Tallinna Ülikool

Informaatika Instituut

Lihtsa arvutimängu kavandamine ja arendamine Unity näitel

Seminaritöö

Autor : Marko Paju

Juhendaja : Martin Sillaots

Tallinn 2015

Sisukord

Sissejuhatus

Arvutimängu tegemine on üks võimalustest olla loov ja oma mõte teostada. Nagu ka kunstiga on arvutimängu loomine loominguline protsess ning nõuab palju originaalsust. Tihtipeale jääb see ainult ideeks või mõtteks, kuna mängu tegemiseks on vaja osata programmeerimist. Siiski ei pea huvitav mäng olema keeruline. Väga palju on olemas mänge, mis on üpris lihtsa ülesehitusega, kuid väga põnevad ning inimesed veedavad tunde neid mängidesMängu eesmärgiks on siiski eelkõige pakkuda meelelahutust ja ajaviidet. Samas soovivad paljud ka mänge müües raha teenida ja populaarsust koguda.

Mänge võib teha ka teistel eesmärkidel, näiteks õppimiseks. Lapsed ja ka täiskasvanud õpivad mängides palju, ka mängudest, mis ei ole õppimiseks loodud. Näiteks simulaatorite ja tõsimängude abil võivad inimesed omandada uusi teadmisi ja vilumusi., siis mängides saavad inimesed uusi teadmisi ja kogemust.

Tihtipeale kannatab mängu sisemine loogika, kuna arendaja püüab näidata oma tehnilisi oskusi. See juhtub siis, kui mängu loomist alustatakse tehniliste võimaluste katsetamisest, mitte mängu ja reeglite kavandamisest (*gameplay*). Minu eesmärgiks on näidata, kuidas mängu kavandamise raamistikku ja arendust *Unity*'s omavahel integreerida.

Oma töös esitan ülevaate ja mängu arendusprotsessist, mis sobiks väga hästi arendajatele, kes ei ole varem mängude tegemisega kokku puutunud ning proovivad seda esimest korda.

Alustasin mängumootori otsimisega, sattusin Unity peale, mis on mõeldud mängude tegemiseks ja sobib nii algajatele kui ka kogenud arendajatele. Eesmärgiks on luua mäng ning harjuda Unity keskkonnaga. Kasutades Unity't, on hiljem võimalik mäng konverteerida Androidile või iPhone'le. Mängu loomise näidiseks valisin kõige tavalisema *Space-Shooter* tüüpi mängu.

1.Arvutimängude üldine arendusprotsess

Mängude arendamise ja tarkvaraarendamise põhimõtted on samad. Mõlemad peaksid teadma kuidas koodi kirjutada, andmestruktuure, veidi matemaatikat, andmebaase ja võrgustikku. Tarkvaraarendajal peaksid olema sügavamad teadmised operatsioonisüsteemidest, süsteemianalüüsist, tarkvara kvaliteedist andmebaasidest ja võrgustikust. Mänguarendajal seevastu peaksid olema sügavamad teadmised graafika programmeerimisest, *AI*-st, mängufüüsikast ja simulatsioonidest, animatsioonidest jne.

Mängu kavandamine koosneb mitmest sammust. Looja peab välja mõtlema väljakutsed mängijale, panema paika mängu mehaanika, valmis saama töötava prototüübi ning saama tagaside. Olenevalt tagasisidest, mängu kas parandama või täiendama, kuni on lõpptulemus käes, mis sobib avaldamiseks. (Pulsipher, 2012).

Kui arendusprotsessi juures töötab mitu inimest, siis on võimalik ülesanded omavahel jagada. Enne mängu arendamise algust paneb disainer paika mängu spetsifikatsiooni. Tehniline arhitekt paneb paika mängu arhitektuuri. Samuti peab keegi tagama, et arendusprotsessi käigus oleks kõikidel töötajatel olemas vajalikud vahendid oma töö tegemiseks. Kui see on kõik paigas, siis teeb disainer mängule kujunduse. Seejärel liigub mäng edasi arendaja kätte, kes siis arendab mängu tarkvara ja paneb selle disainile külge. Samaaegselt töötab ka kirjanik, kes mõtleb mängule sisu ja loo. Juurde tuleks märkida veel, et peale igat staadiumit kontrollitakse mängus vigu või defekte ning kavandatakse muudatused. Edasi järgneb testimine, mida teeb siis kas testija ja/või teised mänguarenduse juurde kuuluvad inimesed (disainer, arendajad jne.). Lõpuks lastakse mäng välja. Muudatusi ja uuendusi tuleb tihti viia sisse ka peale mängu väljastamist.

Disainiprotsess on jagatud kolme põhiosasse. Esimene - Kontseptsiooni faas, Põhiosa, mille peale mäng ehitatakse ning mida hiljem enam ei muudeta. Teine - arendamise faas, kus lisatakse enamus disainiosi ja arendatakse mängu prototüüpide ja testimise läbi. Kolmas - häälestamise staadium, kus uusi põhilisi funktsioone enam juurde ei lisata, küll aga tehakse väikseid muudatusi olemasolevatele osadele.

1.1 Kavandamine

Kõik mängud saavad alguse ideest. Ideest tuleb mängu kontseptsioon mida edasi arendada. Kontseptsiooni osas peaks valmis saama üldine nägemus. Kõiki detailid ei pea paigas olema, kuid peaks olema ettekujutus millest mäng on, mängija roll mängus ning sihtgrupp, kellele mäng loodud on.

Raske on ette arvata, kas mäng osutub populaarseks või mitte. Isegi kõige parema ideega ei pruugi hea mäng valmis saada. Alati tuleks uusi mõtteid otsida. Mängu idee leidmiseks võib ette kujutada "Mis juhtuks, kui ma suudaksin…" või "Kui … oleks võimalik, siis …". Kuna mäng võib olla ükskõik milline, siis võib olla ka idee ükskõik milline, isegi mis ei ole võimalik päris elus. Ideed võivad tulla raamatutest, filmidest, teistelt mängudelt või kust iganes.

Üks esimesi küsimusi mängu kontseptsiooni luues on - "Mis on mängija roll?". Näiteks mängida kuningat või olla võidusõitja. Peab ka mõtlema mida sinu tegelane tegema hakkab või mis ta ülesanded on. Järgmiseks oluliseks osaks on žanri valimine. Nagu raamatutel ja filmidel, on ka mängudel erinevad žanrid. Ei pea kinni pidama ühest žanrist, võib panna ka mitu erinevat kokku. Pannes kokku mitu erinevat žanrit, kasvab ka sihtgrupp, kellele mäng loodud on. Sihtgrupi määratlemine on järgmine tähtis osa kontseptsioonist. Võib teha vea mõeldes, et kui mäng meeldib mulle, siis meeldib see ka teistele. Tuleks mõelda, kes seda mängu ostaks. Samuti võib ka mõelda, et mis on mängu juures sellist, mis võib neile mitte meeldida. Tuleb mõelda, mida inimesed naudiksid, mis hirmutab neid, mis võib olla solvav jne. Kui püüda teha mängu mis meeldib kõigile, pannes see kokku erinevatest žanritest ning elementidest, võib juhtuda, et see ei meeldi kellelegi. Väga oluline on ka mõelda mis seadmetel mängu mängitakse, on see siis kas arvuti, mobiiltelefon või mängukonsool.(Adams, 2009)

1.2 Disain ja arendus

Arendamise staadiumis muutub töö üldisemast spetsiifilisemaks. Esmalt alustatakse prototüübi loomisega. Võetakse olemasolevad ideed ja arendatakse millegiks töötavaks, et näha kuidas mäng välja näeks. Kui kõik sobib, jagatakse töö osadeks, ning mängu tegemine võib alustada. (Adams, 2009)

Tuleb kavandada põhiline mängurežiim kus mängija enamiku oma ajast veedab. Näiteks rallimängus on põhiline autoga sõitmine, auto parandus on teisejärguline. Põhiline pole kõiki detaile paika saada, vaid töötada osadel, mis muudavad mängu tervikliikuks. (Adams, 2009)

Väga tähtsaks osaks on ka luua mängu peategelane (protagonist). Mängides veedetakse palju aega tegelase vaatamisele, mis tähendab, et ta peab olema huvitav. Tähtis on mitte ainult, kuidas tegelane välja näeb, aga ka kuidas ta erinevate asjadega suhtleb, emotsioonid ning keel. (Adams, 2009)

Oluline on luua ka mängumaailm, kus enamus tegevusest toimub. See võib osutuda väga suureks tööks, eriti kui mängumaailm põhineb pärismaailmal. Luues aga maailma mida olemas ei ole, tuleb kasutada väga palju fantaasiat ja püüda võimalikult hästi anda edasi tunne ja atmosfäär. (Adams, 2009)

Edasi tuleks mõelda kuidas mängu mehaanika panna nii tööle, et kõiki tegevusi ja väljakutseid läbi viia. Mängu mehaanika on reeglid, mille järgi mäng töötab. Need reeglid näitavad mängumootorile mida teha, kuid ei ütle täpselt ette kuidas. (Adams, 2009)

Mõned näited mängu mehaanikast :

- **Mängu majanduse toimimine.** Näiteks kust tuleb raha, kuidas seda jagatakse, milleks seda kasutada võib ning majanduse reguleerimine.
- **Aru saama võidust või kaotusest.** Mehaanika tunneb ära kas väljakutse on läbi kukutud või sooritatud, ning mis juhtub reeglite järgi edasi.
- **Mängija tegevustest arusaamine.** Mängija liigutamine ning kuidas mängija ülejäänud mängumaailmaga suhtleb.
- **Juhib *A.I.* tegevust.**

Võib juhtuda, et on vaja rohkem kui ühte Mängurežiimi. Kui mängija on kõik tasemed ära lõpetanud ja mängu läbi saanud, siis luua näiteks ellujäämisrežiim, kus mäng käib punktide peale, ning on lõputu. Lisades liiga palju uusi režiime võib mäng aga muutuda liiga keeruliseks. (Adams, 2009)

Üks viimaseid ülesandeid arendamise staadiumis oleks lugu . See on tähtis, kuna siis tunneb mängija ennast rohkem mängu kaasatune. Samuti on see väga hea põhjus edasimängimiseks, huvi teada saada mis saab. (Adams, 2009)

1.3 Testimine ja häälestamine

Mängul peaks olema rohkem kui üks prototüüp, mille peal testida ja proovida igat uut muudatust, enne kui see mängu lisada. Peale kõikide muudatuste sisse viimist on arendusprotsess lõppenud, ning enam ei lisata mängule ühtegi uut funktsiooni (kui koguaeg uuendada ja lisada uusi asju, siis võib juhtuda, et mäng ei saagi valmis). Tavaliselt on selle jaoks olemas ka tähtaeg, kuid arendusprotsess tuleks lõpetada kui kõik osad omavahel sobivad ja mäng tundub terviklik. Kui see on tehtud, siis edasi tuleb häälestusstaadium, kus tehakse väikseid muudatusi nii tasemetele kui ka mängu mehaanikale ning pööratakse tähelepanu detailidele. Selles staadiumis tehakse mäng täiuslikuks ning eemaldatakse kõik vead mis mängus on. (Adams, 2009)

Kasutatakse erinevaid metoodikaid mängude arendamiseks. Enamasti kasutatakse kahte kindlat mudelit, *Waterfall* ja agiilne mudel ning neid kohandatakse oma vajadustele vastavaks.

1.4 Arendusmetoodikad

Väga oluline on arendusmetoodikat valides see, et valitud metoodikat hästi vallatakse. Kui tegemist on tähtsa projektiga, siis peaks kaaluma uue metoodika kasutuselevõttu. Võttes kasutusele uue metoodika, tuleks kaasata vastavat metoodikat valdav spetsialist. Arendusmetoodika valimisel peaks seda kohandama organisatsiooni ning projektrühmaga

harjumuste ning oskustega. Kindlasti on vaja ka valida õiged arendusvahendid metoodikat arvestades. (Normak, 2012)

Waterfall mudel - Lineaarne arendus, mis oli kuni hiljuti üheks kõige populaaremaks metoodikaks. Selline arendus kujutab endas sirgjoonelist, samm-sammu haaval arendust, kust erinevad etapid tehakse üksteise järel ära. Enne mängu arendamise algust peaks olemas olema juba üpris täpne ülevaade mängust ja mida see sisaldab. Tänu sellele on kohe alguses teada, mis tegema peab ning üllatusi ei tule. Arendajatel on lihtne tähtaegu ette arvestada ning ennustada, millal mingi osa mängust valmida võib. Samuti on võimalik juba enne mängu arendamist ette arvata ja lahendusi leida probleemidele mis võivad tekkida arenduse käigus. Kokkuvõtteks on juba alguses teada, milline mäng olema peab ning arenduse käigus on võimalik alati tagasi vaadata,milline mäng olema peab ning disaini, mis alguses loodud sai. (Thorn, 2013)

Agiilne - Nagu ka Waterfall mudel, algab ka agiilne mudel mängu ette planeerimisest, kuid ei vaja täielikku ning detailset disaini. Agiilne mudel jätab rohkem ruumi muutusteks ning vajab vaid esialgset, töötavat ideed, mida on võimalik mängida ning testida. Kui Waterfall meetodil langesid kõik osad mängu lõpus paika, siis agiilsel arendusel on võimalikult vara mängitav versioon olemas, mida järk-järgult edasi arendada. Igas arendusetapis lisatakse mängu midagi juurde, kuni lõpus ollakse rahul tulemusega. Kuna mängude arendus on üldjuhul pikk protsess, siis agiilne metoodika tagab, et juhul kui peaks kasutusele tulema uus tehnoloogia, siis on võimalik seda poole mängu pealt lisada. Agiilset mudelit kasutades on raske ette arvata projekti maksumust ning võib mõnel juhul olla väga erinev esialgsest disainist. (Thorn, 2013)

On palju erinevaid mängumootoreid mida on võimalik kasutada alustavatel arendajatel. Üheks populaarsemaks näiteks Unity. Kuna Unity on väga populaarne arendajate seas, siis leidub internetis Unity-st palju matrjali ning õpetusi. Unity-l on ka lihtsasti kasutatav ühildatavus pea kõikide platvormidega. Samuti ei võta ka palju aega Unity kasutama õppimine ja mängu avaldamiseks on vaja tasuda vaid ühekordne tasu, pole oluline kui populaarseks mäng osutub, rohkem maksta ei tule. (Jasani, 2014)

2. Mängu Kontseptsioon

Mänguarenduse üldise raamistiku näiteks kavandatakse ning arendatakse käesoleva semestritöö raames lihtne arvutimäng. Kuna on paljusid kes oskavad programmeerida, kuid Unity kasutamine on võõras sai ülesandeks luua klassikaliselt lihtne mäng, mille valmistamisel on võimalik areneda. Kindlasti pidi mäng olema 2D, sest 3D mängu tegemine on esialgu liialt keeruline. 3D mängu tegemisel on väga suur osa modelleerimisel ja tasemete ehitusel. Otsustasin teha 2D liikuvate objektidega mängu, mida saab mängida ajaviiteks.

2.1 Lühiülevaade mänguideest

On palju erinevaid žanreid – tulistamisest hüppamise ja jooksmiseni välja. Võimalik on luua lihtsat mängu peaaegu ükskõik mis žanris. Lõplikuks valikuks osutus arcade-tüüpi mäng nagu *Space Shooter,* mis tundus huvitav ning sinna oli võimalik lisada põnevaid asju ning seda kohandada. Mängu nimeks otsustasin panna "*Space Cats*".

2.2 Mängija roll mängus

Mängija rolliks mängus on juhtida laeva, mis liigub horisontaalselt ning tulistab vastutulevaid vastase laevu. Samuti vältida kokkupuudet vastaste ja teiste objektidega. Eesmärgiks on saada võimalikult suur punktisumma enne elude otsa lõppemist. Samuti on mängijal võimalus kasutada *Slow-mo* seadet, mis peatab aja ning teeb mängimise lihtsamaks.

Peavaenlase võitlused - mängu üheks osaks on ka võitlused suurte vaenlastega, võitluses seisab mängija silmitsi suurema ning tugevama laevaga ja püüab seda hävitada. Enamasti on laeva raske hävitada ning nõuab rohkem vaeva. Vastase laev tulistab vastu, mis muudab võitluse raskemaks. Peale peavaenlase hävitamist liigub mängija järgmisesse tasemesse.

2.3 Mängu sisu – põhilised väljakutsed ja tegevused

Põhiliseks väljakutseks on hävitada võimalikult palju vastase laevasid, enne elude otsa saamist või enne seda, kui laevad mängija selja taha jõuavad. Ettevaatusega tuleb suhtuda ka teistesse objektidesse, millesse otsa põrgates hävineb mängija laev ning kaob üks elu. Iga 500 punkti tagant lisandub mängule 1 vastaste laev, mis muudab mängu raskemaks. Iga 450 punkti tagant lisandub mängijale 1 elu, mis omakorda teeb mängimise kergemaks.

2.4 Žanr

Mängu žanriks on *shoot'em up*, mis on alamliik *shooter* žanrile. Mängija võtab enda kontrolli alla mistahes sõiduki (näiteks kosmoselaev või lennuk), millega ta hävitab tulistades vastase lennukeid, samaaegselt põigeldes nende rünnakurest. *Shoot'em up* mängud on eksisteerinud väga kaua. Tegemist on ühega esimestest arvutimängužanritest. (Battersby, 2015)

2.5 Sihtgrupp

Sihtgrupp on lai. Mäng ei sisalda stseene ega olukordi, mis on keelatud lastele. Põhiliselt tunneksid mängu vastu huvi noored. See sobib kõigile, kellel on vaba aega ja tahtmist ületada sõprade punktisummasid.

2.6 Riistvaraplatvorm

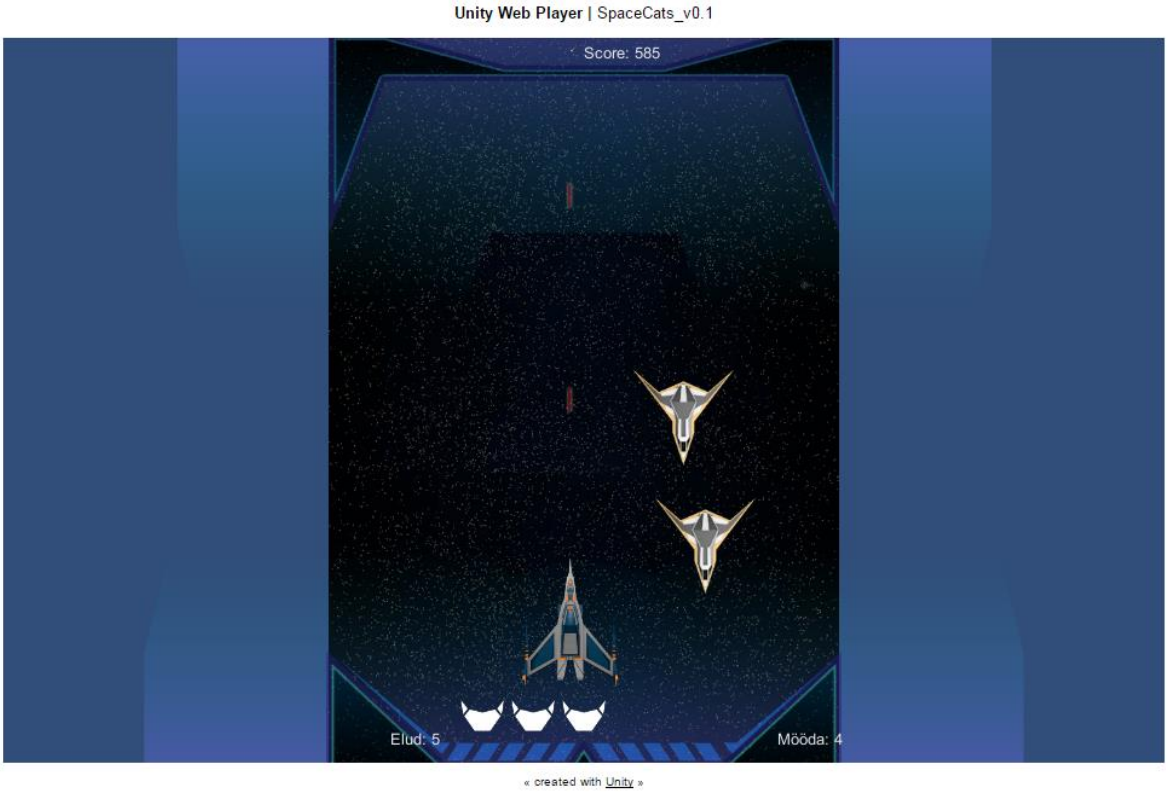
Mäng oli originaalselt mõeldud mobiilsetele seadmetele. Kahjuks polnud testimiseks õigeid seadmeid võtta, seega on mäng hetkel võrgus mängimiseks. Mõni osa mängust on koodi kujunduse poolest mõeldud just mobiilsetele seadmetele. Unity-s on kompileerimine ja ehitamine tehtud väga lihtsaks, olenevalt, mis seadmele on soov mängu teha. Tehes valmis töötava versiooni mängust, näiteks võrgus mängimiseks, pole raske seda kujundada mobiilsele seadmele. Loomulikult on igal seadmel ka iseärasused, näiteks *input* viisid ja mängu resolutsioon.

2.7 Konkurents/Koostöö võimalus

Mängul puudub koostöö võimalus. Mängida saab ainult üksi. Mängu eesmärgiks on saada võimalikult suur punktisumma, mis tähendab, et inimesed konkureerivad üksteisega suurima punktisumma nimel. Mäng salvestab ka sinu suurima punktisumma, et järgmine kord alustades ei pea uuesti nullist alustama.

2.8 Mängumaailm

Mängumaailm on disainitud 2D-s. Tekib mulje nagu laev liiguks edasi tänu liikuvale taustale ning vastutulevatele vastastele. Üleval keskel on kirjas sinu punktisumma, all nurkades elude arv ning mitu vastase laeva endast mööda lasknud oled.



Joonis 1. Mängumaailm

2.9 Turundusstrateegia

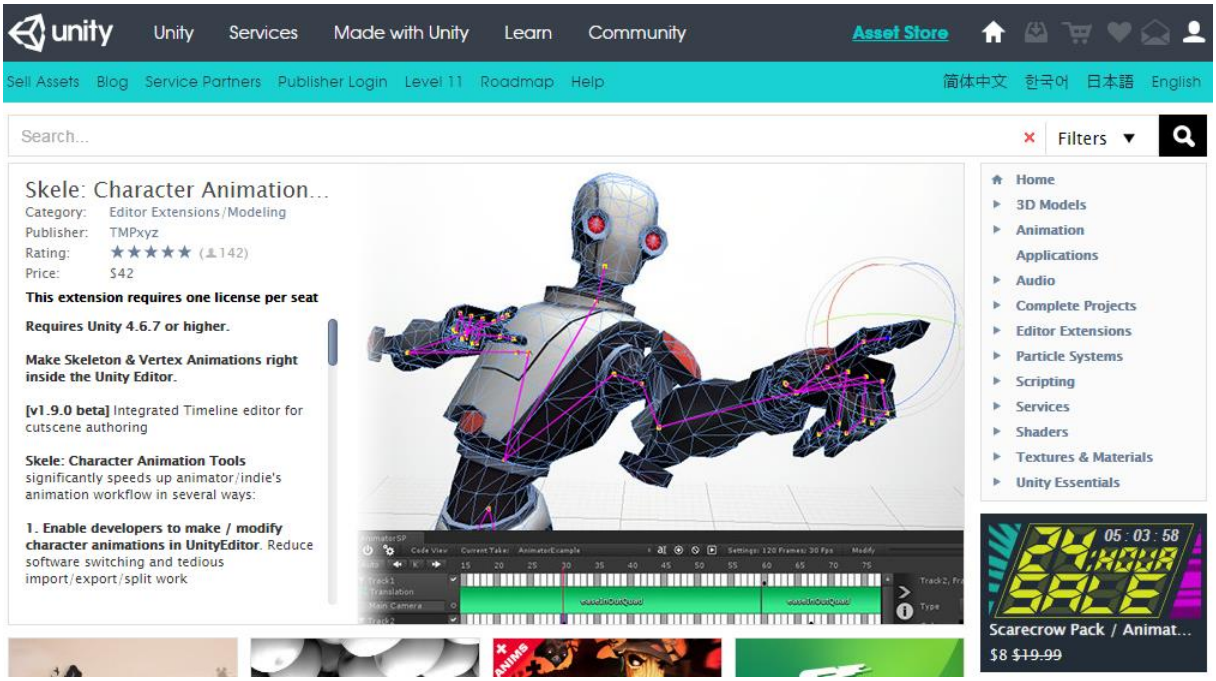
Turundusstrateegiaks oleks vajalik mäng tasuta üleslaadida. Hiljem oleks vaja seda reklaamida ja müüa reklaampinda mängu sees, millest vabanemiseks tuleks osta nö *Full game*.
Tulevikus on plaan avaldada uus, kuid sarnane mäng tasulise 3D versioonina. Usun, et inimesed, kes on mänginud tasuta 2D versiooni ja kellele mäng on meeldima hakanud, ostaksid ka 3D versiooni.

2.10 Edasiarendus ja tulevik

Tulevikuks on plaanis mängule lisada taseme režiim. Näide peavaenlase võitlusest on juba olemas, kuid vajab veel läbimõtlemist ning täiustamist. Tasemerežiim peaks ideaalis välja nägema selline, et peale kindla arvu vastase laevade hävitamist, ilmub leveli lõppu peavaenlane, keda hävitades mängija järgmisesse levelisse pääseb. Peavaenlane on graafiliselt erinev teistest laevadest ning nõuab rohkem pingutust hävitamiseks. Samuti liigub peavaenlane vasakule ja paremale ning tulistab vastu. Tema hävitamiseks peab teda tabama kümme korda. Järgmised levelid on raskemad, ilmub rohkem laevu, laevad sõidavad kiiremini ning nende hävitamiseks on vaja rohkem kui üks lask.

3. Arendusprotsess

Unity on väga hea ja mitmekülgne mängumootor. Suurepärased ehitamise ja kompileerimise võimalused erinevate platvormide jaoks, visuaalne editor, animatsioonisüsteem. Unity kohta leidub ohtralt dokumentatsiooni, millest enamus on inglise keeles. Koodinäited ning muu on internetis olemas. Unity mängumootor on tasuta. Paljud mängud on tehtud Unity-t kasutades. Unity on populaarne. Seda uuendatakse tihti ja palju inimesi töötab selle kallal, et see paremaks muutuks. Kõigele lisaks on seal ka *Asset store*, mis teeb Unity-st väga populaarse valiku mänguarendajate hulgas.



Joonis 2. Unity *Asset Store*

Asset store-st on võimalik alla laadida mudeleid ja programme, mida kasutada enda mängus - osa tasuta, osa tasulised. Kindlasti väga kasulik inimestele, kes soovivad oma mängu lisada midagi enamat. Asset Stores on kõike, 3D mudelitest ja animatsioonidest koodijuppideni välja.

3.1 Mängija laeva liigutamine

Esimese asjana töötasin valmis mängija algelise liikumise. See tähendas laeva liigutamine paremale ja vasakule. Unity's on *update* funktsioon, mida kutsutakse iga kaader. Sinna sisse lisasin horisontaalse liikumise nuppude vajutamisel.

```
function Update () {
      this.transform.Translate(Input.GetAxis("Horizontal"),0,0);
}
```

Koodinäide 1. Mängija laeva liigutamine

3.2 Vaenlaste laevade genereerimine

Järgmisena asusin vastase laevade juurde. Need pidid mööda ekraani allapoole liikuma ja mängijaga kokku põrgates mängija laeva hävitama. Selleks pidin igale vastase laevale ja mängija laevale lisama collideri. Laevu pidi pidevalt üleval erinevates kohtades juurde tekkima ja need pidid omakorda liikuma allapoole, mängija suunas. Vastase laevadel on üleval x-tejel mänguekraanist väljas *spawn* vahemik, ning igakord kui laev juurde tekib, tekib ta suvalisse kohta selles vahemikus. Samuti on vastaste laevadel *constant force*, jõud, mis liigutab neid kindlal kiirusel y-teljel allapoole.

3.3 Liikuv taust

Et jätta muljet nagu laev edasi liiguks, tuli panna ka taust liikuma allapoole. Seda tehes om kaval viis - tehes kaks tausta, mis mõlemad liiguvad kordamööda alla. Jõudes alla välja liigub see tagasi üles samal ajal, kui teine taust nähtav on. Taustade liikumine annab efekti, nagu laevad liiguvad kosmoses, üksteisele vastu.

3.4 Tulistamise funktsioon

Nüüd oli vaja väljatöötada võitlus. Idee oli, et mängija laev peab saama tulistada vastutulevaid vastaseid. Vaja oli luua “rakett”, mis lendab välja mängija laeva esiosast ja liigub ülesepoole, hävinedes ise ja hävitades vastase laeva kokkupõrkel. Tuli luua uus objekt, millele lisasin samuti collideri. Nupuvajutusega tekkis ta mängija laeva esiotsa ja liikus mööda y-telge ülesse *constant force* abil. Vastase laevaga kokku põrgates, hävinevad mõlemad.

3.5 Mänguelementide disain

Kasutasin kõige testimiseks ja arendamiseks erinevaid värvi kaste. Kuna kaste ja ringe kasutades pole mäng pooltki nii põnev, siis sain ka mudelid, milled tegi Kristjan Press ning mida kasutasin edaspidi oma mängus. Lisades need, koos animatsioonidega, oli mäng nüüd mängu moodi ning muutus hetkega huvitavamaks.

Joonis 3. Mängija plahvatus

3.6 UI arendus

Edasi oli vaja arendada mängu UI. Näiteks peamenüü, kus on kirjas mängu mängimise õpetus. Teise seas sai loodud ka kaotus- ja võidumenüü, tekstid ja nupud, mis viisid tagasi peamenüüsse. Kõik need on tehtud erinevateks *Scene*’ks ning nupud muudavad *Scene*. Terve mäng võib koosneda ka vaid ühest *Scene*’st, kuid nendest on kasu, et oma mängu erinevateks osadeks jaotada.

3.7 Punktisüsteem

Mängule oli vaja lisada punktisüsteem, mis loeb palju vastaseid hävitatud on. Igakord kui rakett põrkas kokku vastase laevaga, lisandub 15 punkti, mis lisandub olemasolevale punktisummale. Mängijale lõin ka elud, mida on alguses 6. Vastasega kokku põrgates kaotas mängija ühe oma eludest. Kui elud lõppevad, lõppeb ka mäng. Lisasin veel ka mööda lastud laevade summa, ehk laevad mis jõuavad mängija selja taha. See lõpetab samuti mängu, kui kindel arv on ületatud. Elud lisasin alla keskele, et neid oleks pidevalt näha. Töötasin välja süsteemi, mis lisab mängijale elusid, kui on kindel arv vastaseid on hävitatud.

3.8 Aeglustusfuntsioon

Mängu arendamise käigus tuli mulle mõte lisada aeglustamisfunktsioon. Hoides all *Shift* nuppu, muutub mängu kiirus aeglasemaks ning vastaseid on lihtsam hävitada. Lisasin mängule funktsiooni, mis aeglustab kogu mängu aega poole võrra kui shift nuppu all hoitakse. Selline aegluubis mängimine ei kesta aga igavesti. On kindel aeg, kui palju saab seda võimalust kasutada. Antud funktsioon on võimalus juurde teenida vastaseid hävitades, kellest igaüks lisab hävinedes mängija aegluubiajale väikese osa juurde. Telefoniversioonis rakenduks aeglustusfunktsioon vajutades ja hoides alumist vasakut ekraani osa.

```
function Update () {
             if (Input.GetButtonDown ("shift"))
                    Time.timeScale = 0.7;
}
               else{
                      Time.timeScale = 1.0;
}
```

Koodinäide 2. Nupu vajutusel aja aeglustamine

Kokkuvõte

Eesmärgiks oli luua ülevaade mängu kavandamise ja arendamise protsessile, mida saaksid kasutada inimesed, kellel on huvi mängude loomise vastu. Kavandasin oma *Space-shooter* mängu ning panin kirja olulisemad punktid. Kirjutasin üldisest, kui ka spetsiifilisest arendusprotsessist ja tõin välja *waterfall* ning *agile* arendusmetoodikad. Märkisin ülesse enda tehtud mängu konseptsiooni ning kõik sellega seonduva. Luues enda mängu on võimalik toetuda ja näpunäiteid võtta eelnevalt kirjeldatud arendusprotsessile. Töö oluliseimateks tulemusteks oli kindlasti mängukavandi näide ning teostus Unity keskkonnas.

Mängu tegemine võib esialgu tunduda keeruline. Paika tuleb panna põhipunktid, mida samm-sammult edasi arendada. Väga palju informatsiooni leiab internetist. Tehes mängu, mida teised on varem teinud, on lihtne leida abi ja hästi otsides võib leida isegi valmis koodi. Tagantjärgi võib öelda, et esialgne pilt ja lõpptulemus on üpriski erinevad. Arendamise käigus tuli palju uusi huvitavaid ideid mängu põnevamaks muutmiseks, näiteks lisaobjektid ning nuppu all hoides aegluubis mängimine. Kindlasti on palju kohti, mida edasi arendada ja täiustada, näiteks lisatasemed ning mängu lugu. Väga oluline on tagasiside põhjal muudatusi teha ja neid ellu viia. Samuti on oluline teada ka, et mida kindlamalt on mängu ettekujutus ja kontseptsioon paigas, seda lihtsam on arendusprotsess. Edaspidi võib edasi liikuda keerulisemate mängude loomisele, tuginedes mänguarenduse põhiprotsessidele ja metoodikatele.

Mängu link:

https://www.dropbox.com/sh/3akcpr1c3j9qiik/AACmBQ_LNZfn1ZBv0e7QPBhfa?dl=0

Kasutatud kirjandus

Adams, E (2009). *Fundamentals of Game Design (2nd Edition).* Pearson Education, Inc.© 2010

Battersby, R. (2015). Allikas: https://www.quora.com/What-is-the-game-genre-shoot-em-up

Jasani, T. (2014). Allikas: http://venturebeat.com/2014/08/20/the-top-10-engines-that-can-help-you-make-your-game/

Normak, P. (2012). Allikas: https://www.tlu.ee/~pnormak/PJ-2012/Loengute%20esitlused/8-Tarkvaraprotsess.ppt

Pulsipher, L. (2012).*Game Design: How to Create Video and Tabletop Games, Start to Finish.* Jefferson : McFarland & Company, Inc., Publishers, 2012.

Riel, A., O'Connor, R., Tichkiewitch, R., Messnarz, R. (2010). *Systems, Software and Services Process Improvement*. Berlin ; Heidelberg : Springer, ©2010.

Thorn, A. (2013). *Game Development Principles.* Cengage Learning.

Unity Technologies. *Unity*. Kasutamise kuupäev: 10. November 2014. a., allikas Unity3D: http://unity3d.com/unity