Tallinna Ülikool
Informaatika Instituut

Rakenduste loomine programmi GameMaker abil

Bakalaureusetöö

Autor: Martin Kadarik
Juhendaja: Andrus Rinde

Autor: ..... „ ..... „ 2012
Juhendaja: ..... „ ..... „ 2012
Instituudi direktor: ..... „ ..... „ 2012

Tallinn 2012

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Sisukord

Sissejuhatus

Programmeerimisega alustamine ei ole lihtne, sest see nõuab tõsist tööd ja palju kannatlikkust. Seetõttu on eriti oluline, et esimesed sammud oleksid meeldivad ja motiveerivad. Samuti peaksid esimesed tulemused kiirest näha olema, et kogu vaev saaks tasutud ja tekiks julgus kõrgemaid sihte seada. Lihtsate matemaatiliste võrrandite lahendamisalgoritmide programmeerimine ei pruugi olla algajale programmeerijale piisavalt motiveeriv, sest visuaalselt on näha vaid numbrilist lahendit. Nii võib huvi programmeerimise vastu kergelt kaduda. Arvutimängude loomine pakub siinkohal aga suurepärast võimalust – nende tegemine on väga loominguline ja lõpptulemus pakub samuti lõbu nii tegijale endale kui ka teistele. Samas võimaldavad need rakendada erinevaid interaktsioonistiile, andmestruktuure ja õppida suure hulga muutujate ja objektide haldamist. Oluline on selle juures kohe algusest peale õppida järgima standardeid ja üldisi põhimõtteid.

Kahjuks pole mängude tegemine aga sugugi kerge. Nende loomine nõuab tavaliselt tõsist tööd ja heal tasemel programmeerimise oskust. Mängude tootmisele keskenduvates firmades töötavad suurte kogemustega programmeerijaid, graafikadisainereid ja helitehnikuid. Projektide eelarved on üüratud ning hoolimata suurest meeskonnast, võtab mängude valmistamine aega mitu aastat. Seetõttu oleks väga tore teha enda esimesed mängud lihtsamal moel. Siin tulebki appi GameMaker, mis võimaldab luua arvutimänge ka inimestel, kellel puuduvad head programmeerimisoskused. GameMaker aitab paremini mõista programmeerimise aluseid, sest nagu programmeerimisel tavaks, peab ka mäng järgima kindlaid reegleid ja loogilist skeemi. Lisaks on GameMakerist saadaval vabavaraline versioon, mis võimaldab kasutada seda ka programmeerimise õpetamisel koolides.

Viimasel ajal on Eestis mängude loomine natuke laiemat kõlapinda leidnud. Paljudes koolides kasutatakse programmeerimise õpetamisel mängude loomist, sest see võimaldab rakendada mitmeid erinevaid programmeerimise võtteid, näiteks interaktsiooni loomine. Üks hea näide selles vallas on Tartu Ülikoolis loodud mäng Sohni: Teine teekond Allmaailma – kuid üldjoontes ei ole paljud noored kursis, et mängude loomisega alustamiseks ei pea omama paksu rahakotti ja professionaalset meeskonda. Piisab täiesti natukesest vabast ajast ja heast ideest. Ka maailma mastaabis on suurte mängude kõrval viimasel ajal kuulsust kogunud

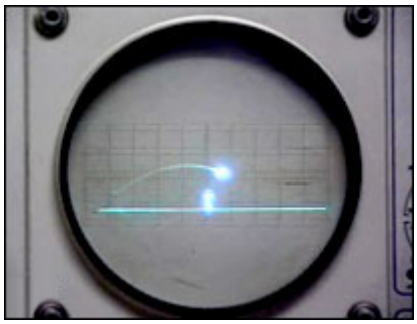
väikesed ja lihtsakoelised, peamiselt pihuseadmetel ja veebiplatvormil mängitavad mängud nagu Angry Birds, FarmVille ja Fruit Ninja.

Antud bakalaureusetöö eesmärgiks on välja selgitada, kas GameMaker on piisav, et rahuldada algaja mängutegija võimalikke soove, olles samas piisavalt lihtsa ülesehitusega, et sobida kasutajale, kellel puuduvad varasemad programmeerimise kogemused. Lisaks kasutusmugavusele peaks programm võimaldama järgida tänapäevase programmeerimise põhimõtteid, et vastava baasi omandamisel oleks võimalik hõlpsalt asuda programmeerima mõnes tänapäevases programmeerimise keeles.

Selle eesmärgi saavutamiseks toob töö autor välja aspektid, millele üks algajale programmeerijale sobiv süsteem peaks vastama. Järgnevalt võrreldakse GameMakeri võimalusi nende nõuetega ja antakse programmist ülevaade, mille juures vaadeldakse millisel viisil võimaldab programm toetada programmeerimise algteadmiste omandamist. Töö lisades on õpetus näidismängu loomiseks, mis tutvustab täpsemalt programmi üldloogikat ning põhilisemaid funktsioone, millest üks GameMakeris loodud rakendus koosneb.

1. Videomängud

Videomänguks (video game) nimetatakse elektroonilist mängu, mida mängitakse simuleeritud mänguruumis, kus mängija ülesanne on täita mingeid eesmärke. Videomängude ajalugu on juba pikem kui pool sajandit ja sai alguse 1958. aastal, kui Ameerika füüsik William Higinbotham leiutas esimese graafiliseks videomänguks nimetatud interaktiivse mängu „Tennis for Two“ (Anderson, 1983). Tegemist oli oskilloskoobi peal kuvatava küljepealt kujutatud tennise mänguga, kus juhtpuldi kaudu sai määrata nähtamatu reketi põrkenurka ja seeläbi muuta pallipõrkamist. Mäng kogus kiiresti populaarsust ja muutus terve laboratooriumi väljanäituse tõmbenumbriks.



Joonis 1 - "Tennis for Two"

Sellest ajast alates on kasvanud videomängud interaktiivse meelelahutuse lahutamatuks osaks. Algselt videomängudes sündinud väljamõeldud kangelased nagu Mario, Max Payne või Lara Croft on tuntud üle maailma ning jõudnud tänaseks päevaks videomängudest ka kinoekraanidele. Sellest hoolimata on videomängud akadeemilises maailmas veel suhteliselt uus valdkond. Seda, miks ja kuidas õnnestus videomängudel akadeemikute tähelepanu orbiidilt aastakümneteks kõrvale jääda, võib seletada mitmeti.

Videomängud on alati olnud popkultuuri, „madala kunsti“ osa, mis kuni viimase ajani pole akadeemilistesse õppekavadesse kuulunud (Randviir, 2011). Lisaks sellele on videomängudele sageli ülalt alla vaadatud ja osaliselt tehakse seda siiamaani, sest neid peetakse millekski triviaalseks – „laste meediumiks“, millest välja kasvatakse (Newman, 2004). Entertainment Software Association’i 2011. aasta uuringuandmete järgi on keskmine USA videomängur 37-aastane. Samast uurimusest lähtub, et lausa 29% üle 50-aastastest ameeriklastest mängib aeg-ajalt videomänge (Entertainment Software Association, 2011).

Seega lähtudes uuringutest tuleb tõdeda, et videomängud pakuvad tänapäeval meelelahutust erinevas eas inimestele.

Videomängud, mida luuakse tänapäeval nii arvutitel, mängukonsoolidel kui ka pihuseadmetel mängimiseks, on viimasel ajal teinud suure edasiarengu ning muutunud üheks kultuuri ja majanduse osaks (Hjorth, 2011). Mänge võib vaadelda ka kui üht uut kunstiliiki, mis ühendab endasse põneva süžee, visuaalselt nauditava mängupildi, meeldiva muusikalise tausta ja inimesi mõtlema panevaid mõistatusi. Kuid tänapäeva mängu kõige põhilisem omadus on siiski interaktiivsus, mis loobki mängijale osalemise mulje. Tänu sellele on võimalik arvutimängudes läbimängida reaalelulisi situatsioone.

Ajajooksul on väljakujunenud terve hulk videomängude žanreid, mida viimasel ajal on jällegi püütud rohkem omavahel segada (Habgood, 2010). Kõige laiemalt vaadates on enamik mänge loomupoolest kiiret tegutsemist nõudvad action-mängud (action games), mis panevad proovile mängija reaktsioonikiiruse ning ka koordinatsiooni. Intellektuaalsed mängud, kus tuleb leida lahendus seoseid luues ja olukordi hinnates, nagu sudoku, male, „viis nuppu ritta“. Majandusmängud, kus tuleb etteantud summadega vastavalt olukorrale targalt majandada. Strateegiamängud, kus tuleb käigupõhise interaktsiooni abil osata ette näha erinevaid olukordi ja nendeks eelnevalt valmistuda. Lisaks kõikvõimalikud simulaatormängud, mis püüavad simuleerida näiteks lennuki või auto juhtimist. Rollimängud, kus mängija peab arendama oma tegelaskuju seigeldes enamasti fantaasiamaailmas. Platvormmängud, kus eesmärgiks on tavaliselt takistuste ületamine ja sihtmärgini jõudmine kas horisontaalselt või vertikaalselt eritasapinnaliste platvormide vahel hüpates. Tulistamismängud (shooter), mis toimuvad enamasti 3D maailmas, kus tuleb täita ülesandeid, hävitada vaenlasi ja jõuda sihtmärgini. Loomulikult on võimalikud ka kõik kombinatsioonid eriliiki mängudest ning see just mängud huvitavaks ja omanäoliseks muudabki (Habgood, 2006).

1.1 Videomängude loomisele esitatud nõudmised

Edukaks mänguks võiks üldiselt lugeda mängu, mida inimestele meeldib mängida. Nagu kõikide teiste meelelahutustoodete puhul on kõike tähtsam esmamulje. Kuid hea mäng peab suutma hoida mängijat enda kütkes ka pärast paaritunnilist mängimist. Kuidas seda saavutada on muidugi hea küsimus. Esiteks peaks mäng olema unikaalne. Kahjuks võib tänapäevast laia videomängude valikut tunduda, et kõik head ideed on juba teostatud ning millegi täitsa

uudsega välja tulla on praktiliselt võimatu. Soovides luua midagi uut, on alati tarvis väga hästi teada, mis on juba olemas. Mängude puhul oleks seega tark tegu tutvuda hetkeseisuga. Millist tüüpi mängud on populaarsed ja mis on nendele taganud edu. Selge see, et luues uus Angry Birdsi kloon, ei korda see esmase edu. Kuid alustuseks oleks väga hea võtta ette mõni lihtne juba eksisteeriv mäng, mille struktuur on selge – Pong, Break-Out. Tähtis on, et mängijast sõltuks mängu lõppresultaat, kuid kindlasti peaks olema ka juhusel oma roll, et vältida ühesuguse tegutsemise korral alati sama tulemust.

Mängu aluseks peaks olema hea idee. GameMakeri puhul tuleks arvestada, et nende mängude mängimist ei võeta enamasti kuigi tõsiselt, pigem pakuvad need kiiret meelelahutust nagu enamus lihtsamaid veebimänge. Seega peab mäng olema lihtne ning mängitav igal vabal hetkel. Ülesehitus peab võimaldama mängida lühikest aega ja nõudma võimalikult vähe süvenemist. Oluline on ka järgida tuntud printsiipi *easy to learn, hard to master* ehk mängima hakkamine peab olema lihtne, aga sellest hoolimata ei tohi mängus tõeliselt heaks saamine olla liiga kerge.

Mängijatele võimalikult elutruu ja kaasahaarava visuaalse kogemuse lubamine, on olnud üks kogu videomängude turustamise ajaloo peamisi hüüdlauseid, mis on teinud mängutööstuses domineerivaks fotorealismi, kuid seda peamiselt suurtes mängudes (Habgood, 2010). Veebimängud seevastu püüdlevad taotluslikult joonistatud multifilmiliku väljanägemise poole. Siinjuures tasubki märkida, et videomängude puhul on äärmiselt oluline ka sihtgrupi leidmine, kas tegu on veebis mängitava mänguga või mõnel tänapäevasel pihuseadmel, vastavalt sellele peaks mäng olema ka ülesehitatud arvestatult seadme võimalustega.

Mängude loomine nõuab väga palju ressursse, sest üks hea mäng koondab endasse ilusa visuaalse pildi, nutika programmikoodi, mille järgi mängus tegevus toimub, helitausta mis aitab mängijal mängu sulanduda. Lihtsamate mängude puhul võib loomulikult ise kõige sellega tegeleda, kuid hoolimata meeskonna tööga seotud erimeelsustele, aitavad need erimeelsused tihti jõuda parema tulemuseni. Seega on väga oluline, et rakenduste loomise protsess oleks võimalikult ülevaatlik.

Samas peaks võimaldama programm luua võimalikult eriilmelisi mänge kõikidest žanridest, piiramata kasutaja loomingut. Programmi ülesehitus peaks olema algajale sobilik, võimalikult ülevaatlik ja hoomatav. Väga oluline on ka valmis rakenduste võimalikult hõlbus levitamine

võimalus. Lisaks peaks programm olema nagu iga teinegi programm stabiilne, tehtud töö ei tohi ära kaduda. Programm peaks võimaldama tutvuda programmeerimisega ja kõik toimingud peaks vastama reaalsele programmeerimisele. Internetis on võimalik leida mitmeid rollimängude tegemiseks mõeldud programme, kuid GameMaker ei piira mängužanri valikut, olles seega heaks valikuks. Samas pakub GameMaker ka HTML5 ja pihuseadmetele mängude tegemise võimalust, mis on midagi täiesti uudset.

2. GameMaker

GameMaker on 1999. aastal Hollandi Utrecthi ülikooli professori Mark Overmarsi poolt loodud programm, mis võimaldab luua rastergraafilisi (bitmap graphics) töölaua rakendusi kirjutamata selleks ainsatki rida koodi. See reklaamlause on saatnud programmi algusaastatest peale, sest eelkõige on GameMaker suunatud noortele, kes tahavad ise innustatult suurtest lemmikmängudest, omanäolisi mänge luua, kasutades selleks lihtsat objektide pukseerimisel (drag and drop) põhinevat keskkonda. Lisaks sellele pakub GameMaker võimalust muuta loodud mängud veelgi interaktiivsemaks, kasutades selleks GameMaker Language (GML) skripte.

2011. aasta juunis on GameMaker jõudnud oma 8.1 versioonini, mis võimaldab töötada ka versioonide 6. ja 7. abil loodud projektide kallal ning mõningatel juhtudel võib see korrektselt töödelda ka 5. versiooniga loodud projekti faile. GameMaker lubab hoida arvutis korraga erinevaid versioone ning need töötavad tõrgeteta. Programmist on loodud eraldi versioon ka Macintoshi jaoks.

Alates 2007. aastast tegeleb GameMakeri arendamisega Inglise mängufirma YoYoGames ning sellest on saadaval kaks versiooni – 39,99$ maksev tasuline Pro Edition ja Lite Edition. Tasuta versioon on suunatud GameMakeriga alles algust tegijatele ning selle funktsionaalsust on mõneti piiratud – lisaks kuvavad tasuta versiooniga tehtud rakendused käivitudes, et antud rakendus on loodud GameMakeri tasuta versiooni abil. Tasuline versioon töötab reklaamivabalt ning võimaldab luua keerukamaid animatsioone, kasutades selleks osakeste animeerimist (particle animation). Lisaks võimaldab see kasutada ka rastergraafikast tuleneva kald- ja kõverjoonte silumist (anti-aliasing), 3D-heliefekte ning erinevaid andmestruktuure. Programmist ülevaate saamise ja programmeerimise aluste mõistmise eesmärki täidab suurepäraselt ka tasuta versioon.

GameMakeriga loodud rakenduste kasutamisele pole piiranguid kehtestatud. Levitamise jaoks ei ole tarvis eraldi litsentsitasu maksta– loomulikult rõhutatakse siinkohal, et eesmärk ja teostusviis peab siiski olema seaduslik ning rakenduses kasutatavad komponendid ei tohi olla kellegi teise õigustega kaitstud. 2007. aastal loodi GameMakeri kasutajatele ühtne portaal YoYoGamesi veebilehel, kus kõik registreeritud kasutajad saavad lisada enda poolt GameMakeriga loodud mänge teistele näitamiseks. Samas saab kõiki rakendusi ka hinnata,

kommenteerida ja kirjutada koguni arvustusi. Lisaks on seal võimalik jagada enda kirjutatud GML skripte ja teisi GameMakeris kasutatavaid ressursse.

2.1 GameMakeri nõuded arvutile

Süsteeminõuded on hoitud madalal. Versioon 8 paigaldamise korral teavitatakse kasutajat, et igal vähem kui 5 aastat vanal arvutil peaks programm ilma vähimate probleemideta töötama. Toetatud on kõik uuemad operatsioonisüsteemi versioonid alates *Windows 2000*. *Macintoshi* versioon nõuab töötamiseks *OS X Leopard 10.5* või uuemat operatsioonisüsteemi. Optimaalseks tööks on vajalik vähemalt 128 MB muutmälu, ekraan minimaalse eraldusega 800x600 ja *DirectX 8* sobituv graafikakaart (Ford, 2009). GameMaker ise võtab installeeritult kõvaketta peal ruumi vaid 15 MB ning sellestki moodustab enamuse programmiga vaikimisi kaasatulevad näidisfailid, mis võimaldavad tutvuda programmi loogikaga. Madalad nõudmised arvutile võimaldavad kasutada GameMakerit ka vanemates arvutites, see hulgas õppetööks üldhariduskoolide arvutiklassides.

2.2 GameMakeri ülesehitus

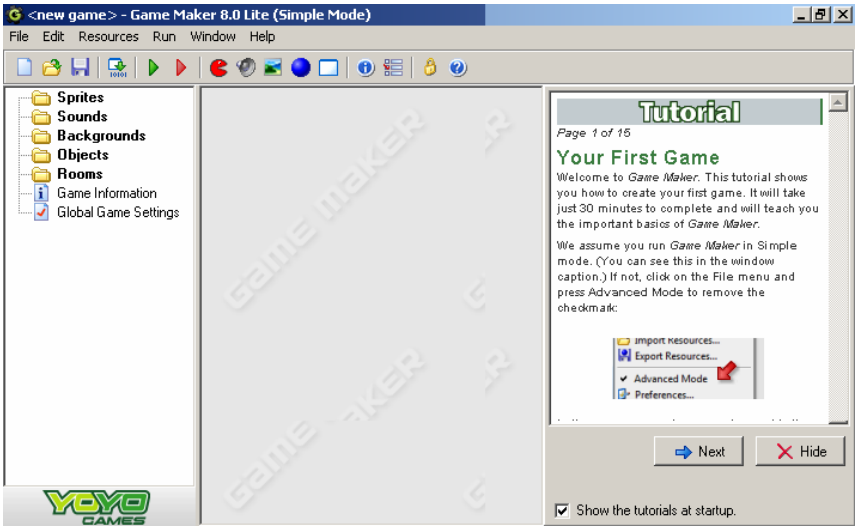
GameMaker pakub lihtsat keskkonda, mis võimaldab ka täiesti algajatel kiiresti mänge luua, selleks sisaldab programm endas juba peamisi mänguloomiseks vajalikke struktuurielemente. Enne töölaua näitamist, küsitakse esimesel käivitamisel kasutajalt üle, kas ta soovib kasutada GameMakeri algaja vaadet või kasutada edasijõudnutele mõeldud vaadet. Nende kahe vaate peamine erinevus seisneb selles, et algaja vaates on paljud valikud peidetud, mis muudab programmist ülevaate saamise lihtsamaks kuid samas võib selline osade valikute peitmine tekitada kasutajas ka segadust, eriti kui kasutada internetis leiduvaid instruktsioone. Nende kahe kuvarežiimi vahel saab igal ajal programmis hõlpsalt liikuda, seega algatuseks sobib hästi ka algaja minimalistlikum vaade.

GameMakeri töölaua ülesehitus on väga lihtne ning kergesti hoomatav - vasakul serval on kaustapuuna näha kõik erinevad kasutatavad ressursid – Sprites, Sounds, Backgrounds, Objects, Rooms ja lisaks rakenduse määrangud ja informatsiooni sisaldavad määrangud. Ekraani ülaservas võib näha standardseid failidega opereerimise nuppe – projektide salvestamiseks ja avamiseks, programmi käivitamise nuppu tavarežiimil ja silumisrežiimil ja lühitee nuppe erinevate toimingute sooritamiseks. Ekraani paremas servas on uuematel

GameMakeri versioonidel Tutorial aken, mis võimaldab samm-sammulise tegevuse tulemusena õppida programmi kasutama.

Erinevalt teistest animatsioonide tegemise programmidest nagu Flash, puuduvad siin ekraanil vabalt ujuvad aknad ning erinevate menüüde ja töölaua akende hulk on minimaalne. See kõik hõlbustab algajal programmist ülevaate saamist. Samas on võimalik ka kindel olla, et õpetuse järgi tegutsedes asetsevad kõik vajalikud vahendid alati samades kohtades.

GameMakeri mängud leiavad aset tasapinnalistes 2D-ruumides (room). Ruumidesse saab paigutada objekte, mille käitumist on võimalik programmeerida. Seega on GameMakeri kõige tähtsamad elemendid objektid. Passiivsel taustal liiguvad interaktiivsed objektid, mis reageerivad eriliiki sündmustele (events). Nendeks võivad olla näiteks kasutaja klahvivajutused, ajalised sündmused. Igale objektile on võimalik määrata pilt (sprite), millisena objekti ruumis visuaalselt kujutatakse.



Joonis 2 - GameMakeri töölaud

2.2.1 Spraidid

Spraidid (*Sprites*) on objektide visuaalsed esitlused. Spraidid on väikesed kujutised, tavaliselt piiratud värvide arvuga ja suurusega (Henno, 2009). Sprait võib olla ükskõik milline staatiline joonistus, mida saab luua erinevate joonistamisprogrammidega. GameMakerisse on sisseehitatud ka lihtsakoeline spraidi toimetamisprogramm, mis võimaldab lihtsamaid spraite programmisiseselt luua ja muuta.

Spraidid on tavaliselt läbipaistval taustal ja vaikimisi võtab GameMaker spraidi läbipaistvaks värviks spraidi vasaku alanurga piksli värvi. Spraidi võib moodustada ka mitu pilti, mis mängus väljendub animatsioonina. GameMakeri mängutegemise alustamist võikski alustada sobiva spraidi loomise või importimisega. Spraidi loomiseks on ekraani ülaservas punane otsetee nupp, mis sümboliseerib mängude ajaloo üht tuntumat mängukangelast *Pacmani*.

Igale loodavale spraidile saab anda nime, mille abil toimub kogu ülejäänud pöördumine loodava ressursi poole. Vaikimisi numereeritakse ja varustatakse kõik GameMakeri eri liiki ressursid nimetustega, kuid ülevaatlikkuse saavutamiseks oleks mõistlik anda igale spraidile oma nimi.

GameMaker toetab suurt valikut erinevaid pildiformaate ning ka pildi animatsioone. Kasutatavad formaadid on *BMP, JPEG, GIF, PNG* ja programmi enda *GMSP* formaat. Viimane neist salvestab lisaks pildile ka informatsiooni selle nullpunkti (*origin*) ja piirik-kasti (*bounding box*) kohta. *JPEG* puhul tuleb arvestada, et tegu on kadudega pakkimisega ning teravate piirjoontega piltide korral tekivad moonutused ja müra. Seetõttu on soovituslik kasutada *PNG* vormingut.

Kõiki pilte on võimalik töödelda GameMakeri sisese spraiditöötluse programmiga. Samuti saab määrata, kas spraidi taustavärv peaks olema läbipaistev või mitte. Taustavärviks võetakse vaikimisi spraidi vasaku serva kõige alumise serva piksli värv. Seetõttu tuleb kindlaks teha, et sprait ise ei koosneks taustaga sama värvi pikslitest või vastasel juhul tekivad spraidi sisse augud. Sprait iseenesest on lihtsalt pilt, ning sellega ei saa GameMakeris veel midagi teha. Mängu interaktiivsuse tagavad mängu objektid.

2.2.2 Objektid

Objektid on mängu elemendid, mis võimaldavad liikumist ja reageerimist mängusisestele sündmustele. Objektide defineerimisel määratakse tegelikult objektiklass – korraga võib olla mängus sama objekti mitu eksemplari korraga. Igal eksemplaril on enda *id*, mille kaudu pääseb ligi selle sisestele muutujatele.

GameMaker võimaldab objektorienteeritud suunitlusega programmeerimist, sest kõik kasutatavad ressursid on paigutatud olemitesse nagu objektid, spraidid ja ruumid. Objekti loomiseks on olemas kiirnupp ekraani ülaservas – sinine pall. GameMakeri loogika järgi on

kõik mängu nähtavad elemendid objektid, v.a. passiivne taust. Loomulikult eksisteerivad ka nähtamatud objektid, kuid ka need reageerivad mängus asetleidnud objektidevahelistele sündmustele.

Kui spraidid on lihtlabased pildid, siis objektid on märksa targemad mängu elemendid. Objekti kirjeldamist tuleb alustada nime andmisega. Kasulik oleks nimes kajastada ka seda, et tegu on objektiga, mitte spraidiga. Selleks võib kasutada Ungari notatsioonist tuttavat ressursi tüüpi näitavat eesliidet. Näiteks nimetus obj_maja annab selge ülevaate, millega on tegu. Seejärel tuleks valida objektile visuaalne välimus, ehk millisena objekti ekraanil kujutatakse.

Igal objektil on olemas sügavuskoordinaat (*depth*), mis määrab ära objektide ekraanile joonistamise järjekorra. Vaikimisi määratakse see kõikidel loodud objektidel nulliks. Suurema väärtusega objektid joonistatakse kõigepealt, võrdse sügavuskoordinaadiga objektid joonistatakse loomise järjekorras. Objektidel võivad olla ka vanem objektid (*parent*), millelt omadusi päritakse ja mask (*mask*). Tavaolukorras kasutatakse objektide asukohtade arvutamiseks nende spraite, kuid teatud juhtudel võib tekkida vajadus vähendada objektide reageerimisala ning selleks on võimalik kasutada maske.

Lisaks on võimalik määrata kas objekt on ekraanil nähtav (v*isible*), püsiv (*persistent*) ehk ilmub igas ruumis ja kas tegu on tahke objektiga (*solid*). Tahkeks objektiks tasuks panna ainult sellised objektid, mis on liikumatud ja millest ei tohiks teised objektid läbi minna. Siinkohal tulebki esile GameMakeri suur pluss, mis muudab sellega mängude loomise kiireks. Kui tavalises programmeerimise keeles kirjutades tuleks põrkavatele objektidele arvutada välja liikumisvektorid enne ja pärast põrkumist, siis siin teeb GameMaker selle töö ise ära. Põrked objektide vahel on üks näide objektide sündmustest (e*vents*).

2.2.3 Objektidega seotud sündmused

Iga objekti puhul on võimalik määrata rida sündmusi (*events*), millele selle objekti eksemplarid peavad reageerima – näiteks teatud klahvivajutused või kindlad ajahetked. Sündmused käivitavad erinevaid tegevusi (*actions*), mis määravad täpselt ära, kuidas ja mis järjekorras objektid käituvad kui mingi kirjeldatud sündmus peaks juhtuma. GameMaker pakub suure valiku mängudes vajaminevaid sündmusi:

- ***Create*** – sündmus, mis käivitub hetkel, kui objekt luuakse, tavaliselt kasutatakse objektile liikumissuuna või atribuutide andmiseks.
- ***Destroy*** – sündmus, mis käivitub vahetult enne seda kui objekti eksemplar eemaldatakse.
- ***Alarm*** – GameMaker pakub igale objektile kuni 12 taimerit, mis võimaldab luua sündmusi, mis käivituvad teatud aja jooksul. Kui kuskil mõne sündmuse peale või mängu käivitamisel pannakse taimer tööle, siis hetkel millal aeg saab täis, käivitatakse see sündmus.
- ***Step*** – siia alla kuuluvad kõik sündmused, mis vajavad jooksvat tegevust, tihti kasutatakse seda kontrollimaks mõne muutuja väärtust. Sammuks (*step*) loetakse iga hetke kui ekraan üle joonistatakse. Seega on tegemist põhimõtteliselt animatsiooniprogrammidest tuttava mõistega kaader (*frame*). Vaikimisi on ühes sekundis 30 sammu, kuid seda saab ruumi määrangutes muuta. Kuna see sündmus leiab aset pidevalt igal sammul siis ei maksa siia alla lisada ressursinõudlikke tegevusi, mis võiks mängu sujuvust häirima hakata. Valides selle sündmuse, palutakse määrata täpsemalt, kas sündmus peaks aset leidma enne kõiki teisi sündmusi (b*egin step*), vahetult enne objektile uue asukoha arvutamist (s*tep*) või pärast seda (e*nd step*).
- ***Collision*** – selle sündmuse alla kuuluvad kõik objektidevahelised põrked. Iga objekt võib käituda eri objektide kokkupõrkel erinevalt. Arvestada tuleb ka sellega, et põrkeid arvutatakse spraidi loomise juures määratud piirik-kasti abil.
- ***Keyboard*** – annab võimaluse lugeda kasutaja klaviatuurilt tulevaid klahvivajutusi. Lisaks konkreetsetele klahvidele saab määrata ka <*No key*> - kui midagi ei vajutata ja <*Any Key*> - kui kasutaja vajutab ükskõik millist klahvi. Sündmusega seotud tegevused käivitatakse igal sammul (*step*) alates klahvi allavajutamise hetkest kuni klahvi vabastamiseni.
- ***Key Press*** – klahvi vajutamise sündmus, mis erineb valikust *Keyboard* selle poolest, et reageeritakse ainult klahvi allavajutamise hetkel. Seega ei saa siia alla panna pidevat allhoidmist vajavaid tegevusi – näiteks liikumise klahvid.
- ***Key Release*** – sündmus, mis leiab aset ainult siis kui teatud allhoitud klahv vabastatakse.
- ***Mouse*** – tegevused, mis toimuvad kui hiire kursor asetseb hetkel vaadeldava objekti peal.

- ***Drawing*** – sündmused, mis käivitatakse igal sammul (*step*), kui ekraanil olevad elemendid üle joonistatakse. Selle sündmuse abil saab muuta spraitide välimust ja parameetreid. Samuti kirjutatakse kõik tekstid GameMakeris alati ekraani ülejoonistamisel.
- ***Other*** – Siia alla on koondatud kõik teised võimalikud sündmused, mis on peamiselt seotud objektide ja ruumidega. Siin on ka mängu alguses ja lõpus ning ruumi alguses ja lõpus toimuvad sündmused, mille abil saab luua omanäolisi üleminekuid ühest ruumist teise.

Kõiki GameMakeri sündmusi kontrollitakse ja nendega seotud tegevusi tehakse tsükliliselt kindlas järjekorras igal sammul:

- Sammu alguse sündmused (*Begin Step*)
- Taimerid (*Alarm*)
- Klaviatuuri sündmused (*Keyboard Events*)
- Hiire sündmused (*Mouse Events*)
- Sammu sündmused (*Step)* - objektidele arvutatakse siin uued asukohad
- Põrked (*Collisions)* – kontrollitakse kas uutes asukohtades tekkis põrkeid
- Sammu lõpu sündmused (*End Step*)
- Joonistamise sündmused (*Drawing*) – joonistakse ekraan üle

2.2.4 Objektidega seotud võimalikud tegevused

Iga sündmuse juurde saab määrata rea tegevusi, mida sündmuse tekkimise korral täitma asutakse. Sündmused koos tegevustega määravad ära objektide käitumise. Tegevuste (a*ction*) lisamiseks on võimalik kasutada GameMakeri pukseerimisel põhinevat keskkonda, kus tuleb soovitud tegevus lihtsalt lohistada tegevuste aknasse. Tegevused on grupeeritud ülevaatlikkuse mõttes erinevatele vahelehtedele, mis hõlbustab algajal vajalike tegevuste leidmist. Erinevaid vahelehti on kokku seitse:

Move – siia on grupeeritud kõik objektide liikumisega seotud tegevused. Kõikide liikumiste puhul tuleb määrata objektile kiirus (*speed*) ehk mitme piksli võrra liigub objekt ühe sammu jooksul. Vaikimisi on ühes sekundis 30 sammu, kuid seda saab ruumi määrangutes muuta. Samuti tuleb liikumiste puhul alati määrata, kas kiirus on konstantne või relatiivne (*relative*).

Main1 – funktsioonid objektide muutmiseks, loomiseks.

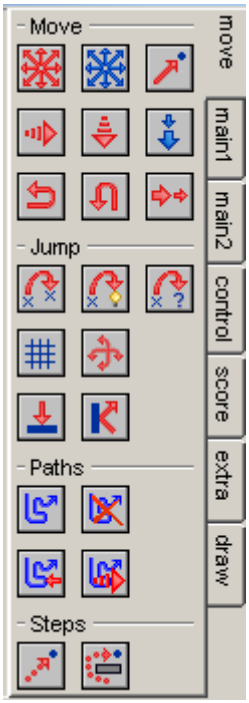
Main2 – taimerid, ajajooned, mängu salvestamise, laadimise ja lõpetamise funktsioonid.

Control – siia alla kuuluvad kõik tingimusi sisaldavad funktsioonid.

Score – funktsioonid, mis on seotud GameMakeri sisese punktide arvutamise süsteemi, tervisenäidiku (*health bar*), elude ning edetabeliga.

Extra – siit leiab osakeste animeerimise süsteemi (*particle animation*), mis võimaldab luua animeeritud elutruud suitsu, tuld, plahvatusi ja lumesadu. Lisaks on võimalik kontrollida CD-plaadi olemasolu arvuti kettaseadmes, seda ettemängida ning muuta hiirekursori välimust. (kõik tegevused saadaval ainult tasulises versioonis)

Draw – joonistamisega seotud käsud, võimalus kasutada järkjärgulisi värviüleminekuid (*gradient fill*), kujundada tekste skaleeritud teksti (*scaled text*) abil ja muuta objekti spraidi välimust. (värviüleminekud on saadaval ainult tasulises versioonis)



Joonis 3 - Actions

2.2.5 Ruumid

GameMakeri mõttes on ruum (*room*) väga tähtis asi. Ruum on koht, kus kõik tegevused toimuvad – seega sarnaneb see mõnes mõttes animatsiooniprogrammidest tuttava mõistega – lava (*stage*). Ruume võib kutsuda ka mängu tasemeteks. GameMakeri puhul on omaette ruumid tavaliselt ka kõik erinevad menüüd ja vaheklipid.

Igale ruumile on võimalik määrata selle suurus pikslites, tausta värvus või taustapilt ja kiirus ehk siis mitu korda sekundis ruumisisu uuendatakse. Ruumi kiirus määrab ühtlasi ka mängu kiiruse, kuna kõik objektide liikumised on seotud sellega. Mida suurem see on, seda sujuvam on liikumine. Ruumi puhul on oluline ka selle suurus, vaikimisi pakub GameMaker ruumi suuruseks 640x480 pikslit. Kuna GameMakeri näol on tegemist rastergraafilise programmiga, siis on sobiva ekraanilahutuse valimine väga oluline. Tänapäeval on ekraanide lahutusvõime paranenud drastiliselt ja seetõttu soovitatakse võtta aluseks juba XGA graafikastandard, milleks on 1024x768 pikslit (Overmars, 2011). Ruumi menüüs saab panna igale ruumile taustapildi (*background*) ja lisada eelnevalt loodud objektid.

Objektide paigutamist hõlbustab muudetava tihedusega taustavõrk (*grid*), seetõttu peaksid objektide spraidid olema alati standardse suurusega – enamasti arvu 2 kordsed. Nagu enamikes graafikaprogrammides loetakse ka GameMakeris koordinaatide alguspunktiks ekraani vasak ülemine nurk. Peale koordinaatidega määratud asukoha on siin objektidel olemas ka suund (*direction*) kraadides. Selle abil on võimalik objekte keerata. 0-suunaks loetakse GameMakeris horisontaalselt vasakult paremale ning see suureneb vastupäeva.

Lisaks saab ruumide all paika seada erinevad vaated (*view*). Mitmete mängutüüpide (enamasti platvormmängud) puhul kasutatakse võrdlemisi suuri mänguruume, kuid mängijale näidatakse ekraani peal vaid hetkel aktiivset osa, kus põhitegevus toimub. GameMakerisse on vastav oskus sisseehitatud ning seega tuleb paika panna vaid vaateakna mõõdud pikslites ning valida vastav objekt, mille liikumise suhtes mänguväli määratakse. Määrata saab ka selle kui lähedale tohib objekt ekraani servale jõuda enne kui vaade liikumisega kaasa tuleb. Erinevad vaated võimaldavad hõlpsalt luua ka jagatud mänguekraaniga (*split screen*) mitmikmänge.

2.2.6 Ajajoon

Paljudes mängudes juhtuvad teatud sündmused kindlatel ajahetkedel. Selle saavutamiseks võib GameMakeris kasutada varem vaadeldud objektide sündmust *Alarm*, kuid teatud juhtudel võib selline lähenemine muutuda keeruliseks. Ajajoon võimaldab kirjeldada objektide omadusi, liikumist ajas. Seega võimaldab ajajoon määrata hõlpsalt teatud kindlatel hetkedel toimuvaid sündmusi. Ajaarvestamise aluseks on võetud jällegi sammud (*steps*). Lisaks on võimalik tegevuste vahelist aega vähendada või suurendada, tehes mängu kiiremaks või vastupidi.

2.2.7 Helid ja efektid

Mängu interaktiivsuse tõstmiseks saab kasutada ka erinevaid helisid. Helide abil on võimalik anda mängijale informatsiooni tema tegevuste kohta nagu ka piltide ja tekstide abil. GameMaker toetab *WAV* ja *MID* formaate. Importida saab ka *MP3* faile, kuid nende lahtipakkimiseks ja taasesitamiseks kasutatakse välist meediamängijat. Seetõttu on *MP3* võrdlemisi piiratud võimalus ning hoolimata väiksemast failimahust võib selle töötlemine GameMakeris rohkelt ressurssi nõuda. Ka mahukad *WAV* failid võivad avaldada mõju mängu sujuvusele ja loomulikult ka valmismängu mahule. Seetõttu on oluline leida optimaalne lahendus.

Üldiselt kasutatakse taustamuusikaks enamasti MIDI faile nende puhul tuleb arvestada, et korraga saab mängima panna ainult ühe MIDI faili. Heliefektid võiks olla *WAV* formaadis ning bitimäär tuleks valida vastavalt efekti sisule. Erinevalt sisseehitatud spraidiprogrammist, puudub GameMakeril helitöötlus võimalus. Seega tuleks kõik helid töödelda juba enne nende importimist. Pakutakse ka võimalust määrata väline helitöötlusprogramm, milles redigeeritav heli avatakse ning programmi sulgemisel naastakse uuesti GameMakeri aknasse.

GameMakeri tasuline versioon pakub lisaks mõningaid heliefekte: kooriefekt (*chorus*), kaja (*echo*), toruefekt (*flanger*), kaikumine (r*everb*), kuristamisefekt (*gargle*). Lisaks saab määrata helitugevust (*volume*), kordamist (*loop*) ning panoraamida (*pan*) heli stereokanalite vahel. Võimalikud on ka kõik kombinatsioonid efektidest. Hoolimata tasulise versiooni võimalustest on targem siiski panna helid õigesti kõlama mõne helitöötlusprogrammiga, mis võimaldab lisaks efektidele ka heli lõigata ja dünaamikat seada. Eriti tähtis on see taustamuusika puhul, mida tavaliselt korratakse terve mängu vältel.

Iga heli puhul saab ka määrata, kas see laetakse mängu käivitamisel mällu või laetakse heliklipp vahetult enne taasesitamist. Viimasel juhul säästetakse küll mälu, kuid laadimise hetkel võivad ilmneda viivitused ning mängu sujuvus võib selle all kannatada. Algajal mänguloojal võib muidugi tekkida kiusatus kasutada oma rakenduses tuntud heliteosed, sest hea taustamuusika, mis toetaks mängu kuid samas ei hakkaks häirima, loomine on keeruline. Siiski leiab internetist ka legaalselt jagatavaid heliklippe, mida tohib vabalt kasutada.

2.3 GameMaker Language (GML)

GameMaker sisaldab sisseehitatud programmeerimise keelt, mis tunduvalt suurendab programmi võimalusi ja paindlikkust objektidega ümberkäimisel. Kõiki tegevusi, mida saab teha GameMakeris pukseerimise abil, on võimalik teostada skriptikeeles. See võimaldab hõlpsalt oskuste paranemisel liikuda samm-sammult programmeerimise juurde. Keele süntaks sarnaneb tugevalt C keele omale, kuid esineb ka teatavaid erinevusi, näiteks saavad massiivid olla kuni kahemõõtmelised ning ei saa sisaldada teisi massiive. Tasuline versioon lisab veel andmestruktuure – pinu *(stack)*, riviloend (*queue)*, loend *(list)*, võrk (*grid)* ja struktuurskeem *(map)*.

Funktsioonide rolli täidavad skriptid, milledele saab kaasa anda argumente. Valmis skripte saab käivitada objektide sündmuste abil. Lisaks on võimalik käivitada neid iga uue ruumi

käivitumise korral. *GML* skriptide puhul tuleb muidugi arvestada sellega, et kõik objektide nimed peavad olema unikaalsed ning ei tohi sisaldada keelatud sümboleid, nagu näiteks punkti või nimetusi *self*, *other*, *global*. Sellepärast on tark anda objektidele alati korrektsed ja ülevaatlikud nimed. Seeläbi õpetab GameMakeri kasutamine mõistma kui oluline on muutujate korralik nimetamine koodilugemise seisukohast.

GML ülesehitus on võrdlemisi lihtne ning olemasolevaid funktsioone on palju. Neil on lihtsad ning kergesti meeldejäävad nimetused. Erinevaid muutujatüüpe eksisteerib ainult kaks – arvud (*real*) ja sõned (*string*). Iga muutuja võib sisaldada vastavat tüüpi väärtust, selleks eraldi muutuja tüüpi deklareerima ei pea. Samas võib selline lähenemine soodustada mitmesuguste halbade programmeerimisvõtete kasutamist, sest GML-laadsest kergest stiilist on raske loobuda.

Eksisteerimispiirkonna ulatuse järgi jagatakse loodud muutujaid lokaalseteks ja globaalseteks Lokaalsete muutujatega tegelemiseks tuleb kõigepealt pöörduda objekti poole. Lokaalsed muutujad eksisteerivad üldjuhul ainult seni, kuni täidetakse skripti. Lisaks on igal objektil kindlad ettemääratud lokaalsed muutujad, nagu koordinaadid – *x*, *y*, liikumiskiiruse komponendid - *hspeed*, *vspeed*, liikumissuund - *direction*. Globaalsed muutujad eksisteerivad kogu programmi töötamise ajal (kõikides ruumides), nad on kätte saadavad tervest programmist.

Nagu tavalistes programmeerimise keeltes on siingi võimalik kasutada erinevaid tingimuslauseid, tsükleid ja jadasid.

```
{
    var i, j;
    for (i=0; i<10; i+=1){
        for (j=0; j<8; j+=1){
            instance_create(40*i, 40*j,obj_muna);
            }
        }
}
```

Joonis 4 - GML skript objektide loomiseks

2.4 GameMakeri kompileeritud väljund

GameMakeris loodud rakendused on võimalik kompileerida *EXE* failiks, mida on kerge edastada sõpradele ja tuttavatele, et oma loomingut näidata. Samas salvestatakse kõik *GML* skriptid ja ressursid lihtsalt ühte faili, mis muudab võimalikuks nende dekompileerimise. Ühelt poolt muudab see võimalikuks teiste valmismängudest õppimise kuidas üks või teine asi on lahendatud, samas tõstatab see jällegi autorluse probleemi. Väljundile saab luua ka enda ikooni ja laadimisakna. Loodud valmismänge on võimalik levitada YoYoGamesi portaali kaudu. Selleks tuleb luua vaid endale kasutajakonto.

2.5 GameMaker:HTML5 ja Studio

2011. aasta sügisel avalikustas YoYoGames GameMaker:HTML5 beta versiooni, mida sai soetada 79$ eest. Täisversiooni hind on kasvanud 99$. GM:HTML5 võimaldab luua valmis HTML ja JavaScripti koodi, mida on võimalik kasutada kõikides HTML5 standardit toetavates veebilehitsejates. Lisaks HTML5 mängudele on võimalus luua ka tavapäraseid Windowsi töölauarakendusi EXE formaadis. Võimalik on importida ka GM 8.1 versiooniga loodud projekte.

Esimese asjana hakkab silma GM:HTML5 värvilahendus, mis erineb kardinaalselt tavaversioonist. Heledad toonid on asendunud tumedatega. Funktsionaalsus on väga sarnane Game Makeri tavaversiooniga. Valmis mängu eksportimisel luuakse HTML fail koos HTML5 canvas elemendiga, kaust rakenduses kasutatavate failide hoidmiseks ning vastav JavaScripti fail. GameMaker:Studio, mis on loodud mitmeplatvormiliste rakenduste loomiseks laiendab võimalusi veelgi. Praeguseks on YoYoGames ametlikult teatanud iOS, Android ja Nokia Symbian operatsioonisüsteemide toetamisest.

Lühikese ajaga on GameMakeriga loodud HTML5 mängud pälvinud palju tähelepanu. Inglismaa populaarne videomängu ajakiri „Pocket Gamer“ valis 2012. aasta parimaks Android platformi peal töötavaks mänguks Karoshi. Mängu autoriks on hollandlane Jesse Venbrux, kes lõi mängu kasutades GameMakeri vahendeid. Tegemist on Super Mario laadse platform mänguga, mille eesmärki on muudetud. Autori sõnul on peategelase ellujäämine, printsesside päästmine ja kangelaseks olemine igavaks muutunud. Seega tema mängu kontseptsioon on vastupidine.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli anda ülevaade programmist GameMaker ja hinnata selle sobilikkust kasutajale, kellel puudub varasem programmeerimise kogemus. Töö teoreetilises osas tutvustati videomängudele ja nende loomisele esitatud peamisi nõudeid. Töö edasises osas vaadeldi GameMakeri sobivust nõuetele.

GameMaker annab väga hea aluspõhja programmeerimisele, õpetades mõistma sellega seonduvaid peamisi teadmisi. Teisalt pakub GameMaker võimalust omandada vajalikud teadmised justkui muuseas ning võimaldab algajal hõlpsalt improviseerida, et näha, mis juhtub siis kui teha teisiti. Sama oluline on ka mõista seda, et soovitud tulemuse saavutamiseks on enamasti mitmeid võimalusi ja neist tuleb lihtsalt vastavalt olukorrale optimaalsem ja lihtsam leida.

GameMakeri ülesehitus on ülevaatlik ja loogiline, puuduvad teistes animatsiooniprogrammides leiduvad vabalt ujuvad aknad, mis võivad algaja kasutaja segadusse viia. GameMaker võimaldab luua esimesed mängud ilma otsese programmeerimiseta, selleks valmis tegevusi plokkide kaupa sündmuste alla pukseerides. Selline lähenemine annab võimaluse mõista sündmuste ja nende toimumisele vastavate reageeringute tähtsust. Samuti õpetab see järjestama tegevusi õiges järjekorras soovitud tulemuse saavutamiseks, mis omakorda õpetab hindama algoritmide efektiivsust. Vastavalt kasutaja oskuste ja teadmise kasvamisega võimaldab GameMaker liikuda samm-sammult üha enam programmeerimise juurde. See aitab algajal kasutajal kiiresti areneda ja jooksvalt hakata üha rohkem kasutama GML skripte.

Valmisloodud rakenduste levitamise võimalused on head. GameMaker võimaldab luua exe laiendiga täitefaile, mida on lihtne teistega jagada ning mingit eritarkvara nende käivitamiseks vaja ei ole. See aitab saada kohest tagasisidet oma rakenduste kohta, mis omakorda motiveerib looma uusi ja paremaid mänge. Seoses GameMaker:HTML5 ja Studio ilmumisega on oodata võimaluste laienemist veelgi. Enam ei pea piirduma vaid töölaua rakenduste loomisega. Saab luua ka veebiplatvormil ja pihuseadmetel mängitavaid mänge ja selleks kasutada sama tuttavat keskkonda.

Negatiivse poole pealt tuleb välja tuua asjaolu, et lihtsustatud kujul programmeerimine GML skriptide näol võib soodustada halbade programmeerimise harjumuste tekkimist, näiteks

muutujate deklareerimata jätmist. Lisaks tuleb arvestada, et GameMakeri valmisrakendusi on võimalik internetis leiduvate programmide abil hõlpsalt dekompileerida, mis tähendab, et GameMakeriga loodud teos ei ole eriti kaitstud lähtekoodi varguse eest.

Positiivse poole pealt tuleb mainida, et GameMaker on tänapäevaks küllaltki populaarseks muutunud ning internetist leiab väga palju kasulikke näpunäiteid ja koodinäiteid iseõppimiseks. Lisaks kõigele, on GameMakerist saadaval ka tasuta versioon, mille võimalused on piiratud vaid mõningal määral, mis ei takista põhilisemate toimingute sooritamist.

Summary

The purpose of this thesis was to make an overview of GameMaker and to check its suitability for a beginner programmer without previous programming experiences. To meet the objective as stated, the author outlines the requirements for which a beginner friendly program should accord to.

GameMaker is a program, written by Mark Overmars, who is a professor at the Institute of Information and Computing Sciences at Utrechts University in the Netherlands. It is available for Windows and Macintosh. GameMaker allows to create exciting computer games, without the need to write a single line of code. For this purpose GameMaker offers an easy drag-and-drop actions system in an intuitive visual interface.

The components in GameMaker are given intuitive names that a beginner can easily grasp: sprites, objects, rooms, sounds. Areas in which a game is played are called rooms, anything inside the room is an object. Sprites are images that can be used as an object's image or mask. GameMaker has an built-in sprite editor, which offers a good deal of control over them, including scaling, rotating, making transparent. GameMaker doesn't have a built-in sound editor, but the full version allows to add some special 3D sound effects, which helps to make games even more great looking.

Using the drag-and-drop interface, it is easy to add events and corresponding actions to an object. It allows to easily understand how the object reacts to an event. Advanced users are allowed to create more complex applications with GameMaker's built-in scripting language – GameMaker Language (GML). Users don't have to worry about declaring data types or bothering with code libraries, which makes programming easy, but can also lead to bad programming practices. All the functions are simple and easy to remember.

GameMaker:HTML5 enables to produce ready-to-run HTML and JavaScript code, allowing games created with GameMaker to be played in any modern browser with HTML5 support. GameMaker:Studio is the perfect development tool for creating cross-platform applications. Currently available platforms include: iOS, Android phones and tablets and Nokia Symbian.

GameMaker is an easy way to start with creating applications. It also has a free version, which is limited in its functionality, but is still appropriate enough to start creating computer games and learning the basics of programming.

Kasutatud kirjandus

Anderson, J. (1983). Who really invented the video game?. *Creative Computing Video and Arcade Games*, 1, 8.

Entertainment Software Association. (2011). *Essential facts about the computer and video game industry*. URL http://www.theesa.com/facts/pdfs/ESA_EF_2011.pdf (viimati loetud 08.04.2012).

Ford, J. L. (2010). *Getting Started with Game Maker*. Boston: Course Technology.

Habgood, J., Nielsen, N., Rijks, M. (2010). *The Game Maker's Companion*. Berkeley: Apress.

Habgood, J., Overmars, M. (2006). *The Game Maker's Apprentice: Game Development for Beginners*. Berkeley: Apress.

Henno, J. (2009). *Mängude programmeerimine*. Tallinn: TTÜ Kirjastus.

Hjorth, L. (2011). *Games and gaming: An introduction to new media*. New York: Berg.

Newman, J. (2004). *Videogames*. New York: Routledge.

Overmars, M. (2011). *Designing successful iPhone and Android Games with GameMaker*. URL http://www.yoyogames.com/docs/designing_successful_iphone_games.pdf (viimati loetud 06.04.2012).

Pocket Gamer Awards 2012. URL http://www.pocketgamer.co.uk/pgawards2012.asp (viimati loetud 22.03.2012).

Randviir, A. (2011). Videomängude esteetika. *Kunst.ee,* 3-4, 28-33.

LISAD

Lisa 1. Näidismängu loomine GameMakeri abil

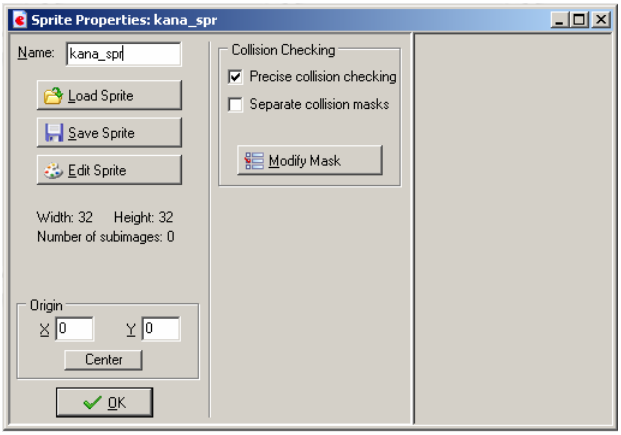
Järgnevas näites luuakse GameMakeri tasuta versiooni võimalusi kasutades lihtne mäng, mis võimaldab tutvuda GameMakeris kasutatavate ressurssidega ning selle üldloogikaga. Loodava mängu peategelaseks on kana, kes on mängija poolt juhitav peategelane. Iga teatud aja tagant muneb kana ühe muna, millest koorub tibu. Esmalt luuakse kasutaja poolt juhitav peategelane kana, kelle liikumine on mängu pildilise ilu suurendamiseks animeeritud.

Animeeritud peategelase loomine

Animatsiooni ehk liikuva pildi mulje loovad kiiresti vahelduvad pildid, millel on eelnevaga võrreldes väike muutus. Selle tegemine GameMakeris käib sarnaselt teiste animatsiooniprogrammidega. Järgnevalt luuakse nelja kaadriga animatsioon liikuvast kanast.

1. Peategelase loomiseks tuleb vajutada *Create a sprite* nupule. 

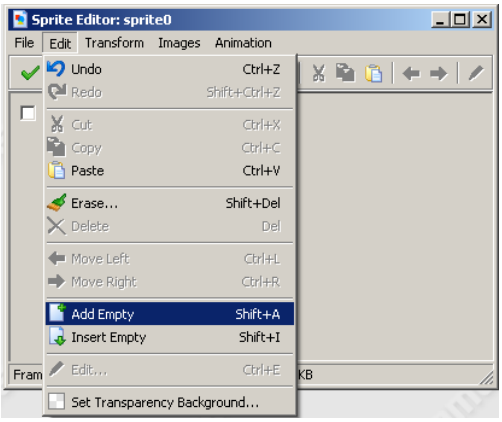
2. Loodavale spraidile tuleb anda nimi (*name*), näiteks **kana_spr**. Vältida tuleks nimedes tühikuid ja punkte, selle asemel võiks kasutada alakriipsu. Samuti ei tohiks nimetus alata numbriga.



Joonis 5 - Spraidi loomise aken

3. Kui peategelane on eelnevalt mõnes pilditöötlusprogrammis valmis joonistatud, siis on võimalik see importida nupu *Load Sprite* abil. Täiesti uue loomiseks tuleb valida *Edit Sprite.*

4. Avanenud spraidi redigeerimise aknas valida *Edit –> Add Empty* ning anda ette loodava spraidi suurus pikslites, näiteks 32x32 pikslit. Standardsete suurustega spraitidega on pärast lihtsam edasi toimetada.

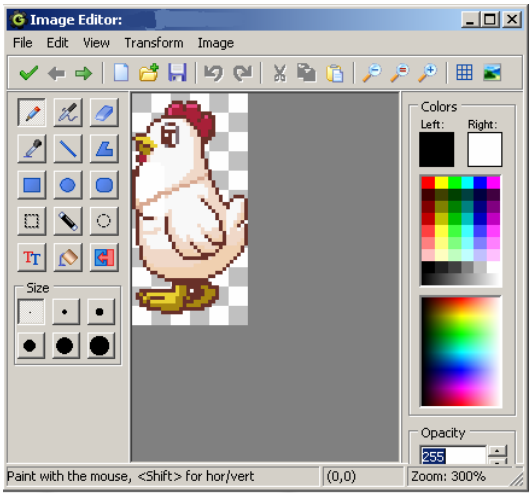


Joonis 6 - Tühja kaadri loomine

5. Selle tulemusena ilmub *image0*, mis ongi spraidi esimene kaader, hetkel on see täiesti läbipaistev.

Joonis 7 - Tühi kaader

6. Tehes topeltklõpsu *image0* peal, avaneb *MS Painti* sarnane joonistamisaken, kuhu saab joonistada animatsiooni esimese kaadri.

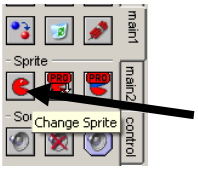


Joonis 8 - Spraidi redigeerimise aken

Hea oleks joonistada nii peategelase külg- kui ka eest- ja tagantvaade. Külgmist vaadet on võimalik hõlpsalt peegeldada (*Transform -> Mirror*) teisesuunaliseks. Lõpptulemusena peab igas neljas suunas liikumise jaoks olema oma animeeritud sprait – **spr_kana_yles, spr_kana_alla, spr_kana_v, spr_kana_p**. Lihtsamal juhul piisab animatsiooniks näiteks neljast pildist, mis annavad edasi peategelase liikumise.

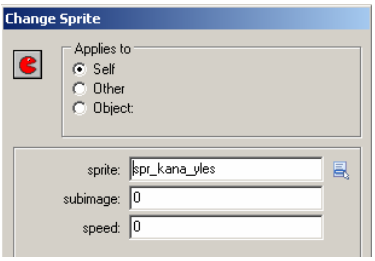
Joonis 9 - Animeeritud kana

7. Järgnevalt tuleks luua objekt, milleks on kana ja spraidiks on eelnevalt loodud animatsioon. Selleks valida *Create an object*  ja spraidiks valida **spr_kana_yles**. Vaikimisi kuvab GameMaker kõik spraidid, mis sisaldavad mitut pilti animatsioonina. Kuna praegusel juhul on animeeritud kana liikumist, siis kohapeal seistes ei tohiks animatsioon veel käivituda. Selleks tuleb lisada loodud objektile juurde tegevus, mis peatakse animatsiooni. Kuna animatsioon tuleks peatada kohe mängu alguses, sobib selleks loomissündmus (*create*), mis leiab aset hetkel kui objekt ekraanile esimest korda joonistatakse. *Create* sündmuse juurde tuleb *main1* vahelehelt lohistada *Change Sprite* funktsioon.



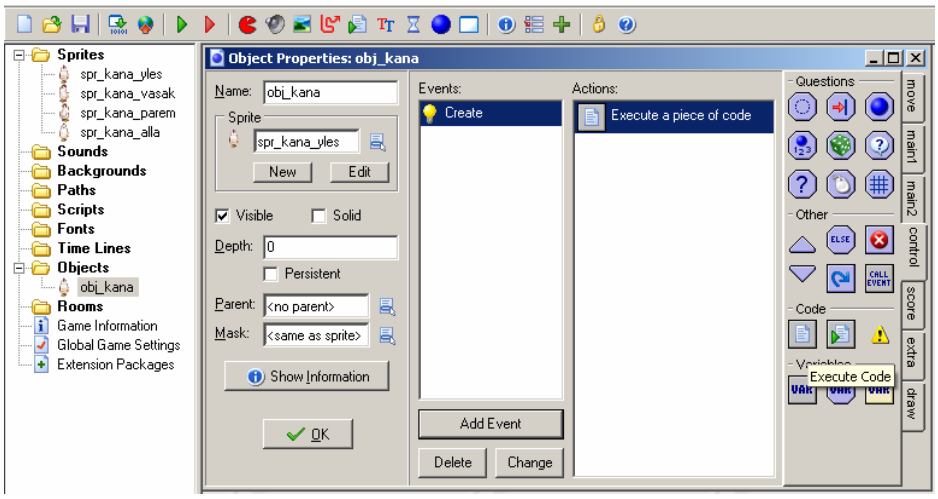
Joonis 10 - Change Sprite

Selle peale avaneb uus aken, kus saab määrata, millise objekti spraiti muutma hakatakse. Kuna hetkel sai loodud see funktsioon objekti kana *Create* sündmuse juurde, siis sobib selleks määrang *Self*. Enamuste tegevuste korral on võimalik valida kas tegevus käib objekti enda (*self*), sündmuses osaleva teise objekti kohta (*other*) või hoopis mõne kolmanda objekti kohta (*object*), mis sündmuses ei osale. Viimasel juhul saab määrata konkreetse objekti. Lisaks tuleb valida rippmenüüst sprait ja määrata ära mitmendat kaadrit animatsioonist näidatakse (*subimage*) ning animatsiooni kiirus (*speed*).



Joonis 11 - Spraidi animatsiooni kiiruse määramine

Edasijõudnud kasutajad võivad muidugi kasutada ka programmeerimise võimalust. Selleks tuleks *Create* sündmuse juurde lohistada tegevuste rühmast *control* reageering *Execute a piece of code*. Selle sisuks tuleks kirjutada - *image_speed=0;*



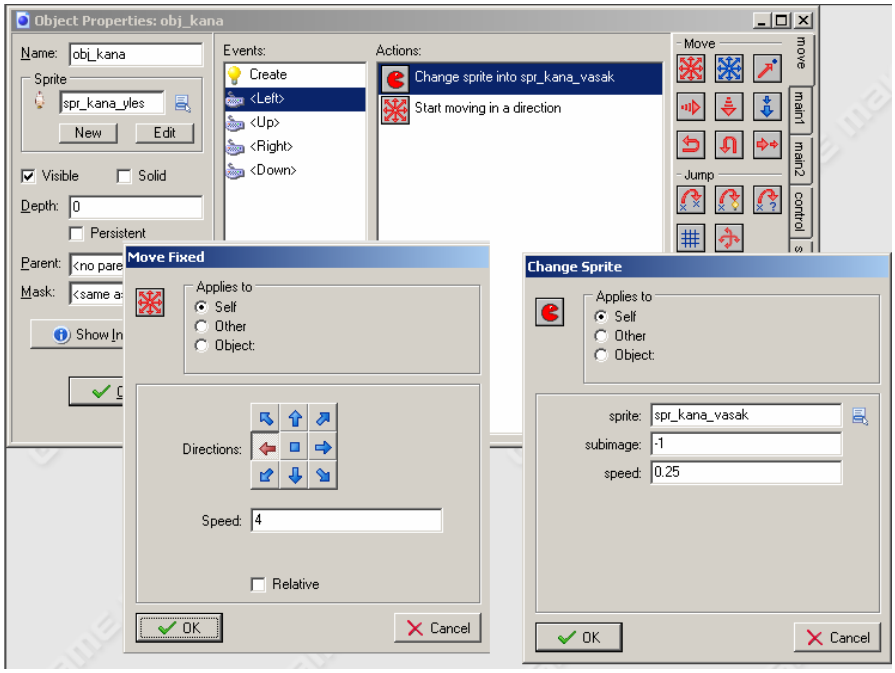
Joonis 12 - Objekt kana Create sündmus kasutades GML võimalusi

Image_speed on üks olemasolevaid objekti muutujaid, mis määrab ära kui kiiresti spraidi eri pilte kuvatakse ehk animatsiooni kiiruse. Nulli puhul näidatakse ainult animatsiooni esimest kaadrit ehk liikumist ei toimu. Tulemus on nii skripti kui ka *Change Sprite* tegevuse lohistamise korral sama.

8. Järgmisena tuleks lisada objektile liikumine. Lihtsamal juhul võib kana liigutamiseks lisada klaviatuuri sündmused nooleklahvide puhul. Liikumise korral tuleb ära määrata objekti liikumise kiirus (*speed)* ja suund (*direction*), kuhu poole objekt liikuma hakkab. Selleks tuleb lisada iga nooleklahvi sündmuse juurde kaks tegevust – *Change Sprite* ja *Move Fixed.*

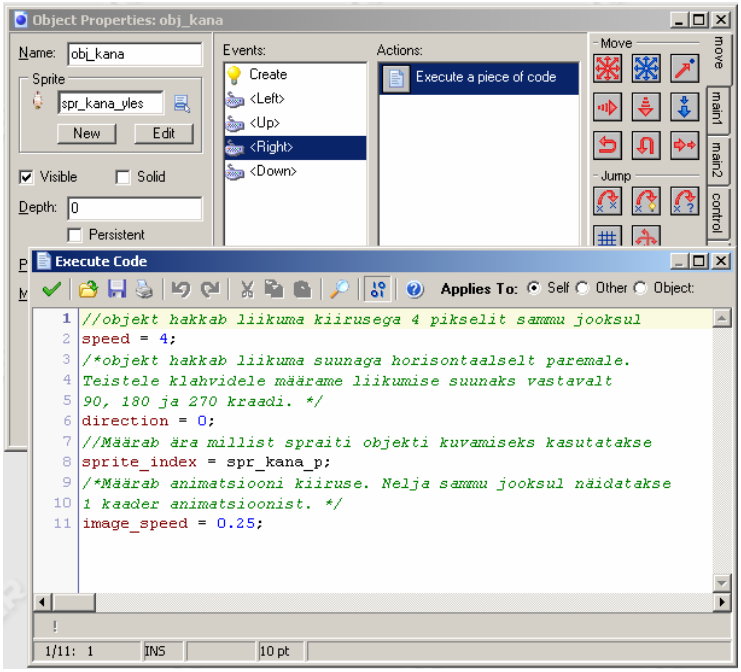
Change Sprite võimaldab hõlpsalt juhtida spraidi animatsiooni. Selleks tuleb ette anda soovitav õigesuunaline sprait ja animatsioonikiirus (*speed*), mis on seotud GameMakeris aja aluseks olevate sammudega (*step*). Väärtus 0.25 määrab ära, et 4 sammu jooksul näidatakse üks kaader animatsioonist. Sedasi saab animatsiooni kas kiiremaks või aeglasemaks muuta. Väärtis *subimage* defineerib mitmendat spraidi kaadrit animatsioonist esimesena näidatakse. Vaikimisi on selle väärtuseks 0 ehk animatsiooni esimene kaader. Probleemiks osutub siinkohal see, et klahvivajutuse sündmused leiavad aset igal sammul ja seega praegusel juhul näidatakse igal sammul esimest kaadrit ja animatsiooni ei toimu. Lahenduseks oleks kirjutada *subimage* väärtuseks *image_index*, mis on üks GameMakeri paljudest juba olemasolevatest muutujatest, mis näitab ära millist spraidi kaadrit tuleks kuvada. Tänu sellele ei minda igal sammul enam uuesti esimese kaadri juurde, vaid animatsiooni mängitakse selle algusest lõpuni ja seejärel hakatakse seda kordama.

Move Fixed nõuab esiteks liikumise suuna (*direction*) määramist, selleks tuleks vajutada õige suunaga klahvi peale. Võimalik on valida ka mitu suunda korraga, sellisel juhul valitakse liikuma hakkamise suund juhuslikult valitud suundadest. Niimoodi on kerge luua arvutipoolt juhitavaid tegelasi, kes liiguvad juhuslikult ruumis ringi. Antud näites tuleks valida ainult üks suund vastavalt sellele, millise nooleklahvi sündmusega on tegemist. Lisaks tuleb määrata kiirus (*speed*), ehk mitu pikslit objekt ühe sammu jooksul vastavas suunas liigub. Sellised tegevused tuleb lisada kõikide nooleklahvide sündmuste juurde ainult vastavate muudatustega suundades. Selle töö lihtsustamiseks on võimalik teha valmistehtud sündmuse peal paremklõps ning valida *Duplicate Event*, mis hõlbustab sarnaste tegevuste sisestamist.



Joonis 13 - Liikumise lisamine

Sama tulemus on võimalik lihtsalt saavutada ka GML skripti kasutades. Selleks tuleb iga nooleklahvi vajutuse korral käivitada alljärgnev GML skript vastavalt väikeste muudatustega liikumise suunas ja vastavasuunalise spraidi valikus. Lõpptulemusena peaks koodikasutamise korral loodav objekt välja nägema selline:



Joonis 14 – Objektiga kana seotud sündmused kasutades GML skripti

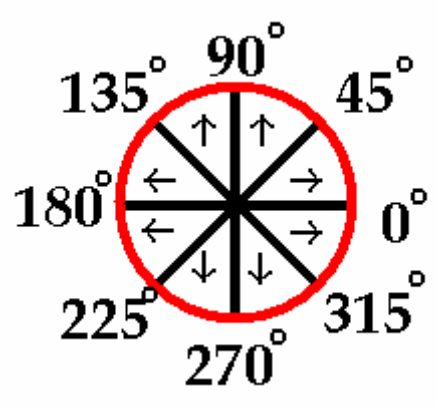
9.Viimase tegevusena jääb üle vaid luua ruum ja objekt sinna asetada. Selleks tuleb valida *Create a room*  ning asetada sinna loodud objekt. Tulemuse vaatamiseks on üleval asuv roheline käivitusnupp *Run the game*  , mis käivitab mängu.

Objekti liikumise täiustamine

Näites loodud kana liigub klahvivajutuse peale, see tähendab, et objekt saab liikuda ainult horisontaalselt ja vertikaalselt. Järgnevalt lisatakse objektile lisaks ka hiirega juhtimise võimalus. Selleks on võimalik kasutada ka ainult GameMakeri pukseerimisel põhinevat keskkonda, kuid töö autori hinnangul läheb taotletava eesmärgi saavutamine vaid GameMakeri valmisprogrammeeritud tegevusi kasutades ebaülevaatlikuks, võimaldamata näidata programmeerimise võimalusi.

Hiire kursoriga juhtimiseks tuleks eelnevalt loodud objektile lisada juurde *Begin Step* sündmus. Eesmärgiks on muuta objekti liikumist vastavalt hiire kursori asukohale. Selleks on hea kasutada GameMakeri funktsiooni *point_direction(x,y,x2,y2)*, mis leiab vastava suuna ühest punktist teise punkti jõudmiseks ning hiirekursori asukoha muutujaid *mouse_x* ja *mouse_y*. Siinjuures tuleb arvestada ka objekti spraidi visuaalse suuna muutmisega. Kuna näites loodi kanast neli spraiti, mis kujutavad kana liikumist neljas erinevas suunas, on

mõistlik jagada 360 kraadi neljaks võrdseks osaks ning vastavalt liikumise suunale kuvada vastavasuunalist spraidi animatsiooni.




```
{
dir=point_direction(x,y,mouse_x, mouse_y);

if dir>315 || dir <=45 sprite_index=spr_kana_p
else if dir>45 && dir<=135 sprite_index=spr_kana_yles
else if dir>135 && dir<=225 sprite_index=spr_kana_v
else if dir>225 && dir<=315 sprite_index=spr_kana_alla;

image_speed = 0.25;
move_towards_point(mouse_x,mouse_y,4);
}
```

Joonis 15 - Suuna muutumine GameMakeris ja GML skript spraidi muutmiseks vastavalt nurgale

Funktsioon *move_towards_point* on kolme argumendiga funktsioon, kus esimesed kaks on koordinaadid, mis määravad ära liikumise suuna ja viimane tähistab liikumise kiirust. Skripti tulemuseks on neljasuunalise spraidiga animeeritud liikuv objekt ekraanil, mis liigub hiire kursori suunas. Liikumise peatamiseks tuleks juurde kirjutada kontroll, mis vaataks, kas objekt on jõudnud hiirekursori juurde ning sellisel juhul edasine liikumine lõpetada ning animatsioon peatada. Kuna koordinaatide täpset kattumist ei pruugi alati toimuda, on mõistlik ette anda väike eksimisruum, näiteks 3 pikslit, mis on väiksem kui objektile määratud liikumine sammu jooksul.

```
if (point_distance(x,y,mouse_x,mouse_y)<3){
image_speed = 0;
speed = 0;
}
```

Joonis 16 - GML skript animatsiooni peatamiseks

Mängule interaktiivsuse lisamine

Igas mängus peaks olema mingi ülesanne, mida mängija peab püüdma lahendada. Kuna eelnevalt sai loodud mängija poolt juhitav objekt kana, siis tuleks mängu samasuunaliselt edasi arendada. Kanadega seonduvad tavaliselt kanamunad. Järgmisena tuuakse sisse uus

objekt – kanamuna. Iga teatud aja tagant muneb kana muna. Selle saavutamiseks on GameMakeris võimalik kasutada taimereid (*alarm*).

1. Kõigepealt tuleks luua uus kanamuna pildiga sprait. Selle loomine käib sarnaselt kana loomisele. Võimalik on ka kanamuna mingil viisil animeerida – näiteks lisada munakoorele läikimise efekt.

2. Seejärel tuleks luua spraidist objekt – obj_muna. Sellega on uus objekt loodud ning edasi tuleb täiendada eelnevalt loodud kana objekti, et iga teatud aja tagant lisataks kana asukohale uus muna eksemplar.

3. Kana objekti *Create* sündmusele tuleks lisada taimer. Selleks kirjutada skripti lõppu juurde *alarm[0] = 50;*

Antud rida käivitab mängu alguses taimeri, mis omakorda käivitab 50 sammu pärast sellega seotud tegevused. Kogu GameMakeri ajaarvestus on seotud sammudega, seega tuleb alati mõelda kui tihti sündmus juhtuma peaks.

4. Kana objektile tuleks lisada uus sündmus *Alarm 0*. Siia alla tuleb paigutada kõik tegevused, mis käivitatakse 50 sammu möödumisel.

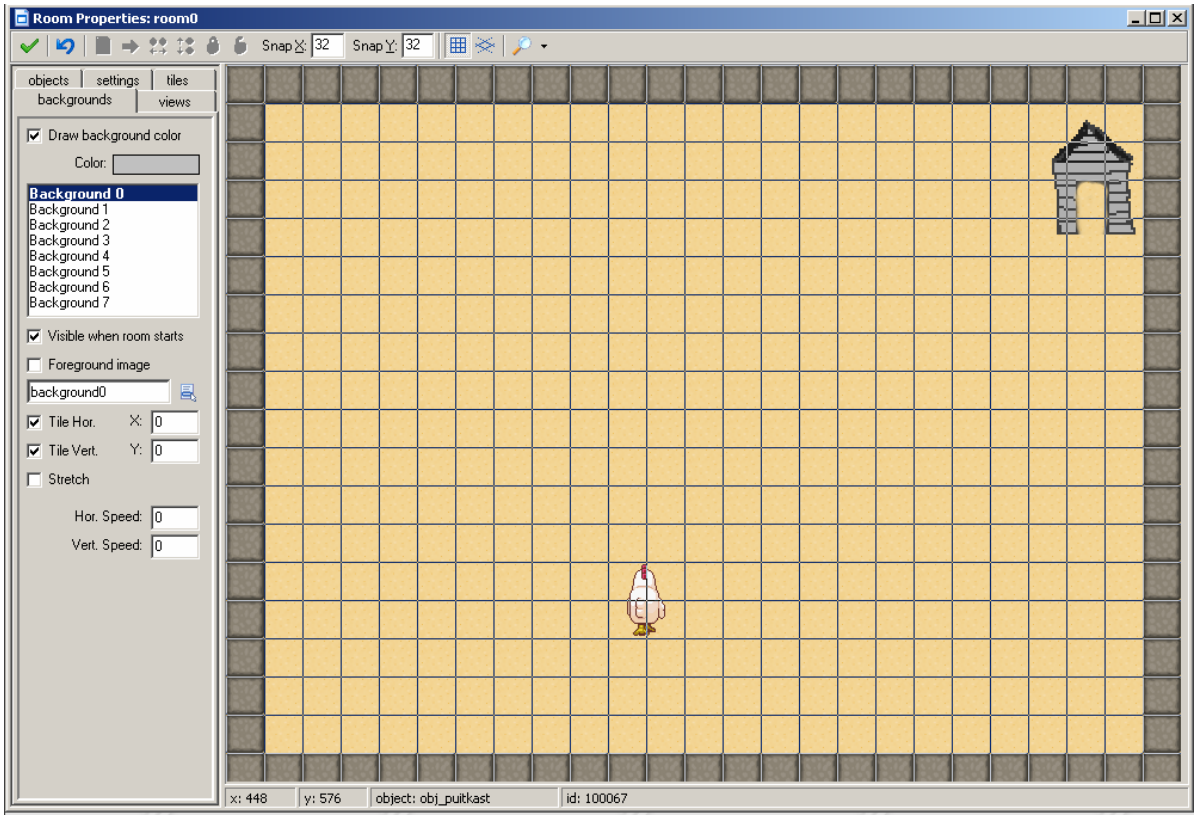
```
instance_create (obj_kana.x, obj_kana.y, obj_muna);
alarm[0] = 50;
```

Joonis 17 - GML skript uue objekti loomiseks

Selle tulemusena luuakse iga 50 sammu järel uus eksemplar objektist muna. Asukoht on määratud objekt kana hetke koordinaatidega. Taimer pannakse uuesti 50 sammu lugema ning 50 sammu pärast tegevus kordub.

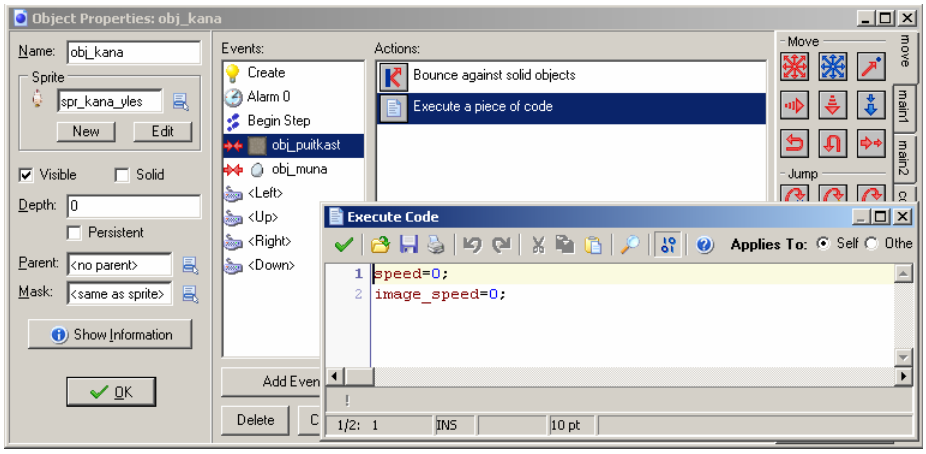
Visuaalse mulje parandamiseks võib joonistada uue kanakuuti kujutava spraidi ja sellest luua uue objekti ning see asetada ruumi. Samuti võib ruumi määrangute alt muuta halli taustavärvi sobilikumaks või lisada taustapilt (*Background*). Selleks tuleb luua uus taustapilt *Create a background*  nupu abil. Nagu teiste GameMakeris kasutatavate ressursside puhul tuleb ka taustapildile anda nimi. Sarnaselt spraidi loomisega on võimalik taustapilt importida või luua uus, kasutades selleks tuttavat spraidi joonistamise akent.

Samuti oleks tark piiritleda ruum ümberringi, et objekt ei saaks kaduda ekraani taha. Selleks tuleks luua jällegi uus kandilise kasti kujuline sprait ja objekt, ning katta ruumi servad loodud objekti eksemplaridega. Ükshaaval lisamise vältimiseks on võimalik all hoida *Shift* klahvi.



Joonis 18 – Piiritletud ruumile taustapildi lisamine

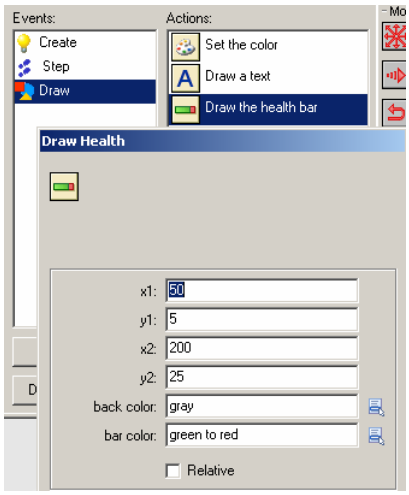
Kastide lisamisest ainuüksi ei piisa, lisaks tuleb määrata objektile kanale põrkamise sündmus kastiga. Kuna põrkamine on alati kahepoolne tegevus siis võib sama hästi lisada põrkamise sündmuse ka kastile. Mõlemal juhul on sündmuse asetleidmise korral oluline peatada kana edasine liikumine ja animatsioon. Jällegi on võimalik kasutada nii GML skripti kui ka standardseid GameMakeri pukseeritavaid tegevusi.



Joonis 19 - Objekti põrkamine seinaga

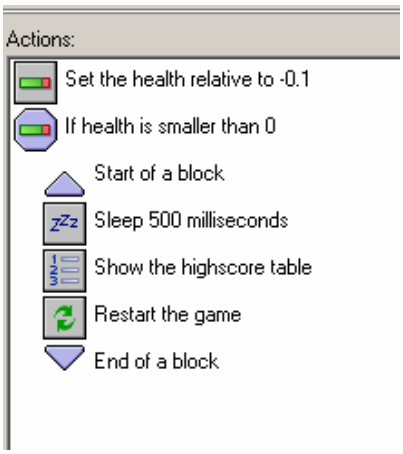
Eelnevaga on loodud mängija poolt juhitav objekt, mis tekitab omakorda teatud aja tagant uusi objekte – kanamunasid. Mängulisuse suurendamiseks tuuakse järgnevalt sisse

tervisenäidik (*healthbar*), mis kuvab ekraanil mänguelude arvu ehk mitu korda mängija tohib eksida enne kui mäng tema jaoks lõpeb ja punktisüsteem. Nii luuakse mängijale motivatsioon kiiremini ja oskuslikumalt tegutseda. Kõik ekraanile joonistamisega seotud tegevused nagu ka tervisenäidiku kuvamine tuleb panna *Draw* sündmuse alla. Vastavad tegevused leiab vahelehelt nimega *score*. Lisaks on võimalik muuta värvilahendust, selleks on võimalik pukseerida vahelehelt *draw* tegevused *Set Color* ja *Set Font.* Tervisenäidiku joonistamiseks on vaja sisestada ristkülikukujulise näidiku vasaku ülemise punkti ja parema alumise punkti koordinaadid. Koordinaatide järgi paigutamist hõlbustab ruumi vaade, kus on hiire kursoriga ruumis ringi liikudes näha selle koordinaadid. Samuti tuleb määrata näidiku värv. Valik on võimalik teha 16 erineva värvilahenduse hulgast. Võimaluste laiendamiseks on alati võimalik tervisenäidiku süsteem ka ise programmeerida GML skripti abil.



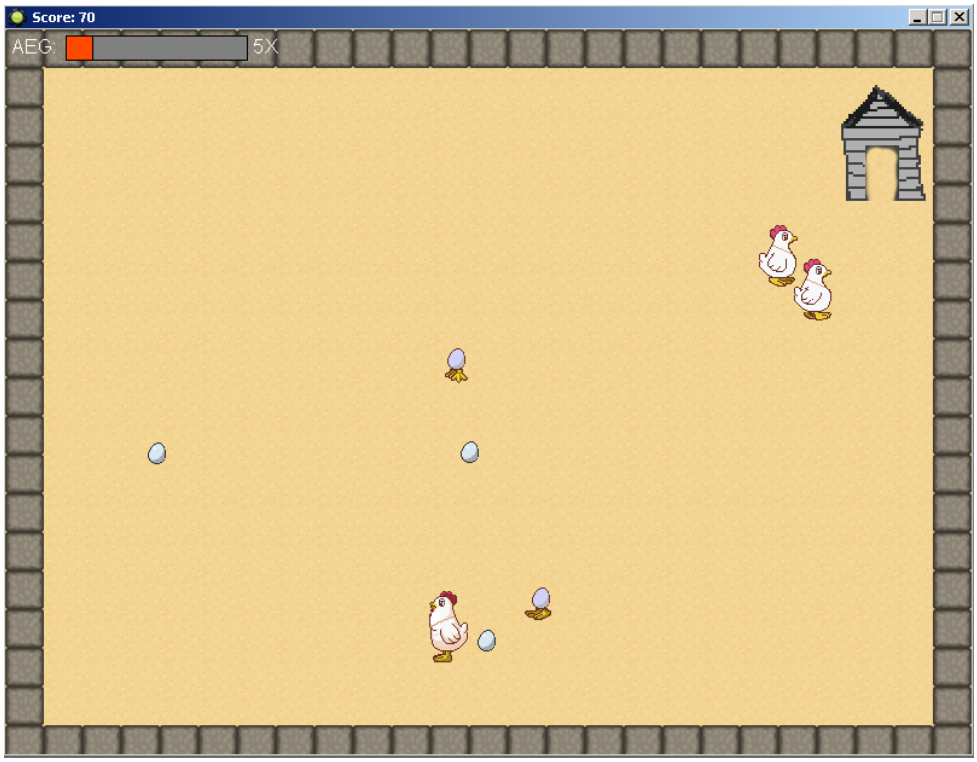
Joonis 20 - Tervisenäidik (health bar)

Alljärgnev tegevuste järjekord on sobilik panna sammu (*step*) sündmuse juurde. Igal sammul vähendatakse mängija elude hulka, mille järel kontrollitakse ega mängijal ei ole elud otsa saanud. Elude lõppemisel peatatakse kogu mäng 500 millisekundiks. Samuti võib esitada mõnda heliefekti, et mängija tajuks mängu lõppemist. Seejärel näidatakse GameMakeri sisest edetabelit, kuhu mängija saab sisestada oma nime. Pärast seda käivitatakse mäng algusest peale. Siinjuures tuleb tähele panna, et GameMakeris on kontrolllaused alati hulknurga ikooniga ning sulgusid asendavad hallid kolmnurgad.



Joonis 21 - GameMakeri sisese eludesüsteemi kasutamine

Nüüd on küll loodud ajaline limiit (mängija elude arv väheneb ajas), kuid mängijast endast sõltub väga vähe – mäng lõpeb ikka ja alati siis kui aeg saab läbi. Edasine mängu kulg ja mängureeglid sõltuvad mängu looja fantaasiast. Näiteks kanaga munade peale liikudes (*Collision* sündmus objektiga muna) on võimalik munasid haududa (tegevus *Destroy Instance* muna) ja nõnda punkte koguda (*Set Score*). Võimalik on lisaks tavalistele munadele tekitada ka halbu mune, millega kokkupuutumist tuleb vältida. Või luua hoopis uus objekt - rebane, kes ilmub välja juhuslikul ajahetkel (*Alarm*) ja liigub lähima muna juurde, et see ära süüa. Samas on võimalik luua hoopis majandamismäng, kus tuleb kanafarmis tagada võimalik hea munade produktsioon, kanade heaolu ning kaubelda turuplatsil munadega.



Joonis 22 - GameMakeriga loodud rakendus