė vykdyti:

- Paminėtų programavimo kalbų kodo atvirkštinę inžineriją;
- Duomenų bazių schemų atvirkštinę inžineriją;
- Kodo bei duomenų bazių schemų generavimą;

MagicDraw UML naudoja ***roundtrip*** technologiją, kuri leidžia keisti tiek objektiškai orientuotą modelį, tiek programos kodą bet kokiu eiliškumu, juos nuolat sinchronizuojant.

MagicDraw palaiko UML 1.4 specifikaciją leidžiančią braižyti dvylikos rūšių standartines UML diagramas bei UML plėtinių diagramas:

- Struktūrinės diagramos.
  - Klasių;
  - Objektų;
  - Komponentų;
  - Realizavimo;
- Elgsenos diagramos.
  - Panaudojimo atvejų;
  - Sekų;
  - Bendradarbiavimo;
  - Veiklos;
  - Būsenų;
- Modelio tvarkymo diagramos.
  - Paketų;
  - Posistemių;
  - Modelio;
- UML plėtinių diagramos.
  - Patikimumo;
  - Web programų plėtinių;
  - Duomenų bazių;
  - COBRA IDL;
  - XML schemų;

- WSDL;
- Turinio;

Rational Rose. Sistemos *Rational Rose* modelių diagramų architektūra palengvina UML[35] kalbos naudojimą, COM [36], OMT [37] ir Booch'93 [38] metodą vizualiam modeliavimui. Semantinės informacijos naudojimas garantuoja konstrukcijos ir vientisumo palaikymo korektiškumą.

Modeliavimas su Rational Rose. Tai vaizdinio modeliavimo įrankis, leidžiantis kurti komponentus, juos analizuoti, projektuoti, peržiūrėti, redaguoti ir manipuliuoti. Yra galimybė grafiškai pavaizduoti programos ar sistemos elgesio bendrą vaizdą.

Rational Rose sistemoje numatytos tam tikros taisyklės reikalingos apibrėžti ir dokumentuoti sistemos architektūrą. Loginė architektūra apima klasių diagramas, kurios savyje talpina klases ir sąryšius atstovaujančius kuriamos sistemos abstrakcijas. Komponentinė architektūra apima komponentų diagramas, kurios pabrėžia esamos programinės įrangos modulių organizavimą per projektavimo terpes. Išsišakojusiose diagramų mazguose žymimi programos procesai – parodantys elementų apdorojimo ir jų programinės įrangos procesų konfigūraciją.

Žymėjimas. Žymėjimas užima svarbią vietą bet kokioje programos vystomoje dalyje – tai lyg klijai, kurie sujungia procesus į visumą. UML (Unified Markup Language) yra numatytas gan stiprus žymėjimas, kuris prasiplėtė nuo analizės iki projekto. Neabejotinai elementų žymėjimas[39] atsirado per analizę. Kitoks elementų užrašymas[40] atsirado per projektavimą.

Žymėjimai sistemoje turi sekančias taisykles:

- Perduodami sprendimai, kurie nėra akivaizdūs ir negalima ką nors pasakyti vien iš jo kodo;
- Paruošta semantika, kuri leidžia parinkti naudingą strategiją ir gerus sprendimus.
- Pasiūlomos konkrečios formos ir įrankiai, kuriais galima manipuliuoti.

Rational Rose numato tokias ypatybes palengvinančias analizę, projektą ir pasikartojančias programos konstrukcijas:

- Panaudojimo atvejų[41] analizė;
- Objektiškai Orientuotas modeliavimas;
- Vartotojo keičiamos aplinkos palaikymas UML kalboje, COM, OMT ir Booch'93 metoduose.
- Semantinis tikrinimas;
- Valdomas pasikartojimų vystymo palaikymas;

[35] UML – Unified Modeling Language (Unifikuota (suvienodinta) Modeliavimo Kalba)
[36] COM – Component Object Modeling (Objekto - komponento modeliavimas)
[37] OMT – Object Modeling Technique (Objektinio modeliavimo technika)
[38] Booch metodas – kai taikomas standartinis UML žymėjimas
[39] Tai yra, panaudojimo atvejai, klasės, ryšiai, sankaupos, paveldėjimas.
[40] Tai yra, indikatorių palaikymas ir savybės.
[41] Use-Case suprantama kaip panaudojimo atvejų

- Abipusė inžinerija;
- Lygiagrečios daugiavartotojiškos saugyklos ir privatumo palaikymas;
- Integracija į duomenų modeliavimo įrankius;
- Dokumentacijos generacija;
- *Rational Rose* kodo rašymas integracijai ir tęstinumui;
- OLE susiejimas[42];
- OLE automatizacija;
- Daugialypė platformų galimybė.

ArgoUML ir Poseidon for UML. Pagal graikų mitologiją, didvyris Jasonas pastatė laivą ir pavadino jį *Argo* ir su savo draugais argonautais, išplaukė ieškoti auksinės vilnos. Poseidonas, jūrų dievas, apsaugojo ir saugiai leido keliauti jūra. [1.9 psl 1.]

Maždaug po 4 tūkstančių metų, Jasonas Robinsas[43] pradėjo kurti atvirojo kodo, UML modeliavimo įrankį ir pavadino jį *ArgoUML*. Dauguma tokių entuziastų kaip jis prisijungė ir šiandien mes turime atvirojo kodo modeliavimo sistemą.

ArgoUML buvo sumanytas kaip įrankis ir aplinka skirta naudoti objektiškai orientuotų programų sistemų analizei ir projektavimui. Ta prasme šis įrankis labai panašus į daugelį komercinių įrankių, kurie parduodami kaip įrankiai programų sistemų modeliavimui. *ArgoUML* turi eilę reikšmingų skirtumų nuo kitų mokamų įrankių:

- *ArgoUML* turi eilę savybių, kurios leidžia greitai susipažinti ir suprasti sistemą;
- *ArgoUML* palaiko standartus kaip UML, XMI, SVG, OCL ir kitus;
- *ArgoUML* yra grynas Java produktas. Tai leidžia *ArgoUML* veikti ant visų platformų, kur tik veikia Java;
- *ArgoUML* yra atviro kodo produktas.

Dauguma šio įrankio savybių išsivystė natūraliai, bet šiek tiek daugiau aukštesnio lygio problemų nėra išspręsta, tai gali sau leisti tik komercinės kompanijos. Kai kurie iš kūrėjų komandos atsiskyrė ir įkūrė kompaniją *Gentleware.* Jie sukūrė patobulinimą šiam įrankių rinkiniui ir pavadino *Poseidon for UML* kaip ir dauguma panašių komercinių kompanijų.

Bazinė *Gentleware* įrankių rinkinio versija buvo pavadinta *Poseidon for UML Community Edition,* ji buvo nemokama. Kuriant šią versiją buvo remiamasi *ArgoUML* ir ypač vartotojo sąsaja ir pagrindinės komandos abiejų įrankių yra beveik identiški. Poseidon for UML turi daugiau savybių ir yra šiek tiek stabilesnis. Jis skirtas būsimiem kasdieniniams darbams tiek komercinėse, tiek profesionaliose aplinkose. *ArgoUML*, kitų rankose, yra atviro kodo ir suteikia galimybę kitiems ją tobulinti, nagrinėti architektūrą ir tęstinumą.

[42] OLE Linking – Object Linking and Embedding (iš angl. k. Objektų susiejimo ir įdiegimo mechanizmas)
[43] Jason Robbins

Reikalavimai. Poseidon for UML parašyta naudojant Java programavimo kalbą ir todėl nuo platformos nepriklauso. Ji veikia beveik visuose, naujesniuose asmeniniuose kompiuteriuose. Kad sėkmingai startuoti *Poseidon for UML* programą reikia:

- Java Runtime Environment ar Java Developer Kit. Rekomenduojamas JDK 1.3. Reikalinga vėlesnė nei 1.2 versija. *Poseidon for UML* neveiks su JDK 1.1.X ar JRE 1.1.X.
- Tam tikra operacinė sistema nereikalinga. Poseidon for UML kiek žinoma veikia tiek ant Windows šeimos, tiek ant Linux šeimos, tiek ant MacOS operacinių sistemų. Labiausiai naudojama ir daugiausiai pratestuota yra ant Linux operacinės sistemos, kaip bebūtų, Windows platforma tinka labiau. Yra keletas klaidų, kai kurie JDK sprendimai, skirtingoms platformoms turi nenumatytų klaidų. Pavyzdžiui, tempimas ir numetimas[44], neveikia ant Linux/JDK 1.3. Naujesnėse versijose šios problemos išspręstos.

Nemokamą programos versiją ir vartotojo vadovą galima rasti kompaktiniame diske, kataloge *Priedai/Nemokamos programos.*

Kurdamas judesio objektiškai orientuoto projektavimo modelius, pasirinkau Poseidon for UML projektavimo sistemą. Mano pasirinkimą lėmė šie veiksniai:

- Ši projektavimo sistema yra atviro kodo, todėl nemokama;
- Sistemos galimybės nėra didelės, tačiau judesio modeliams realizuoti pilnai pakanka;
- Sistemoje yra galimybė klasių ryšių diagramas konvertuoti į Java kodą. Modeliams realizuoti pasirinkta ActionScript objektiškai orientuota programavimo kalba labai panaši į Java programavimo kalbą.

3) Animacijos objektiškai orientuotas projektavimas

Techninės specifikacijos rašymo eigoje naudota UML kalba, leidžianti patogiai ir suprantamai aprašyti judesio objektiškai orientuotus modelius. Diagramoms kurti panaudotas kompanijos *Gentleware* produktas *Poseidon for UML*, palaikantis UML 1.4 versiją. Projekto realizacijos kalba pasirinkta ActionScript, nes šią kalbą naudoja Macromedia Flash MX 2004 sistema.

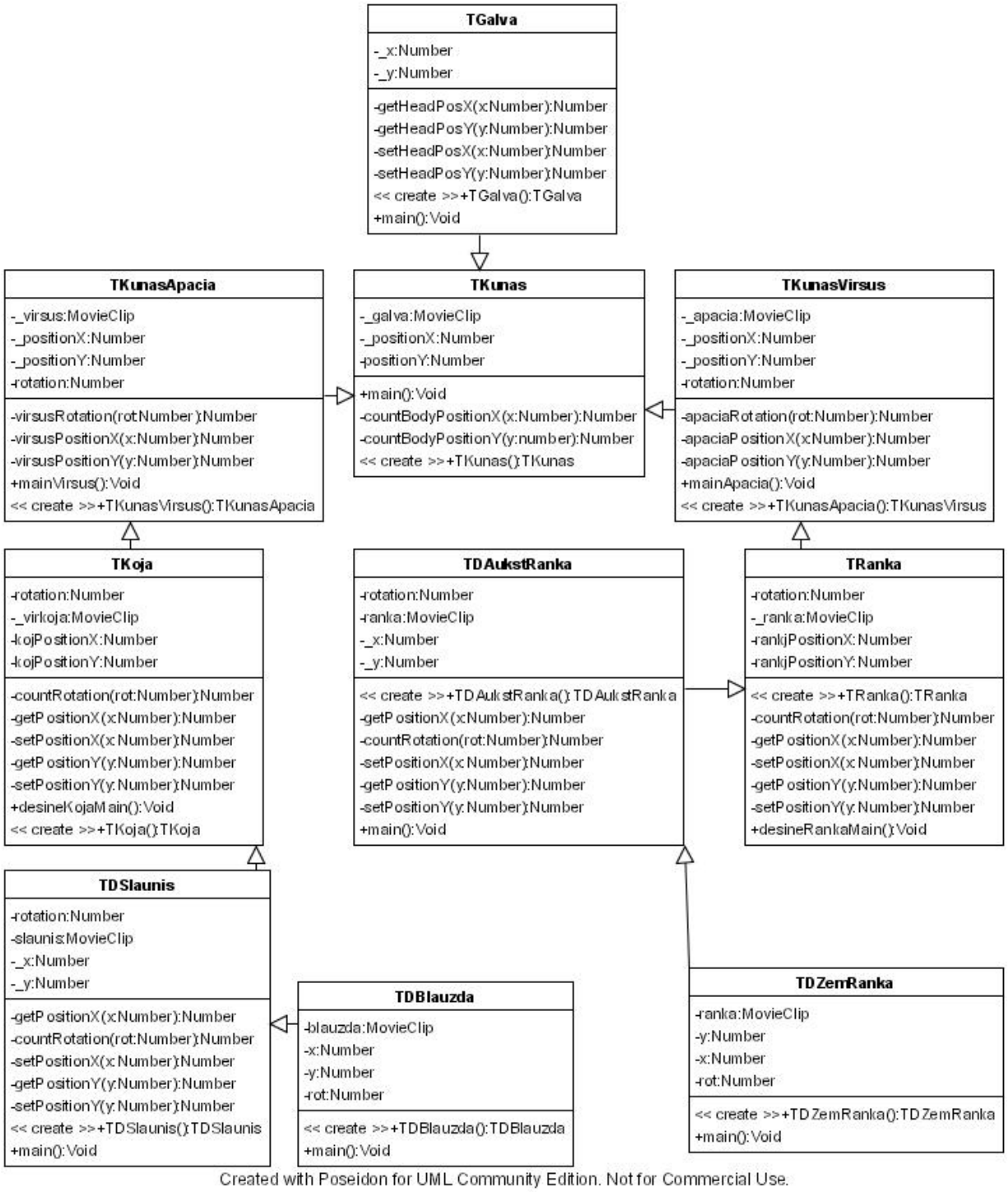
Judesio objektiškai orientuotas projektavimas tai judesius aprašančių klasių kūrimas ir apjungimas į visumą. Kadangi vienas iš darbo uždavinių sukurti keletą judesio objektiškai orientuotų modelių, sukūriau šiuos modelius:

- Žmogaus judesio imitavimo modelis;
- Laikrodis;
- Sniegas;
- Dinaminis prekių pristatymo katalogas;

[44] Drag and Drop

Ø Statybinės įmonės, dinaminė tinklapio antraštė;

Žmogaus judesio imitavimo modelis. Projekto aprašymas ir pats projektas pateiktas kompaktiniame diske, kataloge *priedai/projektai/Zmogaus ejimo imitavimas* byla *Ejimas1.exe.* Žemiau pateikta diagrama (2.3.1 pav.) vaizduoja žmogaus ėjimo modelio klasių tarpusavio ryšį.



2.3.1 objektų tarpusavio sąryšis pavaizduotas naudojant nemokamą projektavimo sistemą Poseidon for UML Community Edition 3.0

2.3.2 paveikslas iliustruoja žmogaus iš profilio judesio momentą. Dėl laiko trūkumo pateikiamas labai supaprastintas dizainas. Kiekvienas žmogaus elementas sudarytas iš vienos ir tos pačios atkarpos tik iš jos kelis kartus padaromas dublikatas, atitinkamai pasukamas ir pastatomas į reikiamą vietą.

2.3.2 Žmogaus judesio imitavimas

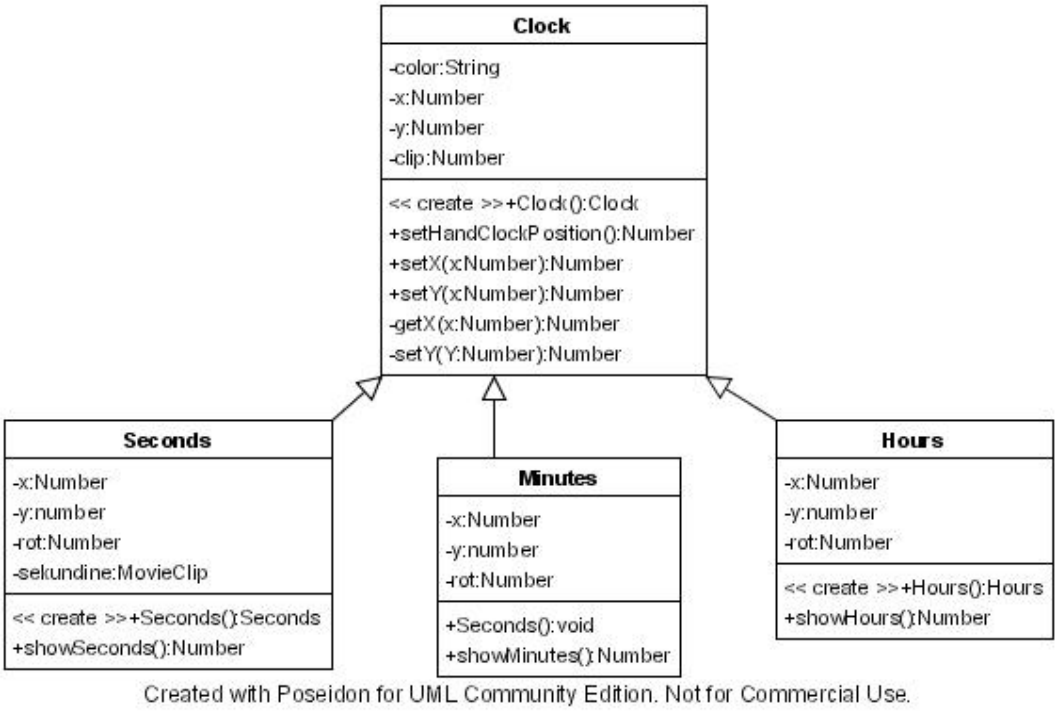
Laikrodžio modelis. Laikrodžio modelis pateiktas kompaktiniame diske, kataloguose *Priedai/Projektai/HandClock/Laikrodis rodantis laika pagal kompiuterio laika* ir *Priedai/Projektai/HandClock/Laikrodis rodantis laika pagal serverio laika.* Tiek viename tiek kitame kataloge bylos pavadinimas *laikrodukas.exe*. Tai yra žymiai paprastesnis projektas nei prieš tai pateiktas. Yra du projekto variantai. Vienas jų kai laikrodis laiką rodo tiesiai iš kompiuterio ir kitas variantas kai laikas specialia funkcija gaunamas iš serverio ir perduodamas pagrindinei klasei (3.3.3 pav.).

Pagrindinė klasė *Clock* susijusi su rodyklių pozicijos ir pasukimo nustatymu. Klasės *Seconds, Minutes, Hours* gauna pradinį laiką ir kas tam tikrą sekundės dalį paskaičiuoja rodyklės pasisukimą ir perduoda reikšmę rodyklėms.

Rodyklės pasisukimas skaičiuojamas pagal formulę:

rodykle._rotation += laipsnis*getTimer()/1000;

kur *rodykle* tai konkrečios rodyklės vardas (instance), *laipsnis* tam tikros rodyklės pasisukimo kampas per laiko vienetą (sekundės 6 laipsniai, minutės 1/6 laipsnio, valandos 1/120 laipsnio).

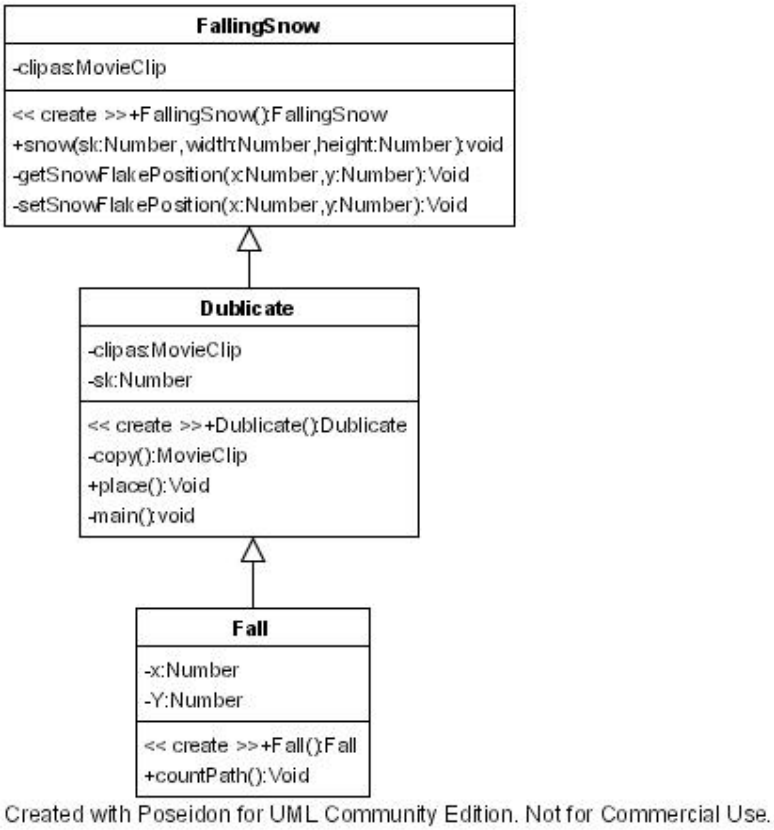


3.3.3. Laikrodžio klasių diagrama

3.3.4 paveiksle pavaizduota Laikrodžio klasių diagramos (3.3.3 pav.) realizacija.

3.3.4. Suprojektuotas laikrodis naudojant objektiškai orientuotus metodus

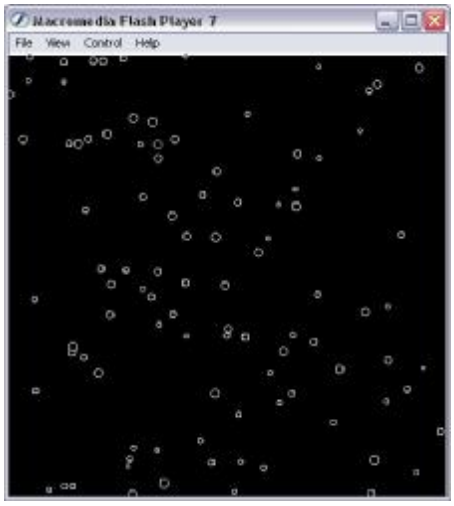
Sniegas. Sniego modelis pateiktas kompaktiniame diske, kataloge *Priedai/Projektai/FallingSnow/* bylos FallingSnow.exe, FallingSnow1.exe, FallingSnow2.exe. Tai vėlgi nesudėtinga klasių struktūra. Sniego kritimas generuojamas atsitiktinai, chaotiškai panaudojant snaigės kritimo keliui generuoti sinusoides bei kosinusoides. 3.3.5 paveikslas iliustruoja atsitiktinį sniego kritimą aprašančias klases.



3.3.5 Snigimo modelio klasių diagrama

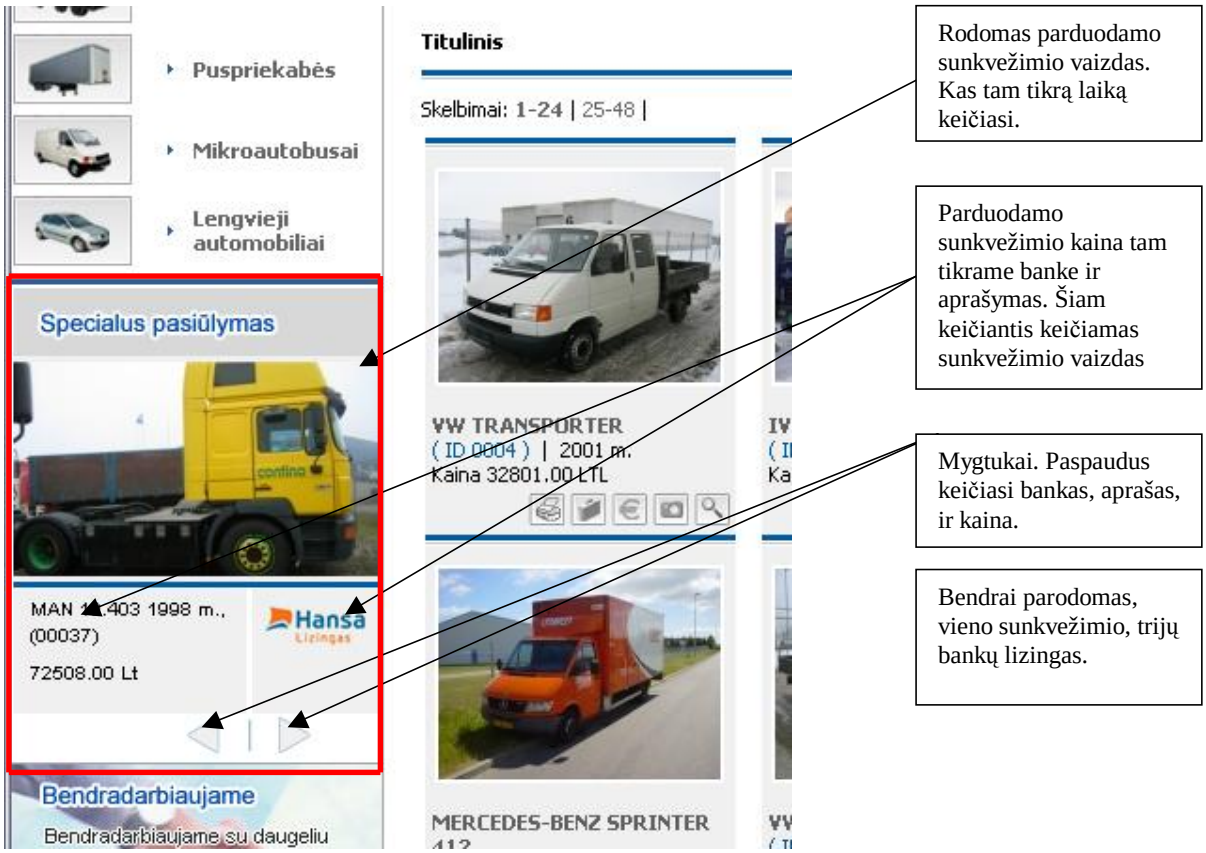
Pgrindinė klasė *FallingSnow* nustato pagrindinio objekto (snaigės) poziciją ir vardą (instance). Klasė *Dublicate* nukopijuoja snaigę ir išdėsto atsitiktiniais tarpais x ašyje, ekrano viršuje. Klasė *Fall* palaiko savybes ir metodus susijusius su atsitiktiniu snaigės kritimo keliu. tvarka

Žemiau, 3.3.6 paveiksle pateikta viena iš snigimo medelio realizacijų.



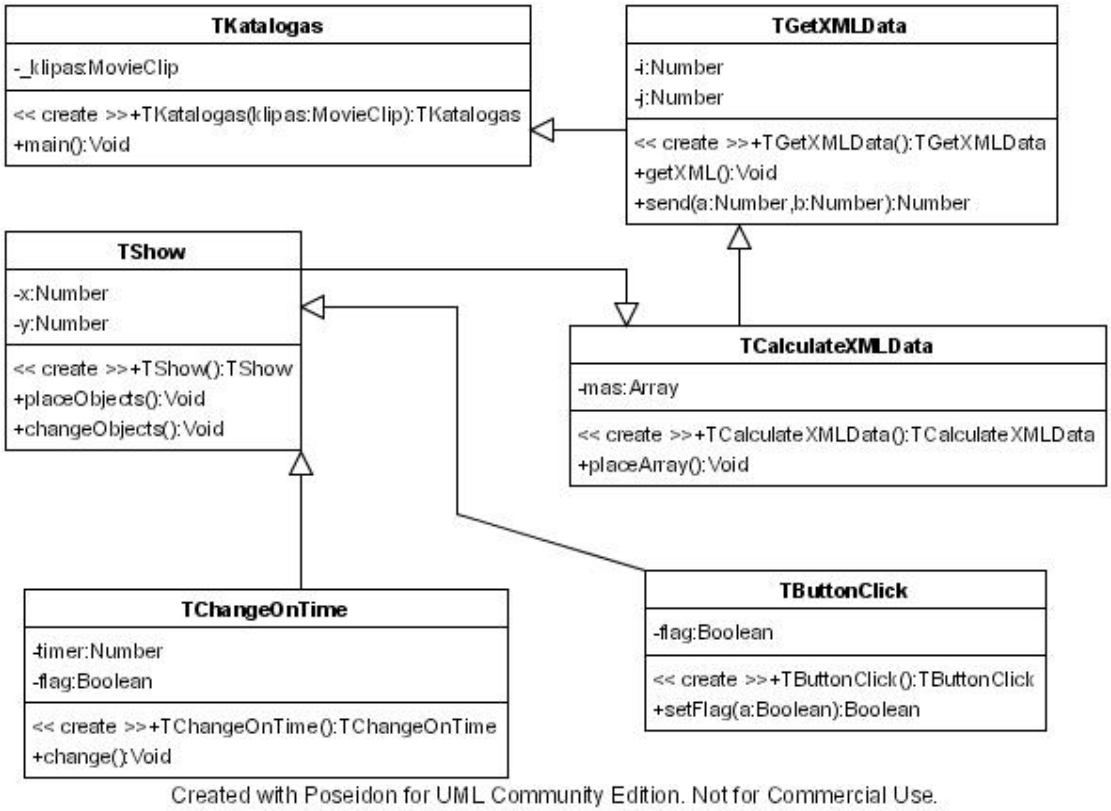
3.3.6 sniego kritimo modelio realizacija

Dinaminis prekių pristatymo katalogas. Šis modelis talpinamas svetainėje http://www.kidtrucks.lt/ 3.3.7 pav.



3.3.7. Dinaminis prekių pristatymo katalogas tinklapyje

3.3.8 paveiksle pateikiama šio modelio klasių diagrama. Pagrindinė klasė pavadinta *TKatalogas.* Klasėje saugomos savybės ir metodai susiję su sunkvežimio vaizdo, aprašymo, kainos ir banko logotipo rodymo vieta. Kita klasė *TGetXMLData,* ji iš savo superklasės *TKatalogas* paveldi savybes ir metodus. *TGetXMLData* skirta duomenų iš XML bylos nuskaitymui. *TCalculateXMLData* susistemina ir sudeda į masyvą *TGetXMLData* klasės gautus duomenis. Gaunamas tam tikrų duomenų masyvas. Klasė *TShow* pagal duomenų masyvo duomenis prisega atitinkamą paveiksliuką, banko logotipą ir sunkvežimio aprašymą su kaina. Atitinkamai klasė *TChangeOnTime* keičia sunkvežimius kartu su visais atributais kas tam tikrą laiką, o klasės *TButtonClick* dėka, paspaudus ant vieno ar kito mygtuko perduodama vėliavėlė, kad galima keisti paveiksliuką t.y. rodyti prieš tai rodytą ar sekantį.

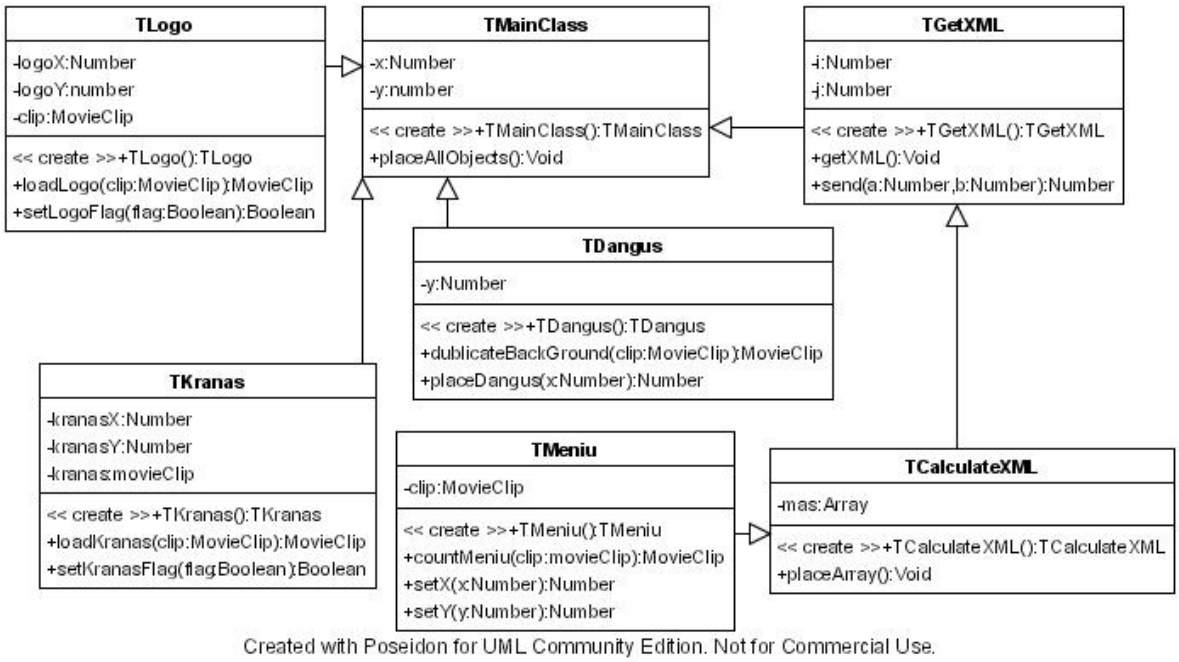


3.3.8 pav. Dinaminio prekių pristatymo katalogo klasių diagrama

Statybinės įmonės, dinaminė tinklapio antraštė. Modelis patalpintas tinklapyje http://www.ns.lt/index.php 3.3.9 pav.

3.3.9 pav. Reklaminė antraštė

3.3.10 paveiksle pateikiama reklaminės antraštės modelio klasių diagrama. Pagrindinė klasė *TMainClass*, kurioje saugomos savybės ir atributai atsakingi už reklaminio skydelio elementų išdėstymą tam tikroje nustatytoje vietoje. Iš ši superklasė turi poklasius *TLogo*, *TKranas*, *TDangus*, kurios kiekviena savyje saugo savybes ir metodus susijusius su atskiro elemento užkrovimu ir parodymu. Klasė *TGetXML* nuskaito iš XML bylos duomenis susijusius su meniu pavadinimais. Klasė *TCalculateXML* meniu pavadinimus sudeda kaip String tipo kintamuosius į masyvą. Klasė *TMeniu* išdėsto meniu tam tikrais tarpais reklaminiame skydelyje.



3.3.10 pav. Reklaminio skydelio pavaizdavimo klasių diagrama

4) Apibendrinimas

Šiame skyriuje buvo pateiktos ir išanalizuotos trys objektiškai orientuoto projektavimo sistemos. Tai:

- MagicDraw;
- Rational Rose;
- Poseidon for UML ir ArgoUML;

Projektavimo sistema MagicDraw UML yra kuriamas UAB „Baltijos programinė įranga“, kurioje dirba ir Lietuvos specialistų, buvo pripažinta produktyviausiu modeliavimo įrankiu pasaulyje. Bet kokiu atveju Rational Rose šiuo metu populiariausias, dažnai naudojamas projektuotojų. Poseidon for UML ir ArgoUML gal šiek tiek mažiau žinomos, o kitiem gal ir iš viso nežinomos. Poseidon for UML pasirinkta todėl, kad visų pirma tai yra atviro kodo sistema. Versijos skirtos ne komerciniams tikslams yra nemokamos. Sistema pasirinkta dar ir todėl, kad ji sukurta

Java kalbos pagrindu, todėl šioje sistemoje klasių diagramų projektavimas labiau pritaikytas Java kalbai. Darbe realizuotose projektuose naudojama vidinė Macromedia Flash MX 2004 kalba ActionScript 2.0 labai artima Java kalbai.

V. Išvados

Apibendrinant visą darbą bei atsižvelgiant į darbo pradžioje iškeltus tikslus, uždavinius, galima išskirti šias pagrindines išvadas:

1. Judesio objektiškai orientuotas projektavimas Internetinėms aplikacijoms Lietuvoje taikomas, bet labai mažai, kuriant dinaminės grafikos svetaines, banerius, animacinius intarpus, logotipus. Tokiem projektam kurti naudojamos UML modeliavimo ir specifikacijų kūrimo kalbą palaikančios sistemos: MagicDraw, Rational Rose, ArgoUML ir Poseidon for UML ir kt. Projektams realizuoti tinkamos Java arba ActionScript programavimo kalbos.
2. Judesio objektiškai orientuotas projektavimas naudojamas ir kitose srityse. Video efektų, kompiuterinės animacijos, kompiuterinių žaidimų, dinaminių Interneto svetainių projektavime ir kt.
3. Visos sistemos, kurios skirtos objektiškai orientuotiem projektam kurti, tinka ir judesio objektiškai orientuotam projektavimui.
4. Darbo eigoje buvo sukurtas keletas judesio objektiškai orientuotų modelių. Keli iš jų aprašyti šiame darbe. Kiti darbai sudėti kataloge *Priedai/Projektai.*
5. Aplamai Judesio OOP gali būti taikomas daugybėje sričių:
   - Internetinės dinaminės grafikos aplikacijose;
   - Žaidimuose;
   - Klimato kitimo modeliuose;
   - Kino filmuose;
   - Animaciniuose filmuose;
6. Pagrindiniai sunkumai rašant šį darbą iškilo dėl kompiuterinių terminų Lietuvių kalba. Kadangi panaudota literatūra pagrinde anglų ir rusų kalbomis teko naudotis kompiuteriniais žodynais, tačiau ne visur ir tai padėjo, daugumą angliškų terminų teko versti tiesiog pažodžiui.
7. Rašydamas darbą pradėjau naudoti projektavimo sistemas palaikančias UML kalbą. Ko iki šiol nenaudojau. UML apibūdina įvairių modelių žymėjimus, kurie gali būti pateikti per objektinę analizę ir projektavimą. Tai pagerina bet kokios aplikacijos kūrimą, ypač dirbant keliems programuotojams ir žinoma pagerina tokios programos (animacijos) kokybę visais atžvilgiais.

VI. Literatūros ir informacinių šaltinių aprašai

1. *Knygos*

1.1. R. Ališauskienė, R. Pocevičienė, A. Malakauskas, L. Ušeckienė (2004), Kursinių, bakalauro ir magistro darbų rengimo vadovas.

1.2. Colin Moock (2004) , Essential ActionScript 2.0, Chapter 2: Object-oriented ActionScript.

1.3. Macromedia (2002), ActionScript Language Reference.

1.4. Macromedia (2002), Using Components.

1.5. I. Petrauskienė (2003), Macromedia Flash 5.

1.6. K. Brodlie, J. Brooke, M. Chen, D. Chisnall, A. Fewings, C. Hughes, N. W. John, M. W. Jones, M. Riding, N. Roard (2004), Visual Supercomputing – Technologies, Aplications and Challenges.

1.7. Douglas C. Schmidt (2004). Object-Oriented Design and Programming.

1.8. User‘s guide (2005). MagicDraw.

1.9. Marco Boger (2001). ArgoUML and Poseidon for UML.

1.10. Rational the e-development company™ (2001). Using Rose.

1.11. Smaltija (1997). Aiškinamasis anglų-lietuvių kompiuterijos terminų žodynas.

1.12. A. Freedman (1999). The Computer Desktop Encyclopedia. 2nd ed.

1.13. K. V. Paulauskas (2000). Aiškinamasis kompiuterijos santrumpų žodynas.

1.14. Raul Rojas (2004). MAAT - Multi Agent Authoring Tool for Programming Autonomous Mobile Robots Diplomarbeit.

1.15. Stefan Bie, Johan Persson (2004). Behavior-based Control of the ERS-7 AIBO Robot.

1.16. Nathaniel Jones (2004). Real-time geometric motion blur for a deforming polygonal mesh.

2. *Interneto adresai*

2.1. Kompiuterijos terminų aiškinamasis žodynas [žiūrėta 2004-10-12]. Prieiga per Internetą: <http://aldona.mii.lt/pms/terminai/term/z2odynas.html.

2.2.Objektinis programavimas [žiūrėta 2004-11-20]. Prieiga per Internetą: <http://www.mif.vu.lt/~ragaisis/Informatika2000/ObjProgr.htm.

2.3. Introduction to Active Contours and Visual Dynamics [žiūrėta 2004-09-21]. Prieiga per Internetą: <http://www.robots.ox.ac.uk/~vdg/dynamics.html

2.4. A. A. Bielskis, Kompiuterinė grafika [žiūrėta 2004-10-10]. Prieiga per Internetą: < http://www.ik.ku.lt/lessons/konspekt/graphics/

2.5. William Robert Stanek, Taškai ir vektoriai: tinklapių grafikos kaita [žiūrėta 2004-05-02]. Prieiga per Internetą: < http://www.nkm.lt/048/komp08.html

2.6. Grafinių bylų formatai [žiūrėta 2004-12-10]. Prieiga per Internetą: < http://gimp.akl.lt/straipsniai.grafiniu%20bylu%20formatai.html

2.7. Animation timeline [žiūrėta 2004-12-10]. Prieiga per Internetą: < http://www.bergen.org/AAST/ComputerAnimation/Hist_Timeline.html.

2.8. First Animation of the World Found In Burnt City [žiūrėta 2004-12-10]. Prieiga per internetą: < http://www.payvand.com/news/04/dec/1249.html

2.9. First Animation of the World Found In Burnt City, Iran [žiūrėta 2004-12-10]. Prieiga per Internetą: < http://www.iranian.ws/iran_news/publish/article_5191.shtml.

2.10. Computer Animation [žiūrėta 2004-11-20]. Prieiga per Internetą: <http://www.scit.wlv.ac.uk/~c9661365/Comp_Anim.htm.

2.11. "Adobe" perka "Macromedia" už 3,4 mlrd. USD [žiūrėta 2005-04-18]. Prieiga per internetą: <http://www.delfi.lt/archive/article.php?id=6499148&categoryID=20172&ndate=1113830296.

2.12. Vektorinės grafikos technologijos tinkle [žiūrėta 2004-11-20]. Prieiga per Internetą: <http://www.reco.lt/technology/vector.php

2.13. SVG zone [žiūrėta 2005-03-25]. Prieiga per Internetą: <http://www.adobe.com/svg/#

2.14. Object-Oriented Design [žiūrėta 2004-11-30]. Prieiga per Internetą: <http://www.sei.cmu.edu/str/descriptions/oodesign_body.html.

2.15. MagicDraw UML pripažintas produktyviausiu 2005 modeliavimo priemonių kategorijoje [žiūrėta 2005-04-25]. Prieiga per Internetą: http://www.ebiz.lt/article.php3/15/7151/6.

2.16. Programų sistemų eskizinis projektas [žiūrėta 2005 – 02-03]. Prieiga per Internetą: <http://www.mif.vu.lt/~moroz/eskizin.html.

2.17. Geografinė informacinė sistema [žiūrėta 2005-02-15]. Prieiga per Internatą: <http://www.elektronika.lt/theory/theme/160/928/.

2.18. FOLDOC. Free Online Dictionary Of Computing [žiūrėta 2005-01-25]. Prieiga per Internetą: <http://foldoc.doc.ic.ac.uk/foldoc/index.html.

3. Straipsniai

3.1. Kirsten Riley, (2003). Spotlight Flash // Elisu newsletter, 5, p 1-3.

3.2. W. James MacLean, Nikos Paragios, David Fleet, (2005). Spatial Coherence in Visual Motion Analysis Computer Vision and Image Understanding Journal (CVIU).

VII. Anotacija (Summary)

Tema „Judesio objektiškai orientuoto projektavimo modeliai“. Pagrindinis darbo tikslas apibrėžti objektiškai orientuotos animacijos kūrimą Internete. Siekiant užsibrėžto tikslo iškelti tokie uždaviniai: išanalizuoti projektavimo sistemas; apžvelgti objektiškai orientuotas programavimo kalbas su galimybe kurti judesį (animaciją) Internetui; sukurti keletą judesio modelių.

Darbo naujumas tame, jog Lietuvoje objektiškai orientuotas judesio modeliavimas naudojant UML ir objektiškai orientuotą programavimą praktiškai netaikomas, o naudojama tik kadrinė animacija.

Darbe aprašomos projektavimo sistemos: Rational Rose, MagicDraw UML, Poseidon for UML ir ArgoUML; judesio kūrimo įrankiai: Macromedia Flash MX 2004, Corel R.A.V.E, 3D Flash Animator; Adobe LiveMotion2; objektiškai orientuotos programavimo kalbos: ActionScript, Java. Darbe pateikti keli objektiškai orientuoti judesio modelių pavyždžiai: reklaminis skydelis (banner), laikrodis, krentantis sniegas sniegas, judantis žmogus, dinaminis prekių katalogas.

Magistrinį darbą sudaro septynios dalys: Įvadas, Kompiuteriniai judesio modeliai ir problemų analizė, Objektiškai orientuoto projektavimo technologijų analizė, Judesio objektiškai orientuotas programavimas, Išvados, Literatūros ir informacinių šaltinių aprašai, Anotacija (summary). Darbo apimtis 46 psl.

Rašant šį darbą išskirti šie etapai: Temos formulavimas, tikslo ir uždavinių iškėlimas, hipotezių formulavimas, literatūros analizavimas, projektų kūrimas ir įgyvendinimas, darbo aprašymas, išvadų formulavimas.

Numatomi rezultatai: išanalizuotos ir aprašytos objektiškai orientuotos programavimo kalbos, projektavimo sistemos, sukurti realūs judesio modeliai. Sukurti pavyzdžiai talpinami svetainėse: http://www.kidtrucks.lt/, http://www.ns.lt/index.php, http://saukenai.siauliai.lm.lt/new/main.swf ir kt.

Summary

Topic: Object-Oriented Movement Projecting Models. The basic aim of the work is to define creation of object-oriented animation on Internet. In order to reach the given target, the following tasks are set: to analyse projecting systems; to review object-oriented programming languages with the possibility to create movement (animation) for Internet; to create a number of movement models.

New in the work is that in Lithuania object-oriented movement modelling using UML and object-oriented programming is practically not applied, using slip animation instead.

In the work I describe the following designing systems: Rational Rose, MagicDraw UML, Poseidon for UML and ArgoUML; movement creation tools: Macromedia Flash MX 2004, Corel R.A.V.E, 3D Flash Animator; Adobe LiveMotion2; object-oriented programming languages: ActionScript, Java. In the work I also give a number of object-oriented movement model examples: advertisement banner, clock, falling snow, moving man, dynamic goods catalogue.

Master's work consists of seven parts: Introduction, Computer movement models and problem analysis, Object-oriented projecting technology analysis, Object-oriented movement programming, Conclusions, Literature and information source references, Summary. The work has 46 pages.

While writing the work the following steps were covered: Topic formulation, setting of targets and tasks, hypotheses formulation, source analysis, project creation and implementation, work description and conclusion formulation.

The results provided: object-oriented programming languages and projecting systems have been analysed and described, real movement models created. The examples created may be fount at:

http://www.kidtrucks.lt/, http://www.ns.lt/index.php, http://saukenai.siauliai.lm.lt/new/main.swf etc.