TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Kaspar Eensalu 142599IAPB

HELITALUVUSEL PÕHINEV MÄNGUMEHAANIKA JA SEDA REALISEERIV PROTOTÜÜP

bakalaureusetöö

Juhendaja: Ago Luberg
MSc

Tallinn 2017

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kaspar Eensalu

19.05.2017

Annotatsioon

Selle bakalaureusetöö eesmärgiks on välja töötada kõikehõlmav helitaluvusel põhinev mängumehaanika 3D esmavaates mängude jaoks ning implementeerida selle prototüüp Unity mängumootori peal. Töö tulemus võiks pakkuda huvi eelkõige tulistamismängude arendajatele, kes soovivad oma mängu lisada realistlikku helitaluvuse kujutust.

Töö käigus luuakse mängumehaanika tööpõhimõtteid kirjeldav dokument, mida saab kasutada kui õpetust helitaluvuse mehaanika loomisel. Lisaks luuakse tööpõhimõtete põhjal helitaluvuse mehaanika töötav prototüüp koos testimistasemega, kus on võimalik prototüübiga lihtsal viisil tutvuda.

Töö kirjeldab ka mehaanika loomisel kasutatavad tehnoloogiad ning meetodid. Uuritud on helitaluvuse kujutamist juba loodud mängudes, helirõhkude levimise füüsikalist põhja ning kõrge helirõhu füsioloogilisi mõjusid inimesele.

Töö tulemuseks on reaalselt töötav helitaluvuse mängumehaanika prototüüp Unity mängumootori peal, mida on võimalik edasi arendada, ning mehaanika tööd kirjeldav dokument.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 6 peatükki, 14 joonist, 1 tabelit.

Abstract

Noise tolerance based game mechanic and its prototype

The aim of this thesis is to develop a comprehensive noise tolerance based game mechanic for 3D first person games and to implement its prototype on the Unity game engine. The solution could be of interest mainly to developers of shooter games who wish to add a realistic depiction of noise tolerance into their game.

The thesis contains a document detailing the principles of operation of the game mechanic which can be used as a guide in creating a noise tolerance based game mechanic. The guide is illustrated with examples from the created prototype. In addition to the document, a working prototype of the developed game mechanic as well as a testing level which can be used to easily explore the intricacies of the mechanic are created.

A great deal of importance is also placed on the real life properties of noise pressure relevant to development of the game mechanic. The thesis details some of the physical properties of noise pressure as well as the physiological effects high noise pressure has on the human body. Depictions of noise tolerance in existing games have also not been left unexplored. A set of requirements derived from the real life properties and existing depictions of noise tolerance is also described in the thesis as well as the technologies used.

The results of this thesis include a working Unity game engine prototype of the developed game mechanic and a document detailing the operating principles of the mechanic. All written code is available online and further development of the game mechanic is made as easy as possible and encouraged.

The thesis is in Estonian and contains 30 pages of text, 6 chapters, 14 figures, 1 tables.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Vaatamata sellele, et valjude helide poolt põhjustatud kuulmisprobleemid on sõjaväes tõsine probleem [5], puudub tänapäeva sõjaväe simulaatorites valjude helide füsioloogilisi mõjusid simuleeriv mängumehaanika. Realistlik helitaluvuse mängumehaanika võiks olla abiks sõdurite treenimisel ning ka tänapäeval aina rohkem reaalsuse poole püüdlevate taktikaliste tulistamismängude juures eksisteeriks ilmselt nõudlus säärase mängumehaanika järele.

Töö eesmärgiks on välja töötada mängijasõbralik, lihtsasti laiendatav ja võimalikult täpselt reaalsust imiteeriv helitaluvuse mängumehaanika ning luua mängumehaanika prototüüp Unity mängumootori peal. Enne mängumehaanika välja töötamist tehti põhjalik taustauuring helirõhu negatiivsete mõjude käsitlusest mängudes ning uuriti helirõhu füüsikalisi ja füsioloogilisi omadusi. Uuringu käigus kogutud informatsiooni tõlgendus on esitatud töö 2. ja 3. peatükis.

Töö tulemusena valmis helitaluvuse mängumehaanika tööpõhimõtteid kirjeldav dokument ning mehaanika implementatsioon Unity mängumootori peal. Lisaks loodi implementeeritud mehaanika demonstreerimiseks mõeldud testimistase. Mehaanika tööpõhimõtteid kirjeldavat töö 4. peatükki võib kasutada juhendina realistliku helitaluvuse mängumehaanika loomisel ning peatükk on varustatud näidetega loodud mehaanika prototüübist.

Loodud mehaanika prototüübi iseärasusi ning testimistaset on kirjeldatud töö 5. peatükis. Lisaks sisaldab peatükk prototüübi kasutusjuhendit. Kogu loodud programmikood on internetis vabalt kättesaadav ning mängumehaanika prototüübi koodi kasutamine/edasiarendamine on julgustatud.

2 Taustauuring

Järgnevas peatükis on kirjeldatud helitaluvuse rolli taktikalistes tulistamismängudes (*tactical shooter*) ning sõjaväe simulaatormängudes (*military simulation*). Taustauuringu lõpus on välja toodud uuringu põhjal tehtud tähtsamad järeldused.

2.1 Helitaluvus varajastes tulistamismängudes

Alustades 1987. aastal välja tulnud Microprose'i "Airborne ranger'ist" ja lõpetades 2000. aastal välja tulnud Valve'i "Counter Strike'iga", pole helitaluvusele suurt rõhku pandud, rääkimata kõikehõlmavast mängumehaanikast. Kõikehõlmava mehaanika all mõeldakse seda, et iga piisavalt kõrge rõhuga heli mõjutab mängijat.
Siiski on ka juba varajastes tulistamismängudes helitaluvuse temaatikat põgusalt kasutatud, seda niinimetatud flashbang granaatide juures.

2.1.1 *Flashbang* mehaanika

Üks huvipakkuv osa paljudest varajastest ja ka tänapäevastest tulistamismängudest on niinimetatud *flashbang* granaadid. *Flashbang* granaadid on olemas pea igas taktikaliste tulistamismängude žanrisse kuuluvas mängus. Nende peamine eesmärk on vastast lühikeseks ajaks pimestada ja kurdistada, kuid tihti võib halvasti visatud *flashbang* sama mõju avaldada ka viskajale endale. Tavaliselt on granaadil mängijale nähtav ja kuuldav mõju. Ekraan muutub reeglina valgeks, mängijal kaob lühikeseks ajaks võime visuaalselt informatsiooni saada ja loomulikult ka täpselt tulistada. Mõju kadu on tavaliselt järkjärguline. Kuuldav mõju on reeglina kõrge sagedusega heli. Mängijal kaob võime auditoorset informatsiooni saada ning nägemise ja kuulmise kadumise koosmõju tekitab segadustunde.

Paljudes mängudes on *flashbang*i mõju binaarne, see kas on või ei ole. Mõnedes uuemates mängudes aga on mõju tugevus sõltuv plahvatuse kaugusest. Kaugel plahvataval *flashbang*il on mängijale väike mõju, sekundimurdosa kestev välgatus ja vaikne vilin. Selline erinevate tasemetega mõju on käesoleva töö raames asjakohane.

Vaatamata sellele, et paljudes *flashbang* granaatidega mängudes on ka palju valjemaid heliallikaid, piirdub mõju kuulmisele *flashbang*idega.

2.1.2 Battlefield 2: Project Reality

2005. Aastal tuli välja laiendus (*mod*) populaarsele mängule Battlefield 2 [1][2]. Ka laiendus ise sai laialdase tunnustuse osaliseks, võites mitmeid auhindu, sealhulgas 2008 ModDB *mod of the year* tiitli [3]. Omal ajal oli tegemist ühega realistlikuimatest mängukogemustest, mis kunagi tehtud. Ka praegu, 12 aastat hiljem, on Project Realityl suur mängijate hulk.

Vaatamata laienduse nimele on heli valdkonnas reaalsusest asi kaugel. Peale *flashbang* granaatide, millest oli 2005. aastaks saanud laialt levinud mängumehaanika, ei ole Project Reality juures helitaluvusele mingit rõhku pandud. Ka Project Reality vaimse järeltulija, 2015. aastal välja tulnud Squadi [4] juures ei tekita kõrged helirõhud mängijatele negatiivseid tagajärgi.

2.2 Helitaluvus kaasaegsetes tulistamismängudes

Kaasaegsetes tulistamismängudes on helitaluvusele rohkem rõhku pandud, mängud muutuvad aina realistlikumaks. Sellest hoolimata ei ole kõikehõlmavat helitaluvuse mängumehaanikat loodud.

Erilist uurimist väärivad mõned tänapäeva tulistamismängude ja sõjaväe simulaatormängude lipulaevad nagu ARMA mängude seeria, VBS simulaatorite seeria ning Call Of Duty mängude seeria.

2.2.1 ARMA ja VBS

2006. Aastal alguse saanud Bohemia Interactive stuudio ARMA mängude seeria väärib eraldi mainimist. Tegemist on sellest ajast kuni tänapäevani ühega tuntuimatest sõjaväe simulaatormängude žanri mängudest.

ARMA mängude heli on realistlikkuse tipp. Heli levib ruumis õige kiirusega, heli levimise kaugus on realistlik. ARMA 3 puhul lisati juurde ka helikiiruse ületamisele iseloomulik vali “plaks”, mida tekitavad nii lennukid kui ka individuaalsed kuulid. Kõigest sellest hoolimata ei ole ekstreemselt valjudel helidel mängijale negatiivset mõju.

Sama mängustuudio teeb ka sõjavägedele treenimiseks mõeldud simulaatorit VBS. VBS1 ja VBS2 on praegu juba ka tavakasutajatele kättesaadav, kuigi väga kõrge hinnasildi taga. VBS3 on hetkel veel tsiviilinimestele kättesaamatu ja ainus informatsioon simulaatori kohta pärineb Bohemia Interactive stuudio enda käest ning on väga puudulik. Ka sõdurite treenimiseks kasutatava simulaatori juures pole valjudel helidel mängijale negatiivset mõju. Tekib küsimus, miks sõdurite treeningvahendite juures helitaluvuse aspekt täiesti välja jäetud on, kui 2012. aastal oli USA veteranide kõige levinumad vigastused tinnitus ja kuulmise halvenemine [5].

2.2.2 Battlefield 3 suppression mehaanika

Üks huvipakkuv mängumehaanika on Dice mängustuudio 2011. aastal välja tulnud Battlefield 3 [6] *suppression* mehaanika. Kuigi tegemist ei ole otseselt helivaljusega seotud mehaanikaga, on sellest nii mõndagi õppida.

Oma olemuselt on mängumehaanika järgmine: vastasmängijate teele pandud kuulid tekitavad mängija lähedalt möödudes mängijale ebameeldivaid tagajärgi. Mängija nägemine muutub kergelt uduseks ja mängija relv ebatäpseks.

Uudne mehaanika sai laialdase mängijate kriitika osaliseks [7]. Mängijatele ei meeldinud see, et *suppressioni* vastu ei saanud midagi teha, tuli lihtsalt oodata ja loota, et vastase salv tühjaks saaks. Veel oli mängijatele vastumeelt see, et relvad kasutult ebatäpseks muutusid. Probleem oli selles, et kontroll võeti mängijate käest ära - relva sihtimine ei muutunud, relv lihtsalt ei lasknud sinna kuhu mängija sihtis. Selle asemel soovivad mängijad, et kontrolli kadumise asemel oleks märgi tabamine raskendatud, mitte võimatu, ja et kogenud mängijad saaksid ikkagi sihtmärgile pihta.

2.2.3 Call of Duty

Ilmselt ideeliselt kõige lähedasem mängumehaanika helitaluvuse mehaanikale on 2003. aastal alguse saanud Call of Duty mängude seeria *shellshock* mehaanika [8] - suuremad plahvatused mängija läheduses kutsuvad esile erinevaid negatiivseid mõjusid. Kuigi mehaanika mõjude täpsed põhjustajad on erinevate mängude jooksul palju muutunud, on mõjud jäänud suures osas samadeks. Mängija nägemine muutub uduseks, liikumise ja sihtimise kiirus väheneb, kuulmine väheneb ning kuulda on nõrka vilinat.

Seeria esimestes mängudes põhjustab *shellshocki* suurtükituli. Mõju kestvus sõltub otseselt sellest, kui palju mängija viga saab. Järgnevates Call of Duty mängudes kasutatakse *shellshocki* põhiliselt etteplaneeritud üksikmängu stseenides, et suurendada mängija sisseelamist mängumaailma, ja tegemist pole universaalse mängu osaga.

Mänguseeria hetkel viimases mängus, 2016. aastal välja tulnud Call of Duty: Advanced warfare'is, on *shellshock* mehaanika suurema tähtsusega, ja seda põhjustavad igat tüüpi plahvatused. Jällegi puudub mängijal kontroll ja mängijatelt võetakse relva kasutamise võimalus.

2.3 Taustauuringu järeldused

Tehtud taustauuringi põhjal on võimalik teha mitmeid helitaluvuse mängumehaanika väljatöötamise koha pealt tähtsaid järeldusi, mida on järgnevalt loetletud.

1. Kõikehõlmavat helitaluvuse mehaanikat varem loodud pole. Mehaanika väljatöötamist peab alustama nullist.

2. Mehaanika väljatöötamisel peab arvestama sellega, et mehaanika mõjustaadiumid kasutajalt liigselt kontrolli ära ei võtaks. Kontrolli kaotamine on mängijatele tavaliselt frustreeriv kogemus.

3. Mehaanika väljatöötamisel saab kasutada Call Of Duty mänguseeria *shellshock* mehaanikale sarnaseid efekte.

4. Mehaanika mõjustaadiumid peaksid sarnaselt *flashbang* granaatide mõjule olema sujuva kadumisega.

5. Arvestades kuulmiskahjustustega veteranide suurt hulka võiks helitaluvuse mehaanika olla soovitav osa sõjaväe simulaatormängudest.

3 Meetodid, nõuded ja tehnoloogia

Järgnevas peatükis on kirjeldatud helitaluvuse mängumehaanika loomisel kasutatavad meetodid, nõuded arendatavale mehaanikale ning töö loomisel kasutatud tehnoloogiad. Meetodid võib laias laastus jagada kahte gruppi: helirõhkude edastamise ja vastuvõtmise loomiseks vajalikud heli levimise füüsikalised omadused, ning mõjustaadiumite loomiseks vajalikud kõrge helirõhu füsioloogilised mõjud.

3.1 Heli levimise füüsikaline põhi

Reaalsust imiteeriva helitaluvuse mängumehaanika loomisel on vajalik helirõhu füüsikalisi omadusi võimalikult täpselt jäljendada. Järgnevas alapeatükis on kirjeldatud helirõhu levimise ja summeerimise meetodid ning välja toodud erinevate huvipakkuvate heliallikate rõhud.

3.1.1 Helirõhu taseme arvutamine

Reaalsusele sarnase helitaluvuse mehaanika välja töötamisel on esmatähtis uurida helirõhu taseme muutust kauguse suurenemisel heliallikast. Ühikuna on tarvilik kasutada detsibelle (dB), sest suur osa lähteandmeid on just detsibellides. Helirõhk langeb 6 dB võrra kauguse kahekordistumisega. Täpselt on võimalik langust arvutada valemiga (1) [9].

$$L_2 = L_1 - \left|20 * \log\left(\frac{r_1}{r_2}\right)\right| \quad (1)$$

L_1 – Helirõhk algpunktis (dB)

L_2 - Helirõhk otsitavas punktis (dB)

r_1 – algpunkti kaugus allikast (m)

r_2 – otsitava punkti kaugus allikast (m)

Korraga on vaja arvestada mitme heliallika mõju, seega on vajalik helirõhu tasemete liitmine. Helirõhu tasemete liitmiseks saab kasutada valemit (2) [10].

$$SPL_T = 10 * \log\left(\sum_{i=1}^{n} 10^{\left(\frac{SPL_i}{10}\right)}\right) \quad (2)$$

SPL_T - Helirõhku kogutase(dB)

SPL_i - Osa helirõhk (dB)

3.1.2 Helirõhu tasemed

Töö raames oleks mõistlik iga kasutatava heliallika kohta välja arvutada helirõhk 10 cm kaugusel, et seda hiljem realisatsiooni käigus kasutada. 10 cm oleks mõistlik minimaalne kaugus heliallikale arvutimängus ja ülejäänud arvutused lähtuvad ette antud helirõhu tasemest 10cm kaugusel.

Tabelis (1) on esitatud tulistamismängudes tihti esinevate heliallikate mõõdetud rõhud koos mõõtmiskaugustega [11] ning valemit (1) kasutades arvutatud rõhud kaugusel 10cm.

Tabel 1 Heliallikatele vastavad rõhud 10cm kaugusel

Heliallikas	**Kaugus heliallikast**	**Mõõdetud helirõhk**	**Arvutatud helirõhk kaugusel 10cm**
9mm püstol	1m	157 dB	176 dB ~ 175 dB
12 *gauge* pumppüss	1m	161 dB	181 dB ~ 180 dB
5.56mm automaat (M4)	1m	165 dB	185 dB
7.62mm/.50 kaliiber kuulipilduja	2m	165 dB	191.02 dB ~ 190 dB
105mm haubits	5m	164 dB	197.98 dB ~ 200 dB
Käsigranaat	15m	164 dB	207.52 dB ~ 210 dB
M3 tagasilöögita tankitõrje relv	1m	190 dB	210 dB
Tüüpiline helikopter	4m	120 dB	152.04 dB ~ 150 dB
Tüüpiline reaktiivlennuk	100m	125 dB	185 dB

3.1.3 Helirõhu taseme muutus kõrvas

Inimese kõrvas langeb helirõhk märkimisväärselt. Kuulmekanali kuju tõttu peegulduvad helilained kõrvas ja trummikilel on rõhk madalam, kui kõrva ääres. Helirõhu langus kuulmekanalis ei ole lihtsasti ennustatav.

Rõhu langus sõltub nii heli sagedusest, kui ka iga kõrva iseärasustest. Helirõhu langus jääb enamasti 10-30 dB vahele [12]. Käesoleva töö raames on vajalik helirõhu langust arvesse võtta.

3.2 Kõrge helirõhu mõju inimesele

Liigselt kõrgel helirõhul on inimesele käesoleva töö seisukohast mitmeid huvipakkuvaid mõjusid. Paljud kõrge helirõhu negatiivsed mõjud on inimeselt-inimesele muutuvad ja isegi subjektiivsed, kuid arvutimängu mehaanika skoobis on neid võimalik piisava täpsusega kirjeldada.

3.2.1 Hirmurefleksi poolt põhjustatud silmade pilgutamine

Üks valjude helide mittevabatahtlik mõju inimesele on silmade pilgutamine. Inimese keha seostab ootamatuid valjusid helisid ohuga, ning silmade sulgemine on keha instinktiivne üritus silmi kaitsta.

Silmade sulgemise refleksi põhjustavad ootamatud üle 80 dB helid [13], kuid kindlalt refleksi esinemise ennustamine on võimatu ja sõltub paljudest, arvutimängu mehaanika skoobis kirjeldamatutest, teguritest. Käesoleva töö raames kasutatakse silmade sulgemise refleksi mängijat hoiatavas mehaanika mõjustaadiumis, kui helirõhu tase ületab 120 dB.

3.2.2 Valulävi

Kuulmise valulävi on helirõhk, mille juures heli muutub kuulaja jaoks valu tekitavaks. Iga inimese valulävi on erinev, kuid tavaliselt jääb see vahemikku 120-140 dB [14]. Helirõhud üle valuläve tekitavad kuulaja kõrvades teravat valu ja heli valjenedes valu suureneb.
Võttes arvesse seda, et käesoleva töö raames välja töötatav helitaluvuse mehaanika on mõeldud sõjaväe simulaatorite ja taktikaliste laskmismängude jaoks ja nende žanrite mängudes on mängija kontrollitav tegelane tavaliselt kogemustega sõdur, on mõistlik valida valuläveks 140 dB.

3.2.3 Tinnitus

Tinnituseks nimetatakse kuuldavat heli, millel pole füüsilist allikat. Tinnitus on täiesti subjektiivne fenomen ja seda kirjeldatakse tavaliselt kui vilinat kõrvades [15]. Tinnitust tekitab ilmselt sisekõrva retseptorite kahjustus, kuid selle täpsed mehhanismid ei ole veel

teada. Tinnitus võib olla nii pika kui ka lühiajaline, kuid käesoleva töö raames on huvipakkuv lühiajaline tinnitus.

Lühiajaliste valjude helide poolt põhjustatud tinnitust on vähe uuritud, ning ei ole leitud täpseid helirõhu tasemeid, mis tinnitust esile kutsuks. On mõistlik eeldada, et tinnitus esineb helirõhkude juures, mis jäävad valuläve ja trummikile lõhkemise vahele.

3.2.4 Akustiline šokk

Akustiline šokk on traumareaktsioon väga kõrge rõhuga helidele. Šoki sümptomid on tavaliselt ajutised. Lühiajalise akustilise šoki sümptomite alla kuuluvad valu/rõhk kõrvades, peavalu, peapööritus ja ka tinnitus [16]. Akustilist šokki kogenud inimeste sõnade kohaselt kaasneb akustilise šokiga ka tugev hirmureaktsioon ja pea ning kaela tõmblus.

Akustilist šokki on raske uurida - uurijad peavad tuginema patsientide endi sümptomite kirjeldustele. Käesoleva töö raames on mõistlik eeldada, et akustiline šokk võib tekkida valuläve ja trummikile lõhkemise vahele jäävatel helirõhkudel.

3.2.5 Trummikile lõhkemine

Kõige ekstreemsem liigselt kõrge helirõhu negatiivne mõju on kõrva trummikile purunemine. Trummikile võib puruneda juba 35 kPa (~185 dB) helirõhu juures ning purunemine on peaaegu kindel 100 kPa (~195 dB) helirõhu juures [17]. Kõrva trummikile purunemisega võib kaasneda täielik kuulmise kadumine, valu kõrvas ja/või sumin kõrvas.

Trummikile purunemise tõsisemad juhtumid vajavad paranemiseks operatsiooni, kuid arvutimängu mehaanika skoobis ei ole ilmselt mõttekas trummikile purunemist püsiva effektina realiseerida. Sellegipoolest on trummikile purunemine viimane ja kõige suuremate negatiivsete tagajärgedega staadium realiseeritavas helitaluvuse mängumehaanikas.

3.3 Nõuded

Käesoleva töö eesmärgiks on luua võimalikult täpselt reaalsust imiteeriv, mängijasõbralik ja lihtsasti laiendatav mängumehaanika. Arvestades eelnevalt tehtud taustauuringu ja meetodite peatükiga, seab eesmärk loodavale mehaanikale mitmed funktsionaalsed ja mttefunktsionaalsed nõuded.

Järgnevalt on loetletud tähtsamad nõuded, millega mehaanika väljatöötamise protsessi käigus arvestatakse. Nõuete juurde on lisatud lühikesed selgitused.

3.3.1 Funktsionaalsed nõuded

1. Helirõhu edastamine toimub ühe funktsiooni välja kutsumisega.
   Mehaanika kasutamise lihtsustamise eesmärgil peab kogu helirõhu edastamise protsess toimuma ühe mehaanika avaliku funktsiooni välja kutsumisega. Kestva heli allika puhul on üks funktsioon edastamise alustamiseks ning teine lõpetamiseks.
2. Edastatava helirõhu arvutamine toimub heliallika komponendis.
   Helirõhu arvutamine heliallika komponendis vähendab koormust heli vastuvõtjale ning ülesanded on hajutatud.
3. Helirõhu suhe kaugusesse on realiseeritud valemi (1) järgi.
   Valem (1) määrab helirõhu langemise kauguse suurenemisel. Realistlik mängumehaanika peaks võimaluse korral seda kasutama.
4. Helirõhkude summa arvutamine toimub heli vastuvõtja komponendis.
   Vastuvõtja komponendis on ka andmestruktuur helirõhkude hoidmiseks, mida kasutades summat arvutatakse.
5. Helirõhkude summa arvutamine toimub kord kaadris.
   Kaader on väikseim ajavahemik, kus helirõhu taseme muutus mehaanika skoobis tähtsust omab.
6. Helirõhkude summa arvutamine on realiseeritud valemi (2) järgi.
   Valem (2) määrab helirõhkude summeerimise päriselus, mida mehaanika imiteerib.
7. Kasutajaliidesel on mängijale nähtav helirõhu summa näidik.
   Mängijal puudub võimalus erinevaid helirõhu tasemeid kuuldavalt piisavalt täpselt eristada, mille tulemusel võib negatiivsete mõjudega mehaanika mängijale frustratsiooni tekitada. Selle vältimiseks peaks mehaanikal olema nähtav helirõhu hetketaseme näidik, mis võib olla nii numbriline kui ka täituva riba kujul.
8. Helirõhu langus inimese kuulmekanalis on implementeeritud.
   Helirõhu langus kuulmekanalis on reaalsust imiteeriv ning lisab mehaanikale ka veidi varieeruvust.
9. Vastuvõtja komponent edastab mehaanika kontrollerile kord kaadris helirõhu hetkesumma.

Kontrolleri ainus ülesanne on hetkerõhu järgi mehaanika mõjustaadiumeid mängida. Kaader on väikseim mehaanika skoobis tähtis ajavahemik.

10. Kõik mehaanika mõjustaadiumid implementeerivad ühte liidest.
    Nii on mehaanika mõjustaadiumite loomine/muutmine lihtne ning arendajatele jäävad maksimaalsed võimalused.
11. Hirmurefleksi poolt põhjustatud silmade pilgutamine on mõjustaadiumis implementeeritud.
    Silmade pilgutamine on hea viis mehaanika hoiatavas mõjustaadiumis. See peaks olema piisavalt pika miinimumintervalliga, et olla mängijale informeeriv, mitte tüütu. Täpse intervalli selgitab hilisem mehaanika testimine.
12. Akustiline šokk on mõjustaadiumis implementeeritud.
    Akustilise šokiga kaasnevad kuulmise vähenemine, tinnitus ja nägemise hägustumine.
13. Trummikile lõhkemine on mõjustaadiumis implementeeritud.
    Trummikile lõhkemisega kaasneb kuulmise pea täielik lühiajaline kadumine, tinnitus, nägemise hägustumine ja tasakaalu kadumist imiteeriv efekt.

3.3.2 Mittefunktsionaalsed nõuded

1. Heliallikate mehaanika komponente on kahtesorti.
   Mehaanika kasutamise lihtsustamise eesmärgil on plahvatuslikel helidel ja kestvatel helidel erinevad mehaanika komponendid. Vastasel juhul peaks kestvate helide korral edastamist manuaalselt välja kutsuma.
2. Mõjustaadiumite nähtavate ja kuuldavate efektide kadumine on sujuv
   Efektide järsk kadumine oleks segadust tekitav ning näeks halb välja.
3. Mõjustaadiumid ei võta mängijalt kontrolli ära.
   Mängijalt kontrolli ära võtmine on mängijas frustratsiooni tekitav. Seda tuleks võimaluse korral vältida.
4. Mehaanika kood on kirjutatud ainult inglisekeelseid muutuja-/funktsiooninimesid kasutades.
   Koodi arusaadavuse seisukohalt on tähtis, et koodi kirjutades on kasutatud järjepidavalt ühekeelseid nimesid. Inglise keel on koodi keeleks loomulik valik.
5. Heli vastuvõtja on mõjustaadiumite kontrollerist loogiliselt eraldatud.
   Loogiliselt eraldatud vastuvõtja ja kontroller parandavad koodi loetavust ja arusaadavust ning lihtsustavad mehaanika edasiarendamist.

3.4 Tehnoloogia

Unity on Unity Technologies' poolt arendatud mängumootor. Käesolevas töös kasutati Unity mängumootorit ja selle arenduskeskkonda mängumehaanika prototüübi loomiseks. Mängumootorit kasutati tasuta, *personal* litsentsi alusel. Kogu programmikood kirjutati C# keeles. Unity mängumootori ja selle arenduskeskkonna kasutamise põhjuseks olid mängumootori piisavalt head omadused, selle tasuta kasutamise võimalus ning selle suurepärane dokumentatsioon ja õppimisressursside arv [18].

Mehaanika prototüübi ehmatuse mõjustaadiumi loomisel kasutati Simple3D assets' poolt arendatud EYE Blink Effect teeki [19]. Veel kasutati mõjustaadiumite loomisel Unity Technologies' Legacy Cinematic Image Effects teeki [20]. Testtaseme loomisel kasutati Into The Void Sound Designi loodud Fog of War Gun Sound FX Free audioteeki [21].

Töös olevate klassidiagrammide loomiseks kasutati tasuta veebipõhist modelleerimisplatvormi GenMyModel veebisaidil *https://www.genmymodel.com*/ [22].

4 Mehaanika tööpõhimõtted

Järgnevas peatükis on esitatud helitaluvuse mängumehaanika täpsed tööpõhimõtted. Peatüki eesmärk on luua helitaluvuse mehaanikale kindel teoreetiline põhi. Peatükki võib kasutada kui õpetust realistliku helitaluvuse mängumehaanika loomisel.

Mehaanika tööpõhimõtete kirjeldamise lihtsustamiseks on lisatud näiteid Unity mängumootori peal loodud mehaanika prototüübist, kuid tööpõhimõtete kirjeldus sobib suvalise objektorienteeritud programmeerimiskeeles loodava 3D esmavaates mängu jaoks.

4.1 Heliallikad ja nende komponendid

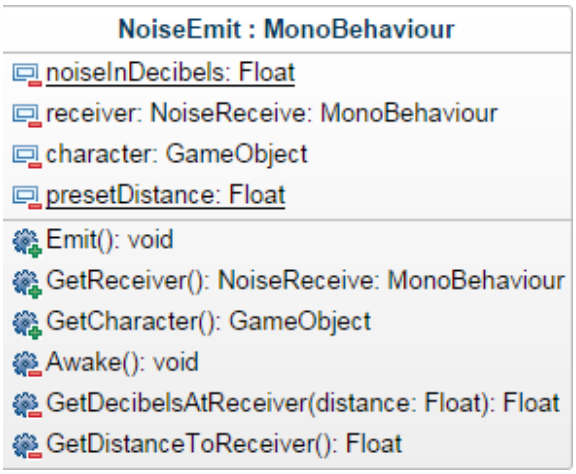
Helitaluvuse mängumehaanika väljatöötamisel tuleb esmalt rõhku panna heliallikate ja vastuvõtjate infovahetusele. Mehaanika kogu ülejäänud töö toetub helirõhkude täpse ja kiire edastamise peale.

Heliallikatena on kasutatud eraldi mänguobjekte, millel on oma positsiooni komponent ja heliallika komponent. Nii on lihtsustatud heliallika täpse asukoha seadmine. Heliallika mänguobjekt on hierarhiliselt heliallika nähtava mudeli *child* objekt, et lihtsasti tagada nähtava mudeli liikumisel ka heliallika täpne liikumine.

Heliallikate mänguobjektidel on peale positsiooni ja heliallika komponentide veel ka heliallika käitumist kontrolliv komponent ja helitaluvuse mehaanika komponent.

4.1.1 Plahvatusliku heli allika helitaluvuse mehaanika komponent

Heliallika helitaluvuse mehaanika komponendi ülesanne on vastuvõtjale/vastuvõtjatele edastada täpne helirõhk, arvestades vastuvõtja kaugust heliallikast. Joonisel 1 on toodud mehaanika prototüübi plahvatusliku heli allika mehaanika komponendi klassidiagramm. Komponendi kirjeldamisel on kasutatud joonisel nähtavaid muutuja ja funktsiooninimesid.



Joonis 1 Näide: Plahvatuse heliallika helitaluvuse mehaanika komponendi klassidiagramm

Komponendis on muutuja (`noiseInDecibels`), mis hoiab heliallika rõhku ettemääratud kaugusel allikast. Ka see ettemääratud kaugus on salvestatud muutujas (`presetDistance`). Komponendi töö alguses salvestatakse muutujatesse ka vastuvõtja mänguobjekt ja vastuvõtja helitaluvuse mehaanika komponent. Lisaks on komponendis funktsioon helirõhu täpseks arvutamiseks (`GetDecibelsAtReceiver()`), mille sisendiks on kaugus meetrites. Kui sisendkaugus on väiksem või võrdne `presetDistance`'ga, tagastab funktsioon muutuja `noiseInDecibels`. Vastasel juhul arvutatakse tagastatav helirõhu tase valemit (1) kasutades. Joonisel 2 on esitatud helirõhu taseme arvutamise näitealgoritm.

```
if (distance <= presetDistance) {
	return noiseInDecibels;
} else {
	return noiseInDecibels - Abs(20 * Log10( presetDistance / distance));
}
```

Joonis 2 Näide: Helirõhu arvutamise algoritm pseudokoodis

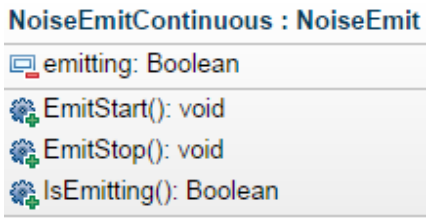
Komponendis on ka funktsioon heliallika ja vastuvõtja vahelise kauguse arvutamiseks (`GetDistanceToReceiver()`). Funktsioon arvutab heliallika ja vastuvõtja mänguobjektide hetkepositsioonide abil välja objektide hetkekauguse meetrites (Unity mängumootoris funktsiooniga `Vector3.Distance()`).

Kuna heliallikaid on laias laadis kahte sorti, plahvatusliku heli allikad ja kestva heli allikad, mis on käitumuslikult erinevad, siis on ka heliallika mehaanika komponente kahtesorti. Eelnevalt kirjeldatu on komponentide ühisosa, kuid helirõhkude edastamine on erinev.

Plahvatuslike helide allikate helitaluvuse mehaanika komponendil on helirõhu vastuvõtjale edastamiseks funktsioon `Emit()`. Seda funktsiooni kutsub välja heliallika käitumist kontrolliv komponent iga kord, kui heli mängitakse. Alguses kutsub funktsioon välja funktsiooni `GetDistanceToReceiver()` ja seejärel saadud kaugust sisendina kasutades funktsiooni `GetDecibelsAtReceiver()` ja edastab tagastatud helirõhu taseme vastuvõtja helitaluvuse mehaanika komponendile, kutsudes komponendis välja funktsiooni `AddNoise`, mille sisendiks on arvutatud helirõhu tase.

4.1.2 Kestva heli allika helitaluvuse mehaanika komponent

Kestva heli allika helitaluvuse mehaanika komponent on plahvatusliku heli allika komponendi alamklass, ja sellel on helirõhu edastamiseks vaja kahte funktsiooni: üks, mis kutsutakse välja heli alghetkel (`EmitStart()`), ja teine, mis kutsutakse välja kui heli on lõppenud (`EmitStop()`). Joonisel 3 on toodud mehaanika prototüübi kestva heliallika klassidiagramm. Komponendi kirjeldamisel on kasutatud joonisel nähtavaid muutuja ja funktsiooninimesid.



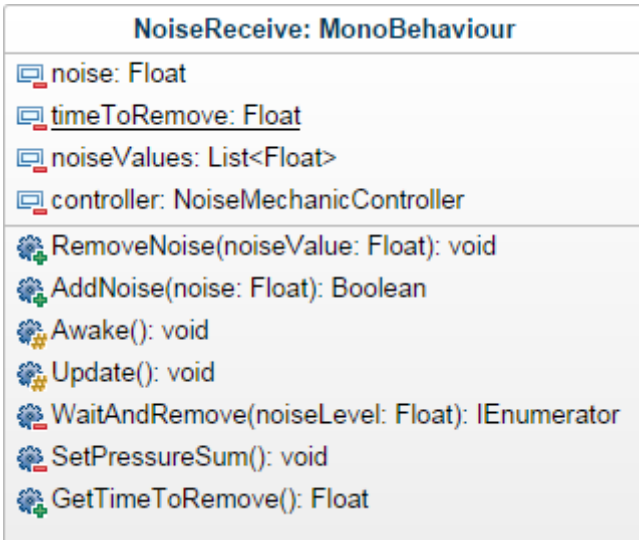
Joonis 3 Näide: Kestva heli heliallika helitaluvuse mehaanika komponendi klassidiagramm

Komponendis on ka tõeväärtus tüüpi muutuja (`emitting`), mis on vajalik käitumusliku komponendi töö jaoks ja näitab, kas hetkel toimub helirõhkude edastamine vastuvõtjale. Muutuja on privaatne ning selle kättesaamiseks on komponendis funktsioon `IsEmitting()`, mis tagastab muutuja.

Funktsioon `EmitStart()` seab alguses muutuja emitting tõeseks ja pärib seejärel vastuvõtja helimehaanika komponendilt, kui kaua see helirõhkusid sisemiselt säilitab (ujukoma tüüpi muutuja `timeToRemove`). Seejärel kutsub funktsioon iga `timeToRemove` sekundi tagant välja ülemklassi funktsiooni `Emit()` (Unity mängumootoris sisseehitatud funktsiooniga `InvokeRepeating()`). Korduv ülemklassi `Emit()` funktsiooni väljakutsumine kestab kuni funktsiooni `EmitStop()` käivitumiseni. `EmitStop()` seab muutuja `emitting` vääraks ning lõpetab `Emit()` väljakutsumise (Unity mängumootoris sisseehitatud funktsiooniga `CancelInvoke()`).

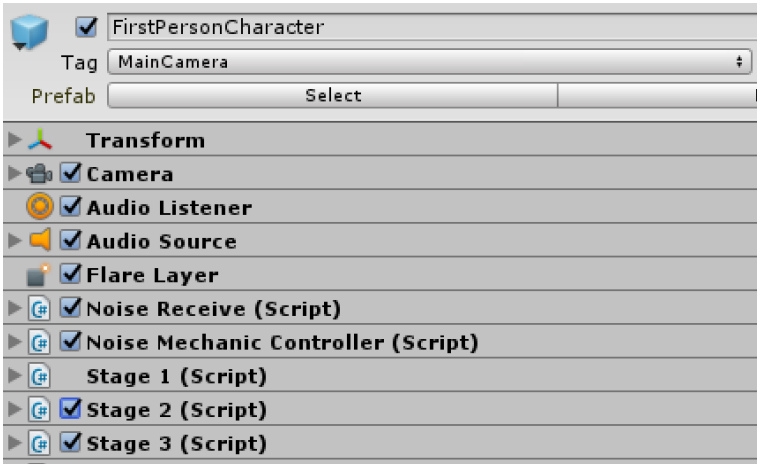
4.2 Heli vastuvõtja

Helitaluvuse mehaanika heli vastuvõtja ülesanne on kõigi mehaanika skoobis kasutatavate helide rõhkude säilitamine ja õigeaegne eemaldamine, rõhkude summa välja arvutamine ja rõhkude summa edastamine helitaluvuse mehaanika kontrollerile. Joonisel 4 on toodud mehaanika prototüübi heli vastuvõtja klassidiagramm. Mehaanika kirjeldamisel on kasutatud joonisel nähtavaid muutuja ja funktsiooninimesid.



Joonis 4 Näide: Mehaanika prototüübi heli vastuvõtja klassidiagramm

Vastuvõtja komponent oleks mõistlik paigutada mängija karakteri mänguobjekti. Nii on lihtsasti kättesaadav kaamera positsioon, mida kasutatakse helirõhkude arvutamiseks, Lisaks sellele on vastuvõtja komponent kaamera küljes ka loogiliselt õiges kohas. Kaamera mänguobjekti sisse on soovituslik paigutada ka muud mängijaga seotud mehaanika komponendid.



Joonis 5 Näide: Helitaluvuse mehaanika komponentide soovituslik paiknemine Unity mängumootoris

Joonisel 5 on näha näidet helitaluvuse mehaanika erinevate komponentide paiknemisest `FirstPersonCharacter` mänguobjekti sees. Lisaks mehaanika komponentidele on mänguobjektis ka kaamera component (`Camera`) ja heli kuulaja (`Audio Listener`) komponent.

Helirõhu hetkesumma salvestamiseks on vastuvõtjal ujukoma arvu tüüpi muutuja (`noise`), ja konkreetsete helirõhkude salvestamiseks andmestruktuur (`noiseValues`). Muutujatesse on salvestatud ka helirõhkude säilitamise aeg sekundites (`timeToRemove`), mis on staatiline, ning helitaluvuse mehaanika kontrolleri komponent (`controller`).

Helirõhu salvestamise jaoks on vastuvõtjas avalik funktsioon (`AddNoise()`), mida kutsub välja heliallikas. Enne salvestamist lahutatakse sisendrõhust inimkõrva kuulmekanalis aset leidvat helirõhu langust simuleeriv väärtus. See väärtus ei tohiks ületada 30dB. Töös on kasutada juhuslikku väärtust väikesest vahemikust (näiteks [10, 20]). Juhuslik väärtus aitab simuleerida reaalsust, kus helirõhu edasikandumisel on mitmeid väikeseid häiringuid. Arvutatud helirõhk on vaja salvestada `noiseValues` andmestruktuuri ning seejärel tuleb oodata `timeToRemove` arv sekundeid ja väärtus listist eemaldada. Ootamise realiseerimiseks on mitmeid võimalusi, kuid Unity mängumootoris on mõistlik kasutada kaasrutiine (*Coroutine*) nagu on näha joonisel 6.

```
private IEnumerator WaitAndRemove(float noiseLevel) {
    yield return new WaitForSeconds(timeToRemove);
    noiseValues.Remove(noiseLevel);
}
```

Joonis 6 Näide: Ootamise lahendus Unity mängumootoris

Helirõhkude summa edastamine kontrollerile toimub kord kaadris. Enne summa edastamist toimub summa arvutamine. Summa arvutamine toimub vaid juhul kui helirõhke hoidvas andmestruktuuris `noiseValues` on rohkem kui 0 väärtust, vastasel juhul edastatakse kontrollerile 0. Helirõhkude summa on arvutatav valemiga (2). Valemi koodis realiseerimise viise on mitmeid ja koodi arusaadavuse eesmärgil on mõistlik kasutada abifunktsiooni. Joonisel 7 on välja toodud helirõhkude summa arvutamise algoritm.

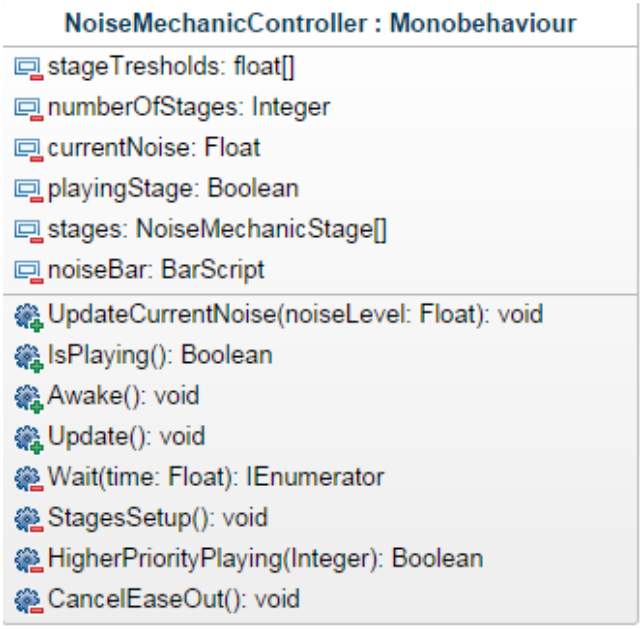
```
noise = 0f;
for (value : noiseValues) {
       noise += Pow(10, (value / 10);
}
noise = 10 * Log10(noise);
```

Joonis 7 Näide: Helirõhkude summa arvutamine pseudokoodis

Lisaks on vastuvõtja komponendis abifunktsioon säilitamisaja edastamiseks (`GetTimeToRemove()`) ja funktsioon(`Awake()`), mis kutsutakse välja mängustseeni initsialiseerimise ajal. Funktsioonis `Awake()` instantsieeritakse andmestruktuur `noiseValues` ja salvestatakse `controller`.

4.3 Helitaluvuse mehaanika kontroller

Helitaluvuse mehaanika kontrolleri ülesanne on mehaanika mõjustaadiumite õigeaegne käivitamine ja peatamine. Lisaks on mõttekas kontrolleri ülesannete hulka lisada ka kasutajaliideses nähtava hetkerõhu näidiku kontrollimine. Hetkerõhu näidik võib olla numbriline, riba kujul vms. Joonisel 8 on toodud mehaanika prototüübi kontrolleri komponendi klassidiagramm. Komponendi kirjeldamisel on kasutatud joonisel nähtavaid muutuja ja funktsiooninimesid.



Joonis 8 Näide: Helitaluvuse mehaanika kontrolleri klassidiagramm

Kontrolleris on float tüüpi muutuja heli hetkerõhu hoidmiseks (`currentNoise`) ja muutuja hetkerõhu näidiku kontrolleri hoidmiseks (joonisel 8 `BarScript` tüüpi `noiseBar`). `currentNoise` uuendamiseks kutsub vastuvõtja komponent kontrolleris välja funktsiooni `UpdateCurrentNoise()`. Juhul, kui sisendväärtus on suurem/võrdne 100dB, salvestatakse see muutujas `currentNoise`. Vastasel juhul on rõhk liiga väike, et mehaanika mõjustaadiumeid esile kutsuda, ning `currentNoise` seatakse nulliks. 100dB on valitud näidiku miinimumrõhuks, sest helirõhu negatiivsete mõjude koha pealt on huvipakkuv just vahemik 100-200dB.

Mängustseeni initsialiseerimise käigus kutsutakse kontrolleris välja funktsioon `Awake()`. Selles funktsioonis toimub hetkerõhu näidiku salvestamine, mehaanika mõjustaadiumite arvu määramine ning staadiumite seadistamine. Staadiumite seadistamise loogika on koodi arusaadavuse eesmärgil viidud eraldi abifunktsiooni (`StagesSetup()`). Abifunktsioonis instantsieeritakse staadiumite arvu (`numberOfStages`) suurused massiivid, mis hoiavad staadiumite klasse ja staadiumite mängimise miinimumpiire detsibellides (vastavalt `stages` ja `stageThresholds`). Enne tuleb salvestada staadiumite klassid ning seejärel nende kaudu miinimumpiirid.

Kord kaadris välja kutsutavas funktsioonis `Update()` on vaja alguses `currentNoise` muutujat kasutades uuendada näidikut, ning seejärel otsustada kas ja millist mehaanika mõjustaadiumit mängida. Kontroller teeb kindlaks milline staadium vastab hetkerõhule, ning kontrollib, et ükski kõrgema prioriteediga staadium hetkel ei mängi. Kõrgema

priotiteediga staadiumi mängimist kontrollitakse funktsioonis `HigherPriorityPlaying()`, mis itereerib üle kõigi sisendstaadiumist kõrgemate staadiumite ning tagastab tõese väärtuse, kui leiab, et mõni neist staadiumitest mängib hetkel. Joonisel 9 on toodud kõrgema prioriteediga staadiumi mängimise leidmise algoritm.

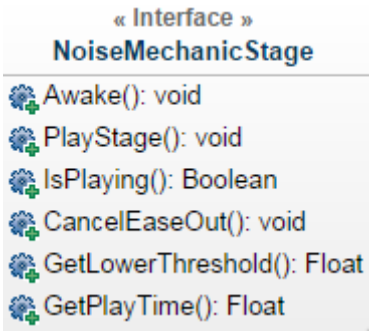
```
for (int i = currentStage; i < numberOfStages; i++) {
      if (stages[i].isPlaying()) {
            return true;
      }
}
return false;
```

Joonis 9 Näide: Kõrgema prioriteediga staadiumi mängimise leidmine

Olles kindlaks teinud, et staadiumit on vaja mängida, kutsutakse välja funktsioon kõikide staadiumite sujuva kadumise katkestamiseks (`CancelEaseOut()`) ning pannakse vastav staadium mängima. Funktsioon `CancelEaseOut()` itereerib üle kõigi staadiumite ning kutsub igas staadiumis välja funktsiooni sujuva kadumise lõpetamiseks. Samal ajal seatakse ka staadiumi mängimist näitav tõeväärtus tüüpi muutuja `playingStage` tõeseks ning staadiumi mänguaja möödudes uuesti vääraks.

4.4 Helitaluvuse mehaanika mõjustaadiumid

Kõik mehaanika mõjustaadiumid implementeerivad ühte liidest, mis määrab kindlaks staadiumis sisalduvad kohustuslikud funktsioonid. Nii on tagatud mõjustaadiumite paindlikkus ning kontrolleri sõltumatus staadiumitest. Joonisel 10 on toodud mehaanika prototüübi mõjustaadiumi klassidiagramm. Komponendi kirjeldamisel on kasutatud joonisel nähtavaid muutuja ja funktsiooninimesid.



Joonis 10 Näide: Helitaluvuse mehaanika mõjustaadiumi liidese klassidiagramm

Funktsioon `Awake()` kutsutakse välja mängustseeni initsialiseerimise käigus. Selles funktsioonis on mõistlik salvestada erinevad staadiumi mängimisel vajalikud parameetrid. Funktsiooni `PlayStage()` kutsub välja mehaanika kontroller staadiumi mängimiseks. Selles funktsioonis tuleb käivitada kõik animatsioonid/helid vms, mida staadiumis kasutatakse ning pärast seda oodata staadiumi mänguaeg sekundeid. Pärast ootamist käivitub sujuv staadiumi mõjude kadumine (selle olemasolul).

Lisaks on mõjustaadiumitel kohustuslikud hulk abifunktsioone:

- `GetLowerTreshold()`, mis tagastab staadiumi miinimumrõhu detsibellides.
- `GetPlayTime()`, mis tagastab staadiumi mänguaja sekundites.
- `CancelEaseOut()`, mis katkestab mõju sujuva kadumise.
- `IsPlaying()`, mis tagastab tõese väärtuse, kui mõjustaadium hetkel mängib.

Mõjustaadiumite implementeerimine on iga mängumootori peal erinev ning peale liidese järgimise on arendajal vabad käed. Soovituslik oleks mõjustaadiumid implementeerida päriselu sarnaselt, peatükis 3.2 välja toodud nähtuste baasil.

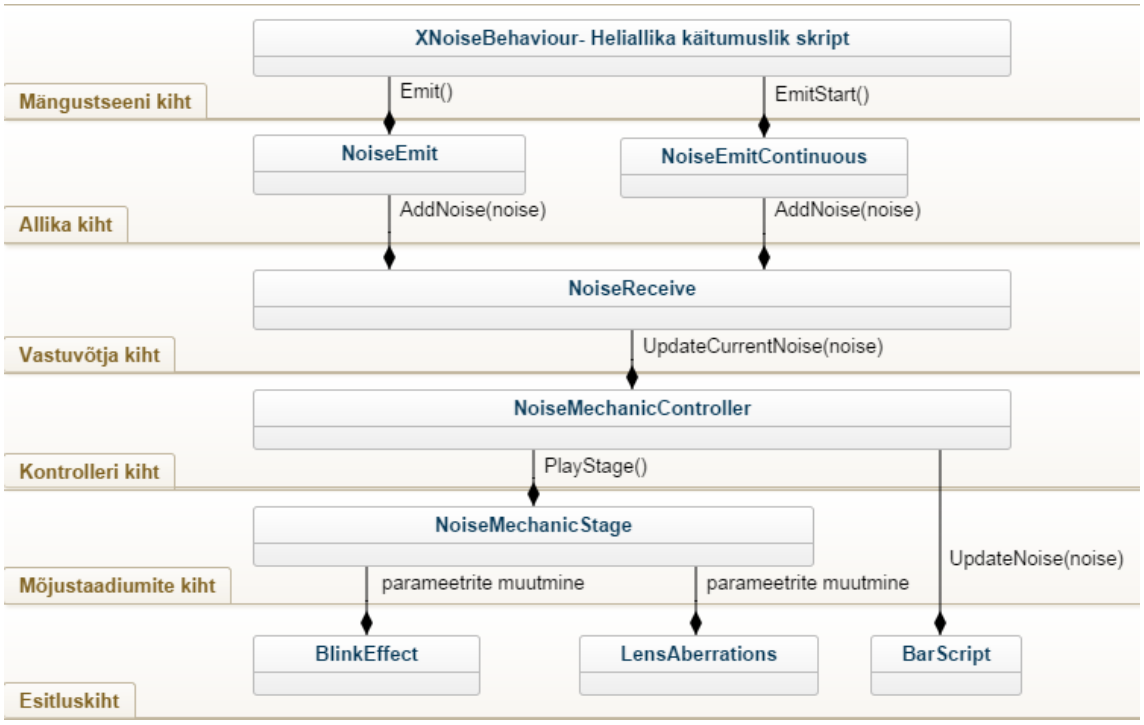
5 Mehaanika prototüüp

Järgnev peatükk tutvustab töö raames loodud helitaluvuse mängumehaanika prototüüpi. Prototüübi loomisel on järgitud eelnevas peatükis kirjeldatud mehaanika tööpõhimõtteid. Kogu programmikood on kirjutatud *C#* keeles ning prototüüp on avatud lähtekoodiga. Prototüübi kood on saadaval aadressil:
https://Kaspar.Eensalu@gitlab.cs.ttu.ee/Kaspar.Eensalu/helimehaanika.git

5.1 Ülevaade

Mehaanika prototüüp on üles ehitatud kihiliselt. Iga kihi klassid kutsuvad välja vaid endast madalama kihi klasside funktsioone. Igal kihil on oma ülesanne, kuidas kõrgemalt kihilt saadud informatsiooni töödelda ning selle alusel madalama kihi klasside funktsioone välja kutsuda.

Loodud mehaanika kihilist struktuuri kirjeldab joonis 11.



Joonis 11 Prototüübi kihiline struktuur

Mängustseeni kihi ülesanne on heliallikate käitumise kontrollimine ning allika kihi klassidele rõhu edastamiskäsu andmine. Olenevalt heliallika liigist kutsutakse kas `Emit()` funktsioon või `EmitStart()` funktsioon.

Allika kihi ülesanne on vastuvõtjale edastada täpne helirõhk. Allika kihi klassides arvutatakse täpne helirõhk, arvestades allika helirõhu ning vastuvõtja kaugusega. Arvutatud rõhk edastatakse vastuvõtjale funktsiooniga `AddNoise()`.

Vastuvõtja kihi ülesanne on allika kihi klasside poolt edastatud helirõhkude säilitamine, õigeaaegne kustutamine ning kontrolleri kihile täpse rõhkude hetkesumma edastamine. Helirõhkude hetkesumma arvutamine toimub samuti siin. Arvutatud helirõhkude hetkesumma edastatakse kontrollerile funktsiooniga `UpdateCurrentNoise()`.

Kontrolleri kihi ülesanne on kontrollida mehaanika mõjustaadiumite tööd ning ka kasutajaliidese hetkerõhu näidikut kontrollivale klassile hetkerõhu edastamine. Kontroller käivitab hetkesituatsioonile vastava mõjustaadiumi funktsiooni `PlayStage()` välja kutsudes. Näidikut kontrollivale klassile edastab kontroller kord kaadris hetkerõhu funktsiooni `UpdateNoise()` välja kutsudes.

Mõjustaadiumite kihi ülesanne on esitluskihi klasside `BlinkEffect` [19] ja `LensAberrations` [20] parameetrite muutmine. Lisaks mõjustaadiumi mängimisele tegeleb mõjustaadiumite kiht ka mõju sujuva hajutamisega.

Esitluskihi klasside ülesanne on mängijale otseselt nähtavate/kuuldavate efektide kontrollimine.

5.2 Mõjustaadiumid

Prototüübi mõjustaadiumite loomisel on imiteeritud peatükis 3.2 välja toodud mõjusid. Mõjustaadiumeid on kolm:

- ehmatuse staadium, mille käivitumiseks vajalik miinimumrõhk on 120dB (Vt peatükk 3.2.1).
- akustilise šoki staadium, mille käivitumiseks vajalik miinimumrõhk on 150dB (Vt peatükk 3.2.4).
- trummikile purunemise staadium, mille käivitumiseks vajalik miinimumrõhk on 180dB(Vt peatükk 3.2.5).

Mõjustaadiumitel on erinevad mängimisajad, sujuva hajumise parameetrid ja kasutatavad efektid. Järgnevalt on mõjustaadiumeid täpsemalt kirjeldatud.

5.2.1 Ehmatuse staadium

Ehmatuse staadium on ideeliselt mängijat hoiatav mehaanika staadium, mille ülesanne on mängijat potentsiaalselt ohtlikult kõrge helirõhu olemasolust teavitada. Staadiumi käivitumiseks vajalik miinimumrõhk on 120dB, mis on ka valulävi väidetav miinimumrõhk.

Staadium kasutab EyeBlink efekti hirmurefleksi poolt põhjustatud silmade pilgutamise simuleerimiseks.

Staadium kestab 3 sekundit ja efektide sujuv hajumine puudub, sest kasutatud on vaid silmade pilgutamise efekti.

5.2.2 Akustilise šoki staadium

Akustilise šoki staadium on ideeliselt mängijat eemalepeletav mehaanika staadium. Staadiumi mängimine tähendab, et mängija on liiga lähedal ohtlikult kõrge rõhuga heliallikale ning peaks positsiooni vahetama. Staadiumi käivitumiseks vajalik miinimumrõhk on 150dB.

Staadiumi käivitumisel kuuleb mängija 60% valjusega pininat, mis simuleerib tinnitust. Mängija kuulmine halveneb staadiumi mänguajaks 40%. Valutunde ja nägemise hägustumise simuleerimiseks kasutab staadium `LensAberrations` klassi `vignette` komponenti. `vignette` komponendi `blur` (maksimum väärtus 1) ja `intensity` (maksimum väärtus 3) atribuutideks seatakse 0.5. Atribuudi `blur` suurendamine muudab ekraani hägusemaks ning `intensity` suurendamine muudab `vignette` värvi intensiivsemaks. Ehmatuse simuleerimiseks kasutatakse ka EyeBlink efekti, mis käivitatakse `fadeOut()` funktsiooni välja kutsudes.

Staadium kestab 2 sekundit ning efektide sujuv hajumine toimub kord kaader väljakutsutavas funktsioonis `PerformEaseOutStep()`, mis on välja toodud joonisel 12. Erinevat `-Interval` lõpuga muutujad on konstandid, mille väärtuste võrra vastavaid parameetreid kord kaadris muudetakse.

```
private void PerformEaseOutStep() {
	if (lensAb.vignette.intensity >= intensityEaseInterval) {
		lensAb.vignette.intensity -= intensityEaseInterval;
	} else {
		lensAb.vignette.intensity = 0;
	}
	if (lensAb.vignette.blur >= blurEaseInterval) {
		lensAb.vignette.blur -= blurEaseInterval;
	} else {
		lensAb.vignette.blur = 0;
	}
	if (AudioListener.volume < 1f) {
		AudioListener.volume += 0.05f;
	} else {
		AudioListener.volume = 1f;
	}
	if (audio.volume >= tinnitusEaseInterval) {
		audio.volume -= tinnitusEaseInterval;
	} else {
		audio.volume = 0;
	}
}
```

Joonis 12 Akustilise šoki staadiumi efektide sujuv hajumine

5.2.3 Trummikile lõhkemise staadium

Trummikile lõhkemise mõjustaadium on ideeliselt mängijat karistav mehaanika staadium. Staadiumi mängimine tähendab, et mängija on kogenud trummikile purustamiseks piisavalt kõrget helirõhku. Staadiumi käivitumiseks vajalik miinimumrõhk on 180 dB.

Staadiumi käivitumisel kuuleb mängija 100% valjusega pininat, mis simuleerib tinnitust. Mängija kuulmine halveneb staadiumi mänguajaks 90%. Tugeva valutunde, nägemise hägustumise ja peapöörituse simuleerimiseks kasutab staadium `LensAberrations` klassi `vignette` ja `distortion` komponente. `vignette` komponendi `blur` (maksimum väärtus 1) ja `intensity` (maksimum väärtus 3) atribuutideks seatakse 1. `distortion` komponendi `amount` atribuudiks seatakse 30 (võimalikud väärtused [-100, 100]), `centerX` atribuudiks seatakse suvaline väärtus vahemikust [-0.2, 0.2] (võimalikud väärtused [-1, 1]). Atribuut `amount` määrab ekraanimoonutuse intensiivsuse ning `centerX` moonutuse algpunkti x-koordinaadi. Ehmatuse simuleerimiseks kasutatakse ka EyeBlink efekti, mis käivitatakse `fadeOut()` funktsiooni välja kutsudes.

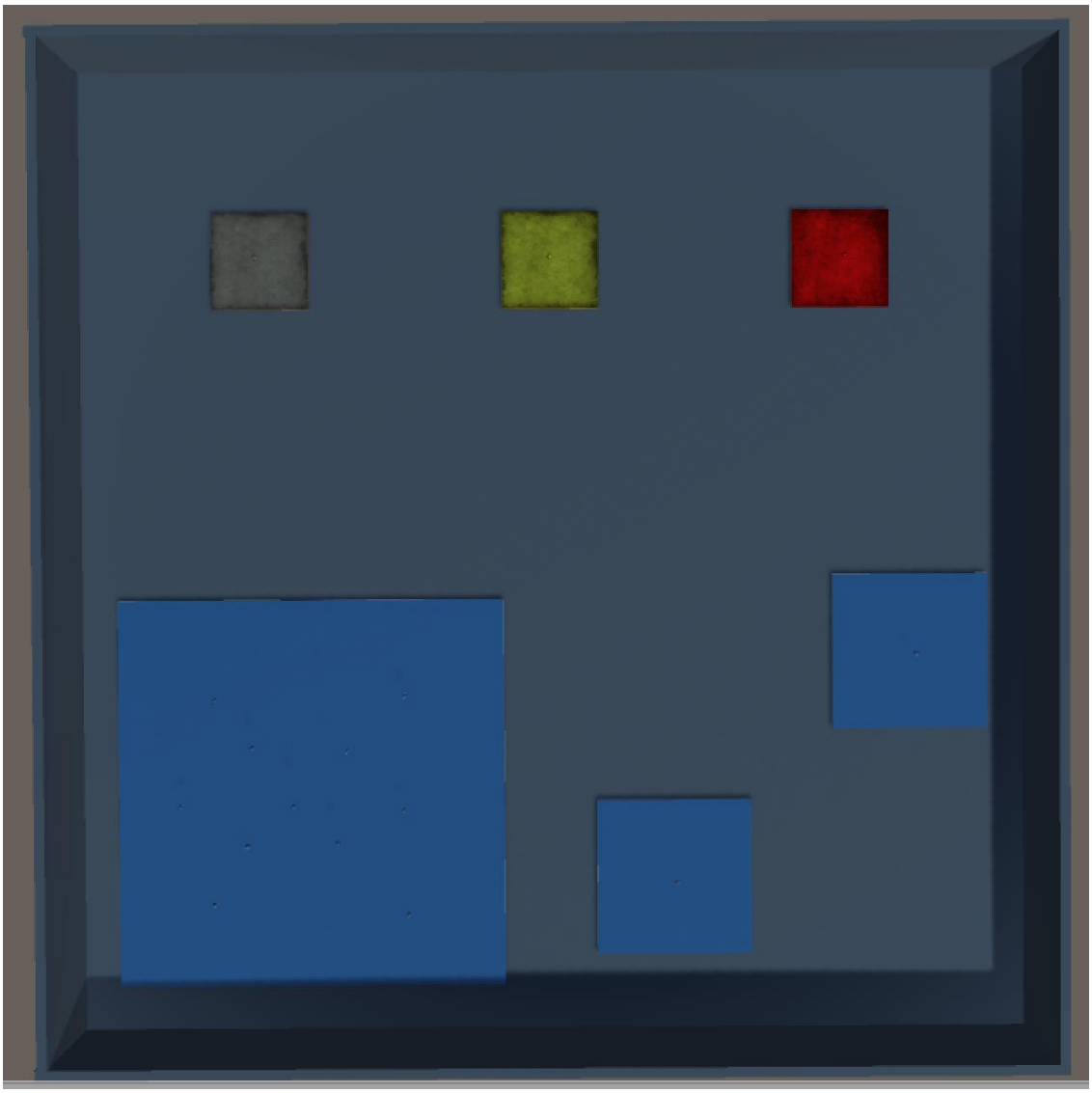
Eraldi väärib märkimist see, et staadiumi mängimise ajal toimub `distortion` komponendi atribuudi `centerX` muutmine iga 0.1 sekundi tagant. Selline käitumine imiteerib peapööritust. Staadium kestab 7 sekundit ning efektide sujuv hajumine toimub kord kaader väljakutsutavas funktsioonis `PerformEaseOutStep()`, mis on välja toodud joonisel 13. Erinevat `-Interval` lõpuga muutujad on konstandid, mille väärtuste võrra vastavaid parameetreid kord kaadris muudetakse. Efektide hajumine on aeglasem kui akustilise šoki mõjustaadiumis.

```
private void PerformEaseOutStep() {
	if (lensAb.vignette.intensity >= intensityEaseInterval) {
		lensAb.vignette.intensity -= intensityEaseInterval;
	} else {
		lensAb.vignette.intensity = 0;
	}
	if (lensAb.vignette.blur >= blurEaseInterval) {
		lensAb.vignette.blur -= blurEaseInterval;
	} else {
		lensAb.vignette.blur = 0;
	}
	if (AudioListener.volume < 1f) {
		AudioListener.volume += 0.05f;
	} else {
		AudioListener.volume = 1f;
	}
	if (audio.volume >= tinnitusEaseInterval) {
		audio.volume -= tinnitusEaseInterval;
	} else {
		audio.volume = 0;
	}
}
```

Joonis 13 Trummikile lõhkemise staadiumi efektide sujuv hajumine

5.3 Testimistase

Testimistaseme loomise eesmärk oli luua kontrollitud keskkond, kus kasutajad helitaluvuse mängumehaanikaga lihtsal viisil tutvuda saavad. Suurema huvi korral on kasutajatel võimalus ka mõnede heliallikate rõhkusid muuta ja nii mehaanika iseärasusi täpsemalt uurida.



Joonis 14 Testimistaseme ülaltvaates ekraanitõmmis

Joonisel 14 on näha loodud testimistaseme kuus platvormi. Igal platvormil on heliallikas, mille käivitamiseks peab kasutaja platvormil seistes vasaku hiireklõpsu tegema. Sinistel platvormidel olevate heliallikate rõhk on muudetav 10dB sammude kaupa klahve F1 kuni F10 vajutades, kus F1 vastab helirõhule 150dB ja F10 helirõhule 250dB. Taseme sulgemiseks tuleb vajutada Escape klahvi.

Hall platvorm on mehaanika ehmatuse staadiumi testimiseks. Platvormi keskel hõljub heliallikas, mille rõhk 10cm kaugusel on 180dB. Nii on ehmatuse mõjustaadiumi käivitumine garanteeritud.

Kollane platvorm on mehaanika akustilise šoki staadiumi testimiseks. Platvormi keskel hõljub heliallikas, mille rõhk 10cm kaugusel on 210dB. Nii on akustilise šoki staadiumi käivitumine garanteeritud.

Punane platvorm on mehaanika trummikile lõhkemise staadiumi testimiseks. Platvormi keskel hõljub heliallikas, mille rõhk 10cm kaugusel on 235dB. Nii on trummikile lõhkemise staadiumi käivitumine garanteeritud.

Suur sinine platvorm on mehaanika testimiseks paljude heliallikate korral. Platvormi kohal hõljuvate üheteistkümne heliallika rõhud on 10dB sammudega muudetavad vahemikus 150dB kuni 250dB.

Väikesed sinised platvormid on mehaanika testimiseks ühe heliallika korral. Ühel platvormidest on plahvatusliku heli allikas, ning teisel kestva heli allikas. Heliallikate rõhud on 10dB sammudega muudetavad vahemikus 150dB kuni 250dB.

Kasutaja informeerimiseks kuvatakse ekraanil abiteksti. Kui kasutaja ei seisa ühelgi kuuest platvormis, kuvatakse talle üldist abiteksti, vastasel juhul aga igale platvormile omast abitekst. Mehaanika mõjustaadiumite mängimise abil kaob abitekst ekraanilt, et mitte varjata mõjustaadiumite visuaalseid efekte.

5.4 Prototüübi kasutusjuhend

Veebilehel https://gitlab.cs.ttu.ee/Kaspar.Eensalu/helimehaanika on saadaval loodud helitaluvuse mehaanika prototüübi programmikood. Kogu loodud programmikood on kaustas ***Assets***. Kaustas ***NoiseScripts*** on kõik ise loodud helitaluvuse mehaanikaga seotud klassid, ning kaustas ***TestLevelScripts*** on kõik ise loodud testimistasemega seotud klassid.

Testimistaseme käivitamiseks tuleb allalaadida veebilehel http://www.tud.ttu.ee/web/Kaspar.Eensalu/HeliTaluvus/TestLevel.zip saadaval olev zip fail. Edasi on vaja käivitada kaustas olev fail ***NoiseMechanicTestLevel.exe***. Seejärel tuleb ette tulevast menüüst valida resolutsioon jms. ning vajutada nupule „Play!“. Edasised juhised toimimiseks kuvatakse rakendusesiseselt.

Mehaanika prototüübi kasutamiseks eraldi projektis tuleb alla laadida kausta `NoiseScripts` sisu ning see oma projekti importida. Edasi tuleb heliallikatele lisada helitaluvuse mehaanika heliallika komponendid (olenevalt liigist kas `NoiseEmit` või `NoiseEmitContinuous` klassid). Samuti tuleb igale heli mängimise väljakutsumisele lisada `Emit()` funktsiooni väljakutse heliallikate komponentidest. Järgmisena tuleb

esmavaates karakteri mänguobjekti lisada `AudioSource` komponent ning hulk mehaanika komponente:

- Vastuvõtja komponent `NoiseReceive`
- Kontrolleri komponent `NoiseMechanicController`
- `LensAberrations`[20] komponent, mis on vaja eraldi importida
- `BlinkEffect` [19] komponent, mis on vaja eraldi importida
- Mehaanika mõjustaadiumite komponendid `Stage1`, `Stage2` ja `Stage3`

Juhul, kui soovitakse mõjustaadiumid ise luua, pole `LensAberrations`, `BlinkEffect` ja mõjustaadiumite komponente lisada vaja. Sellisel juhul peavad loodavad mõjustaadiumite klassid kindlasti implementeerima liidest `NoiseMechanicStage` ning loodud klassid tuleb lisada `NoiseMechanicController` klassi funktsiooni `StagesSetup()`.

6 Kokkuvõte

Käesoleva töö eesmärgiks oli välja töötada realistlik helitaluvuse mängumehaanika, mis oleks ka lihtsasti edasi arendatav ning mängijasõbralik, ja implementeerida välja töötatud mängumehaanika prototüüp Unity mängumootori peal.

Eesmärgi saavutamiseks uuriti helitaluvuse varasemat käsitlust mängudes, kõrge helirõhu füüsikalisi omadusi ning füsioloogilisi mõjusid inimesele. Tehtud uuringu põhjal töötati välja helitaluvuse mehaanika tööpõhimõtted ning loodi tööpõhimõtteid kirjeldav dokument.

Välja töötatud mehaanika tööpõhimõtete alusel loodi helitaluvuse mängumehaanika prototüüp ning mehaanikat demonstreeriv testimistase Unity mängumootori peal. Loodud protüübis toimub helirõhkude edastamine füüsikaseaduste järgi ning loodud mõjustaadiumid imiteerivad kõrge helirõhu reaalseid füsioloogilisi mõjusid.

Kogu loodud prototüübi programmikood on kirjutatud C# keeles ning on internetis vabalt kättesaadav. Prototüübi vaba kasutamine ja edasi arendamine on tugevalt julgustatud ning edasiarendamise võimalusi on palju. Loodud mängumehaanika pakub tugevat vundamenti, millele edasiarendusi luua.

Üks edasiarendamise võimalustest on mehaanika liitmine vigastuste süsteemiga (*damage system*). Trummikile lõhkemine peaks kindlasti ka mängija tervist (*health*) vähendama ning mehaanika kasutamiseks võiks liitmine vigastuste süsteemiga implementeeritud olla. Lisaks saaks mehaanikale lisada helirõhu vastu kaitsemeetmed, olgu see siis kõrvade katmine kätega, kõrvatroppide lisamine vms.

Välja töötatud helitaluvuse mängumehaanika täidab siiani tühjana seisnud koha ning pakub uudse viis helitaluvuse realistlikuks kujutamiseks mängudes. Loodud mehaanika sobib kasutamiseks suvalises 3D esmavaates mängus ning võib erilist huvi pakkuda sõjaväe simulaatorite ja taktikaliste tulistamismängude loojatele.

Kasutatud kirjandus

[1] Battlefield 2 koduleht [WWW]
https://www.battlefield.com/games/battlefield-2 (13.04.2017)

[2] Project Reality koduleht [WWW]
http://www.realitymod.com/ (13.04.2017)

[3] 2008 Mod of the Year Winners feature [WWW]
http://www.moddb.com/groups/2008-mod-of-the-year-awards/features/players-choice-best-released-mod (13.04.2017)

[4] Squadi koduleht [WWW]
https://joinsquad.com/ (13.04.2017)

[5] Jenica Su-ern Yong and De-Yun Wang, "Impact of noise on hearing in the military," *Military Medical Research*, 2015.

[6] Battlefield 3 koduleht [WWW]
https://www.battlefield.com/games/battlefield-3. (14.04.2017)

[7] Battlefield suppression mehaanika wikia leht [WWW]
http://battlefield.wikia.com/wiki/Suppression (14.04.2017)

[8] Call of Duty mängude seeria koduleht [WWW]
https://www.callofduty.com/ (14.04.2017)

[9] Sound pressure and the inverse distance law [WWW]
http://www.sengpielaudio.com/calculator-distancelaw.htm (21.04.2017)

[10] Adding acoustic levels of sound sources [WWW]
http://www.sengpielaudio.com/calculator-spl.htm (21.04.2017)

[11] Elliott H Berger, Rick Neitzel, Cynthia A Kladden (2016). Noise Navigator® Sound Level Database [WWW]
http://multimedia.3m.com/mws/media/1262312O/3m-noise-navigator.xlsx (22.04.2017)

[12] Christopher Bergevin (2014). External and middle ear sound pressure distribution and acoustic coupling to the tympanic membrane. - The Journal of the Acoustical Society of America, 134 (3), March 2014.

[13] David Fernando Rammirez-Moreno, Terrence Joseph Sejnowski (2012). A computational model for the modulation of the prepulse inhibition of the acoustic startle reflex. - Biological Cybernetics, 106 (3), 169-176, April 2012.

[14] John R. Franks, Mark R. Stephenson and Carol J. Merry (1996). Preventing Occupational Hearing Loss. [WWW]
https://www.cdc.gov/niosh/docs/96-110/pdfs/96-110.pdf (25.04.2017)

[15] "4 Tinnitus." Institute of Medicine (2006). Noise and Military Service: Implications for Hearing Loss and Tinnitus. - The National Academies Press, 2006. [WWW]
https://www.nap.edu/read/11443/chapter/6 (25.04.2017)

[16] Acoustic Shock Disorder (ASD) and Tonic Tensor Tympani Syndrome (TTTS) [WWW]

http://www.dineenwestcottmoore.com.au/uploads/ASD_TTTS_guide_medical_professionals.pdf. (26.04.2017)

[17] Eardrum Rupture – At What Pressure? [WWW] http://hearinghealthmatters.org/waynesworld/2012/eardrum-rupture-at-what-pressure/. (26.04.2017)

[18] Unity – Game Engine [WWW] https://unity3d.com/ (12.04.2017)

[19] EYE Blink Effect [WWW] https://www.assetstore.unity3d.com/en/#!/content/61275 (30.04.2017)

[20] Legacy Cinematic Image Effects [WWW] https://www.assetstore.unity3d.com/en/#!/content/51515 (26.04.2017)

[21] Fog of War Gun Sound FX Free [WWW] https://www.assetstore.unity3d.com/en/#!/content/66100 (27.04.2017)

[22] GenMyModel koduleht [WWW] https://www.genmymodel.com/ (23.04.2017)